CHAPTER 1

Digital Layout

Chapter Preview

Here's what you're going to see in this chapter:

- Close look at automated layout software
- Why automated layout only works with certain cells
- Knowing the circuit really does what it should
- How to know in advance if your floorplan choice is good
- Automated programs getting stuck
- Troubleshooting tips
- Which nets to wire first
- Which nets to wire by hand
- Techniques to guarantee rule-perfect layout
- Flowchart of digital layout procedures
- Lots of feedback loops
- How to keep the power moving through big cells
- Chicken or egg wiring and timing circle
- Did you really build what you designed?
- How to build quickie chips for testing

Opening Thoughts on Digital Layout

The majority of integrated circuits built today are large. I mean really huge CMOS digital chips. One chip might have literally millions of transistors in it. It's beyond any single mask designer's capabilities to lay out a chip like that by hand—in any reasonable time frame, at least. Consequently, the majority of large digital chips are laid out with the assistance of computer-aided tools.

Understanding how these automated digital layout tools operate allows you to develop skillful daily habits in your work—even in your analog work. If you understand how the software operates, you can lay out better circuits faster, compensate for software inadequacies, and steer clear of roadblocks before they happen.

Design Process

Let's build a digital chip. In this chapter, we will follow a design team as they progress from concept, through circuit testing, and finally to the actual gate placement and wiring of a digital chip, using a suite of software tools.

Let's start. It's the circuit designer's move first.

Verifying the Circuitry Logic

Circuit designers typically use languages called **VHDL** or **Verilog** to design their enormous digital circuits. VHDL stands for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language, an IEEE standard since 1987. Verilog is another proprietary logic description language. We will use VHDL in our examples.

Circuit designers use the VHDL language to create a chip that exists first as only a database of numbers. The circuit designer's VHDL files are very C-like.[1] The files essentially say, for example, "I want a circuit function that adds two 16-bit numbers together." In this way, the VHDL files describe our microprocessor, our digital functions, or whatever functions we need.

These VHDL data files are then submitted to a computer simulator, which tests the chip circuitry while it is still in software form. The logic functions of the VHDL code run very quickly, much faster than a traditional transistor level SPICE simulation (but not as fast as the real silicon.)

The VHDL simulator needs to have process-specific software descriptions of each logic function it wants to use, such as rise time, fall time, gate propagation delays. This information, as well as other device parameters, is stored as a series of files that the VHDL simulator can access. Along with these electrical descriptions, there are also physical representations of each of the gates that the simulator and logic synthesizer can use. All of these files are collectively known as a **standard cell library** or **logic library.**

[1] The computer language, C.

VHDL Code Segment

```
architecture STRUCTURE of TEST is
 component and2x
 port(A,B,C,D: in std_ulogic := '1';
 Y: out std_ulogic);
 end component;
 constant VCC: std_ulogic := '1';
 signal T,Q: std_ulogic_vector(4 downto 0);
begin
T(0) <= VCC;
 A1: and2x port map(A=>Q(0), B=>Q(1),
Y=>T(2));
 A2: and2x port map(A=>Q(0), B=>Q(1),
C=>Q(2), D=>Q(3), Y=>T(4));
 Count <= Q;
```

The company that is supplying your silicon usually provides a standard cell library. Theoretically, you are given a library, which is perfect, and will stay perfect. However, updates to the library can occur quite frequently. Changes to the library can cause a once-perfect chip to stop functioning, especially if mistakes have been made in the updates. Updating a library mid-project is usually a bad move.

By looking at the results of these VHDL simulations, we can make adjustments to the circuitry before we commit the chip to actual silicon. This is a great saving.

Compiling a Netlist

Once the circuit designer has finished verifying his logic design, he will put his VHDL code through a **silicon compiler** or **logic synthesizer.** The compiler translates the high level C-like code into a file that contains all the required logic functions, as well as how they are to be connected to each other.

The file basically says, "In order to add two 16-bit numbers together, I need 25 gates and here's how they should be connected." In this way, all our logic functions are created and cross-referenced.

At this point, we know what gates we need, and we know how they must be eventually wired to each other. This file, called a **netlist,** will drive your automated layout tools.

Netlist Segment

```
module test ( in1, in2, out1);
 input in1,in2;
 output out1;
 wire \net1 , \net2 , \net3 ;
 AND2_2X U1 ( .Z(net1), .A(net2), .B(net3) );
 AND4_2X U2 ( .Z(net1), .A(net2), .B(net3),
.C(net2), .D(net1) );
endmodule
```

As the circuit designer begins to compile the VHDL code, he will control various switches. The switches control parameters such as area, power, and speed. Depending on chip requirements, the circuit designer might decide to compile the VHDL to prioritize only speed, only area, only power, or some specific combination of these interests. Results will vary depending on these priorities, so he inputs these choices to the compiler before it begins.

Drive Strength

The compiler can create nets that are extremely large. There may be hundreds of thousands of cells on one particular net, for instance. The more cells we have on a net, the more power we need to drive them. If we try to drive too many gates from a single source, we might overload our driving transistors. Our circuit will not work.

Therefore, before we can start layout, we need to modify the netlist to make sure that these large nets are adequately driven. To do this, we replace the cells that are driving the net with cells of identical logic function that have larger driving capability. Driving capability is referred to as the **drive strength,** or **fan out,** of the cell. The fan out number indicates how many devices a gate can drive. A driving gate can be any cell in a library.

For example, we might see that our cell library has 10 or 15 different sizes of inverters. These inverter selections might be referred to as *1x, 2x,* or *4x* inverters. These designations on the inverters refer to the drive strength of each inverter. Since a *1x* can typically drive two gates, a *2x* would drive four gates, and a *4x* would drive eight gates.

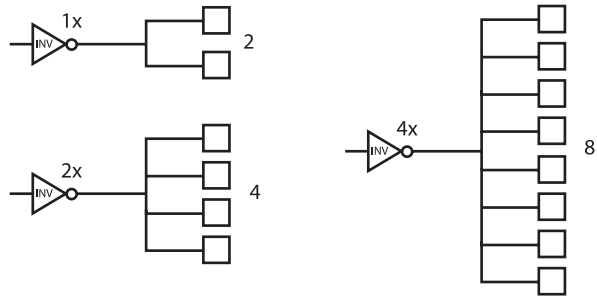


Figure 1–1. *One inverter drives two loads, so a 2x drives 4 loads, and a 4x drives 8 loads.*

You might wonder why we don't just build one huge gate to cover all circuit eventualities. We could do that. However, we would waste circuit area and burn more power than is necessary. The wisest technique is to be sure that you can drive what you need, and no more. So, during the compilation process, the compiler examines the number of gates on each net and adjusts the size of the gate driving each net accordingly. If the net is too large to be driven by our maximum drive strength device, the compiler will break the net into smaller sections that are easier to drive.

Buffer Cells

If the compiler breaks a large net into smaller, more easily driven sections, it will insert additional gates to drive each smaller newly created net. These extra gates were not part of the original logic. The circuit designer did not add them. You did not add them. The computer made the decision by itself.

These extra gates are called **buffer cells.** Buffer cells help drive gate and wiring capacitance. A buffer cell has no logic function associated with it. Whatever logic signal is fed into the buffer cell appears at its output.

In the next section, we will see an example of how the compiler uses these concepts to drive a large clock net.

Clock Tree Synthesis

Most digital circuits have a clock waveform that clicks away in the background. Every function is synchronized to that click. The wiring nets for this clock timing signal are called **clock nets.** A clock net is usually very large. Typically, the net connects to thousands of gates.

It is impossible to create a cell with enough drive strength to drive all the gates on a clock net, so we have to do some extra work to get the clock net to function. We split the clock net into smaller sections and add buffer cells, as mentioned previously. The net is split into a branching-out pattern, called a **clock tree.** Establishing the tree is called **clock tree synthesis.**

To illustrate how the tree concept works, let's look at a small example. Let's say a certain clock net has six gates on it, and the maximum drive strength our library offers is a fan out of only three. Therefore, we cannot expect one gate to drive the entire clock net. So, we break the net into two smaller sections, and drive each section separately.

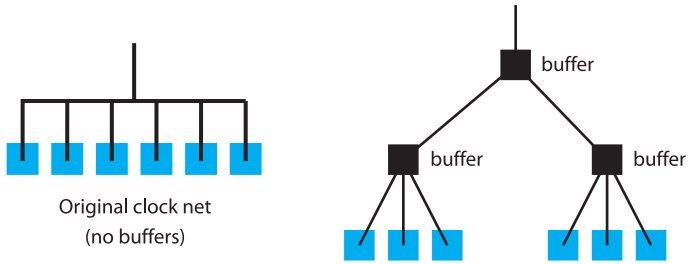


Figure 1–2. *Adding buffers to smaller sets of gates to help drive the signal.*

You can see in Figure 1–2, that the compiler added two lower level buffers to the circuit, one buffer to drive each set of three gates. The compiler also added another higher level buffer to drive the two lower level buffers. So, three extra cells have been added to our circuit.

If our clock net was even larger, the compiler would continue branching in this manner, splitting the net and adding additional buffer cells, each one driving no more than three others. You can see how this would form a very large tree with many levels.

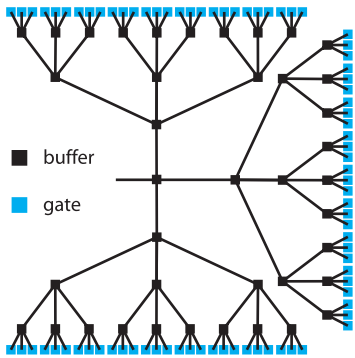


Figure 1–3. *Large nets are broken into many smaller sections that can more easily be driven.*

With a large number of added buffer cells, the extra cells will introduce extra delays that were not accounted for in the original simulations. Not only that, but other large nets may require this same sort of tree synthesis as well, adding even more buffer cells, also creating delays. Therefore, once the clock net is synthesized, and any other large fan out nets are buffered, we need to re-simulate our design using the compiled net list. Compiling creates a need to re-simulate. This sort of iteration is common in chip development. The good news is that it is not a never-ending story. At some point, you will have a finished netlist.

We are now ready to start the layout process. We begin with floorplanning.

Layout Process

We are now ready to use a suite, or package, of automated software tools called the **place and route tools.** Place and route tools cover the gamut of higher level and lower level software assistance leading to your final layout. As the name implies, these programs generally *place* the gates and *route* the wires, in addition to other helpful functions.

Floorplanning

The first piece of software we will use from the place and route tool suite is called the **floorplanning tool.** It will help you create areas of functionality on your chip, determine the connectivity between these areas, determine your I/O pad placements, and give you feedback on how easy your floorplan might be to wire. The floorplanning tool gets its connectivity and gate information based on the netlist file, created by the compiler software.

Let's follow the floorplanning tool in more detail, beginning with your initial decisions.

Block Placement

Typically, your chip will be divided into various functional areas. For example, if you are working on a large digital chip, there might be a microprocessor unit in your chip (MPU), perhaps a floating point unit (FPU), maybe a RAM block and a ROM block.

Where you locate each area of functionality is your decision, not the computer's. You might say, "Ok, all of the gates for the microprocessor I want in the bottom left hand corner. All the gates for the RAM I want in the top right corner." And so on. You will have a chance to change these decisions later, once you see how your decisions might affect your layout, particularly the wiring.

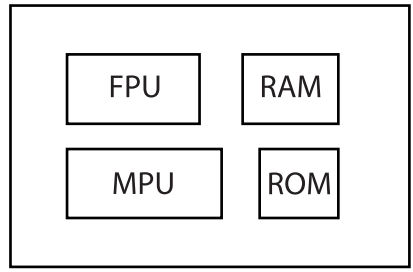


Figure 1–4. *Chips have well-defined areas of functionality.*

Gate Grouping

Once your areas of functionality are specified, the first task you would want to do is gather together, to some degree, the gates used in each block. You would not want FPU gates scattered throughout the ROM or RAM blocks, for example. Associated gates should all be located near each other.

The floorplanning tool begins by helping you gather your gates together. The exact placement of each gate is not determined at this point. We do not yet need this level of detail. Besides, we might be changing our block placement decisions at some later point. So general vicinity placement is good enough for now.

Block Level Connectivity

Next, your floorplanning tool will help you place the input and output (I/O) cells of your chip. For instance, you would want all the inputs that go to the FPU close to the FPU block in the corner. To help you with this, some tools will actually place the I/O cells in the appropriate areas automatically; other tools will provide graphic feedback for you based on your placement decisions.

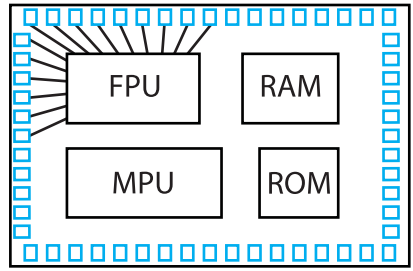


Figure 1–5. *Inputs and outputs can be located near their appropriate cell block.*

The floorplanning tool also shows basic wiring connections that must travel between blocks. It will show connections between the FPU and the RAM blocks, for example.

The automated software programs such as the simulator and place and route tools make choices a conscientious human with enough time would make, given the same information. (We hope.) However, you can see that constant human intervention and monitoring are essential.

The software never operates without human supervision (you). You have broadly defined where you want your high levels of functionality and your inputs and your outputs. You have predetermined some basic layout instructions for the software, depending on the chip specifications, the size of the final package in which they will be placed, the specific circuitry, and ultimately, on your understanding of how the software operates.

The tools never run completely by themselves. The human brain must oversee the working of the tools or the tools become useless.[2]

Using Flylines

Typically, the floorplanning tool will show you all the wiring lines coming from each block connecting to the I/O pads and to other blocks. All these myriad of wiring lines are what most tools call **rat's nests** or **flylines.**

As you click, drag, and resize blocks around your computer monitor, you will see all these wiring connections moving around in real time with your cursor.

As you drag your blocks around with your cursor, watch the lines. If the lines become badly crossed-over and generally messy, you know that it will be tough to wire the circuitry. If there are no cross-overs of the flylines, then it will be easy to wire.

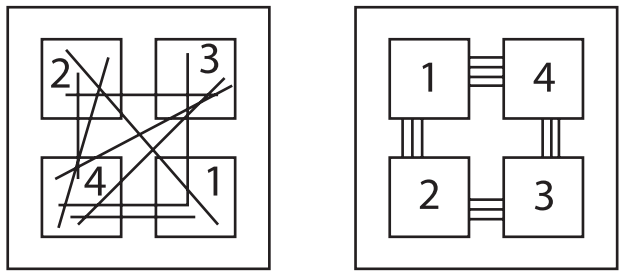


Figure 1–6. *Neat flylines indicate good floorplanning.*

[2] Just as spell checkers knead a human aye.

You will make changes to your block floorplan so that your rat's nest eventually looks as clean, nice, and wireable as possible.[3] You might decide to relocate entire areas of functionality. You might bring one small block across to the other side and fit it between two larger blocks. You might bring a center block to the outside, or an outside block to the center.

When you finally have a block diagram which gives you nice, simple wiring, you save your floorplanning output files.

Timing Checks

Since the final floorplanning tool output files specify where the gates will be generally located, the placement tool roughly knows how long all the wires will be. These wiring length estimations are based on the physical dimensions of the digital library.

Using this information, your floorplanning tool can output an estimated wire length file that goes back into the digital circuit simulator. You now can run some simulations to determine how your estimated wiring lengths will affect your digital circuit. You must check the possibility that long wires will slow the circuit signals too much, affecting the circuit timing.

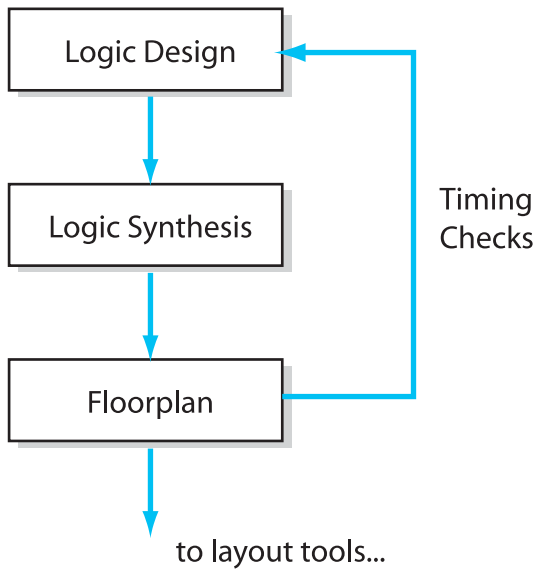


Figure 1–7. *The floorplan/timing loop.*

[3] As Chris always says, "I don't care if it works, as long as it looks good."

If the wire lengths are indeed overly affecting the circuit timing, the designer will need to modify his design, based on your floorplan. He will change the netlist. He might place higher-powered cells in the block to drive the extra wiring capacitances, for example. As the designer works to better organize the design, not only is it easier to wire, but you will find the chip operates better in the end.

You go around this floorplan and timing check loop a couple of times. The simulator will eventually let you know when you have met the timing criteria.

So, at some point, you decide you finally have a good design. You then move on to cementing your devices in place, so to speak. The fine-tuning now begins that we have been putting off.

Placement

We can now nail down the exact positions of all the logic gates within each block, using a placement tool. In the next section, we will then connect the gates with an accurate and final wiring plan, using a routing tool. The placing tool and the routing tool are two of the many software programs that make up the generally named place and route tool suite.

The task of writing one piece of software to do all the placing and routing functions at once would be enormous. Place and route tools are typically broken into individual pieces of software that address specific areas. One piece of software will perform the placement, one piece will perform the wiring, one the block diagramming, and so on.

Some of the software tools have a nice, fancy front end to them that makes it seem as though one program goes off and renders all the output for the chip at once. But, in the background are still many individual programs feeding their outputs to each other, operating one at a time.

There are various differences in the ways place and route tools appear to the user. However, if you pull back the secret curtain, you will see they all essentially work the same way. It is primarily just the user interface that appears slightly different in each of the various software packages sold.

The placement software starts by selecting one block to work with first. It looks for components associated with the selected block. The tool sees 5,000 gates associated with the floating point unit, for example, and wants to place them.

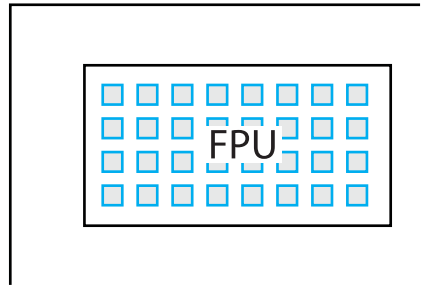


Figure 1–8. *Place and route tools place gates in their exact locations.*

It might look at the 5,000 gates and basically say, "Well, I see from the floorplan that these 25 gates in the netlist are all to be connected together, so I'll place those gates as close to each other as I can. That way I won't have wires going all over the place." The placement software continues placing logic gates based on their connectivity and the output files from the floorplanning tool.

The initial placement scheme can be considered just a first pass. Depending on the tool, device placement might require multiple passes. The tool says to itself, "Ok, I'll come back and improve the placement, look at it again, improve it a little more, and keep working the placement a little at a time until I get a placement I think is easy to route." The final placement will predetermine much of your eventual layout.

There are placement tools available that can make gate placement decisions based upon the signal timing of a design. This placement approach is known as **timing-driven layout** and has quickly become standard practice. The end result is usually far superior to more traditional techniques.

I/O Drivers

Not only are your gates inside the chip placed at this time, but all the **I/O drivers** are placed at this time, as well. These I/O drivers are the special cells that will drive the input signals, provide outputs, contain ESD protection and test circuitry.

These drivers are placed using separate placement tools that know about the I/O rules. They place the I/O pads separately from the placement of the standard logic cells.

Finally, after all these automated tools and all the timing feedback looping, you have the best placement you think you can reasonably make. Next you will route the wires.

Routing

With your gates and I/O cells nailed in place, you will now start to wire everything together. You will take the file that popped out of the compiler, and drop it into another software program. Again, we rely on automation.[4]

Your wiring software has two priority nets—power and clock signals. It will route these two types of nets first since they are the most critical.

After the power rails and the clock signals have been placed, your wiring software will continue to wire the remainder of the circuitry, beginning with any other circuitry you declare as critical.

We will next examine these specific wiring concerns, in order of importance, beginning with the power nets.

Power Nets

There are certain rules for connecting power to logic gates. Wiring must be centered in certain places and run in certain directions. You end up with power rails running through the middle of your gates, as in Figure 1–9.

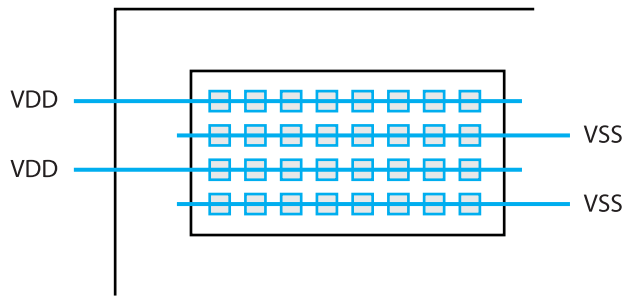


Figure 1–9. *Power rails running through our gates.*

You can intervene in the automated wiring at any time, of course. Perhaps you want to do something special with your power wiring. Maybe you want to move certain blocks around. Maybe you want to put extra power wiring in a particular portion of the circuit because you know the area will demand more power under certain circumstances. It helps if you understand the function of your circuit so that you can make these decisions.

[4] We would not be able to automate so much of our digital layout process without the standardization techniques found in Chapter 2. While these are options available for analog layout, standardization techniques are absolutely required in the digital world. See Chapter 2. (In fact, see all the chapters. After all, you paid for the whole book.)

The wiring software is driven by the net list, which is aware of every component. Therefore, it can tell you when it has completed wiring the power nets.

Strapping

In Figure 1–10, notice the highlighted cell at the far upper right corner, farthest from the VDD input pad. As the power comes into the circuit on the lower left, it must travel through the existing rails. You can see that the supply current has a lot more metal in series to go through to get to our little end cell in the upper right corner.

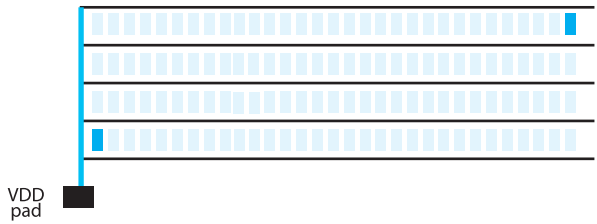


Figure 1–10. *Distant cells see more resistance through the rails due to the length of wire. If only we could lower the resistance to those remote cells.*

The other highlighted cell nearest the pad would see a lot less resistance, being so close. Let's see how we can alleviate this difference.

By laying straps of metal across your power rails, you create a big waffle iron grid of multiple paths. Now it's like having resistors in parallel. The overall resistance reduces.

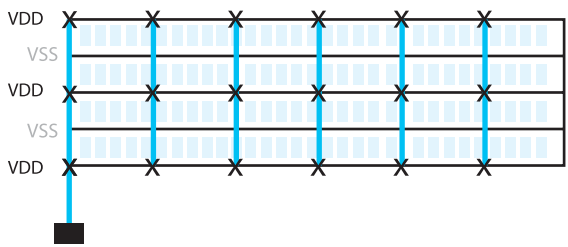


Figure 1–11. *Strapping our rails.*

We have given the current multiple paths. We just blanket the chip with straps and rails to give the chip the most parallel routes for power as possible. The more the merrier.

The older power routers had a uniform spacing rule for power strapping. The power straps were placed every so many microns, regardless of need. You would just say, "Oh, I'll have a metal strap every 60 microns or every 100 microns." Perhaps you thought these were good average numbers, so you would just go with it.

However, newer tools actually look at the drive strengths in the cells before they place their power straps. The router can calculate, for example, that there is a large concentration of huge drive strength cells in certain areas that require more power. So, the router places more power strapping in that area.

Power net wiring is more important than it used to be. More intelligence is being built into the tools to know more about what's going on in the circuits, to give us more intelligent auto-placed and auto-routed chips. It's taking some of the mystery out of the process.

Digital mask designers need to know more about their tools and techniques than they did before we had so much sophistication. They have to understand why the tools are doing what they do. There is more need to understand when the tool messes up, misinterprets, fools itself, or gets stuck. At times we must override the tool's choices. Rules are getting intricate. Tools are getting intricate.

For example, a mask designer might say, "Ok, the power router hasn't done a particularly good job around here. I know this is a very power hungry block, so I'm going to have to go in here and manually put in some extra power strapping."

There can be a lot of manual intervention with digital layout. Using the tools, rebelling against the tools, improving what the tools have routed, finding work the tools left behind, and giving the tools some human touches. People drive the tools.

Clock Net Wiring

Once you have finished running the power rails, your software usually offers a specialized tool just to wire all the clock nets, the circuitry distributing the timing signal. Remember the clock tree synthesis we generated earlier? This is a very critical net.

There are various approaches to wiring a clock net. Each tool has its own particular way of operating.

If you are unfortunate enough to have to hand wire a clock net, you can use the **Central Clock Trunk Approach.** There is usually a clock driver cell that has enough drive strength to drive the top level clock buffers. Place that cell centrally within your design and create a large central trunk that branches out to join to all the clock buffers.

As the net reaches further out from the main driver, it continually splits into more and finer branches. The wire widths at the outer edges become smaller and smaller. The large central region resembles a thick tree trunk, hence the name *Central Clock Trunk Approach.*

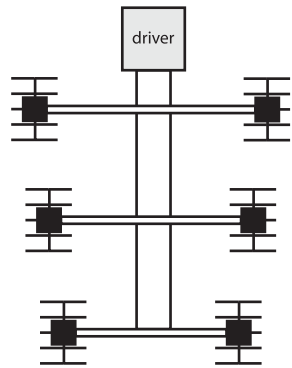


Figure 1–12. *Central Clock Trunk Approach to wiring clock nets.*

This Central Clock Trunk Approach is very easy to wire. The clock signal distributes very easily.

Other Critical Nets

At this point, we turn to any other nets that need special attention. You wire the nets that you are most worried about first, maybe by hand. You should have been given a list specifying the critical nets of the chip. (If not, ask.)

You plug the critical nets file into the auto-router and it goes off, routing the critical nets while wiring is still relatively easy, before we wire the bulk of the chip. As always, you can intervene at any time until all the critical nets are exactly what you want.

Remaining Nets

The last thing you do is wire the rest of the circuitry. If you are lucky, the tool will know how to automatically wire everything else on its own. Then it says, "I'm done. I'm finished."

Finishing the wiring can take days on really big chips. Typically, the software will go away and work silently on its own. You just sit there and play backgammon. Then just as you discover that elusive guitar chord you've been trying to find, the tool beeps and says, "Finished."[5] Seriously though, there is usually plenty of time to do more productive things while your software is running.

Finishing the Wiring by Hand

When the auto-router finishes as well as it can, you might end up with all the wiring hooked up on your chip. Or, more likely, you might not. Usually, the routers just cannot get to some areas. It can wire itself into some blind alleys and helplessly sit there thinking it is finished. You look at the auto-router's work and ask it, "Ok, how many of the nets couldn't you do?"

Usually you will have to use the human eye, break some nets, and move stuff around, in order to complete the wiring. You can count on some final intervention on your part to finish all the nets and do all the wiring the computer could not do.

For example, the router might say there are five nets it just could not complete. It will highlight for you where these open nets are located. Zoom into the area where the pin is located that it says it cannot wire. Usually in these cases, you will see that the router has placed the wire close by, but it stopped because there is a bunch of stuff in the way. Usually it's something really obvious—a bunch of nets that were pre-wired earlier may be in the way, for example.

Nine times out of ten it's pretty straightforward to move some wires out of the way or do a little re-wiring to free up some area. Then you can normally run the wire where you need to.

It's unlikely the auto-router will do 100% of the wiring in one pass.

With experience, you will often manually pre-wire some signals because you know the auto-router will have problems with them. It saves a lot of time as you learn to preempt these problems in advance.

There are some times, if the chip dimensions are too small, you may not be able to wire the chip at all. You just cannot place 5,000 wires in a tiny 100-micron space.[6] Or, you might have lots of room, but there might be just too many crossed nets, or a bad floorplan. For whatever reason, sometimes you

[5] Chris has a thank you on our band's CD to a certain tool that allowed him the time to practice his guitar licks at work. It's a tool with a reputation for being slow. I'm not sure if they still use that tool, but Chris' guitar licks are still hot and polished.

[6] Yet.

just cannot do it. In that case, start over. Go back to your floorplanning and begin from scratch.

Eventually, you really are finished. All the gates are wired. At this point, you have the computer spit out a wiring file that says, "Ok, here are the real wire lengths, and the real wire capacitances." You have them in hand. No more fine-tuning. No more estimating. This is the real stuff.

Prefabricated Gate Array Chips

All of the above techniques will also be used on a special type of chip, called a **gate array.** A gate array is a predefined and partially prefabricated chip, literally an array of gates. The semiconductor manufacturer processes wafers up to just before the metal is deposited, then stops and stores them until needed.

You will still use floorplanning tools, placement tools, and wiring tools. However, you are not placing any diffusion or poly, only metalization and contact layers. This type of chip is useful for prototyping circuits. Instead of waiting twelve weeks or more for your chip to be fabricated, the processing time for a gate array only takes a few weeks, since you are only fabricating the back end of the process.[7]

Typically, you can choose from only a few fixed-sized gate array die—small, medium, and large. The manufacturer tells you a certain gate array will handle, say, 5,000 gates, this other one will handle 10,000 gates, and this last one will handle 50,000 gates. You select from the offered sizes.

If you have a 30,000-gate design, but your supplier does not have a gate array designed for 30,000 gates, you have to move up to the next size, the 50,000-gate array. You waste a lot of space. Also, the 50,000-gate might have provision for 150 I/O pads, but your design only needs 50 I/O pads. So, with all this extra baggage on the gate array, you end up with a much bigger chip, but you get them built quicker.

When you have finished your logic design, are happy with it, and you have proven your concepts using a gate array, you then convert the design into a real fixed device, a full-custom chip that goes into production.

[7] This reminds me of the old cardpunch days, taking a week to see how my Fortran programs turned out. And I'm talking a week for each attempted fix to the program. Imagine. You kids don't appreciate how we trudged ten miles in the snow, marched thousands of cold cement steps to the dark basement, to drop off a batch job at the one room-sized computer on the UC Davis campus. (Well, it snowed in the California central valley pretty bad in those days.) Now you can make a whole chip in that time.—*Judy*

Verification

It's time to verify that what the team originally had in mind is what was actually built. You will now verify the design.

Design Verification

You feed your new wiring file back to the simulation people again. This time they simulate with the actual wire data. No more estimation. The wiring is physically in place this time.

If you are lucky, everything's happy. If your tools are good enough, and your models are good enough, then the wiring will be fine. If you are not lucky—for instance, the router has done a bad job, or you have done a bad job—the circuit designers may have to change the design again.

If you need to redesign, it is not necessarily a total loss. The team may be able to keep some of the original work—just merge in some new logic, rip out some cells, replace some cells, and re-wire. However, there are times you might have to actually redo the whole enchilada.

The digital mask designer makes these changes. It's a matter of running the tools and software. You typically do not pull out one or two little gates and replace them by hand. The software is intelligent enough to merge in new connectivity and do the majority of the restructuring.

As you can see, a good layout person has to know a lot about the tools and what they are trying to do in the circuit in order to be successful. A good mask designer can put a good floorplan together from the start, especially if they know some details about how the circuit functions. They can preempt problematic issues that they know might develop.

Digital layout is a skill. It is not just a case of pressing the buttons and playing backgammon all day.

You have to know where the tools are going. You have to foresee issues that cause the tools to stumble, and work in advance to prevent those issues. You have to nurse the chip through these processes. You have to direct the flow.

As you master good layout skills, make them part of your daily habit. Soon it all just seems to happen by itself.

Eventually, when all the timing is done, all the wiring is placed, and the chip has been re-simulated, everyone is happy. You have finished the top-level layout of your chip. You have converted a database from a conceptual format into a real mask design.

Physical Verification

Up to this point, you have not been working with real transistors. You have just been working with little boxes with points on them that say, "This is the input. Here is the output. We do not care what is inside." There are no real transistors in those boxes.

GDSII File

To complete your mask design, you take this abstract, high-level database, and replace the boxes with the real logic gates. You merge the real components from real libraries with the wiring and placement data from the place and route tools. As you replace the abstract components with real library components, you produce a **GDSII** stream file of your chip. This is a file that has all the components, all the wiring of your cells, all your vias, everything.

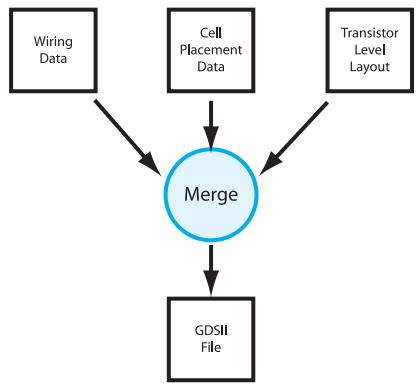


Figure 1–13. *The data is merged with the actual transistors to produce a GDSII stream file.*

DRC and LVS Checks

By the time you produce your final GDSII output file, the chip has gone through the wringer. Typically, there have been so many operations performed on these databases, from start to finish, that you no longer trust the final output. Has the software kept up with all the loops and changes well enough? Those nice software people do make mistakes, you know, especially when we run lengthy and complex iterations on a project.

Once you get your GDSII stream file, you then will want to run checks to be sure that the wiring is correct and complete. At this point, we check all the process design rules using Design Rule Check (DRC) software.

At the same time, we also check that the wiring and transistor connectivity correctly match the connectivity defined in our netlist. We use Layout Versus Schematic (LVS) software to perform this connectivity check.

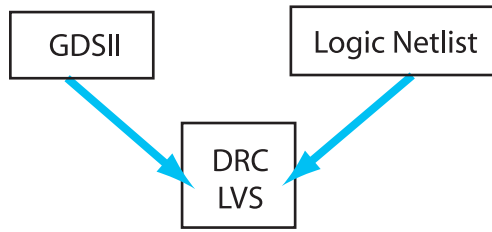


Figure 1–14. *How did we do? Let's run a check.*

Library Management

What could possibly go wrong in this highly automated, computer-controlled nirvana? Plenty.

The place and route process relies on synchronization of all the various database files for all the tools. It relies on the fact that the layout for your inverter is correctly represented by the schematic for the inverter.

Having various representations of the logic gates creates very complicated library management issues. For example, you could have a cell called *Inv1x*, but it might not LVS to the schematic of *Inv1x* because someone has made a mistake in the compiling of the library.

As another example of a possible problem, the place and route tool might not have up-to-date information. The place and route tool might think everything is fine according to what it knows. However, when you merge your cell layout data with the wiring data you could find wires just dangling in free space. The problem could be, for example, that the representation of the inverter the place and route tool uses shows a pin one grid lower than the representation in the GDSII file.

Good library management for digital place and route is very important. An incremental change to one cell in a library could cause six or seven files to require updates in multiple directories. There are so many files, which have

to be 100% perfect and synchronized with each other, mistakes are easily made.

However, if your tools are good, flows are good, and libraries are good, your design should pop out DRC and LVS clean, first time out of the box. If not, you have a bit more work to do. We have a section later in the book to help you resolve these errors. It is not always as difficult as it may first appear.

Summary and Flowchart

So, there you are. That's how large digital chips are created. You have created your netlist. You have placed your gates and routed the wiring. You have considered critical elements of your circuit. Finally, at some point, you are finished.

Tape out. Clean chip. Pass Go. Collect $200.

That was really something, wasn't it?

I feel a flowchart coming on... Yes, let's look at all that again as a flowchart.

The flowchart represents the steps of a typical digital layout process. Follow along in the chart as you read each paragraph below. (See Figure 1-15.)

First you design your logic, synthesize it, draft a floorplan, and then do some timing checks around that loop for a while.

Then you run your place and route tools and do some timing checks. You keep going around that loop until you are happy. You may even have to go around the floorplan loop a couple of times again.

At this point, you need the digital libraries. By library I mean the AND's, OR's, input cells, output cells, and all the real transistor level components. You need the digital libraries to make your final GDSII file.

Finally, you run the DRC and LVS checks against the netlist you started with, out of your logic synthesis. Then you get the final chip done.

You can see from the massive use of computerized tools, that your digital library devices must be put together with a lot of standardization. Your cells, your wiring, and your logic must all be pre-designed with computerized placement in mind.

Analog mask designers get to use standardization techniques as options when appropriate for their projects. However, digital mask design absolutely

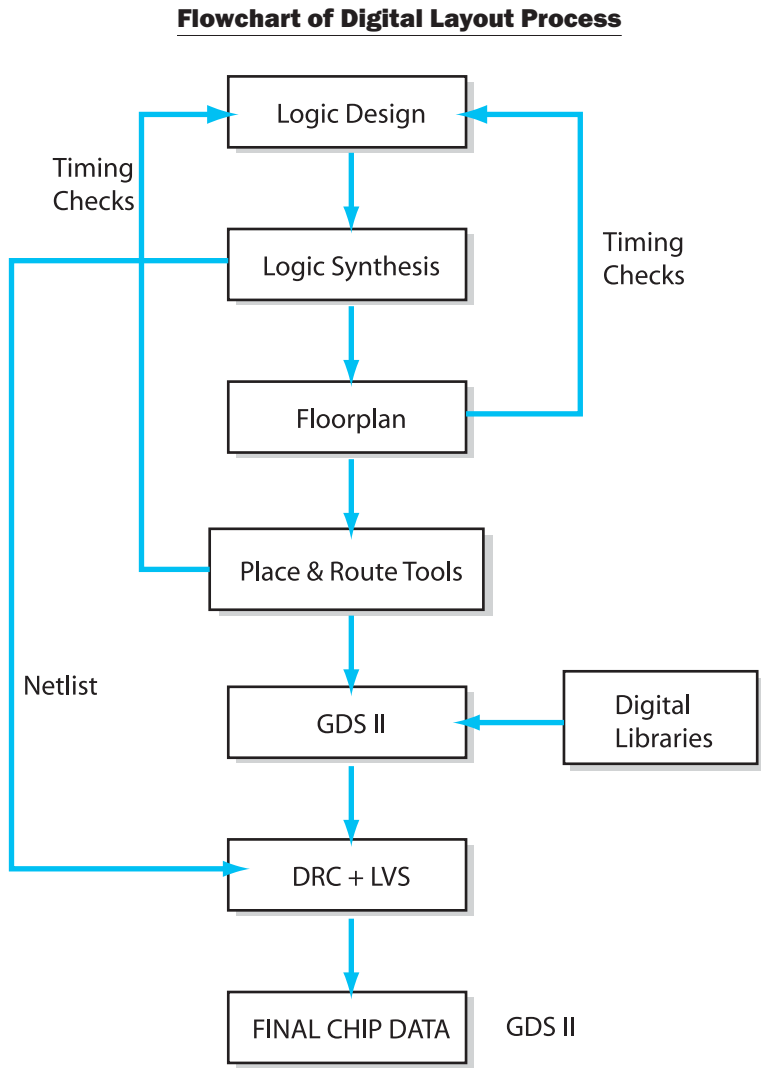


Figure 1–15. *Our original design undergoes many iterative loops and manual interventions before we have our final data for the chip layout.*

demands use of standardization techniques. Automated software tools would not work without it. That's where we go in the next chapter.

Closure on Digital Layout

As you can see, automated digital layout can be very complicated and convoluted. In reality, though, all we are doing is using very simple concepts, just on a massive scale.

Most companies that use these tools to design large digital chips also have very well documented procedures that are designed to take you through every stage of the layout process. There are so many feedback paths and decision points in a large digital design that it is otherwise impossible to keep track of where you are in the flow. You can use the procedures to help you take the project a step at a time.

On the face of it, digital layout may appear as though we are just turning a handle and spitting out fully finished chips. However, there is a lot of manual intervention required. The mask designer can stop the process, make adjustments, preset priorities, work ahead of the automation to avoid problems before they happen. The more you know about the automated procedures, the more effective you will be at the helm.

Here's What We've Learned

Here's what you saw in this chapter:

- Place and route tools
- Floorplans and flylines
- Priority nets
- Feedback loops in the design and layout process
- Troubleshooting automated procedures
- Placing buffer cells according to drive strengths
- Gate arrays
- GDSII from place and route tool
- Netlist from logic synthesizer tool
- DRC and LVS checks
- Flowchart of digital layout procedures

CHAPTER 2

Standard Cell Techniques

Chapter Preview

Here's what you're going to see in this chapter:

- Why standardization is mandatory in digital layout
- Advantages of standardization techniques in analog layout
- Why we butt some cells together
- Tips if you have few metals
- Tips if you have lots of metals
- Digging channels for our wires
- When to run big power lines
- Getting signals in and out of tight spots
- How to guarantee a good fit between cells
- How to guarantee rule-perfect layout
- How to save time
- How to protect your gates from zaps

Opening Thoughts on Standard Cell Techniques

In order for automated layout tools to be able to place and connect components, you need rules. Rules for the cells. Rules for placement. Rules for connectivity.

Think of plastic Lego bricks for a second. These uniform little play blocks all have connections in exactly the same places, spaced at predictable intervals. They all fit on that flat, square sheet of green plastic with gridded bumps that we all called the lawn or the garden.

The blocks all fit with each other. They all fit on the grid.

The reason Legos fit so well together is because the blocks are all standard length, standard width, standard height, with connection bumps in standard places. You cannot just throw any random block design into this process.

Cell libraries that are designed to be used with automated layout software follow guidelines and rules the same way as Lego bricks. Standard cells also need to fit together in predictable patterns. We use a variety of standardization techniques to build a library of specially designed cells just for this purpose.

These techniques are useful in analog mask design, even if you are not using automated layout software. However, because these techniques are almost mandatory in digital layout, the emphasis and examples in this chapter will be primarily from the digital mask design world.

Let's have a closer look.

Standardized Grids

This standardized layout system enables automated wiring and guarantees overall operable placement of standard cells, by aligning everything on a standard grid. This grid is like the flat square sheet of green plastic used as a Lego base. This is what we will discuss first.

After we talk about this grid system, we will then discuss the standard cells that operate on this grid. If we design our cells uniformly, watching grid alignment and rules, we can allow an automated tool free reign over placement and wiring. We will know that our circuit will be built correctly regardless of the software decisions.

Grid-Based Systems

The classic router software is **grid-based.** A grid-based router has two constraints. Wires can only be certain widths, and can only be placed on a predefined coordinate grid. You cannot just freely design anything you want in a grid-based system. You must follow grid alignment rules.

Determining Grid Size

Let's suppose the process manual states the minimum Metal One wire width[1] is 1 micron. It also says your minimum spacing is 1 micron. Therefore, our mini-

[1] Minimum Metal One wire width: Say that five times real fast!

mum distance for two parallel wires would require a distance of 3 microns. One micron for each wire, and 1 micron for the space between the wires.

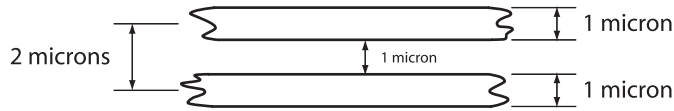


Figure 2–1. *Coarse grid example. Minimum wire is 1 micron. Minimum spacing is 1 micron. Therefore, our two wires use 3 microns, and we have established center-to-center grid spacing of 2 microns for this process.*

As you can see in Figure 2–1, the distance between wire centerlines is half the top metal width, plus the minimum spacing between metals, plus half the bottom metal width. In this example, that's a total of 2 microns. The centerlines of the wires will be 2 microns apart. This makes a 2-micron grid.

Determining our minimum wire widths and the minimum spacing between wires dictate how coarse or fine our grid will be. The wider our minimum metal, or the further our metals must separate, the larger our grid will be.

In our example, a grid spacing of 2 microns guarantees metalization that will always be design rule correct for this process.

Design rules determine grid size.

We can do the same for Metal Two and any other metals that are in the process.

We have now defined x (horizontal) and y (vertical) sets of grid lines across the whole of the chip. The grid-based router can only place wires along our grid lines, from intersection to intersection.

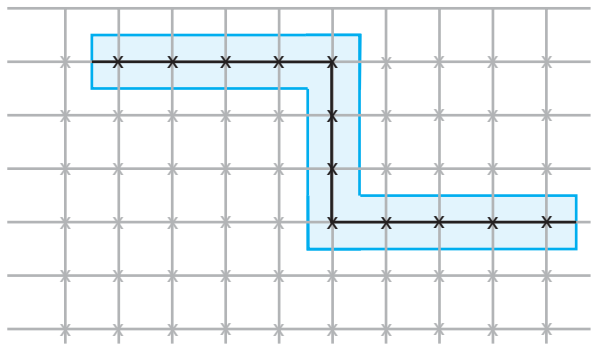


Figure 2–2. *Auto-routers only run wires which are centered on both x and y gridlines.*

Following grid lines limits our wiring possibilities, but allows automated tools to do the routing for us, at least as well as they can. A grid-based router relies entirely on these grids.

Can you have different grid sizes on different layers in a grid-based router?

You can have different sizes of grids for each layer, but that could be awful trying to line up all our contacts, our vias, matching both horizontal and vertical grid intersections in each layer.

Try it. Draw one grid and overlay a slightly differently sized grid. You will see that hardly any intersections or lines lie directly on top of each other. You would have almost nowhere to connect.

After trying that exercise you should appreciate what rule-based routers can do. (See next section.)

Rule-Based Routers

In modern processes, any two grid pitches in a process are typically not the same dimensions. Metals and spacings on different layers have different minimum widths. If we forced all grids to the same dimensions, we would have to universally use the largest requirements of any layer of the chip. This would waste space in the other layers. For example, if a layer can tolerate 1-micron spacing, why force it to use 2-micron spacing? Let's see how to solve this problem.

An improved version of the grid-based router is called a **rule-based router.** For each layer containing wiring, instead of using a fixed grid, the computer actually uses the real design rules for that layer.

Most people use the grid-based approach since it is simpler to use. It makes the router's job easier. It doesn't have to integrate an extra set of rules for each layer. With the good old grid-based router, we just tell the software where the grid is, and it always places its wire on that grid. The wiring job is a lot easier, so the software is a lot simpler to write.

However, you can get much more compact wiring using a rule-based router, since Metal One width and spacing might not be the same as Metal Two width and spacing. Some wiring levels might be more squishable. They might have tighter grid spacing, so you save space in your layout.

Directional Layer Technique

Here you are with your computer happily wiring away all over the gridlines of Metal One. However, if you try to wire everything on your chip completely in Metal One, you will get landlocked.

It's like playing that pencil and paper game with a friend, trying to trap the other player on a grid. I believe there was a game in the movie *Tron* between two players on motorcycles that was much the same idea. The two riders were essentially drawing grid walls as they raced, in order to trap each other within the walls. Very fast. Very good wall splats. Cool movie.

Imagine trying to wire hundreds or thousands of devices all on the same layer. If you run your wiring around haphazardly, you will find yourself blocking other wire paths quite quickly.

In order to wire so many components, you must come out in Metal Two somewhere. We need another layer to save us from becoming trapped in Metal One.

We will connect our layers with what we call a **via,** meaning a passageway between metals. Since we are using a grid-based system, the vias from Metal One to Metal Two can only exist on this grid, as well. Well, duh, that's where the wires are.

Your technology will drive the grid spacing you can use, but let's continue using our example of a 2-micron grid. As we mentioned, our Metal One wires can only live on this particular grid. Let's constrain our Metal Two wires to these gridlines as well.

If we were to use this grid on both metals, and not constrain ourselves further, we could end up with Metal One and Metal Two still darting in haphazard directions.

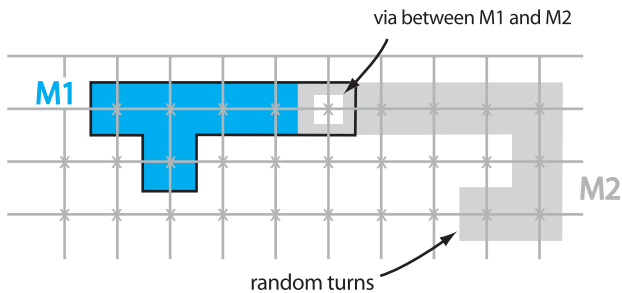


Figure 2–3. *Have you ever played a trapping game like this with a friend? It's easy to find yourself running into a dead end in no time, even in two layers.*

Running rampant all over two layers of metal does allow more wiring possibility than running rampant all over a single layer. So, having a second layer does help us move about between connections. But we still can trap ourselves in no time.

Running haphazardly requires more and more layers of metals, to continue escaping dead ends. However, we just might not have a few thousand metal layers available. For the complex requirements of larger chips, there is a better method.

How about running all Metal One horizontally and running all Metal Two vertically? Ingenious. Simply ingenious.[2]

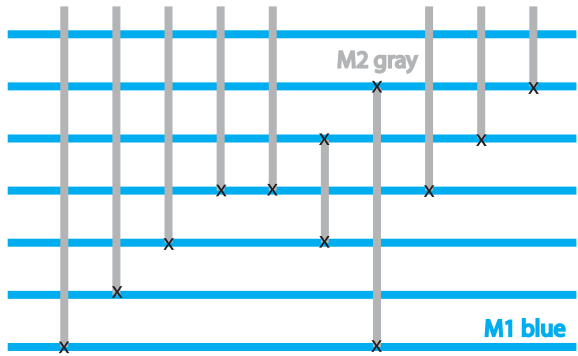


Figure 2–4. *Technique allowing wires to run in complex knots using only two metals, without becoming trapped. Metal One runs horizontally only. Metal Two runs vertically only.*

Whenever you change directions, you will change metal layers. When you need to wire from one point to another, first run horizontally in Metal One, for example. Then change to Metal Two and run upward to your connection point. The horizontal wires will not run over each other, and the vertical wires will not run over each other. Using this technique, you can run all your wiring without becoming trapped, using only two layers of metal.

Do you think we still use Metal Two for our vertical run if we are only moving, say, one or two grids? In Figure 2–5 we see five parallel wires that all jump up one grid in about the same general area. What do you think about skipping our Metal Two for these short vertical runs?

[2] Airliners use the same concept. North-South flights and East-West flights run at different altitudes; different layers, if you will.

Figure 2–5. *What do you think for very small jumps, stay in Metal One, or still use our alternating metal technique?*

If the vertical jump is only one or two grids, you generally stay in the same metal. (Congratulations if that was your answer.)

Rule of Thumb: Don't bother changing metals for short jumps.

Basically, popping upward into Metal Two is to allow more horizontal wiring to run in the same direction between our signals. However, if our wires are only one grid apart, there is no way another wire could ever be placed between any two of them anyway. Therefore, we do not gain any extra wiring advantage.

Not only that, but there is a possible disadvantage. Every time you use a layer of metal, you block any other wires from being able to use that layer of metal in that location. There is the possibility that you might need that same area for some other wiring. You have taken up the option of using Metal Two by having placed all these little, short jumps that, as it turns out, are not providing any extra wiring allowance between them anyway.

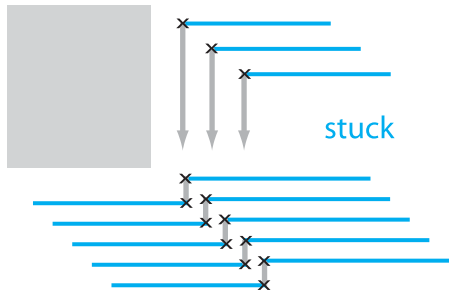


Figure 2–6. *Metal Two is already used in this area.*

You also get some reliability improvements by using this technique. The vias we introduce for such a small run can potentially introduce high resistances. Not only that, but vias can sometimes not etch properly. So, for small jumps of only one or two grids, it is not as reasonable to use your second layer of metal. Stay in Metal One.

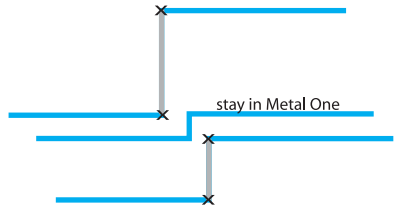


Figure 2–7. *Stay in the same metal for small jumps.*

Library Rules for Grid-Based Systems

When we use a grid-based router, we devise a set of rules for everything dealing with our layout. Typically, we construct an entire standard cell library according to these rules. Every cell, every inverter, every NAND gate, absolutely everything conforms to the rules.

Input and Output Alignment

Figure 2–8 is the schematic of a standard inverter cell. The input is marked A. The output is marked Z.

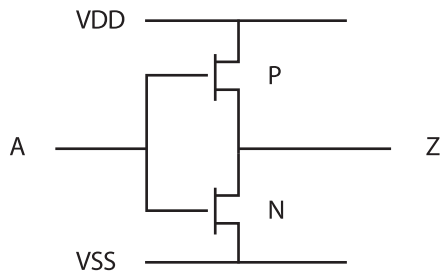


Figure 2–8. *Schematic reference for our standard inverter cell. We need input A and output Z to align perfectly with our grid wires, or they will miss the connections placed by the auto-router.*

Figure 2–9 shows how such an inverter might look in your standard cell library. Notice the input and output, A and Z, located in the center of the cell.

The input and output connections, A and Z, cannot be placed haphazardly. They must be located exactly on the same grid as all the wiring. How else would our wiring attach?

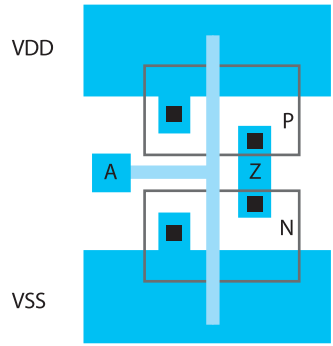


Figure 2–9. *Standard inverter cell. How can we guarantee good hookups for this cell if wires only exist on given gridlines?*

Let's superimpose our grid over our inverter layout, to see that it aligns well with the wiring grids.

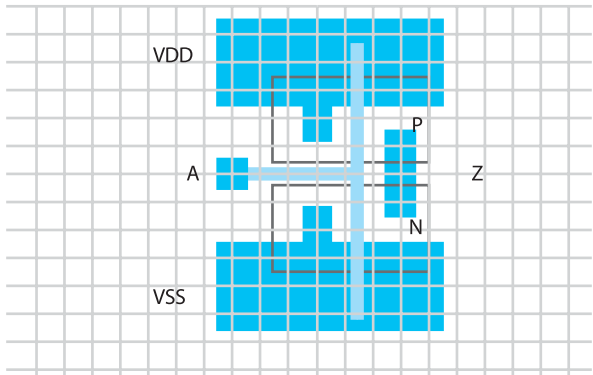


Figure 2–10. *Grid system overlaid on our standardized inverter. Notice that the input and output align with gridlines. This cell will work with an automated wiring program.*

By inspecting the alignment, it appears we should be able to connect A and Z quite easily with wires that only live on the x and y gridlines.

We have to match all our standard cell components to our grid in this same way. The wires, the cells, the intersections—all layout entities need to obey the rules, such as alignment and isolation distances. Otherwise, we cannot guarantee that our automated system will give us a DRC and LVS clean job.

Here's a no-brainer reminder. Not only do we need to make sure that all our library cells conform to the grid rules, but also that the devices inside the cells interact nicely with the grid. It seems common sense, but you would be surprised how easily we can forget to nudge a device squarely onto the grid after it is placed.

This happened recently. We got caught out on it. Someone had not laid out some internal devices exactly on-grid. It was just slightly off. When the auto-router began placing wires, it thought this certain open intersection was fair game for a wire. Plop, it puts the wire down. No problem, it thinks.

However, if the cell had been laid out correctly, then the open grid point would have been far enough away from the internal devices to not cause any errors. We had to re-layout the internal devices to fix the problem.

As a final note, make sure all your inputs and outputs for your standardized cells are not only on x grid, but on y grid as well. Make sure the automated software can find them both horizontally and vertically. This sounds simple, but it is a common oversight.

Fixed Height, Variable Width

If each gate in our library had differently sized power rails, and each gate was a unique height, our wiring would be messy and our software tough to run, even if they all did align to a grid. Notice in Figure 2–11 how the small, medium and large versions of our inverter cause our power rails to wander.

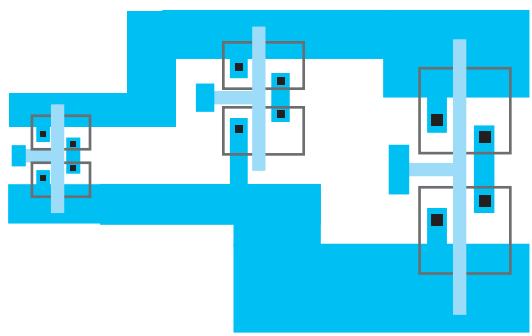


Figure 2–11. *Varying heights of layout items such as cells or power rails causes headaches.*

Therefore, to keep our structure uniform, we force all our gates to conform to what we call a **fixed height set.** A specified fixed cell height drives the entire library.

If we need bigger logic gates with larger transistors to drive large loads, then we just make the cell wider and split the transistors to fit inside the rails. But, we still maintain the fixed library height.

The minimum cell height is typically determined by the size of the transistor, which we obtain through simulation, as well as the grid we have chosen for the library. Typically, you select a height that is a bit larger than the minimum, because you want to have a power rail that is larger than minimum.

A fixed height library has an advantage. If we place all our logic gates side by side, the power rails wire easily. In fact, we could butt each cell right up against the next cell, and connect our supplies for free. They would all line up.

For digital layout, particularly in standard cell layout, fixed height, variable width is the only way we can work because we are constrained by the place and route tool. This method guarantees you will be able to place all these cells next door to each other and be DRC clean. It's a fairly common technique that is used all throughout the digital layout world. Most libraries you see will be fixed height, variable width. Our cells are selected from the library, which is already built around this grid. Just find the one you want in the library and place it.

Digital libraries: fixed height, variable width.

Fixed height, variable width is also a very useful technique to use in analog layout. It's something people do even in full custom analog circuits. If you have a very repetitive design, with lots of similarly sized cells, for example, you may as well use the approach to make your cell placement and wiring easier. You create rows and columns of cells. You get advantages such as contiguous supplies and routing channels, and can treat it as if it was a digital layout.

Determining Wire Gauge

Our 1-grid wire sets the minimum distance between grid lines, as we saw earlier. One wire runs along one grid line. We can also make wires of larger gauges. The power rail in our inverter example is what we call a 3-grid wire. Typically, power rails are either 2- or 3-grid wire.

To build a 3-grid wire, place three single grid wires on-grid, running side by side, and then fill the gaps between them with metal.

Common N Well

Suppose we want to place four gates next door to each other. Typically, we want to place them as close as we can to maximize the number of transistors that we get in our circuit.

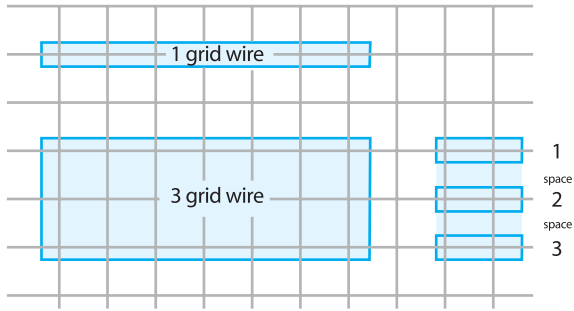


Figure 2–12. *Three-grid wire, built as though we laid three wires and filled in the spacing gaps between them.*

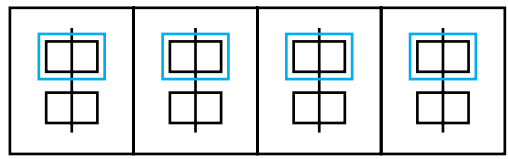


Figure 2–13. *Placing four gates next to each other, each in a standardized library cell.*

A typical CMOS process usually has an N well spacing rule that is very large. If we were to space our inverters such that we had a minimum N well spacing between all the devices, we would waste large amounts of space. Luckily, because most logic circuits have the PMOS devices connected to VDD along with the N well, we can create one large single N well and save space.

Having all our devices in one large N well now means that our limiting design rule is the transistor-to-transistor rule, which is much smaller. We can place our devices closer together by sharing the N well.

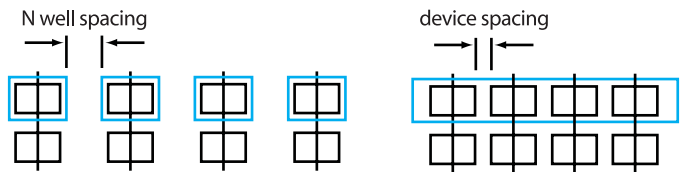


Figure 2–14. *Making a continuous N well allows us to use device spacing minimums rather than N well diffusion minimums.*

Consequently, our cells are designed so that when they are placed next to each other, we get the devices inside the gates spaced at the minimum transistor

spacing. The N well and the power rails butt against each other forming long, continuous N well and power rail strips. (See Figure 2–15.)

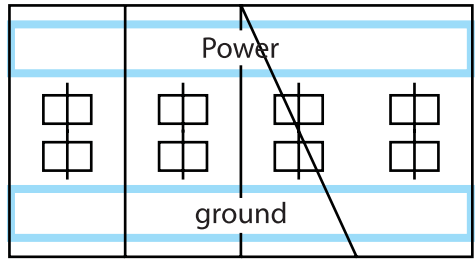


Figure 2–15. *If the rail material extends to the edges of the cells, butting the cells together forms one long power rail.*

Half-Grid Cell Sizing

We will be butting cells up next to each other. As we just saw, this allows our power rails and N well strips to be connected for free. They just form a long strip. However, if we intend to butt the cells next to each other, how will that affect the spacing between our internal components? We run the risk of butting the internal components next to each other as well. We can't have that. Internal components must obey their minimum spacing rules.

Here is a good solution. If we keep all internal components far enough away from the edge of each cell, then we are free to butt the cells next to each other.

There is a convenient and most efficient way to do that. We keep all internal wiring on-grid. Then, we have the ends of the cells that butt against each other falling between gridlines, on the half-grid. This guarantees the metals will stay apart by exactly the minimum spacing needed. See Figure 2–16.

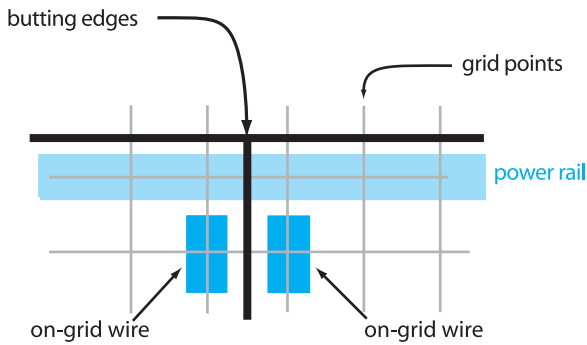


Figure 2–16. *Metals fall on-grid. Cell edges fall off-grid. This keeps metals apart.*

Cells butt against each other in all directions, so every edge of a cell—top, bottom, left, and right—needs to end on a half-grid. That keeps our internal components properly spaced on all sides.

Half Design Rule

As we stated earlier, we like to place our transistors at a minimum spacing in order to maximize the number of transistors on a chip. If our cell ends on a half grid point, then in order for our transistors to be placed at minimum spacing between two butting cells, each transistor is placed at least one-half design rule distance from the edge of the cell.

For example, your design rule for active diffusion to active diffusion spacing might be 1.4 microns. If you keep the edge of your active diffusion 0.7 microns, or half the minimum spacing, away from the edges of each cell, then two cells placed together will provide the 1.4-micron separation needed.

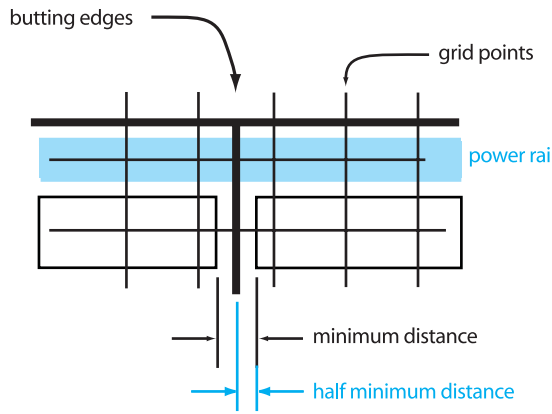


Figure 2–17. *Placing components one half minimum design rule distance from each edge puts one full design rule distance between components.*

Now, there are times when you are laying out standard cells that the cell turns out to be just a little wider than the half-grid line. You cannot meet the half design rule. Your internal items would be too close. Warning lights will flash. In that case, you widen your cell by one full grid. You just put some slop on each end, and live with the fact that it's slightly bigger than it needs to be.

We can see that our grid drives the design of our library cells. Everything must stay on the grid. All cells must obey the half-grid rules. Internal components must remain one-half the minimum distance away from the edge of the cell. The whole library is built around these constraints.

You can write DRC rules that check the various library cell requirements when you lay out your cell. You can check your minimum distances, determine if devices are too close to the boundary, check if you have employed the half design rule spacing from the edges.

Routing Channels

The number of metals available in your process determines how you design your standard cell library.

If you have five or six layers of metal, then you have a lot of freedom to wire the way you want. In our grid-based approach, you might say, "Ok, Metal One will run horizontally and we'll run power on Metal One. Metal Two runs vertically, Three horizontally and Four vertically." Some people might only run power on Metal One. Some might decide to run power on Metal Five. You have all the latitude and creative discretion you desire if you have that many metal layers to work with.

However, if you have only two metals in your process, you need to rethink some of these wild and carefree options.

In your standard cell, your power and ground wires might be at the extents of your cell (run clear out to the edge). This option works very well if you have lots of metals, but let's see what happens if you try to use this design when your number of metals is limited.

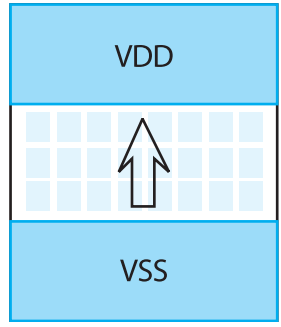


Figure 2–18. *Power and ground rails are placed all the way to the extents of your cell. The middle is full of transistors.*

When you place a bunch of these cells together in a row, you have the power and ground rails butted next to each other, forming long VDD and VSS strips. One long strip for power and another long strip for ground.

With this technique, if you only are placing a single row, everything is fine.

However, typically you have rows and columns of cells. Every other row would have to be flipped upside down, so that the VSS's point toward each other—touch each other. After flipping and placing enough rows, you end up with quite an array, as in Figure 2–19. (The arrow in the figure points to the UP orientation of each cell. You can see that alternating rows have been flipped.)

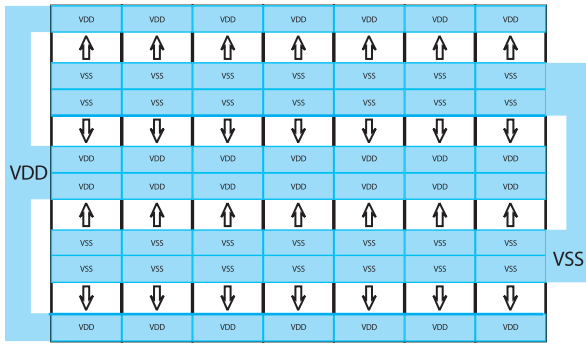


Figure 2–19. *Since the centers of your cells are filled with components, we have no room to run additional wiring. If we have very few metal layers to work with, this is a big problem.*

It is interesting to note that the top and bottom rails of the array are the same height as the height of the rail in the original cell. However, the inner rails are double that width, since each rail is actually made from two rails set together, one from the row above, and one from the row below.

This arrangement is good for your power net. You just connect your VDD's to each other in Metal One at the ends of the rows, as in the Figure 2–19. Or, you could run some Metal Two outside of each end, and all your VDD's and VSS's are hooked up as in Figure 2–20. Done. Easy.

Using Metal Two to strap power rails to each other is one of the ways that the power router can operate. The software knows that the power rail always runs in certain directions. The VDD's are connected to each other at the ends of the block along with the VSS's, after the cells are placed.

As we mentioned, having your power rails hit the extent of your cell is great if you have lots of metals. You get really compact designs. Almost no area is wasted whatsoever. We can use Metal Three or Four or Five for our additional wiring needs.

However, if you only have two or three layers of metal, this method gives you no room to wire in Metal One at all. You can use Metal One for the rails, and

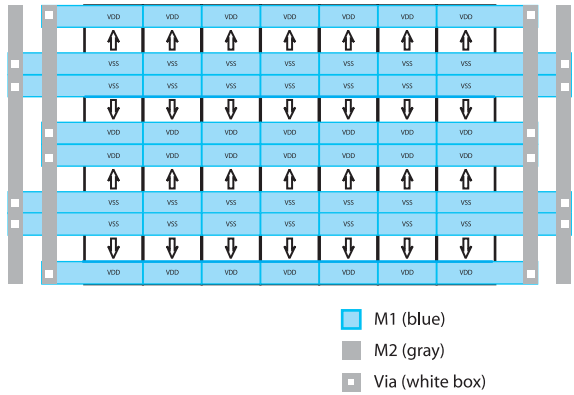


Figure 2–20. *Typical power ring using Metal Two buses.*

you can still use Metal One for the insides of the cells, because that's the lowest level. But that's the end of it. You have used up all your wiring space for rails and cell contacts.

To solve this problem, one strategy for a standardized cell library that only has a few metals is to leave some airspace above and below the cell structure, outside the rails. These air spaces are known as **routing channels.** The concept is to leave yourself channels in your standardized cell to run additional wiring later.

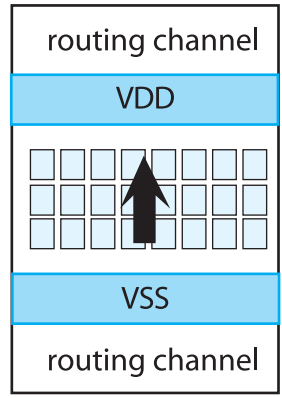


Figure 2–21. *Leave room at the extent for additional wiring.*

Watch what happens when we tile these cells in rows and columns. (See Figure 2–22.) Just look at all that clean, fresh air just waiting for some metal wiring. Notice we do not need to flip every other row since the power rails are sepa-

rated. With airspace between cells, we can leave our VDD and VSS as they are, all right side up.

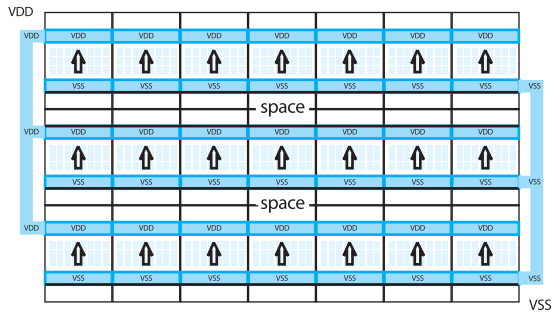


Figure 2–22. *This arrangement requires no flipping of rows, and allows room to wire in lower metals.*

The routing channel approach also makes it easier for the placement software. Since the software does not have to worry about alternating orientation of cells, the coding for that piece of software is easier.

Let's look at a close-up of two of these cells placed together. Notice the join between cells offers lots of clear space for additional M1 wiring.

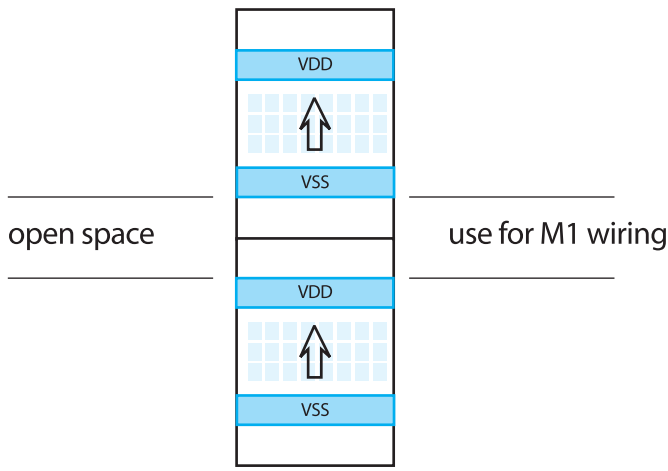


Figure 2–23. *Bringing our rails back from the cell extent we open new spaces for wiring in lower metals.*

To summarize, if we have only a few metals in our process, we will need to use Metal One for wiring as well as for power. So, we make the extent of the cell bigger than we need. We give ourselves one or two extra grids to wire outside of the power rails, instead of sharing the power rails between flipped rows.

Leaving room at the cell extent this way only guarantees horizontal room to run wires. However, if we use our alternating metal directions, we can run Metal Two vertically. This should give us all the wiring room we need.

The routing channels, by the way, can be any height we want. We can make cells any dimensions to fit our needs.

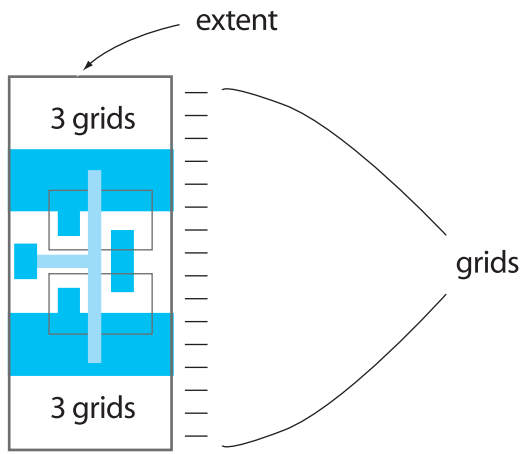


Figure 2–24. *Inverter cell using 3-grid routing channels.*

This approach is fairly standard for CMOS cells, designed to work with auto-routers and auto-placers. You can see how the limitations of using automated software have driven our library and our layout for digital mask design projects.

Channel Routers

You can use standard cells in a different way.

Let's assume that your design has an incredibly large number of wires. Many more than usual. Even with pre-defined routing channels you still don't have enough room for all your wires. In a special attempt to solve this problem, there are some routers called **channel routers.**

Channel routers create channels between cells. You will see a whole bunch of cell rows, then a big gap, then more cell rows. Channel routers build in chan-

nels for wiring between rows of cells. This is similar to the pseudo-channels we made by leaving gaps at the extent of our cells in the last section.

A **channel-based** approach deliberately leaves room for wires to be placed along designated channels. Of course, if you have only a few nets (wires), this approach wastes area.

You can see in Figure 2–25 that the wires and their gaps are evenly spaced. This is called **fixed-channel routing.** This would be simpler to automate, but could be wasteful. Every channel does not necessarily need the same gap for wiring.

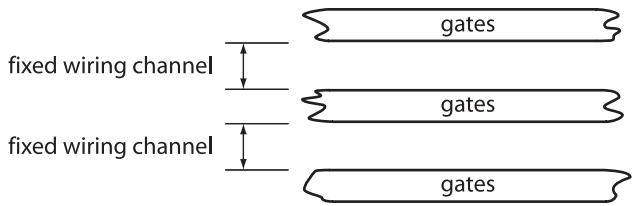


Figure 2–25. *Fixed spacing channel-based layout.*

Newer software looks at the design, the wiring density, and the number of nets per area, and says, "I think here I can place my cells really close together, but out here I think I need more room." Our channels can vary in size according to need. This is called **variable-width routing.**

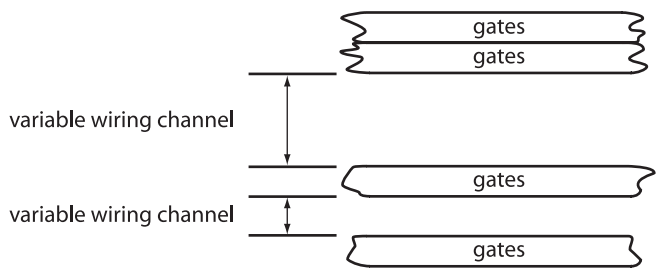


Figure 2–26. *In variable-width channeling, our open spaces can be altered according to the needs predicted by the software.*

Varying your channel width according to need can give you a much more compact die, a much smaller chip. But, the software that drives the placement of cells has to be much more sophisticated, able to handle much more information.

Antenna Rules

One issue you do have to worry about, particularly with modern processes, is the **antenna rule.** An antenna rule is a design rule check that makes sure that any CMOS gate is tied to a diffusion before Metal One is processed.

In order to make sure that the gate is tied, you have to add a small reverse biased protection diode. You have to design your standard cell in a way that guarantees that any input will be tied down, i.e., protected. You will place these small protection diodes usually on the inputs to an FET gate.

The output of a device can provide protection for the gate it is driving. The internal FET gates of a big flip-flop, for instance, are usually protected automatically by this connection. But for the inputs of an inverter, for instance, you have to build these protection diodes into your cells up front.

Some people call these protection diodes **NAC diodes.** NAC stands for **Net Area Check.**

You have to build NAC diodes into all your logic gates, particularly the inputs of standard cells. You cannot guarantee they will be driven directly from a diffusion in Metal One. For instance, an input gate could be accessed in Metal Two. So, we have to add gate tie-down diodes into the standard cell.

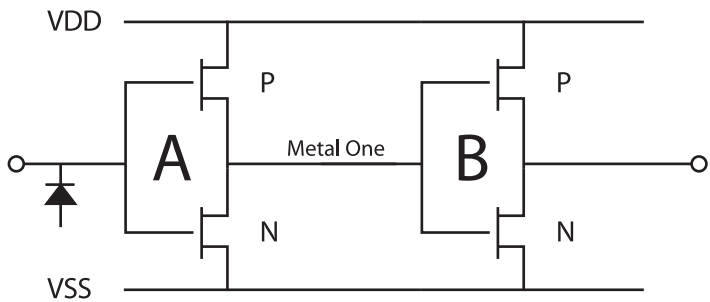


Figure 2–27. *The input of Inverter A is floating, so it needs a tie-down. However, the input of Inverter B does not need a NAC diode, since it is tied down by the output of Inverter A.*

Standardized Input and Output Cells

Just as the standard cells in the chip must conform to the grid rules, the cells that drive signals in and out of chips have to conform to rules as well.

Here is a sample I/O cell. There will usually be multiple power rails in an I/O cell. In this example, the bottom two rails could be the driver supply. The top two could be ESD supply.

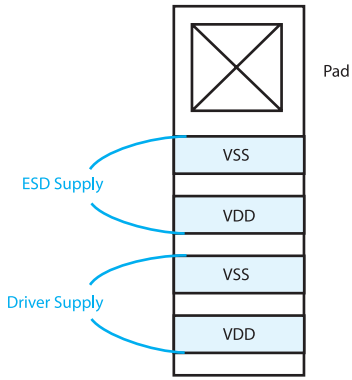


Figure 2–28. *I/O pad at the top. Rules decide how it can connect to the rails.*

The same kinds of constraints exist for I/O cells as our standard cells. The only difference is that the I/O cells are placed in a ring around the outside of the chip. We still follow the half-grid and half design rules.

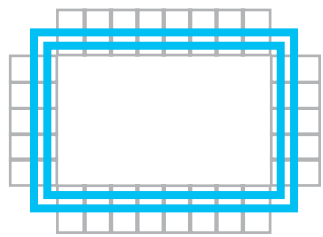


Figure 2–29. *I/O pads surrounding the chip with their power rails.*

There are many placement rules for I/O pads, just as we had for grid-based inverters, wires, and other cells. The rules are very similar to the rules for other components, mainly to protect proper alignment and design rule clearances.

Using Standardization in Analog Mask Design

You can use all of these techniques in analog mask design, not because you need them for automated layout software, but because they are just good tech-

niques. If they make things easier for a computer, they will also make things easier for you.

There might be a time when you have a big analog circuit to put together and you decide to use standard cell techniques to make your power bus wiring easier, for example. Or, perhaps you have a very tight compact chip with lots of circuitry in it. Using the horizontal and vertical metal rules might be the only way that you can put that kind of chip together.

Sometimes you might want to pick and choose just one or two of these techniques to use by themselves. For example, even if the only standardized technique you use is to have wires on a grid, even that alone can speed things up.

So, you can either use all of these techniques in analog mask design, or just the ones you think will be useful for you. Analog mask design is a world of choices and decisions. Be sure to add standard cell techniques to your bag of tricks.

Closure on Standard Cell Techniques

The subject matter in this chapter is covered in two-year college occupational IC layout programs. When most people come out of these programs, the first thing they'll be assigned is library development. They'll be laying out standard cells following these types of rules. If you understand what you're doing, how the tools work and how the cells will be used, it will be a lot easier to understand what's going on, what your tools are providing you, and why.

All these notional cell-building rules are only there to make the automated software easier to write. The rules are not there because they are the only way to make a cell. They may not even be the best way to make a cell. Your abilities as a human are much more sophisticated. Therefore, you must know how and when to take command over the rules, and how and when to follow them to the letter.

For example, if you understand that you are using a grid-based router, you know why you have to put inputs and outputs on grid. If you place them slightly off, your supervisor will emphatically chastise you, saying, "No, it has to be exactly on grid!"

Understanding the rules, and what you're doing, makes your life a lot easier. Not only is there less difficulty, but you will be able to make better decisions throughout the design process.

The standard height, grid-based cell structure is a technique I use even in full custom, hand-wired chips. Standard height, variable width, Metal One running

horizontally, Metal Two running vertically, butting to the edge, half design rule from edge, and so on. Most companies use this universal technique in some form.

You can use this technique not only in digital layout, but it is useful in analog circuits as well, as we have mentioned. It's useful all over the complete layout board.

Using and understanding standardization techniques empower you to do more complicated, routine tasks more easily and with better quality results.

Here's What We've Learned

Here's what you saw in this chapter:

- Advantages of using standardization in analog layout
- Why standardization is mandatory in digital layout
- Standardized wire gridding
- Using repetition to your advantage
- Horizontal and vertical layering technique
- Creating a contiguous N well
- Grid-based routers vs. rules-based routers
- Wiring channels
- Fixed height, variable width libraries
- Using repetition to your advantage
- Half design rule
- Channel routing
- Antenna rule
- NAC diodes

CHAPTER 3

Analog Layout

Chapter Preview

Here's what you're going to see in this chapter:

- Basic differences between digital and analog layout styles
- Importance of knowing how circuits function
- Assistance for the new analog mask designer
- Three key questions to ask your circuit designer
- How the answers directly affect your layout
- Dialogue samples between mask designer and circuit designer
- Importance of mask designer as problem spotter
- Advantage of attending design review meetings
- Expectations of an analog mask designer
- Walkthrough of pre-layout communication with your circuit designer and what it will mean to you

Opening Thoughts on Analog Layout

In the previous chapters, we looked at the constraints you work with in order to create cells for use in big digital blocks. We saw a multitude of uncompromising rules that allow automated tools to toss our cells together by the millions for us, with DRC clean guarantees. Digital mask design proved to be rule-based. Very rigid. Very demanding. Lots of procedures, checklists, and flowcharting.

Some of the rules that most people are used to, like keeping elements on-grid or running metals horizontally and vertically in different layers are not so much of a worry in analog mask design. These techniques are relegated to the position of being mere options, choices you keep in your bag of tricks. You are

free to consider boundless sizing options, placement options, neighboring and pairing options, or noise options, for example.

Considering the fact that we have so much control and so many choices, this can be a frustrating transition for people coming into analog layout from digital mask design. This chapter assists in that transition.

After this introductory chapter, we will discuss matching at depth, parasitics at depth, what you can do to improve or reduce them, and other techniques for good all-around mask design. You will be developing your awareness of options and developing your understanding of integrated circuits.

Our complete mask design bag of tricks will apply not only to analog mask design, but increasingly to good digital design as well. The two worlds are beginning to converge.

In addition, this chapter guides you through the development of communication skills you will need to use with your circuit designer. Communication is vital in mask design, particularly as skilled layout becomes increasingly important to the quality of the product.

You are now entering a design arena highly appreciative of true skill and expertise. The analog layout world. Say goodbye to work. Enter creativity and reward.

Digital Skills vs. Analog Skills

As we just mentioned, digital and analog mask design are converging. As microprocessor clock rates get higher, a CMOS digital chip becomes more analog-like. Also, as we have mentioned, the standardization techniques routinely used in the digital world can be highly valuable as tools for the analog mask designer.

All the techniques covered in this book are applicable in some degree to either domain. So, the skills used are not necessarily the biggest difference between digital and analog mask design.

Let's look at a few of the more pronounced differences between digital and analog mask design.

Difference of Scale

In digital IC's, you might have 10 million inverters all next door to each other in one chip.

However, in analog IC's, you will not see 10 million amplifiers. You will have one amplifier, or maybe three or four.[1]

Difference of Primary Objective

As you work on your digital CMOS chips, your primary objective is to optimize chip size and compactness. You want to make sure your digital inverter, for instance, is as small as possible. One inverter may only be 10% bigger than it needs to be, but with 10 million of them, that compounds into a substantial increase over the whole chip. Very bad news. Your die will cost more, and your work will fail to impress.

In analog, whether CMOS or even Bipolar, your primary objective is not the size of the chip. Your primary objective is to optimize circuit performance, matching, speed, and all types of functionality issues. For example, is the wiring sized correctly for analog current consumption? Are the parasitics too high? Are the matching techniques going to be adequate?

Space is still an issue, to some degree, of course, but not the overriding issue anymore. In analog mask design, performance matters more than size.

Difference of Teamwork

With a digital project, to a certain extent, you can go off and hide in a closet and do the layout. You can practically do your work in total isolation. No communication. Shut the door. Almost the only information you need to communicate is where the inputs are, where the outputs are, and where the power rails are. That is all you have to tell the automatic placement software: "Here are the inputs, here are the outputs, and here are the power rails."

However, with an analog project, the first thing you do is communicate with your circuit designers. How could you possibly begin to place wiring or polygons without knowing how much room you must allow for shielding or for matching or for special orientations or for specially paired double signal wires for differential signaling? There is too much information you need to know before you begin.

Even after you have your preliminary floorplanning ideas, you continue to communicate with your team from front to back of the project. There is just too much at stake to think one person would be as creative as two would be. You are communicating constantly with your circuit designers, listening to their feedback to make sure your circuit will perform the best it can, given the options available to you.

[1] It's ok. Breathe slowly. Find your chair again.

Difference of Completion Schedule

With a digital mask design project, it's pretty likely that the vast majority of the chip will be designed and completed by the time you start work on the chip layout.

However, with analog mask design, it's very likely that the circuit design is evolving at the same rate as you are laying out cell blocks. There is a lot of uncertainty as to what the final circuits will really look like. This uncertainty can lead to a lot of worry, panic, and frustration. You will be asked to come up with accurate schedules and chip sizes based on some hand-wavey-pie-in-the-sky-don't-worry-Chris-it'll-be-ok statements from a bunch of extremely vague hand-wavey-pie-in-the-sky-don't-worry-Boss-it'll-be-ok design engineers. Having the chip evolve as you work affects confidence levels, but the team somehow brings it all together in the end.

Difference of Innovation

In digital circuit design, most of the circuits have been designed and laid out many, many times in the past.

In direct contrast, almost everything you lay out for an analog chip has never been designed or laid out before. You are always breaking new ground. Each project goes where no mask has gone before.

Difference of Constraints

If you have spent all your life doing CMOS digital layout, your initial reaction to a new schematic is to ask, "Ok, what's my standard cell height? What's my grid? What metal runs horizontally? What metal runs vertically? Are there any rules that I need to follow? Where are the procedures? Where are the documents? Where is my closet?"

Provided you follow the heavily written procedural rules that have been given to you for cell height, the grid and all, you can successfully complete your DRC-clean digital chip.

But in an analog circuit, it is very different. There are very few rules. You are more worried about how the circuit will perform than you are about all those nitpicky little rules ensuring that the cells will all fit together.

In the place of rules, all of which must be obeyed in digital mask design, you now have options in analog mask design, any of which may or may not apply. Instead of rote procedure, you have creative toying.

In the place of rules, all of which you had to apply,
you now have options, any of which may apply.

Depending on the company you work for, and how your analog cell will be used, there may well be some rules.

Your layout might be used in a mixed signal chip.[2] Or, there could be a lot of digital content in your chip. You may have to interface your cell into a big digital chip. In which case, it will probably be wired using all the place and route tools. So, you might need to conform to a standard cell height. You could be working with a mixture of some standardized cells and some full custom cells.

But, purely analog layout is wide open to your imagination. The structure is yours.

Difference of Understanding Circuit Techniques

Because we are more worried about optimizing circuit performance, analog mask designers need to know a bit more about circuit techniques than a pure digital-only mask designer. They should know more about how a circuit works, about voltage and current, and their relationship to each other. They should understand why differential pairs need to match each other. They should learn about signal flow, about reducing parasitics, about current densities, about device orientation, about wiring concerns. They should learn when to use that $2.99 Scooby Doo calculator sitting on their desk.

Sometimes needing to understand circuit techniques creates confusion or frustration for people new to analog mask design. Until a mask designer grasps what understanding is expected, it can seem daunting. However, like many technical skills, a little can get you started, and the learning never stops. That is what makes this job so much fun. There is no end to improvement and creativity.

The next section helps you get started.

Three Key Questions

Let's pretend you are the new analog mask designer. The schematic for a CMOS op amp circuit has been placed on your desk. (See Figure 3–1.)[3] This is the first time you have been given an analog circuit to lay out.

What kinds of things do you need to know? What kinds of things do you need to ask? What kinds of differences are there in this new field? Where do you start?

[2] Mixed signal: mixture of analog and digital.

[3] This schematic is similar to our first Case Study.

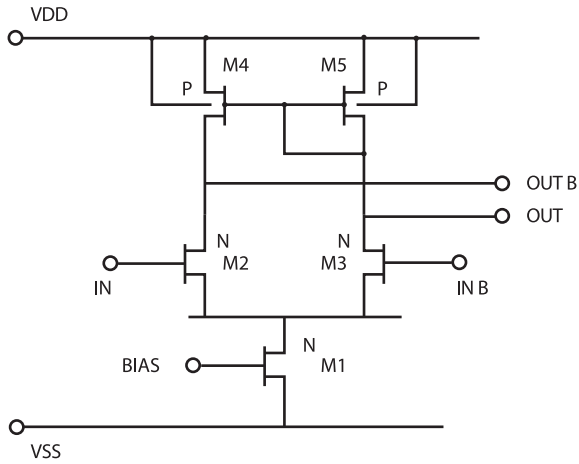


Figure 3–1. *This shows up on your desk. Where do you start?*

In dealing with analog layout, the first thing you do is get over the shock of being told, "There are no rules." This adjustment takes longer for some people than others.

That done, your next and most important lesson is: Communicate with your circuit designer.

Communicate with your circuit designer.

Following is a set of questions, or a checklist, to get you started. Post it where you can see it. It's your communication checklist. It's your starting point for any project.

I'm not sure you heard me. The following communication checklist is your starting point for any project. Any project. Every project. You start here. Got it? (Ok, that's better.)

If you ask the circuit designer the questions from this checklist up front, and you understand a bit about how the circuit works, then you can make some highly appropriate choices when you do your layout.[4]

Where Every Good Mask Designer Starts

QUESTION 1: What does this circuit do?

[4] *Make some highly appropriate choices* is polite for *kick ass.*

QUESTION 2: How much current does it take?

QUESTION 2a: Where are the high and low current paths?

QUESTION 3: What matching requirements are there?

CATCH-ALL QUESTION: Is there anything else?

How these questions and their answers guide your layout is covered partially in this chapter, and in depth throughout the rest of this book.[5]

Let's just see a few quick examples of where these questions might lead your thinking, and how the answers might affect your layout decisions.

QUESTION 1: What does this circuit do?

The first thing to ask is "What does this circuit do?"

Referring to the schematic in Figure 3–1, let's suppose your circuit designer answers, "This is an amplifier."

Depending on your experience, a very simple answer like this might be all that you need. However, if you are new to analog layout, you may not understand what an amplifier is or how that might affect your layout. If this is the case, then once you are told, "This is an amplifier," then your next question is, "What's an amplifier?" Don't be afraid to ask.

Ask questions like, "What's the frequency this amplifier runs at, Kenneth?" "How much gain does it have, Susan?" "Kenneth, why do you want me to call you Susan?" The more questions you ask, the more you will learn. Try to get as much information about the circuit as you can.

Knowing the function of the circuit is crucial to your layout. You will make decisions based on this information. Circuit function determines how you handle such issues as:

- isolation
- matching
- placement
- balance
- overlaps

[5] Your layout choices, based on what you know about the circuit, require an entire book. That tells you something about how many decisions are made in layout. It's great. I love it.

- protection schemes
- location of I/O wires
- device splitting
- floorplanning
- . . . and many, many other techniques covered in more detail throughout the rest of this book.

We will leave examples of these techniques to other chapters, where they can be covered in more depth.

QUESTION 2: How much current does it take?

The next question is "How much current does it take?"

In the ideal world, the schematic will have a bunch of notes on it telling you how much current each section of the circuit takes. However, most circuit designers get so consumed in the minutia of their circuit function, that they don't have enough time—or more accurately, they *feel* they don't have enough time—to annotate their schematic with all the individual currents.

Sometimes circuit designers just think, "Hey, you can run a simulation on the circuit, and if you're really interested in the currents then you can figure it out." Well, sure, many people could, I suppose. But mask designers do not necessarily have the time or knowledge to do that. We really need the currents annotated on the schematics.

Let's assume that our sample schematic has no annotations on it anywhere regarding current. (Baaaaaad circuit designer. No pizza for you.) So you have to ask that question: "How much current is the total analog block taking?"

The answer you get will drive a lot of your device choices, a lot of your metal sizing choices, and to a certain extent some of your placement choices. These are very important decisions.

As we look at our schematic, the circuit designer tells us, "Hey, no worries. This particular circuit is only taking 200 micro-amps." That's a fairly low current.

If your circuit is running just a couple hundred micro-amps then it's a no-brainer question. It's low current, so your minimum metalization dimension should be able to handle it.

Anything above one or two milliamps should ring another bell in your brain. One or two milliamps is pretty high. You would need to calculate some current densities.

Calculating Current Densities

We have just been told our circuit is taking 200 micro-amps. Shall we just throw down some minimum wire and hope for the best? (*Hint: No.*) That would not be good mask design practice. I'll show you how to be more certain of your wiring.

The process design manual contains the numbers that tell us how much current a piece of metal can safely handle. This is usually expressed in milliamps per micron. In a typical CMOS process, we will have current handling capability of about 0.5 milliamps per micron. We will use this figure for this example.

We can use this information to determine the width of metal required to handle a given current. It's a pretty straightforward formula to calculate. The amount of current a wire can handle (I) equals the width of the metal (W) multiplied by the current handling constant (I_h) found in your process manual.

$$I = W \cdot I_h \qquad amps$$

In our example, we were running 200 micro-amps. That's what we were told the entire circuit was taking. We set this current equal to our width, which is what we're trying to find, multiplied by our magic number from the process manual, which we said will be 0.5 milliamps per micron in this example.

$$0.200\ mA = W \cdot \frac{0.5\ mA}{1\ \mu m} \qquad amps$$

We can rearrange our formula to find the width of metal we need to wire our circuit. In this case, it turns out that we need a width of 0.4 microns. (Notice we changed 200 micro-amps to 0.200 mA, for consistency of units.)

$$W = 0.4\ \mu m \qquad microns$$

We now look at our process manual, and ask, "Ok, what's our minimum metal width? Ah, I see that our minimum metal width is 0.5 microns." Oh, good, that's more than we need for this example. The minimum metal width is 0.5 microns and we only needed 0.4 microns to be safe. We can wire our circuit using the minimum width wire.

You can see why the amount of current in the analog block is so important. We might have calculated a wider necessary width than the minimum wire for our process. That would be critical information to have before we begin our layout.

For example, you might calculate that you need 0.6 microns to be safe, yet if you had run a minimum wire, you would only have 0.5 microns. Unreliable wire. Products that break down. Bad company reputation. Lousy pieces of junk. And you could have used your little calculator to quickly determine the need for a larger wire, had you spent the time.

A good mask designer is key to helping the team put out fantastic, robust products.

We are now starting to make layout decisions based on circuit function. In a digital standard cell environment, you just wire everything up with minimum. You make it as small as possible.

In analog, however, we are starting to get away from this minimum rules habit. The mask designer is spreading his wings, doing some of the calculating, checking, and decision-making.[6]

Or, let's try another example. Say the circuit designer came back and said, "Ok, this circuit takes 5 milliamps."

You run the same equations. (*Go ahead—run the calculation. I'll wait. Determine the necessary wire width.*)

Entering a 5-milliamp current into your equation, you solve for width again. The answer is 10 microns this time. You need a wire that is 10 microns wide, in order to carry 5 milliamps of current.

Seeing a *10-micron* value come out of your equation, you could just blindly go away and say, "Ok, let's wire the whole cell with 10-micron-wide wires." You would be safe everywhere in the circuit. However, that's wasteful. You only need the 10-micron wire in those locations running 5 milliamps.

That gets us to the second half of Question 2. If there is a big current, where is that big current flowing?

QUESTION 2a: Where are the high and low current paths?

From our last example, you will go back to the circuit designer with your answer of 10 microns, and ask to be shown exactly where those 5 milliamps will run.

The circuit designer might say, "Well, this is an amplifier. All of that 5 milliamps is flowing through this transistor (M1), but sometimes it flows through the left (M2), sometimes through the right (M3)." Now you know the two paths that need your 10-micron width for wiring. There is no need to run such large wire elsewhere in the circuit.

[6] His *or her* wings, of course. I find the *his/her, he or she, s/he* attempts simply distracting, so I'm not putting them in the book. So, don't blame Chris. After all, he's British, and you know how polite they are.—*Judy*

There may be multiple circuit paths, each with its own current flow considerations. There may be some paths that take only 1 milliamp, some that take 10 milliamps, and some that take 100 micro-amps. Both high and low current paths need your attention. Find out where they are. Find out how important they are.

Let's carry this example a little further to see another example of how knowing the high current path can affect your layout decisions. This time, instead of wire width, we will look at device orientation.

Device Orientation

In Figure 3–2 we have a four-fingered FET. When we auto-generated the transistor layouts from our circuit schematic, this is the device the tool gave us. We asked about current, and were told by Circuit Designer Bob that the bottom transistor in the circuit diagram, M1, is drawing 5 milliamps. (*Refer to device M1 in previous schematic.*)

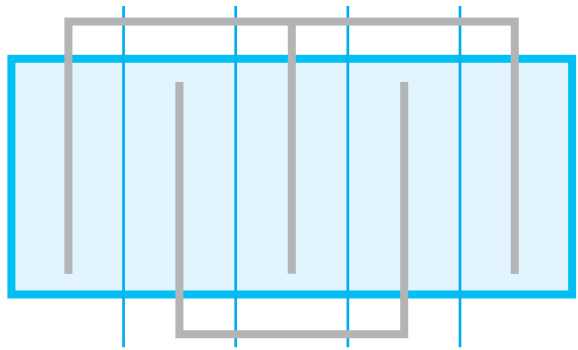


Figure 3–2. *Plain vanilla four-fingered FET.*

After doing our calculations, we know we need a 10-micron wire hooking up to this device. Ok. Thick wire. Make the signal thick. But that's not the whole story. Keep looking.

In a typical situation, the layout for this library transistor has been made as compact as possible, so the metalization that runs down into the source-drain regions is usually minimum width.

The question I want you to consider now is: When we lay out the cell, how do we hook into this device?

So, option one is to go right ahead. Let's make this signal, which is coming from the left, 10 microns wide. That's just a first option, not necessarily the best. (*Hint: It's not.*) All the current is flowing in on the left side. If we just work with that option, we can say we sized our wire at 10 microns, so we're happy.

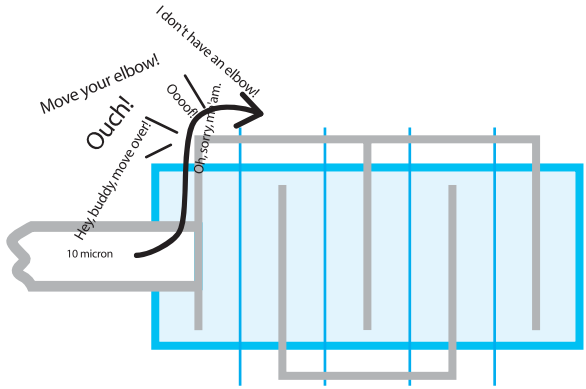


Figure 3–3. *Lots of pushing and shoving around that tight corner.*

But, look where that current flows into the device. It all travels through a tight, little bottleneck going up and over the top. That's a mighty skinny chunk of metal going around that corner.

So, we say, "Ok, well, let's bring our 10-micron wire in at the top."

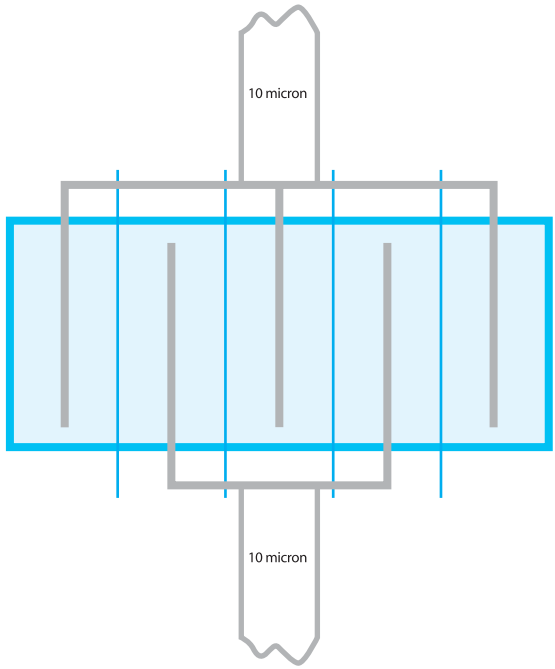


Figure 3–4. *Routing the wire through the top of the device.*

Try It

Jump ahead of me, here. Stop to think about these two questions before reading on.

1. Why would you benefit from running the signal at the top?
2. How would orientation of the device be affected by knowing the current?

Answer both. (Stop here to think.)

Running the signal into the left side of the transistor makes the electrons compete for a spot around that tight turn, like water trying to get through a kink in the hosepipe.

If you enter from the top of the device, the electrons are allowed to spread out. The electrons may travel into any of the three fingers. With three path choices, not all electrons are going to pick door number two, so to speak. So, effectively we have reduced resistance. That's the benefit of running the input at the top. That answers the first question.

Now, with your wire coming in from the left, why not just rotate your library device 90 degrees counterclockwise instead of trying to configure the wire up and over the top? Rotating the orientation gives us a ready-made way to assist current flow. That is how orientation of the device is affected by knowing the current. That answers the second question.

We are feeling rightfully smug at this point. Let's recap:

- ✔ We have communicated with our circuit designer. (pat on back)
- ✔ We have calculated wire width. (pat on back)
- ✔ We have rotated our device. (pat on back)
- ✔ We are masters of layout.

Just as we did not stop after calculating wire width, we will not stop after rotating our device. There is always something more to think about. It's like solving a puzzle, painting a picture, or sculpting. It's an art. One person might not stop as soon as another. Let's look a bit further.

Current comes in on a 10-micron-wide wire. It will pop out of the device on a 10-micron-wide wire as well. Is that all we worry about? Input and output wiring? Think of what is going on inside the device itself.

We have the device current being conducted through the metalization of two source-drain connections. What is the width of these wires on our standard cell

that we were given by our layout tool? Not only do we need to worry about the amount of current going into and out of the entire device, but we have to look at the current densities on the metalization within the device as well. Now you're earning your money.

We have been told we need a four-finger device. We have two drain regions where the current falls out. Theoretically, these metals should be 5 microns wide if we need a 10-micron-wide wire to handle the total current. Each drain finger should be half the output wire width.

Let's go in and actually measure the width of the fingers of these devices. Let's be very sure of what the tools gave us. We measure to verify. We find the wires actually are the minimum 0.5 microns. Then you think, "Oh my gosh, that's much too small. This device has to be reworked considerably in order to meet just the current density rules."[7]

At this point we would have to go back to our circuit designer and say, "Hey, you told me that device M1 was taking 5 milliamps, yet the number of fingers that you have assigned to that device are such that the current densities won't handle it. So we need to work this out."

This is another example of why you cannot just go work in your isolated closet until you are done. You are an integral and interacting part of the circuit design process. The circuit design is not finished until you are finished.

If worse comes to worst, the circuit guy might say, "Well, my transistor has to be that size." You could still work with the transistor the layout tool gives you. You could flatten it—make a custom device, a nice, big device with big, fat fingers in there to do what you want. Maybe you need two of them, or change fingers. Who knows what you might come up with.

If you re–lay out the device, you have to keep the same effective gate area, since that defines how the transistor works. But, there are an infinite number of ways to lay it out. One way is to go inside, do some hacking and come up with a fully custom set of devices that are optimized for metalization from a power point of view.

You work with the circuit designer. You have to get agreement, you can't just say, "Phooey, the circuit designer doesn't know what he's doing. I'm just going to go in there and I'm going to change this." You have to talk it over first.

Say something like, "I have this problem with the layout. I'm trying to do analog layout now. My book tells me I have to worry about thus and such."[8] The

[7] Now you might be starting to understand how an analog layout of 4 or 5 transistors could take longer than a digital layout of one million transistors.

circuit designer might come up with a solution for you, or you could figure it out together.

You may end up resolving the finger width issue with the circuit designer through five or six iterations. The two of you might come up with some fancy layout in order to get the current through. You might split this device into two. Maybe split it into multiple fingers. Maybe you will try something altogether novel. You will resolve it together.

Another option is that you sit with it for a while by yourself first, trying to think of a proposal on your own before you meet with the circuit designer. You could go away and do some what-if pieces of layout. You say to yourself, "Ok, I know I need to maintain the gate area, so why don't I split the device up?" And off you go working up a proposal.

Perhaps your thinking goes something like this: "I think I'll try multiple power wires. Big, fat buses with a bunch of connections. I know I need a 10-micron input and output wire. I'm splitting my current into four devices, this internal bus can be 2.5 microns wide."

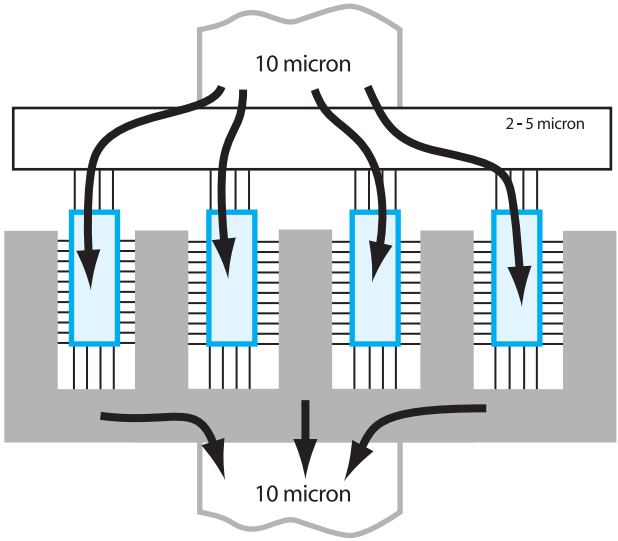


Figure 3–5. *Possible solution, perhaps your invention.*

In the possible solution shown in Figure 3–5, everything spreads out, gets shared and comes back together.

[8] You are, of course, referring to either our *IC Layout Basics* book or this *IC Mask Design: Essential Layout Techniques* book. Bless you.

Using only what you know about metal widths, you could say to your circuit designer, "Here's a proposal. Here's a piece of layout that meets all of the current density rules. What do you think?"

The circuit designer might say, "Yes, that's perfect" and he'll go change his circuit diagram to reflect what you've laid out.

Or, you may not understand how this particular transistor gets used, so there might be other options your circuit designer can suggest. For example, he might say, "Yes, I know I told you 5 milliamps, but that's if it's a Wednesday and it's hot and you're at 50,000 feet. For 99% of the life of this chip, it's really running at 2 milliamps, but it could occasionally get higher. Use the 2-milliamp figure. I'm sorry, Chris." So you work with 2 milliamps.

Changing the device orientation was just one more example of how Question #2 could directly affect your layout decisions. There are many more examples, as we will see in the rest of this book.

Here is the third key question from your checklist:

QUESTION 3: What matching requirements are there?

The third key question to ask your circuit designer is, "What matching requirements are there?"

"Well, these two input devices, M2 and M3," the circuit designer answers, while pointing at the schematic, "need to be matched very, very well."

You then ask sub-questions about the matching requirements. As we've learned, one answer often just leads to more related questions. You might ask, for example, "How good do you want the matching?" "Is just having them located next door to each other good enough?" "Do you want any special matching requirements?"

With all the answers you receive from this line of questioning, you begin to understand what you need from a matching point of view. You then use as many of the matching techniques as you need.

We cover matching techniques more thoroughly in Chapter 5 and in the Case Studies. We will leave the examples of how matching requirements affect your layout for those later sections of the book.

Additional Questions

Along with your three key questions, you will ask a variety of miscellaneous questions, as many as relate to the circuit. For example, "Does this

cell need to hook up with an auto-router at all?" Or, "Is there a wiring grid?"

There are many questions that you ask, and tricks you set in place, even before you see your first polygon.

This is very unlike the digital method of layout. In digital, you assume blindly that your circuit designer knows all the rules and has covered them. Full stop. You take whatever you are given. You blindly go off and lay it out according to your rigid design rules and procedures.

You have to stop and ask
a lot more questions in analog mask design.

Ask and listen. When you know the answers, always stop to rethink and plan. It's better for you to spend two weeks at your computer trying various options than it is for you to charge ahead for the sake of just getting it done.

Success depends on your creativity, your experience, and your skills. Take your time.

Bipolar Analog

Bipolar analog layout, to some extent, is slightly easier than CMOS layout. For example, with Bipolar transistors, you do not worry about source-drain sharing issues.

There is usually less flexibility with Bipolar transistors. You might be given only one flavor of Bipolar transistor. There might be only four sizes of that transistor ever used. There might be a 1-micron, there might be a 5-micron, there might be a 10-micron, there might be a 20-micron, and that's it. No in-betweens.

Also, Bipolar transistor cells usually have all the layers pre-defined for you, so you only have to worry about the metal connections.

If you've never done any Bipolar analog layout, you might start to panic. You go flailing from room to room asking, "Where are all the rules?"

When most CMOS layout people start to transition into Bipolar layout, they expect 30 or 40 internal device design rules to deal with. They might look for gate overlap, or P implant overlap or other similar rules. Nope—don't exist. You are given a box with three pieces of metal and told to just go wire them up. Well, they do exist but you don't mess with the insides of a Bipolar transistor like you do with a CMOS transistor.

You still have to go back to the basics, even if you just expect to wire the metals and not deal with the transistor design rules. If that circuit diagram we had at the start of the chapter was a Bipolar op amp, you would still ask the same three questions we covered previously:

- What does the circuit do?
- How much current does it take?
- What are your matching requirements?

Once you take care of these questions, your life becomes slightly easier. It's just a case of worrying about the wiring of the transistors to each other and the signal flow of the layout.

When you ask the first question, "What does it do?", you could be told it's an amplifier. This answer triggers all you know about amplifier circuits that you have previously seen in your work. The answer will directly affect your layout.

Then, just as before, you ask, "What frequency is it working at?" because nobody really ever cares about ending with a preposition.

The frequency of the circuit directly affects what you do with your layout. If it's a low frequency, you do not have to worry too much about the way you wire your devices because parasitics do not affect operation as much. However, if you are working up in the hundreds-of-megahertz range, then those frequencies do start to have a major affect on how you lay out your circuit.

We will discuss some of these circuit techniques and the way to handle some of these layout concepts in further chapters. However, the main point is that your questions and concerns remain the same, even in Bipolar analog layout.

Expectations of an Analog Mask Designer

The other trap CMOS mask designers tend to fall into big time is that they believe the circuit designer. They expect schematics to be finished and perfect. What the designer drew is the final word.

- **Rule of Thumb: Don't believe your circuit designer.**

An analog circuit designer is much more unsure about his schematic. There are many different ways to achieve the same circuit functions. Some circuit device size choices are 100% dependent on the layout that surrounds them. The circuit designer usually changes the schematic once he sees what you have laid out.

In fact, your job can become very exciting when you are making real-time decisions about the chip alongside the rest of the team. Your choices. Your skills. Your concerns. Your solutions.

How can we handle these noise issues? Is this the correct width? How do we want to place these devices? How can we make this circuit more reliable? The mask designer answers a lot of those questions. A good mask designer is invaluable to a company.

Since so much is expected of your skills, if you see something in a schematic that doesn't look quite right, stop everything. Do not assume it is what the circuit designer wants.

There have been many occasions when I thought something did not look right in a schematic. I returned to the circuit designers, asked the question and I got the response, "Doh!"

This happened at a big three-day review.

It's 6 a.m. in the morning, I'm sitting there staring at the schematics. I'm kind of thinking, "That doesn't look right."

There were some PMOS transistors that had their bulk, N well, connection pulled out independently. Now, usually, you connect the N well of the PMOS device directly to VDD. Most of the time, one side of the PMOS device is connected to VDD, or the most positive point of the circuit. But, this particular circuit designer had pulled out the bulk to this completely independent wire. It wasn't connected to VDD.

As a layout person, that means to me I have to run a separate wire as a separate connection. I can't do my usual trick of, "Well, if the bulk and one side of the transistor are connected to VDD, just connect the two."

If we had left the circuits hooked up that way, it would have caused the layout engineer a lot of problems. They would have had to run extra wires for the bulk connection, and eventually at the very top level of the circuit the bulk wire would be connected to VDD.

So, I raised the issue. I said, "This doesn't look quite right, guys. Do you really intend to do this?" And it was a fairly inexperienced circuit designer who said, "Well, we were told to bring out all the bulks independently."

I said, "Yes, do that for the substrates, but not for the well." I asked everyone else on the team what they thought, and they all said, "Yes, that's perfectly right." So just by spotting that one issue, I saved one of my layout people probably a week's work.

It's a function of experience. You can't do this from day one.

But, if anything looks out of the ordinary, or you've been given a circuit that looks like a similar circuit you did for a designer the week before, and this one looks totally funky, wacky, and way out there, go back to the circuit guy. Ask him, "When I did this mixer for Bob he did it like this. Why aren't you doing it like that?"

And you might spot a problem. But you might learn something, too. Good news either way.

There have been times when problems have remained on the schematic through several levels of team review. Apparently, many people feel that since a circuit has passed through so and so and this person and that person, it surely must be correct. Mistakes have been caught by mask designers at all points of the production cycle.

Never assume. Of course, you will get better at this with experience. Just by exposure, and asking your three key questions, you will get some appreciation of the circuit techniques people use, and what certain circuits should look like.

Circuit techniques are the same the world over, so you will see the same techniques repeated, repeated, repeated. You will eventually get to the point where you will notice when someone seems to be doing something wrong.

To this sort of expectation, a new mask designer might ask, "Are you expecting me to be a circuit designer?"

To a certain extent, yes. Not a circuit designer, really, but someone who has an appreciation of circuit techniques.

A mask designer might want to only push polygons, not get involved with this higher level of creativity and decision-making. If someone expects to churn out repetitious digital layout for fifteen years, I'd tell that person they need to sign up for some courses. We need to bring them up to speed.

If they aren't doing analog layout now, they are probably doing at least high frequency layout, maybe mixed signal layout, something that uses many of these analog techniques. They need to move on. The field will pass them by.

Mask design is relatively young. Up until recently, there weren't any courses on layout. Some of the courses mentioned at the back of this book have only been available 5 or 6 years, or so. The CAD tools have only been available for

maybe 20 years. It's a profession that's become recognized in only the last 5 to 10 years as being worthy in its own right.[9]

Anytime you see a new field develop you will find many of the original groundbreakers at some point are left behind. Younger upstarts come along with new training. The field advances quickly. Tools develop. Books are written. Employers demand higher levels of training. Degrees are offered at colleges. It leaves many original players in the dust, unless they do something to keep up.

A lot of the original mask designers are now very experienced. They may have been in the field since it was new, but they still might have very little circuit theory. That will hold them back the same as anyone else as the profession demands more skill.

I don't mean everyone should understand how to calculate the logarithmic forward gain frequency response of a bicoupled photon quantum pulse inverter doodah. (Pause for effect.) But, there are some very simple techniques all mask designers can benefit from. There are some basic courses that enhance layout skills immensely. For example, just Ohm's Law can get you 90% of what you need.

You have to learn. You have to progress.

Of course, there is a lot to be said for on-the-job training. A lot of our good analog mask designers know to ask these three key questions because they've been beaten up so much in the past. They were forced to improve their skills.

When they first got into analog layout, they blindly went off and worked on their own. Then they took their layouts back to their circuit designers and said, "Finiiiiished!" The circuit designer looked at the layout and ripped it to shreds.

So they went off and did some more of the stuff the circuit designer wanted. The mask designer would come back again after a bit of time, waving his papers. "Finiiiiished!" thinking he was done for sure this time.

The layout still got ripped to shreds.

Top-level analog mask design can be a very frustrating transition. A lot of digital mask designers kick back and say, "Well, I can't be expected to know all of this circuit function stuff (grumble grumble)." They are used to having procedures and a whole bunch of aids to help them.

[9] See *Beginnings* at the back of this book.

But, most companies don't have that. Companies expect people doing the layout to know a bit about what's going on, especially as technology and tools continue to advance. There is just no standing still. The profession is advancing quickly.

You don't have to learn circuit design, just some simple circuit techniques. The fundamentals of these techniques could be shown in a few days, or you could read some basic books (again).

Then, once you have learned the basic techniques, effective mask design is something you can really only master by experience. It can take a few years to get the right kind of initial exposure to these circuit techniques, to get the right kind of experience.

But, there is an immediate place to start, to help you become more effective beginning today. If you ask the three golden questions—what does it do, how much current is it taking, what are the matching requirements—if you get into the habit of asking those questions, you can make your life a lot easier.

Reading, studying, keeping your eyes and ears open, asking questions, taking the time to think, learning all the techniques you can—that's what is expected.

Closure on Analog Layout

You could be used to very well defined rules. *Don't go off the beaten path* type of layout. When you start moving into the analog world, you still can use some of those tried and true digital layout techniques. However, now there is a whole new universe of scary stuff that opens up. That's when the fun begins.

Hopefully this chapter has provided some exposure to a few of the analog layout concepts that you will see in more detail in other areas of this book. This is basically a transition chapter into those other areas.

All mask designers will find themselves dealing with more and more of the chapters of this book as technology advances. Even if you primarily live in the digital environment, analog will become a larger part of your life. Analog teaches us skills we increasingly need in the digital world. Besides, analog is where the big money is.

Here's What We've Learned

Here's what you saw in this chapter:

- Analog versus digital issues and techniques
- Advice for the transitioning mask designer
- Primary focus areas for communication before layout
- Effect of input from circuit designers
- Dialogue samples between mask designer and circuit designer
- Speaking up about errors
- Expectation to know circuit techniques
- Brief examples and overview of material covered in-depth later

Appendix: Key Questions Discussion

Judy: Could you walk me through some more of the answers you might get to these questions? Maybe tell me what other questions I might ask, or how the answers might affect your decisions in layout?

Chris: Sure. Let's draw a diagram, and I'll walk you through it.

(Follow along in the diagram on the next page.)

Chris: Here are our three questions.

- What does it do?
- How much current does it take?
- What's the matching required?

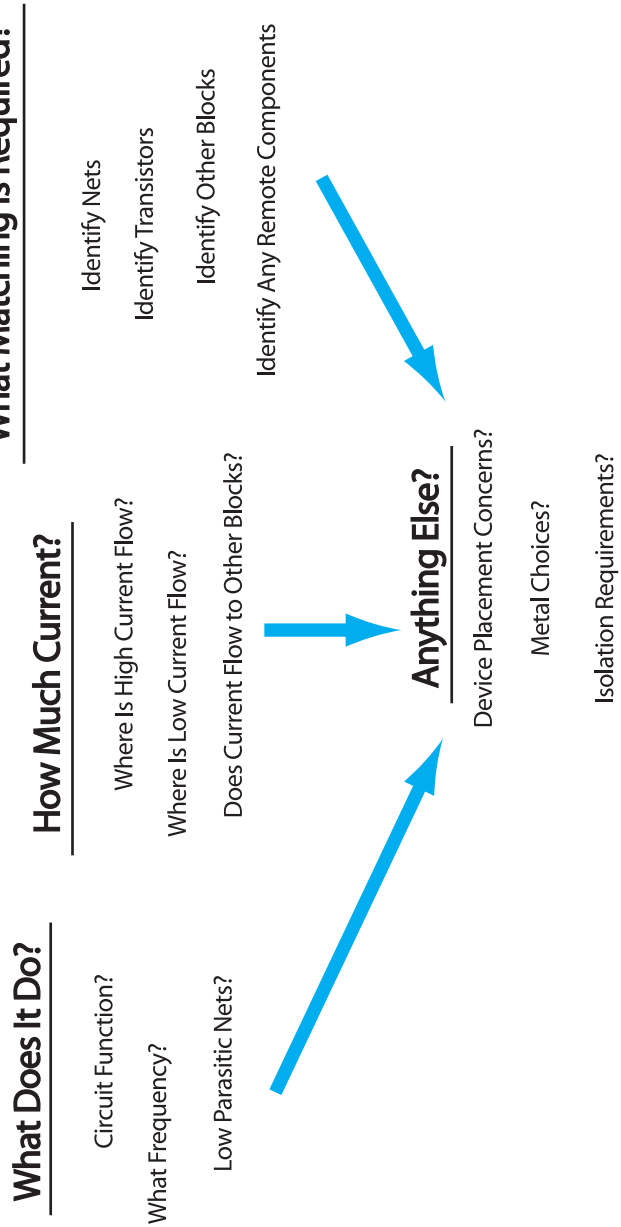
And all of these fold into a fourth question:

- Is there anything else that I need to worry about?

What does it do? This question gets an obvious answer, the actual function of the circuit itself. It could be an amplifier, could be a mixer, could be a charge pump, could be an op amp, whatever. The answer itself doesn't tell you much. It just gives you a feel for what the circuit will be used for. It should put your next set of questions into your head.

Judy: So the first time in your job that you hear the answer, it might be meaningless? You hear that it's a something-something pump, and it doesn't mean anything now, but the next time you are given a something-something pump to work on...

Chris: You'll then go and look in your brain and say, "Ok, the last time I had to lay out a something-something pump, these were the things I had to worry about." So, it then turns your mind to the thought, "Ok, now I know some specific questions I had better ask before I lay out this something-something pump."



Judy: Ok, so if they tell you it's an op amp, you're not supposed to necessarily know how op amps work?

Chris: Correct.

Judy: You're just making a little file folder for reference in your brain?

Chris: Yes.

For me, the next question is, *what frequency?* How fast is that op amp, or charge pump, or inverter, or whatever, operating at? What is the frequency that is involved?

Judy: What does that tell you?

Chris: That should start to guide you more. It might lead you to ask, "Ok, if it's a really high frequency involved, then are there any low parasitic wires that I need to worry about?" It should then point you to some of the matching requirements you will need to deal with. When I ask, "How much current and how much matching is required," those questions have already been colored by what the circuit function is, and what frequency it runs at.

Judy: So, knowing that a circuit is high frequency just sets your mind going, thinking, "I need to remember I've got high frequency concerns as I ask all the rest of my questions."

Chris: Yes. And the high frequency concerns to keep in mind are, "Ok, if it's a real high frequency, I then have to start to worry about parasitics. Ok, what nets do you want to have low parasitics?" Once you find that information out, it gets you to started thinking about layout techniques that reduce parasitics.

Judy: And for that you go to your books or notes. So, the answers don't tell you what to layout, they just direct you to the correct issues that need to be discussed?

Chris: Yes. You are learning where the concerns are. The solutions will be up to you.

Once you get a feel for what the circuit does and how fast it's operating, the next question is, *How much current is in there?* If it's a high current circuit, where does that high current flow? Identify the high current paths in the circuit diagram, because as we've seen in the example, if there's high current flowing then you have to size your metal accordingly. So, there's a layout choice there.

And where is the current flowing? If you have fat metal to handle big currents, then that affects how these high current blocks interact with each other. So, current flow concerns directly affect your device placement as well.

Likewise, there may be some of the circuit that has a low current flow, in which case you can put the devices off to one side where you really don't care about what's going on.

Then the other big question is, *Is any of this high current flowing out to any of the other circuit blocks?* You might size one piece of your layout for just itself, and not realize that 5 milliamps is also flowing out to these other circuit blocks. You have to make your output of the cell big enough. And make notes for the next level up, when you are doing the top-level hookup.

Judy: Should I know what high current and high frequency numbers are, or is it all relative?

Chris: It's all relative. What might be high current for me is low current for someone in the power amplifier space. What I think is low current for me might be high current for the low power CMOS digital people. So, it's all relative.

And then, once you find out how the circuit's working, where the current is flowing and some of your metal sizing, you might say, "Ok, *what matching is required?*" "Are there any nets that need to be evenly matched?" "Are there any transistors that need to be evenly matched?" "Are there any other transistors or components in other circuit blocks, that someone else down the road is doing, that eventually need to match back to this one?" So, there are a whole bunch of matching issues.

Once you've asked these three areas of questions, then you ask that phantom fourth question, *"Is there anything else I need to worry about, anything that isn't covered under these three main basics?"*

And, yes, there might well be. Something might occur along the lines of, "Well, within my circuit I want to keep these two transistors as far away from another pair of transistors as possible." Or, "I want the inputs to be on the left and the outputs to be on the right" because of some higher-level stuff that isn't obvious.

You might be specifically told, "I want this particular net on the circuit diagram to be in the highest metal." Now, that one could lead to more questions about parasitics.

You'll find that many of these *anything else's* are often covered in the top three questions. But, there are interactions. For example, because of the matching

and the current we have to do some of this such and such stuff. It's not because of the matching, and it's not because of the current, but because of the two combined. So, we need to pull that out separately.

Likewise, ask if there are any device isolation requirements. "Ok, I want a pair of substrate contacts right next to transistor M1 on the circuit diagram." Or, "I want to put a Wall of Death between this half and the other half." (You know about the Wall of Death—you cross this line and I'll kill you—remember in the Libyan Gulf of Sidra? There was this Col. Qadhafi line of death. If any boat crossed this imaginary line, he would blow them up. I do actually call them Walls of Death.)

Judy: What are Walls of Death on the circuit?

Chris: Just isolation structures.

Judy: Now, you don't need to lead your circuit designer through any of these scenarios, do you? You just need to ask, "Is there anything else?" and let them think? You don't lead them into each issue, do you?

Chris: Oh, definitely. You ask about everything. Don't wait for the circuit designer to bring it up.

Judy: Oh, you still try to come up with as many what if's as you can?

Chris: Yes. And to start off with, you won't know what questions to ask. If you are brand new to analog layout, you won't have a clue.

You may be lucky, and get a circuit designer who realizes you are new to analog layout, holds your hand and takes you through every little subtlety and nuance of "This is what I want." Or, you may get a circuit designer who is overloaded, way too busy, and just throws you the circuit diagram and expects you to know it all. That's a recipe for disaster.

If that is the case, and you are new to analog layout, don't go off into your closet and hide. Every hour, if you need to, ask for validation: "Is what I'm doing what you want?" Until you get some confidence and some cells under your belt, communicate, communicate, communicate.

Judy: Wouldn't you be afraid of being fired for appearing stupid?

Chris: Well, that can happen. People do feel like that.

Judy: How realistic is it?

Chris: Not very.

Judy: You won't be fired for coming back every hour with another question?

Chris: Well, if you're a new analog mask designer, then in my mind there is no such thing as a dumb question. The only dumb question is the one you don't ask.

Judy: Do circuit designers feel the same way: "I'm glad you came back to check that"?

Chris: Some of them do. Others will be total jerks, and be totally stressed out anyway, no matter what you do or don't do.

You have to play it by ear. If you feel you're not getting the kind of answers or the kind of support that you want, go to your supervisor and say, "Look, I'm trying to do this. I haven't a clue what's going on. Every time I ask Bob this question, he blows up at me." Something needs to be sorted out.

Judy: Talk to me more about communication. You mentioned once about sitting in the design review sessions.

Chris: Well, that's part of the *What does it do* question.

Judy: A lot of people wouldn't think to get involved until after they are given a circuit. But your idea is to go sit in the design reviews.

Chris: Yes, I would rather have a mask designer sit in a design review for an hour and just get exposure. It's all about exposure. Particularly when you're learning, as you start getting used to what's going on.[10]

Ninety-nine percent of what you hear in a design review will be highly technical and totally over most mask designer's heads. But, there will be the odd, one little throw-away line that says, "Yes, we need to worry about that in layout."

If you weren't in that meeting, everyone else will be thinking, "Yes, we need to worry about that in layout." But they aren't doing the layout!

The prime reason for sending mask designers to technical circuit reviews is for that one-in-a-hundred chance that something comes out that's just a throw-away comment about worrying about it in layout.

---

[10] Coincidental that as I edit this section, Chris is in a design review meeting. He will be in this meeting for three days. "Starts at 6 in the morning and doesn't end until 11 in the afternoon!" as he puts it. Every once in awhile I hear him give some input, spot mistakes, or be asked a specific question as the layout representative. I go check things out if I hear a good loud laugh.

It's the layout person's responsibility to do the layout, so they should be there to pick up on those things, and say to themselves, "Ok, after this meeting I'll go and find out what that is that I need to worry about."

It could get to the layout review and no one has told you, or they forgot. All they were counting on was this memory of someone saying, "Yeah, we need to worry about that in layout". Well, since everyone agreed in the meeting, the world must know. All of the sudden there's the screaming, "Well, we talked about this in the design review. Why hasn't that been looked at!"—Oh no! The tapeout! Death! Doom! Despondency! Fingerpointing! (And that happens.)

As you get more experienced in analog layout, when you go to these design reviews, you will be able to contribute. You will be able to ask the right questions because you will have had exposure to circuit techniques. You will say, "Well, Ok. What's the frequency of this?" "Are we worried about matching?" "Are we worried about the amount of current it's going to take?" "Did you really want to bring that out like that?"

And you'll get a lot more useful information up front before you see the schematics come across your desk. Later, when the schematic does come across your desk, you've been to the design review of it, you know some of the things that they're worried about, so that again drives your questions you will ask your circuit designer. "Ok, in the design review you mumbled something about the way that's going to happen, so can we talk a little about that? Is there anything in that bit that's going to cause my layout trouble?"

And the circuit designer may reply, "Oh, yes, well, in that case, because we're worried about gain blah blah we have to worry about matching of these things."

You can lead the circuit designer into giving you the answers you need by just having had exposure to the design cycle. But it takes 2, maybe 3, maybe 5 years of that kind of exposure to get real good at contributing during the meeting.

Judy: You're seeing things that will affect the layout you will be doing. A heads-up right off. Just from what they said and how they said it.

Chris: Yes. We'll be emphasizing communication in every chapter. And being there listening is part of communicating. It's a start.

CHAPTER 4

Parasitics

Chapter Preview

Here's what you're going to see in this chapter:

- Parasitic types and causes
- Parasitic locations
- Effect of parasitics on a circuit
- Special high frequency considerations
- Wiring options
- Counter-intuitive metal selection
- Communication examples with your circuit designer

Opening Thoughts on Parasitics

Nothing in an integrated circuit operates perfectly.

An IC is built of layers. You have metals running over other metals. You have transistors next to other transistors. You have transistors built in substrates. Whenever you introduce two different materials like this, you end up creating extra capacitances. It's like we deliberately placed lots of tiny capacitors all over our circuit. And worst of all, we cannot get rid of them.

An IC runs current through wires, implants, and through all sorts of materials. Wherever you expect current to flow you experience the resistance of the material. You end up with unwanted parasitic resistance. This is the same as placing tiny extra resistors in the circuit. And as with parasitic capacitance, you cannot get rid of them.

These and other extra parasitics act as unwanted physical components. They tend to slow your circuit, change the circuit's frequency response or cause a multitude of other nasty things to happen.

When the circuit designers begin their design, they have to take those parasitic components into account. If the parasitic, for instance, is reducing the bandwidth of an amplifier by 10%, then you have to over-design the amplifier to account for the reduction. You incorporate those parasitic reductions into your design.

Let's look at the three primary parasitics—capacitance, resistance, and inductance—and what can be done to alleviate their annoying presence.

Parasitic Capacitance

Where do parasitics come from?

They are everywhere, really. As we said earlier, every time you run a wire or you run a gate stripe or you create anything in a chip, you get some kind of parasitic.

Integrated circuits are loaded with parallel conductors running over each other, or even side by side. You can get a parasitic just by implants being next door to each other, or just by an implant existing in the substrate.

To illustrate how prevalent parasitics can be, let's look at four metal traces above two other metal traces (small blue boxes in Figure 4–1). Between each of these wires, there is effectively a parallel plate capacitor. There is also a capacitance from each of the four wires down to the lower layer, and from the lower layer to the substrate. We also have the fringe capacitances all the way down. Every little piece of your circuit speaks to every other little piece of your circuit, through some kind of a capacitance.

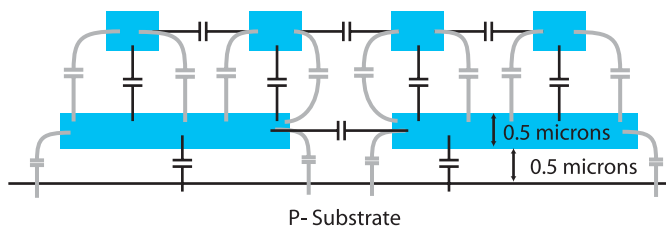


Figure 4–1. *Capacitance is everywhere. Everything is talking to everything else.*

Now, the values of these parasitics are somewhat small, due to the dimensions involved. For example, the metal could be only 0.5 microns thick, with perhaps a 0.5-micron insulator between the metal and the P-substrate. So, we're talking about a wire trace that could have a capacitance of 10 femtofarads (10^{-15}). That is small. However, they all add up.

If you have a circuit that is very insensitive to capacitance, like a power regulator, or something else that is quite hefty in the circuit, then you really do not care about these little extra capacitances all over the place.

However, the faster you go, the higher the frequency, the higher the speed of the circuit you are trying to work with, the more important these capacitances become. They do matter.

In most circuits, if you do not pay attention to parasitics, then the parasitics can kill your chip. Typically, when you do analog layout, whether it's CMOS or Bipolar, if there is any reasonably high frequency involved, maybe 20 megahertz or higher, you will have to worry about parasitics of some sort.

Ignoring parasitics can kill your chip.

Knowing when to worry about parasitics falls under one of our three questions from the previous chapter. Probably the first one, *What does the circuit do?*

Let's say someone tells you the circuit is an amplifier that is giving you a certain gain under certain conditions, and this circuit runs at a high frequency. Once you have learned it's a high frequency circuit, then you start to ask questions like, "Ok, how worried am I about the parasitics in this amplifier?"

You may well be told, "Yeah, it's really quite sensitive to parasitic capacitance on the input, so we need to reduce the input lines to be as short as possible."

Sometimes you are asked to utilize a certain technique, such as shortening the wires as in the above example. Sometimes you already have an idea how to handle the capacitance from experience, so you make your own adjustments.

Wire Length

If you are told some areas of wiring need to be low parasitic, one of the easiest ways to accomplish that is to keep the wire as short as possible, as mentioned above.

If you reduce the length of the wire, you are reducing the overlap between the wire and substrate, or the wire and something else that happens to be conducting.

Metal Selection

Another solution depends on the metal system that you have available to you.

The dominant capacitance issue is usually the capacitance of the wire going down to substrate. That capacitance is the one you are most interested in because substrate goes everywhere. It runs under the entire chip, so any announcement made to substrate is carried to every other component.

Worry a lot about substrate—it goes everywhere.

In Figure 4–2, we see two circuits, each placed in a P-substrate. You can see that each circuit has a capacitance to substrate. We also have the parasitic resistance of the substrate itself. The parasitics can couple the noise from circuit 1 into circuit 2 through substrate. This could be a real problem if you are trying to keep circuit 2 isolated from noise.

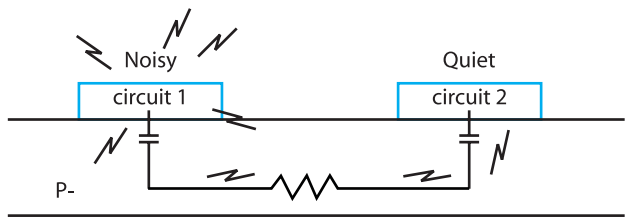


Figure 4–2. *If one circuit snores loudly, the substrate keeps everyone awake by telegraphing the noise through inherent capacitance.*

You can see that reducing the capacitance from circuit 1 to substrate reduces the probability of circuit 2 being affected by the noise. Work to keep all noise out of substrate.

Depending on your metal processing, a second way to reduce parasitics is to use the highest-level metal, the metal furthest away from the substrate. Typically, the further you get from substrate the less capacitance you have, because the distance between the two plates is a lot further. Capacitance is inversely proportional to the distance between plates, like other types of radiation. A little distance makes a lot of difference.

Unfortunately, just saying that the highest-level metal has the least capacitance is not always true. This is where your particular metal processing could make a difference.

The design rules for the metals may conspire against you. You have to look carefully through your process manual to calculate which metal is the lowest

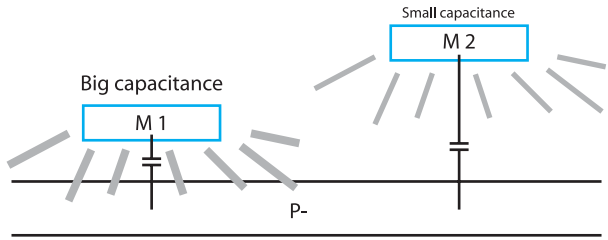


Figure 4–3. *Usually, closer metals have larger capacitance to substrate.*

capacitance, particularly since the minimum width for these metals may be unique.

Calculate, don't assume.

Here's an example with Metals 1, 2, 3, and 4. The minimum widths are 0.8, 0.8, 2.4, and 6.5 microns, respectively, as you can see in the table.

Metal	M1	M2	M3	M4
Min. Width	0.8	0.8	2.4	6.5
Cap/Unit Area (fF/μm^2)	5	3	2.5	1.5

Each of your metals not only has a minimum width, but each one will also have a different capacitance to substrate. If you look in the process manual, you can typically find a capacitance per unit area value for each of the metals. That number represents the capacitance per square micron from the metal to substrate. (*See last line of above table.*)

Metal 1 is closest to substrate. Therefore, it has the largest capacitance per unit area. Then, as you go up to Metal 2, Metal 3, and Metal 4, you see the capacitance per unit area drops.

However, the minimum metal width gets larger as the metal level increases in this particular process. Metal 4 is used for power busing and Metal 3 is used for secondary power. This is just an example, but it shows that you have to stop to calculate the lowest parasitic metal. There are too many factors to think you could use the same layer in every process.

In order to choose the lowest capacitance metal let's calculate the capacitance for a 10-micron-long wire. Multiply the length of the wire by the minimum width of the wire to give us the area. Multiply the metal area by the capacitance per unit area value. We see the results in the last line we have added to the table.

Metal	M1	M2	M3	M4
Min. Width	0.8	0.8	2.4	6.5
Cap/Unit Area (fF/μm^2)	5	3	2.5	1.5
Capacitance 10-μm wire	40	24	66	97.5

In this example, it turns out that Metal 2 is the lowest capacitance, 24 fF. This should seem counterintuitive. Metal 2 is not the metal that is furthest away from substrate. The factor that is killing us is the larger widths of the higher metals. Although the capacitances per unit area are smaller as you travel away from the substrate, the minimum widths are wider. So, if you want to create some wiring that is the lowest capacitance, then in this particular process, use Metal 2.

You can see that this example is one occasion where knowing a bit about your process gives you more knowledgeable choices for your layout. We are not using just a blanket approach. We are not simply advocating keeping everything as short and as small as possible. We make choices based on circuit function and processes available.

Circuit function told us this was a high frequency circuit. Being a high frequency circuit told us to keep the parasitics down. Knowing we must keep our parasitics down told us to make our wires short and run them in Metal 2. Circuit function drives layout choices.

You and your circuit designer may not even be talking about a high frequency circuit for these issues to arise. The circuit may just be particularly sensitive to capacitance. It could be a low frequency circuit, but the circuit designer tells us, “Hey, I’m really worried about the capacitance. Can you keep it as low as possible, please?”

Or you may have asked, “Are there any nets that you want to worry about from a capacitance point of view?” and been told about some circuits to watch. Ask your questions. Know about your circuit functions. That will tell you how to do your layout.

Metal over Metal

Up to this point, we have been talking about capacitance to substrate. As we mentioned at the start of the chapter, there are capacitances from everything to everything. For example, consider a circuit with a wire that runs over the top of another circuit. Parasitic capacitance develops between that wire and everything in the underlying circuit.

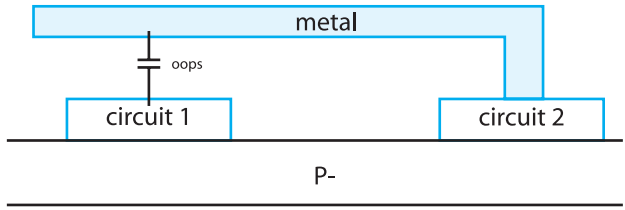


Figure 4–4. *Running metal over a foreign circuit.*

Running metal over circuitry is something they do in the digital world all the time. They have to, because the logic gates must be located as close to each other as possible so they can get millions of them into one die. Consequently, there is very little room to run wires in between circuits. They have all sorts of metals running over the top of each other. So there will be parasitic capacitance to your inverters, NAND gates, flip-flops, and so on.

There can be times when you will have critical wires in a digital circuit that are very sensitive to noise. However, the auto-router says, "Hmmph. I'm going to put that wire anywhere I darn well feel like." After all, the auto-router is not paid to think, but to route. And route it will, regardless of the consequences.

You can kill your chip just by letting the auto-router place your wires without supervision.

With analog circuits, we typically want to keep sensitive signals away from each other. So, if we had a chip with wires all over the place, it may not work as well as if we had the individual circuits spaced away from each other. With no wiring going over the circuits, just wiring in between the circuits, the parasitics are much more controlled.

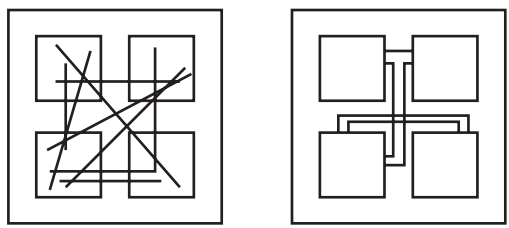


Figure 4–5. *Avoid running wiring over the tops of your circuits.*

Instead of running a wire over the top of a circuit block, you may have to wire entirely around a block because it's a very sensitive node.

Figure 4–6. *Top view. Option of running wires around circuits instead of over them.*

Again, you are making decisions based on key questions you ask your circuit designer about function. "What's the function of the chip?" or "What's the function of the block?" or even "What's the function of the net?" The choices you make depend on what the chip is doing. You may just not care about parasitic capacitances with some functions, with others you do.

If you are just designing low-level cell blocks, then the choices are more straightforward. But, when you begin to wire those cell blocks to each other, you have to ask all sorts of questions about an individual wire. "What does that wire do?" And, depending on the function of the wire, such as how much current it handles or what isolation it needs, you make a bunch of layout choices. This is very different from the digital world where 90% of the wiring is thrown together and who cares about the function.

This is an oversimplification, of course, but the point is to let the circuit designer lead you. Ask about all levels of function. After all, you have to know.

Parasitic Resistance

Another parasitic mentioned at the beginning of this chapter can also come along and bite you—parasitic resistance. Each wire has a parasitic resistance associated with it. And again, your handling of this parasitic depends on what the circuit does. This time we concentrate on our second question, "How much current does it handle?"

If you recall, we looked at current densities to see how the amount of current affected our wiring width choice. In addition to wiring width choice, current affects cell-to-cell wiring choices as well.

Calculating IR Drops

Let's say you have a wire that runs from one cell to another cell, which must handle 1 milliamp of current.

We look through our process manual to find the current density capability for this wire. We see that the metal we want to use can handle 0.5 milliamps per micron. That number tells us that we need to make that wire a minimum of 2 microns wide if we expect it to handle 1 milliamp.

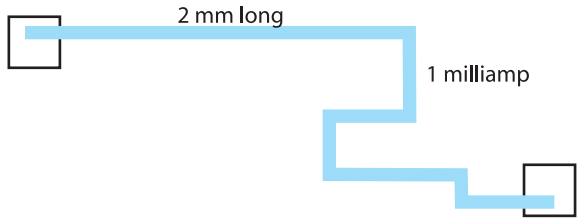


Figure 4–7. *How much will voltage drop as it travels this lengthy wire? Can our circuit handle the drop?*

So, ok, we draw a 2-micron wire. We're happy. However, what we weren't told by the circuit designer is that he is worried about the resistance of that wire as well. We calculate the resistance of the wire and let's say, for example, that the length of our wire from one side to the other is 2 millimeters long. Being 2 microns wide, that equals 1000 squares. (Dividing length by width gives you the number of squares.)

$$2\ mm \div 2\ \mu m = 1000 \qquad squares$$

Knowing the number of squares in our wire, we go to our process manual to find the resistance of that particular metal in ohms per square. We read that this metal is 50 milli-ohms per square. So, the resistance equals 1000 squares times 0.05 ohms per square.

$$\begin{aligned} R &= 1000 \cdot 0.05 \qquad ohms \\ &= 50\Omega \end{aligned}$$

The resistance through the wire is 50 ohms. Fifty ohms is a significant resistance. That wire is carrying 1 milliamp. Using *VIR*, you calculate that the voltage drop across that wire is 50 ohms times 1 milliamp. That is a 50-millivolt drop.

$$\begin{aligned} V &= IR \qquad && volts \\ V &= 50 \cdot 1 \qquad && (ohms)(milliamps) \\ V &= 50 \qquad && millivolts \end{aligned}$$

The difference in voltage level due to the current in this one piece of wire, is 50 millivolts. If the circuit at the other end of the wire is sensitive to voltage offsets, then we have trouble.

So again, it's a case of going back to your circuit designer and saying, "I'm just finishing up this chip. I've got this really long wire from one side of the chip to the other. You told me it was taking 1 milliamp, so I'm getting a 50-millivolt drop, is that too much?"

Rule of Thumb: Communication is key.

And the circuit designer will say, "Crikey, that's huge! That's enormous! I'm sorry. I forgot to tell you. I need a maximum of a 10-millivolt drop on that wire or my circuit won't work properly."

That means you have to make your wire 5 times wider. So, instead of running our 2-micron wire, we apparently need a 10-micron wire. That will lower the drop to only 10 millivolts, which is within the requirements for this particular cell.

These resistance parasitics typically manifest themselves in power wiring because power supply currents are usually pretty big. You can have 20 to 30 milliamps in one power supply. If you have a lot of circuits all connected to the same power supply, it needs to be sized to handle the right amount of current.

So, what do we do about that? That's next.

Wiring Options

You need to know the IR drop limitations and the amount of current flowing in your circuit. When you look at your top-level circuit, you may realize you have to split your power supply wiring into multiple pieces of wire just to handle these conditions.

In Figure 4–8 is an array of circuits. Their power supply runs along from the bond pad into each circuit as shown. Our circuit designer tells us that the currents for the various blocks are 1 mA, 5 mA, 10 mA, 1 mA, 1 mA, and 1 mA, as noted.

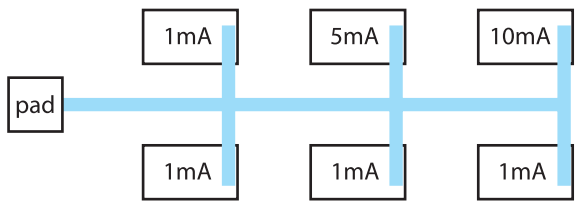


Figure 4–8. *Large power drain located furthest from pad creates a problem. (See upper right block)*

It appears we have a total of 19 milliamps all coming in from the outside world through the pad on the left. Unfortunately, the block needing the most current is furthest from the pad.

We could size our metal, all the way back to the last block, based on a total current of 19 milliamps. Let's use our 0.5-milliamp-per-micron example for our wire. In that case, the wire width we need is 38 microns to be reliable. (Total amps divided by amps per micron.)

Just give yourself a big, fat chunk of metal. The whole thing is 38 microns wide.

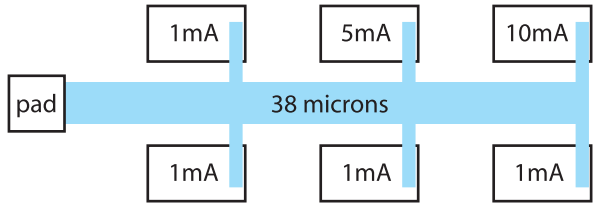


Figure 4–9. *Option of running thick power wire all along route.*

However, you might notice that toward the end of the line we are merely supplying 11 mA. It seems wasteful to use our fat chunky wire at the end. So, what we could do is taper the width as we move along the route.

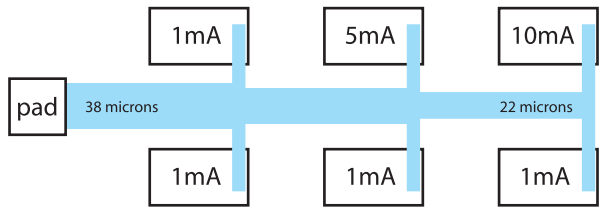


Figure 4–10. *Stepping down the width as we go saves room.*

Decreasing the width saves room, just in case we are strapped for real estate. We could start the wire at 38 microns, and then reduce the width as needed.

Here is another option. Why not bring the high current path back to the bond pad independently from the other wiring? You may need to use this option because you can get voltage drops caused by the 10-mA current that affect all the other blocks on the supply. This technique, of course, requires the chip real estate above the array of blocks.

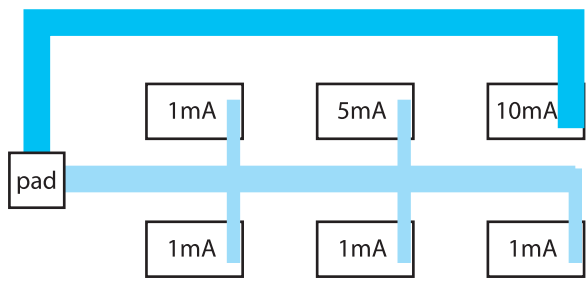


Figure 4–11. *Up and over.*

There are all sorts of options. Which method you use depends on the requirements of the circuit, which is based on the circuit function, and on what you've been asked to do.

Hopefully you are seeing that knowing a bit about the way the circuit works affects your layout choices. Mask design is not just a case of hooking things up and hoping for the best.

How did you learn these wiring techniques?

On the job training. I was asked by a circuit designer to specifically do these sorts of wiring techniques. He knew his circuit requirements. He knew it was essential, so I was told what to do.

After you see circuit designers make choices based on circuit function, you can pretty soon start to call the shots before they do. Pretty soon you know how and when to make calculations, or use a useful technique, and take care of the issues by yourself. I learned by watching and listening. Putting two and two together. Being told. By learning more about circuit function.

Of course, that took years. There weren't any layout books out there to just read. There wasn't anybody to give you formal training in this. We just learned by doing, by being told, by trying new things, and by making mistakes.

As a Rule of Thumb, I use a 10-millivolt drop as my cutoff. If anything is bigger than a 10-mV drop, then I go back and ask the circuit designer if it's a worry or not. There's a saying I have: "Only a fool breaks the 10-millivolt rule."

Rule of Thumb: If your IR drop is bigger than 10 millivolts, check with your circuit designer.

In order to reduce parasitic resistance, make sure that you use the thickest metal. You can usually find the thickness of a metal in the process manual. If the metal thickness is not explicitly stated, then the metal resistance usually is. The thickest metal has the lowest ohms per square value.

If your metals all have the same thickness, then you can sandwich chunks of metal on top of each other, as in Figure 4–12.

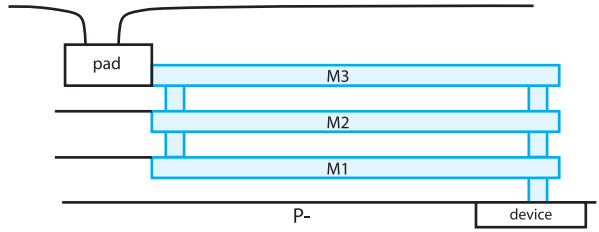


Figure 4–12. *Cross-section. Running three metals in parallel to save real estate and reduce series resistance.*

In this particular case, you effectively have three metals in parallel. You have reduced the resistance by a factor of 3 for this wire because the current path is shared. In very high current situations you may do your current density calculations and find, for example, that in order to be reliable, you need a wire 500 microns wide! Running strips in parallel is a good technique to reduce the resistance of high current paths and save yourself some space.

Parasitic Inductance

When you work with really high frequency circuits, the wires in your circuit start to have a parasitic inductance as well. The way to handle parasitic inductance is to try to model it, so that the inductance is calculated as part of the circuit.

Work with your circuit designer right away. Try to develop a floorplan of the chip very early so that the circuit designer can see how long the wires will be. He will incorporate some estimates of the inductances involved.

You may have to choose wires that are much wider than expected. You may have to leave room around certain wires because they are very inductive and very wide. You do not want them to inductively couple into other parts of the circuit.

Should you wait for the circuit designer to fix inductance problems? Just assume he'll work it into his design?

If you're worried about inductance on a particular wire, you ask your circuit designer, "What do you want me to do? Are there any special circuit techniques?"

It's a hand in hand working-through. You work on the floorplanning together. Your skills in mask design will help make the decisions.

Is there any time you can use parasitics to your advantage?

On the whole, no. Parasitics are evil.

You can design a circuit that relies on parasitics as part of your circuit, but it's very dangerous. You might read the parasitic numbers from the book and say, "Ok, I'll rely on the parasitic capacitance to give me some circuit function."

But, usually those parasitic capacitances are not controlled at all well. They can vary by plus or minus 50%. If you're trying to design a parasitic into your circuit that is a dominant component, and your circuit relies on that component, it's going to crash and burn. Death, doom, despondency. You won't get your laser-printed certificate. Cars will stall. Satellites will fall from the sky.

However, you can use parasitics to get some little extras for yourself. For instance, if you want a lot of capacitance, and you don't care how much it is, then you can design in extra parasitics to help out. For example, running supply and ground wires over the top of each other give you free supply decoupling capacitance.

Device Parasitics

Up until now, we have primarily talked about parasitics of elements that sit above the substrate. But, let's look inside. Look at the devices we have built in the substrate. You will see a mess of parasitics happening there, as well. The devices themselves have parasitics.

CMOS Transistor Example

Here's our old, faithful CMOS transistor. It sits in this big N well. You can see it has a capacitance from the well to the substrate, a capacitance from the gate to the well and a whole bunch of other ancillary capacitances.

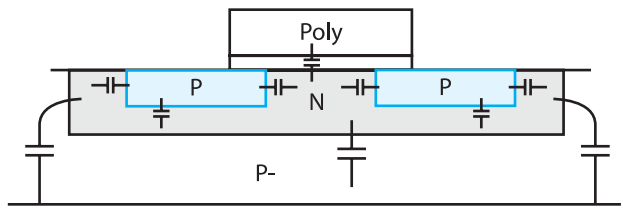


Figure 4–13. *Capacitance here, there, and everywhere.*

As voltage changes on a source or drain, it is slowed by the well capacitance. When a voltage is applied to the gate, it is slowed by the gate capacitance. The series resistance of the gate stripe, in conjunction with the gate capacitance, forms an RC time constant that slows the device even further. Almost every part of the device has some sort of capacitance that slows its operations in some way.

The only technique you can use to reduce CMOS device parasitics is to reduce the series resistance of the gate stripe. None of the other inherent device parasitics can be changed. If we reduce the series resistance of the gate stripe we can reduce the RC time constant of the gate, improving device speed. We can reduce the resistance by splitting the gate stripe into multiple fingers and wiring them in parallel.

Just by splitting a device in two, for example, reduces your RC time constant by a factor of 4. The resistance of each gate finger becomes halved. Plus, since they are now in parallel, that reduces the resistance by another factor of two.

Splitting devices and source-drain sharing can improve the parasitics on a CMOS transistor a lot. It also makes life easier for you when you lay them out. Long, thin transistors are hard to work with, having bits that poke out. Nicely squared, split transistor arrays pack more easily around other objects.

Bipolar Transistor Example

In Bipolar transistors, the collectors have parasitic capacitance from the implanted N directly down to substrate, similar to the capacitance across the well diode in our CMOS example.

Unfortunately, with Bipolar transistors, there isn't a lot you can do about it. Parasitic capacitance is just inherent to the size of the device. Luckily, though, the parasitics of the transistor are measured and modeled, so they are automatically taken into account when the designer does the simulations.

Proximity of two transistors to each other can hurt your circuit. In Figure 4–14 we see two Bipolar devices. The huge collectors of the two Bipolars sit side by side. We see in the figure that we have capacitance from the collectors down to substrate, and resistance along the substrate between the two transistors.

You might want to do some special stuff with substrate to help reduce the communication between these devices. What you can do depends on your processing options. Let's look at just a few.

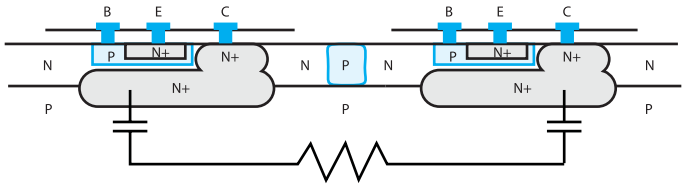


Figure 4–14. *Two Bipolars talking to each other through substrate.*

Full Custom Options

Devices can be made smaller using some full custom techniques. It's usually a matter of combining several smaller devices into one big device, usually in one common N well. That way your N well is smaller overall, reducing the capacitance to substrate.

If you have a lot of Bipolar transistors in parallel, instead of having lots of individual transistors to wire, think about combining the collectors into one. Fold it over, and merge the collectors so that the components are closer together.

Always remember to go back and check with the circuit designer. Just ask, "How worried are you about . . . ?" You may well have to come up with a custom transistor because somebody's very worried about the parasitic capacitance somewhere. Open the door with a simple question. Find out.

A full custom device could incorporate anything you and your circuit designer agree on. Your imagination is the limit. Be creative.

Closure on Parasitics

There are very few pleasant parasitics. We typically want them all eliminated, or at least reduced. We don't have many options, but that makes the few we do have all the more important. Communication with the circuit designer is likewise important.

Speaking of communicating with our circuit designer, in this chapter we have again used our three key questions, and used the answers to help determine what we draw.

- Knowing about the circuit function helps reduce parasitics
- Working with the circuit designer on the floorplan from the beginning helps to reduce parasitics

- Depending on the circuit requirements, you may have to do a full custom layout to bring the parasitics within reason
- Certainly, the circuit designer incorporates the parasitics into his simulations

I want more options. How about coming up with some? Don't ever think we are at the end of clever tricks and techniques. People come up with creative innovations every day. And eliminating nasty parasitics is a good one for you to work on. We mask designers everywhere thank you in advance.[1]

Here's What We've Learned

Here's what you saw in this chapter:

- Parasitic resistance, capacitance, and inductance
- Parasitics between wires, devices, circuits
- Effect of parasitics on a circuit
- Special high frequency considerations
- Metal sizing, routing, and tapering options
- Determining optimal metal layer
- Effect of circuit function on parasitic considerations
- Importance of working with your circuit designer

[1] You'll love your job even more when you select challenges for yourself to work on. Be creative. Have fun. That's what it's all about. That's how we've come this far in this profession in only a few decades—people like you feeling excitement over a new solution, a new technique all their own. Give your new idea a name and send it to us. Who knows? If it's helpful, it might become a whole section for our next edition.

CHAPTER 5

Matching

Chapter Preview

Here's what you're going to see in this chapter:

- Real world effect of poorly matched circuits
- How processing can keep devices from matching
- The difference device placement makes
- Why layout can make or break an end product
- How to avoid having to use parts of this chapter
- Importance of communication
- Many tricks you can do to improve matching
- The common mistake most people make using root devices
- Splitting components
- Wrapping components
- Why you would make a device knowing you will not use it
- Boxing halves of components across from each other
- How changing a design can help matching
- Why differential logic needs such good wiring

Opening Thoughts on Matching

Imagine you have just been to the music store where you bought the latest CD of your favorite band. You get home, put your headphones on, put the CD in your player, and start listening. Great music, as always, but something sounds a bit off.

You notice the left channel is slightly louder than the right channel. So, you get your CD player, and adjust the balance such that both channels play the same. Then, after 5 to 10 minutes, when your CD player is warmed up, you notice that now the right channel is louder than the left channel. So, you go back to the CD player and adjust the balance again. Every 5 or 10 minutes, depending on what's happening in the CD player, the balance on the CD changes.

Your CD player has a very badly designed circuit. As someone opens a window, or turns on an air conditioner, the temperature in the room changes, and so does the sound of your CD. What a bother. And so unnecessary.

The two amplifiers that drive the left and right channels seem to be reacting very differently to temperature. Probably, as the voltage on the battery drops, the frequency response of the amplifiers changes differently. So, you end up with one channel not only sounding quieter, but with not as much treble response.

When engineering circuits, you want partnered devices to react exactly the same way, whether in a CD player, a cellular telephone, or a Battle Bot receiver. In our CD example, you want the frequency and amplitude response of one amplifier to **match** and **track** the frequency and amplitude response of the other amplifier exactly. There are ways to do this. This is called **matching.**

By the way, all of the rules of matching, which are emphasized throughout the text, are gathered for you at the end of the chapter. No need to copy them down as you read.

Importance of Layout

In an IC, perhaps you are concerned about two devices matching due to the quirks you know will happen to components during processing. For example, let's say you might want one resistor to match another resistor. However, if your resistor process over-etches slightly, then to guarantee the best matching, you want all resistors to over-etch the same way, by the same amount.

This, and other matching procedures, can be accomplished successfully by the layout engineer. Or, ruined successfully by the layout engineer, for that matter.

Layout and matching are very tightly bound together. A bad piece of layout, from a matching point of view, can ruin a very good design. Conversely, a good layout can immensely enhance a design.

Let's look at the CD example again. You have two single transistors in two amplifiers. Those transistors directly affect the volume that you hear in the headphones. If the transistors are physically placed a long distance from each

other on the IC, depending on other nearby circuitry, one of those transistors could become much hotter than the other one. The resulting difference in temperature alone can make the characteristics of those two transistors very different from each other.

Accidentally placing matched components a good distance from each other can happen quite easily, since we are constantly laying out our components in, around, and about other circuitry. Sometimes a chunk of circuitry just gets in the way, and off you go, placing a transistor miles away. Keep them together.

Rule of Matching: Place matched devices close to each other.

Many years ago, when IC's were first becoming popular, manufacturing results varied substantially. Even if you placed two transistors close together, there were still no guarantees that those two transistors would look or behave the same. The processing technology was just not as good. They did not have as much control over the photolithography. They did not have as much control over the implants and diffusions.

Two identical pieces of CAD layout could act and work very differently by the time they were manufactured—this is called unrepeatability. Although everything in your layout would look identical, for some reason you simply could not repeat the same characteristics in your two devices.

Many layout techniques were developed to overcome process inadequacies such as these. With the advent of new processing technologies, a lot of the unrepeatability of circuits has gone away. That is not to say you do not have to worry about basic matching anymore. On the contrary, matching remains a prominent concern.

Skilled layout is very important for good device matching.

Let's look at some of the basic matching techniques that most people use. You will see that the skill behind your layout directly affects the way your circuit works.

One final thing before we move on. Although modern processes can produce very consistent results, with the increasingly smaller sizes of the devices we now design, you still have to be careful about matching issues in your layout. However, provided you know your process very well, and your circuit designer knows his process very well, just by being careful you can avoid some of the more laborious matching techniques. Some of the advanced matching techniques can become very painful. Learn them because you will use them, but also learn how to think creatively and avoid unnecessary work.

Good habits can eliminate some laborious matching work.

Learn the good rules of matching and use them constantly. Make them your daily habit in all your layout work. Not only will you avoid some of those really messy matching problems, but your everyday circuits will typically perform better as well. Be the one layout designer who consistently applies the rules that others keep on the shelf for emergencies. Your work will show the difference.

Importance of Communication

How do you know when to apply matching techniques to your layout? That depends on your circuit designers. After you have worked with your circuit designers for a few years, you will start to know what they like and how they like their layouts worked. Until then, you simply communicate a lot.

Communicate frequently with your circuit designer.

The best way to guarantee attention to all the matching concerns, is to see all the matching information written on the schematic itself. If the circuit designer wants a pair of transistors to match very well, he should tell you. He should write it on the schematic. If he doesn't, then he is running the risk of producing a circuit that does not work well.

However, even if nothing is written on your schematic, there are basic rules of matching that a layout person can utilize in all layout work. Provided you follow these rules of matching, you will automatically incorporate some reasonable matching for free. We have seen one rule already in our introduction. Let's examine more of those matching rules for everyday work.

Simple Matching

As with all everyday matching rules, once you incorporate these techniques into your daily layout design, you will automatically provide the best end product for your company, without conscious effort. Make these everyday matching rules part of your standard layout attitude.

The first rule we discussed in our opening CD audio example referred to placement. Keep your partnered components near each other. We saw that two similar devices could react differently, simply due to the temperature differences caused by separation of the devices. So we learned to place matched components near each other.

Pay attention to devices that are adjacent to the matched components. Even though two matched components might be located next to each other, the one

on the right, for example, might be just a little closer to a heat source. There you go again, different conditions. So be careful of neighboring components.

Rule of Matching: Watch the neighbors.

You can see in Figure 5–1 a drawing of two transistors. We see that the transistors have been placed directly next to each other. Well, good, you think. At least I have insured the proximity rule of matching. However, if you continue thinking about these two transistors, you notice another problem.

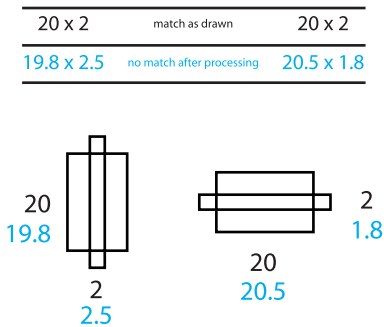


Figure 5–1. *What looks the same on the screen may result in different dimensions due to orientation-specific processing error.*

Just the fact that they are close to each other is good. Their configuration, however, is bad. Let's look at why this creates a matching problem.

For a CMOS transistor, the parameters that most affect the characteristics of the transistor are the gate length and the gate width. Some etches used in processing etch preferentially in one direction. That is the problem. One device is placed sideways. What etching errors occur in one transistor's width, will occur in the other transistor's length.

You could end up with oddball lengths in a poorly oriented device. For example, even though both devices begin with a drawn width of 20, one device ends up with a width of 19.8 after processing, and its partner ends up with a width of 20.5. Their characteristics would be very different even though they are the same device, taken from the same layout library in your CAD tool. The above example is extreme, but it demonstrates the principal we are discussing.

Our first rule for simple matching was keep things close. The second was to watch the neighbors. This new rule must quickly follow the first two: Keep devices in the same orientation. Now we have three rules for you to use every day in your layout work.

Rule of Matching: Keep devices in the same orientation.

If you follow these three basic rules, throughout all of your layout, you are guaranteed a certain amount of good matching, plus the advantage of better device performance.

There will be times when trying to keep all your transistors, resistors, and capacitors in the same orientation makes your layout very difficult, due to the sizes of the devices. That sounds like a good time to split up and reshape your devices. Once they are split, you will still apply the basic matching rules.

There will be times when you cannot break up your devices sensibly, or you are not allowed to. In these cases, how much you know about what the circuit is doing can help your layout. Find out what components are the least important in the circuit and maybe, just maybe they can be rotated to make your layout smaller.

For example, you might have a problem device that just will not fit well in your layout. If you think the device is non-critical, it may be a good candidate to rotate. Go to your circuit designer and ask, "Is it ok to rotate this transistor?"

If ever you want to rotate something through 90 degrees from the majority of the circuitry, always ask. Sometimes they will say, "Yeah, that's ok, don't care about that one." Sometimes they will help you find other solutions. Always communicate with your circuit designer.

Rule of Matching: Always communicate with your circuit designer.

Sometimes you may have to worry about not only having all the transistors within a circuit block match each other, but there might be a transistor half way across the chip that has to match back to these as well. Remembering that mask designers are not psychic, the burden falls fairly and squarely on the circuit designer's shoulders to communicate this to the layout engineer.

Rule of Matching: Mask designers are not psychic.[1]

If the circuit designer does not tell the layout engineer that a transistor has to match from block to block, they are asking for trouble. Circuit designers should be aware of the matching they want, and make sure everyone knows about it.

[1] MDANP. To be placed somewhere on every chip, hung in executive washrooms, and written on a sandwich board to be worn for one day by any circuit designer who fails to write adequate instructions on a circuit design.

Root Device Method

Sometimes we have more than two devices that must match each other. There might be five or six devices, all needing to match.

As an example, you might have a circuit with row after row of different value resistors, that all must match.

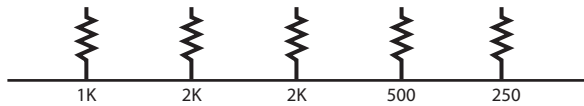


Figure 5–2. *Resistors of varying values. Have fun matching.*

The first strategy you can use is placing the resistors as close together as you can. This comes naturally, since you now practice the rules of matching all the time. What more can we do?

The second strategy is to keep them in the same orientation. Ok, no problem, you tend to think about orientation all the time, anyway. Now let's look at another technique that many people know about, but seldom tend to do correctly.

A good way of matching these resistors to each other, in this example, is to use what is called a **root component.** By a root component, I mean one resistor from which you will make all the others. Pick one of the devices, and make all the rest out of that same library device.

Using a root component, if the resistors are all the same size, all the same shape, and all the same orientation, and they are all close to each other, you get very good matching. If the resistors over-etch, they all over-etch the same way and still match each other.

Now, that's the classic statement. But, let's see what most people do wrong.

Pick a root device. (Go ahead, really. Pick a root device from the example in the figure. We'll wait.)

(Ok, we waited long enough.) Even now, and I've been working with people with 10 years experience, they make the same mistake. The root device they tend to choose is always the lowest value from which all other values are multiples. That's the mistake.

In this example, the lowest common factor is 250 ohms. Well, yes, you can make all the other resistors out of 250-ohm resistors. I didn't say it was impossible, I just see a better way. I'll show you in a minute. First, let's use 250 ohms, as most people do.

We could redraw our circuit like this:

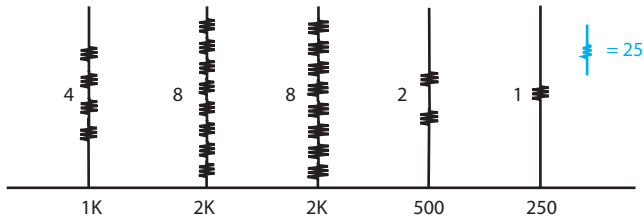


Figure 5–3. *All components are 250 ohms for better matching*

Everyone says, "Hey! That's perfect matching. They're all made out of a single resistor. Matching job done." And that's true. It's a pretty good statement.

However, you have some large resistors, like the 2K resistor, for instance. Consider this, contacts on a resistor are typically quite variable and using a 250-ohm resistor as the root component could have a significant portion of its resistance made up from contact resistance. The contact resistance could then create a significant amount of the total resistance on your larger resistors, which have eight times the number of contacts.

Even if the contact resistance is negligible, we can improve the situation.

Rather than think, "Let's find the smallest common value and have everything in series," I say let's find a value somewhere in the middle of all the values. Let's have some of them in series and some of them in parallel. That's it. There's your magic.

Rule of Matching: Choose a middle value for your root component.

A much better way to use the root component strategy is to pick a medium value. Let's pick one from our previous example. Let's choose 1K as our root resistor. The 2K's would each be two resistors in series. The 500-ohm is two resistors in parallel, and the 250 would be four resistors in parallel. We have made all our required values based on a 1K resistor. We used both series and parallel arrangements.

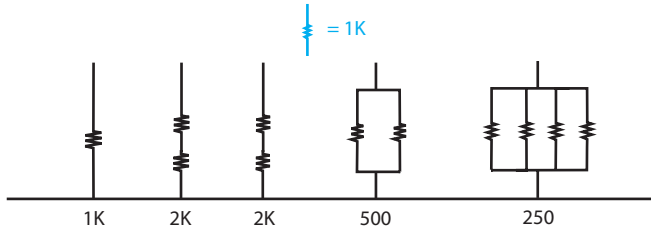


Figure 5–4. *A better root value is something in the middle of the range.*

The physical chip area will be the same as in the first attempt. It has to be. Even though there are fewer resistors, they are larger resistors. It all evens itself out. We still occupy roughly the same amount of chip area. Perhaps less, due to fewer gaps between resistors, so you win there.

Notice that we have reduced the total number of contact resistances, and the resistors are bigger, so the contact resistance contributes less overall. Our resistors are now dominated by the sheet resistance of the resistor body. By being practical, thinking about how you choose your root component, you can make a big difference.

This isn't a Saint Special, it's just common sense. I have seen other people choose a medium value root component, but too many people, even experienced designers, still generally pick the lowest workable number.

In fact, just the other day I showed this technique to a circuit designer who was having difficulty. He said he had been instructed by a Ph.D. to pick the lowest common number. Well. Sometimes that's a good reason to do things differently right there.

The root device method can be used with any type of device, not only resistors. The same issues are still relevant. Choose your root device sensibly.

Interdigitating Devices

We could improve the matching of our resistors even more. Typically in circuits, a whole bunch of components must match a given device called the defining component. Somewhere on the chip, or in the circuit block, there

might be a single resistor that starts the ball rolling. Everything must match, all the way back to that single resistor.

First, we must find our defining resistor. Let's, for argument's sake, say that in Figure 5–5 resistor A is the defining resistor.

Remember our rules. Keep things close. If we were to lay out our components left to right, like the circuit diagram shows, then resistor D will be the furthest away from resistor A. That's pretty far away. We want D as close to A as possible. But, then what about B? And C? Everybody wants to be the neighbor.

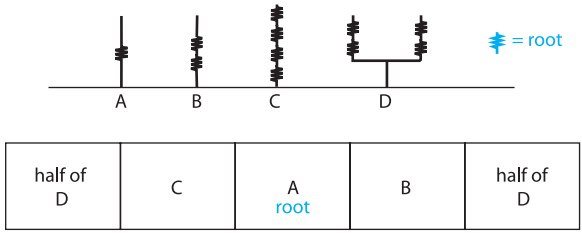


Figure 5–5. *Placing the defining component in the center keeps all matching components as close as possible.*

So, how can we place all of the components as close to the defining resistor as we can? Well, let's have resistor A in the middle. Then, wrap B and C around it. Then continue placing on either side until you run out of components, keeping the defining component in the center.

Resistor placement is a layout choice you make. You were given a circuit diagram, and hopefully, the circuit designer has devised a sensible way of implementing the components. It is now up to you as to how to lay these components out. Decisions. Your decisions. If you are lucky, sometimes you might even be given a diagram of how the circuit designer wants the components laid out.

Wrapping the root component, keeping it centered, is a very good solution to the above matching problem. This is also known as **simple matching.** Some people call it **interdigitation.**

Rule of Matching: Interdigitate.

Notice that resistor D was split into two areas. One half of the resistor was placed to the far left. The other half was placed to the far right. We wrap our devices around the root, even if we must slice a component in half to do it.

Let's have a look at a simple circuit. We are told that these two resistors need to match very well. They need to be interdigitated, as we saw above.

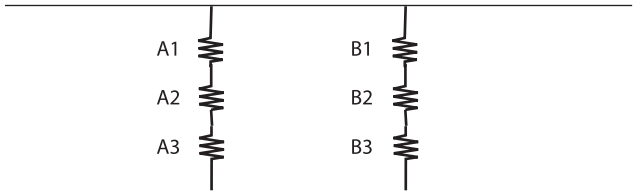


Figure 5–6. *We will interdigitate these resistors for best matching.*

Figure 5–7 is another example of wrapping. We have effectively interleaved two resistors with each other to achieve our simple matching. Follow the labels. We not only alternate A with B resistors, but we feed from opposite ends of the series. A1 sits next to B3.

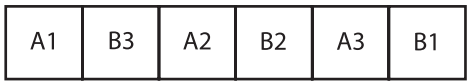


Figure 5–7. *Two resistors interdigitated.*

Interdigitation is a good technique and can be used not only with resistors, but with any device. You can interdigitate all sorts of things, provided you have two or more of them.

The above resistor placement follows our rules of matching: The devices are close to each other and they all follow the same orientation. Now, how would you wire them?

In this particular case, we could snake a line under and over to wire all the A's together. Then do the same for the B's, on a different metal, of course.

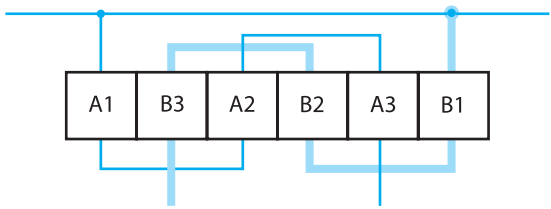


Figure 5–8. *Wiring the interdigitated components.*

Interdigitation is a very good, simple technique. You only alter the way your components are laid out, nothing more. You can use this technique with any number of components. You just need to choose how you want to interleave them, and how many of them to include.

Dummy Devices

Let's see how we can improve our matching example even more. The devices have the same orientation, are close to each other, and are interleaved as neighbors to each other. We have satisfied the three main rules.

However, if you look at resistors A1 and B1 in Figure 5–8 from the last section, they each have a side that is hanging out in free space, exposed. However, none of the central resistors has a side hanging out in free space. Aha, we have found something that makes our resistors different from each other. This is no joke. One little difference like this could ruin your very critical matching attempt.

When these components are etched, the ones in the middle of the block see very different conditions during processing than the ones on the ends. The resistors on the ends might etch more, making them slightly narrower than the ones in the middle.

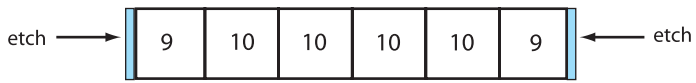


Figure 5–9. *Edges of blocks etch differently than middles of blocks.*

An easy way to etch all the components identically is to draw dummy devices on the ends that we do not actually wire into the circuit. You end up with some extra devices, not used at all as far as the circuit is concerned. They are there purely to give the real resistors a cushion against over-etching on the ends.

Figure 5–10. *Dummy devices take the hit of the over-etching, protecting our real devices in the center.*

It is very important to space the dummy components at the same spacing as the rest of resistors. Likewise, all the resistors should be spaced the same to ensure they see same conditions.

Now, if you are ultra-paranoid about the matching requirements of your devices, you can completely ring them with dummy devices on all four sides.

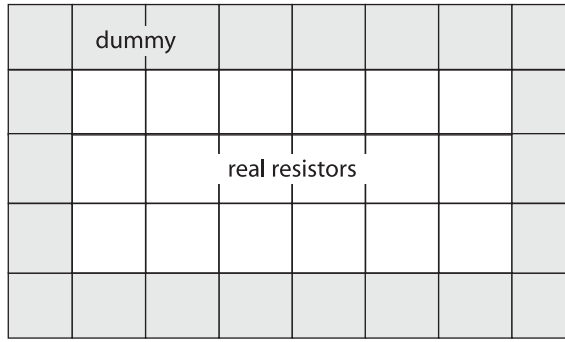


Figure 5–11. *Encased within a ring of dummy devices, the 18 real devices in the center are protected from over-etching on all four sides.*

Now we have dummies above, below, and on the right and left sides. They do not have to be the same size as the real devices, just adjacent.

The real resistors we are worried about are in the middle, and surrounding them on all sides are dummies. This happens every once in awhile, depending on what the circuit is trying to do and the precision you need.

Rule of Matching: Surround yourself with dummies.

Like all the matching techniques we have discussed, this technique can be used with any device, not just resistors.

Common Centroid

Placing devices around a common central point is known as **common centroid** placement. Even placing devices in linear symmetry is considered a use of the common centroid technique.

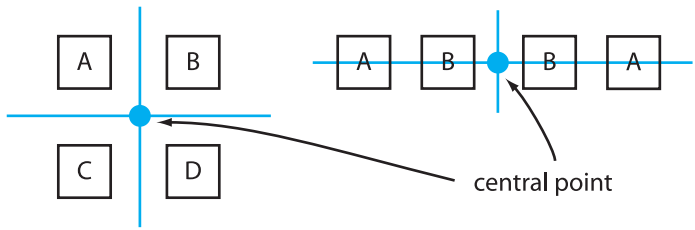


Figure 5–12. *Placement around a common central point.*

The common centroid technique is very good at reducing the effect of thermal or process linear gradients that may be present in an integrated circuit. A thermal gradient, for instance, is generated by a hot spot on the chip that can change the electrical characteristics of a device. Devices close to the hot spot will be affected more than devices that are further away. This technique distributes the gradient effect more evenly among the devices.

Even if the schematic calls for three devices, or five, or any other number, you might still be able to use the common centroid technique. Figure 5–13 shows more examples.

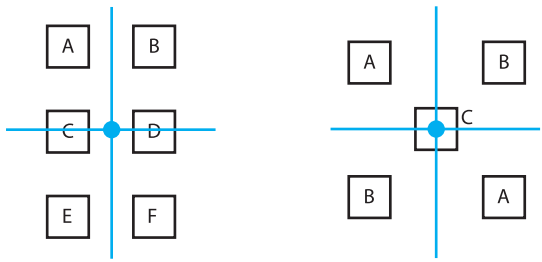


Figure 5–13. *More examples of common centroid placement.*

Cross-Quading

Although placing devices around a common centroid helps matching to some degree, we can use a special case of common centroid placement to even greater advantage if we have exactly two devices.

We split each device in half, and place the halves diagonally opposite each other. This special use of the common centroid technique, called **cross-quading,** is worth a closer look.

Suppose we are given an amplifier circuit with instructions written on it that say, "match very well" across a pair of transistors.[2] (See Figure 5–14.)

Cross-quading would work very well in this case. We will split each transistor into two halves, and place the halves diagonally across a common, central point. Diagonal emitters are connected. Diagonal bases are connected. Diagonal collectors are connected. We connect the diagonal halves in parallel, of course, so that the halves operate together as one device.

[2] Good circuit designers communicate their instructions as part of their circuit design.

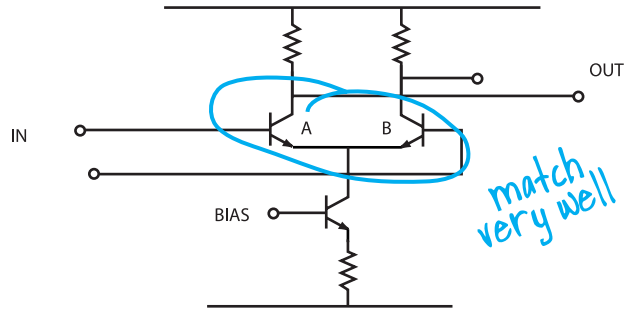


Figure 5–14. *Two devices to be matched very well. This sounds like a job for cross-quading.*

Rule of Matching: Cross-quad your device pairs.

You can only use cross-quading with two devices that have been split into halves. If someone asks you to cross-quad a single device you cannot do it. If someone asks you to cross-quad four totally different devices, you cannot do it. Your diagonal halves must always form a single device across the center point to be true cross-quading.

Cross-quading looks like a box. Notice in Figure 5–15 the halves of the devices are cross-corner from each other.

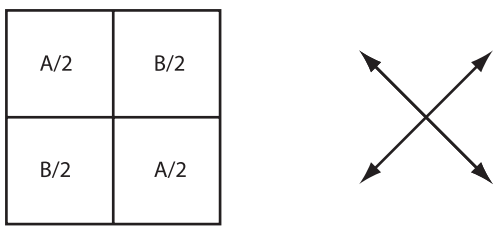


Figure 5–15. *Cross-quading technique. Split devices in half and place cross-corner to each other. Only works for exactly two devices.*

The technique is called cross-quad because there are four sections (*quad*), placed across from each other (*cross*). The cross-quad could be a pair of any type of devices, not necessarily just transistors, as in our example.

Figure 5–16 is an example layout of a completed cross-quad and its schematic.

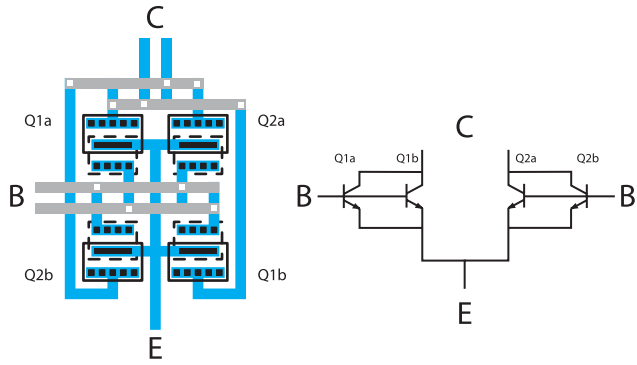


Figure 5–16. *Cross-quad Bipolar transistor layout.*

Take some time to trace through the wiring. The emitters are wired through the center of the layout. All four emitters are connected, as required by the schematic. The collectors are wired to the top of the layout, one output per pair, also required by the schematic. And the two base connections are output to the left of the layout.

If you look at Figure 5–16 more closely, you will notice there are some extra overlaps in the collector and base wiring that do not need to be there. The extra overlaps help balance some of the crossover parasitics. We try to make the wiring the same length, with the same overlaps, on the same metals—everything identical.

Rule of Matching: Match the parasitics on your wiring.

Here is a much simpler version of the cross-quad technique that yields reasonable matching performance. This linear configuration takes less time to lay out and saves space, so it is sometimes called the **poor man's cross-quad.** This option uses the A-B-B-A linear version of the common centroid technique.

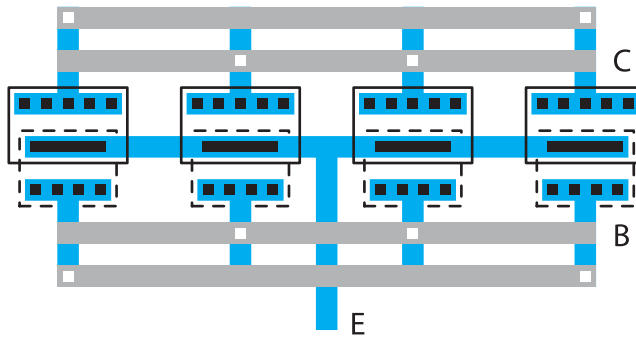


Figure 5–17. *Poor man's cross-quad.*

As we did in Figure 5–16, we have included extra overlaps on the base and collector connections to equalize the wiring parasitics.

At times, you might even approach your circuit designer with the idea of combining or splitting devices to create a pair that can be cross-quaded. He might think that is a good idea and go about changing his schematic to suit your suggestion. With experience you will have a better idea when this will be worth suggesting.

Symmetry

Symmetry is a major concern in matching devices. Let's say you are working with some big circuit blocks that need to be matched. Without paying attention to lines of symmetry, you are working against yourself.

In Figure 5–18, the output from block A is correctly wired to blocks B and C. It's a fair layout. It will work.

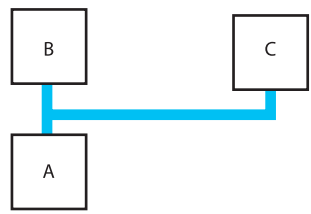


Figure 5–18. *Casually placed layout, poor matching.*

However, block C has a much bigger parasitic on it because you have given it a longer wire. A nicer way to wire these blocks is to imagine a line of symmetry between the blocks, then place the blocks in mirror image on either side of the line.

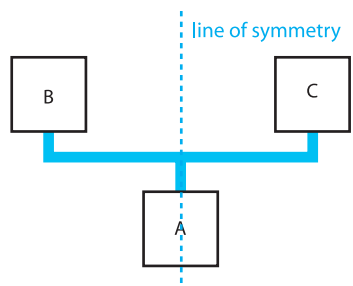


Figure 5–19. *Imaginary line of symmetry helps components match.*

Particularly in high frequency circuit layout, if you want to match parasitics, you have to lay out blocks across a line of symmetry. It is essential to do this mirroring.

Rule of Matching: Keep everything in symmetry.

You may have to observe multiple lines of symmetry at different magnifications in order to keep the circuitry well-matched. Draw your layout around lines of symmetry all the way down the hierarchy. Find symmetry within devices. Find symmetry within blocks. Find symmetry within sections of blocks. Find your lines of symmetry at all levels.[3]

For one thing, not only will your devices match better, but you could save yourself some layout time by cutting and pasting, as well. Just a thought.

Matching Signal Paths

One circuit technique that needs very good matching is called **differential logic.** If you hear the word *differential,* pay close attention to matching.

In CMOS logic, there is a 0 and a 1 represented by a high or low voltage. In CMOS logic, you only have one wire per signal. That is all you need. One wire can carry the low or high condition that is necessary to determine the logic state.

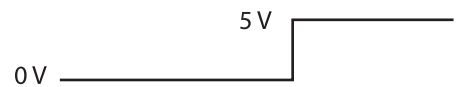


Figure 5–20. *CMOS logic. Low or high. That's all you get.*

However, in differential logic you have two wires per signal.[4] Determining the difference between the two signals on the two wires gives you the logic state. We still see low and high voltages, but we determine the logic state by subtracting one voltage from the other.

See if you can follow Figure 5–21 before reading on.

Figure 5–21 shows the two waveforms of a single differential signal.

[3] I use the word *find* rather than create. Symmetry already exists in any circuit, we just have to find it. Sort of a zen tool, helping the layout come to its destiny.

[4] We will talk about one advantage of differential logic in the Noise chapter.

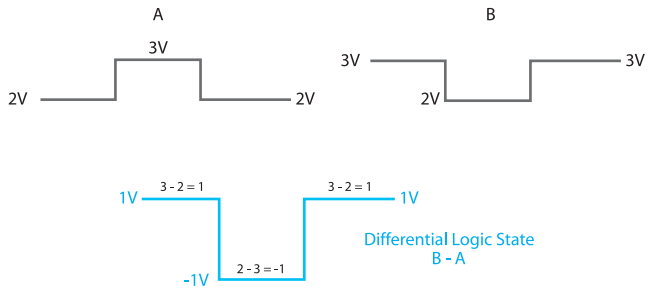


Figure 5–21. *Differential logic. Low and high are determined by the difference between two voltages.*

Signal A is always the inverse of signal B, and both waveforms change state at the same time.

Remember, the difference between the two voltages defines our logic state, voltage B minus voltage A.

If we look at the difference between the two voltages at the beginning of our waveform, we subtract 2V from 3V and get a value of 1V. That would be our high logic state.

In the center of our figure, we subtract 3V from 2V to get a value of -1V. That would be our low logic state. At the far right of the figure, we again see a resultant difference of 1V. The logic state at the end is, then, high.

Notice that the differential voltage swing is actually twice the voltage swing you see on either one of the waveforms, A or B. The input waveforms only rise or fall by one volt, but the differential voltage between the two waveforms rises or falls by as much as two volts.

For differential logic to work well, you must match the two signal lengths in your layout. If you are trying to drive a differential signal from block A to block B, you need to wire the two signals as nearly identically as you can. If the two wire paths are the same, the parasitics will be the same, the time constants will be the same, and block B will see both input signals rise and fall at exactly the same time. Differential logic relies on identical wiring.

Figure 5–22. *Run two signals along matched wires. Comparing the signals using differential logic increases accuracy.*

The parasitic capacitance and resistance of the wires cause the voltage waveforms to rise and fall slower than we would like. They really do not appear as boxy or clean-cut in the real world. Figure 5–23 is what our real differential signals look like.

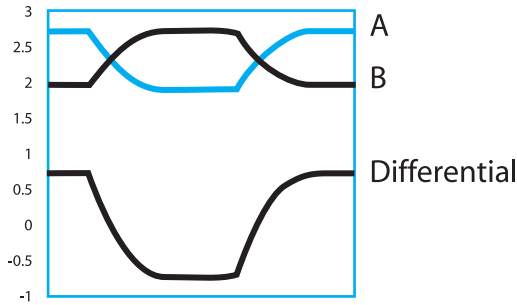


Figure 5–23. *Real world differential signals. Not perfect, but we do see a flattened bottoming out where our differential signal is low.*

Often, some circuitry can get in the way of our wiring, forcing one wire to be drawn longer than the other is drawn. The capacitance and resistance on the wires now differ substantially. Bad news for matching.

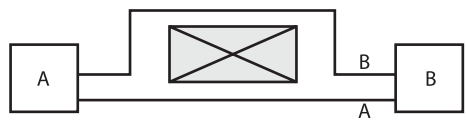


Figure 5–24. *Unfortunate wiring around an obstacle ruins matching.*

In Figure 5–25 you can see the voltage waveforms that would be produced if one wire had a much larger parasitic than the other. Signal B is much longer and consequently has larger parasitics. The rise and fall times of signal B are much longer than the rise and fall times for signal A. Our differential signal now has a non-ideal shape that can cause problems.

Rule of Matching: Make differential wiring identical.

In differential logic, it is essential to have well-matched path lengths and wires. As mentioned in an earlier section, not only do you have to match components to each other, but sometimes you have to match wiring signals to each other.

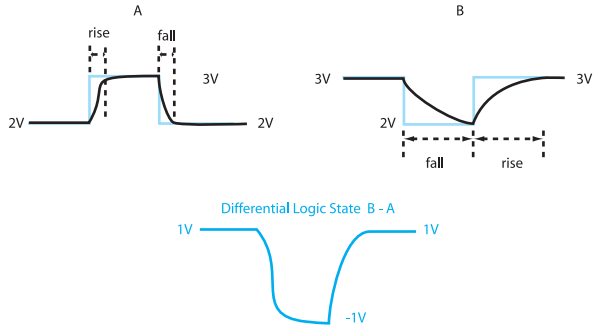


Figure 5–25. *Realistic wave forms, showing time constant delays. The differential signal almost never flattens out to a constant low value.*

Device Size Choices

One final section on matching. Device size.

If you need two devices to match very well, choose the devices to be the same size.

As an example, if you want four resistors to match each other, then do not select one of those resistors as 5 microns wide, another 10 microns wide, another 2 microns wide and a fourth one as 20 microns wide. They all need to match each other, so choose one universal width that is sensible, and have them all the same width. Vary only the lengths.

Rule of Matching: Match device widths.

If you made all your resistors only 2 microns wide, they would match very well, but their inherent matching is still poor because of the variability of the processing. They are likely to change by plus or minus 0.1 microns. And, if we're dealing with only 2 microns to begin with, a change of 0.1 microns is a reasonably high percentage error. If you make your resistor wide enough in the first place, then some of the photolithography effects will not hurt you.

If you were to have very wide resistors, like 10 microns wide, then that 0.1-micron etching error is a much smaller percentage to worry about.

Rule of Matching: Go large.

Good matching not only is a layout issue, but it is also a circuit design issue. So, if you see on your instructions, "These resistors need to match really well,"

but the device sizes are all different lengths and widths, wacky all over the place, go back to the circuit person.

Say to the circuit designer, "Before I waste my time laying this out, you say you want good matching here, but you haven't done your circuit design particularly well for matching. Can we work together on this and choose some device sizes that give you the kind of matching that you want, because it's one of those things that could kill your chip?"

The designer is likely to change widths. Physical widths and lengths can be altered without changing the necessary circuit values. You can make a 1K resistor that is 10 microns wide or 1 micron wide. The dimensions are different, but the value remains the same. For better matching, you would use the larger dimension in order to reduce the effect of the etching error.

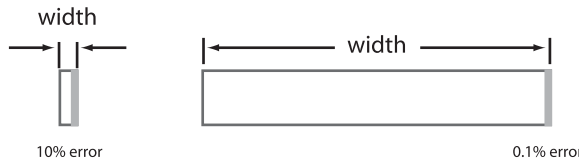


Figure 5–26. *Same processing error, but different percentage error overall.*

A bigger resistor is less susceptible to error as the processing variations have a much smaller percentage effect on a large device than a small device. If you cannot use the root component approach, then choose a reasonably wide resistor and then just live with the fact that there will be some inherent mismatching. If you make the resistor wide enough most of the problem goes away.

Rule of Thumb: Minimum resistor width is 5 microns. Minimum resistor length is 10 microns.

Over-sizing is one of those practical approaches people often use. With some people you see these huge resistor networks, and you ask, "Why do you need to do this?"

They respond, "Well, they all need to match and I've got all these weird values. The only way I can get a decent match is to have them all made with large widths."

Not only does the device size choice affect resistors, but the same rule applies to every component. If you want two components to match, make them a reasonably large size, meaning non-minimum. If you expect minimum devices to

match each other, you're sailing into the sunset. You're gonna die. You'll burn your feathers off.

Closure on Matching

What I try to teach people is, even before you start layout, just sit and look at the circuit. Spend the time to find any symmetry you can. Start there. If you can find lines of symmetry, you can lay out half the cell, copy and flip that much work, and save yourself some effort, as well as ensure that the parasitics are even.

You also need to keep in mind what blocks and what components this piece of layout will interface with in the grander scheme. So you have to know a wee bit about the chip and the system you are working in.

Matching in its purest form is very simple. It is one of those subjects that generates a lot of mumbo jumbo. You will hear people spout off, "Oh, the matching is really tough to get." In reality, it is just a matter of being thorough, understanding what you are working on, and caring about your work.

Speaking of caring, one of the biggest mistakes you can make is if you really do not care about what you are laying out. You are almost certainly going to make the chip not work as well as it could.

I just got off the phone with a designer. He'd been looking at the layout and he'd noticed that the metalization from two transistors was running north/south.

The output from one of the transistors had more metal in it than the other one did. So, he felt that there would be extra resistance in the northernmost transistor. "Could we re–lay out the metalization so that it came out on one side, and both outputs had the same metal resistance?" he wanted to know.

When I looked at the layout and measured the distances involved, we had just over one square of metal extra in one of the transistor outputs. However, the metal that was joining the two transistors was over 20 microns wide and had a sheet resistivity of 0.01 milliohms per square.

This effectively means that one transistor has just over 0.01 milliohms extra resistance. Technically, he was right. One transistor did have extra resistance in it, so the matching was not optimal. However, examination of the practical side of the layout showed that a difference of 0.01 milliohms was a non-issue in this case. Puny. Who cares?

Lesson: Above all these rules, be practical.

Here's What We've Learned

Here's what you saw in this chapter:

- Stereo channel amplification example
- Over-etching and under-etching effects on matching
- Proximity and similarity of matching components
- Importance of layout on matching
- Importance of daily matching practices
- Importance of communication
- Interdigitation, root device method, sizing, and other techniques
- The common mistake most people make using root devices
- Splitting components
- Wrapping components
- Advantage of using dummy devices
- Cross-quading, common centroid technique
- Redesigning a component into two components for splitting
- Differential logic wiring needs

Rules of Matching

- Place matched devices close to each other.
- Keep devices in the same orientation.
- Choose a middle value for your root component.
- Interleave or interdigitate.
- Surround yourself with dummies.
- Cross-quad your device pairs.
- Match the parasitics on your wiring.
- Keep everything in symmetry.
- Make differential wiring identical.
- Match device widths.
- Go large.
- Always communicate with your circuit designer.
- Layout designers are not psychic.
- Watch the neighbors.

CHAPTER 6

Noise Issues

Chapter Preview

Here's what you're going to see in this chapter:

- Worst kind of circuits for noise problems
- Causes of zaps, spikes, and *krrrr* sounds
- Key analogy to remind you of common sense solutions
- Why you might suggest a certain library
- How to create a Wall of Death
- Or two Walls of Death, one for you
- Timing solution
- Location, location, location
- How to make a 360° shield
- Running two lines instead of one
- Placing a capacitor in the neighborhood
- Stacking your power rails
- Unwanted secret signals created by your main signals

Opening Thoughts on Noise Issues

Noise in an integrated circuit can be a pretty big problem, particularly when you have very sensitive circuitry that is trying to pick up some very low value signals, located next to noisy circuitry that's doing all sorts of computations, control logic and noisy switching. We have to pay very special attention to our layout and our floorplan. We need to understand where the signals are wiring to and from. We need to be real careful.

Noise becomes very painful to deal with, particularly in mixed signal chips. Mixed signal circuitry has the most problems with noise because analog and digital circuits operate at very different noise levels.

In a digital circuit, you have two transistors that connect across a power rail. Every time they open or close, there is a possibility that for a very small amount of time both those switches can be open, causing a dead short across the power rail. Multiply that times 10 thousand. Ten million, even. Likewise, every time a logic state changes, an FET gate capacitor is charged or discharged. This charging current flows through the transistors to a rail causing another current spike.

So, every time something happens in the digital world, you get a spike of current. That spike manifests itself as noise. It's like a lightning zap. Wherever you are in the AM dial, whenever there is a lightning strike nearby, you will hear a *krrrr* zap on the radio.

The same sort of constant *krrrr* zapping and popping occurs in a chip. Digital logic is inherently very noisy. It's the Walter Matthau of *The Odd Couple*. It snores loudly and leaves dirty socks everywhere.

Usually in a mixed signal chip, you are trying to pick up an analog signal, like a radio signal, for instance. These are very weak, very tiny, very neat signals. Jack Lemmon signals. People add lots of amplification to the design to make these weak signals louder. Consequently, you amplify all the unwanted noise that exists around the signal as well.

Depending on the application, if you are working with a purely analog chip, it could well be that the output of your analog chip is also the unwanted noise. You will try to isolate the input of the chip from the output of the chip, so that it can pick up a quiet signal without having to listen to itself. You do not want to pick up the chip's own output. It's like putting a microphone in front of a loud speaker. The whole thing takes off in a spiral and you'll never get it back again.

Noise can ruin a chip. It's crucial that the mask designer know as much about cutting down the noise as possible. Let's start with a little story to illustrate that your common sense solutions to everyday noise problems are the same solutions you can use in your chip layout.

Noisy Neighbors

As an example of the kinds of remedies you can use to reduce noise, let's look at the Noisy Neighbors Example, a particular favorite of my examples, because it involves a rock band.

It's a nice, sunny afternoon. You're sitting out in your back garden talking with your silver-haired grandmother. She's 93 and speaks very softly, very quietly. You are enjoying her reminiscing about the good old days; back before airplanes and auto-routers, when her job was to hitch the team every morning, when people only bought flour and fabric.

Suddenly, your neighbors come home with their teenage son's rock band in tow. They go out on their patio, set up their equipment and start playing their latest tunes.

The noise from the neighbors totally overwhelms what your grandmother is explaining. You can't hear a word she is saying anymore. So what do you do about it?

The first thing to do is pop next door, and say, "Excuse me, can you turn your music down a bit, please?"

Being nice, well-brought-up teenagers, they say, "Oh, of course we'll turn the music down." So, you go back home. The music is quieter, but you still can't hear what your grandmother is saying.

You struggle through for a while, but soon you go next door again. You say, "I know I asked you to turn it down, but do you mind moving the band inside?" So, again, since they are well-brought-up teenagers, they spend 15 minutes moving all their equipment back into the house.

You go back home, and you can just about hear what your grandmother is saying now. However, the band forgot to close the windows of the house, so it is still difficult to hear. You go back for the third time and you ask, "Can you close the windows, please?"

They are inside the house. They have turned down their music. And, they have now closed the windows. However, when you go back into your garden, you still find it difficult to carry on a quiet conversation. The thumping of the bass and drums is very distracting.

Having thought you have done everything you can do, you decide, "Well, I really want to hear what Grandmother wants to say because she's giving me some wisdom here. She was just about to tell me her theories regarding placement algorithms." You and your grandmother go inside your house. You shut all your doors and windows. Finally, everybody is happy now. You can clearly hear what your grandmother says and the teenagers next door are able to rock out.

If they are misbehaved teenagers, if they refuse to go back inside, or if they instead turn up their music just to upset you, then you can call the

sheriff.[1] The sheriff comes along and insists that the music stop. The teenagers shut down the band, but that only works for a while. Soon the band decides they want to rehearse again.

With the help of the sheriff, you work out an arrangement such that the band is to stop playing after 9:30 at night, and they agree to only play on Wednesdays and Saturdays.

Knowing this rehearsal schedule for the band, you can arrange to visit with your grandmother during other hours or other days, when they are not rehearsing. In fact, since you know their schedule in advance, you arrange with your grandmother to take her shopping and to the movies while the band rehearses. You were well brought-up, too, you see.

Now, a drastic final option is that you move house completely. You just find a quiet neighborhood somewhere else and buy a house there.

From turning down the volume to moving, these are very real world examples to reduce the noise of a neighborhood rock band. Each tactic is, of course, common sense.

We will next discuss each one of these tactics as it relates to mask design. Each scenario directly applies to what you can do in layout to reduce noise in your integrated circuit.

Common Sense Noise Solutions

Let's go through each one of our real world rock band solutions, one at a time. We will see how they map across to mask design techniques—what we can do in both the design world and the layout world to achieve some of these noise reductions.

Turn Down the Volume

Our first request was to ask the rock band to turn down their music. Turning down the volume would be like reducing the signal swing in a circuit. In layout there is not a lot we can do to reduce the signal swing.

By **signal swing** I mean the amplitude or value of the voltage that is being wired around the chip. In a digital circuit, for instance, the *zero* state is represented by 0 volts, whereas the one state can be represented by 5 volts. This is a voltage swing of 5 volts.

[1] Chris actually said *sheriff.* Really. I didn't have to translate from bobby or constable or something. Jolly good. California is rubbing off on the ol' boy from London town.—*Judy*

Now, if we were to reduce that voltage swing to only 2 volts, then we would be switching less energy each time a flip-flop flips. Or flops. That is the direct equivalent of turning the noise generator down, turning the volume down.

As we mentioned in the opening thoughts to this chapter, mainly the digital part of a mixed signal chip is the section that needs to be quieted. So, if you can get a digital logic family that is inherently quiet, i.e., that has very small voltage swings, then that helps keep the overall noise down from the start.

Voltage swing is not primarily a layout issue, but it can be, to a certain point. Suppose your circuit designers come along and say they have a chip for you to lay out, with a reasonably high chunk of digital circuitry. They have indicated that they plan to use this certain library, which you know is a 5-volt swing library. You remember that you used a digital library on a similar chip some time back that had a 2-volt logic swing.

You can turn around and say, "Well, are we worried about noise? Because there is this 2-volt swing library that could do the kinds of things you want. That might help."

And they might agree. The amount of voltage swing is mainly a circuit designer's decision, but a good mask designer knows the options and when to suggest them.

Rock Band Moves Inside Their House

The second item we talked about was asking the rock band to go inside their house. The house provided some isolation, some sound insulation. The sound had to get through the walls of the house before it got to our ears. Now, this is something we can definitely devise in layout.

We can effectively hide a noise-generating block behind a Wall of Death, as I call it. It's a Wall of Death because it stops anything trying to get past. We mentioned this earlier.

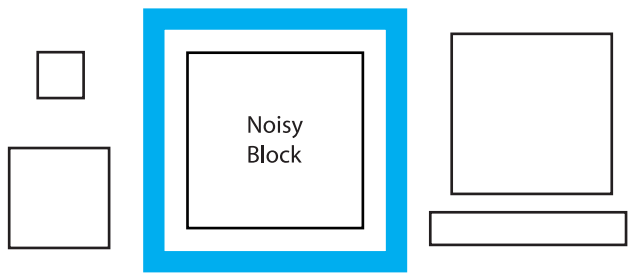


Figure 6–1. *Put the noise inside a little house.*

So, you make a wall. You make all the noisy circuitry go inside its own little house.

There are various layout techniques to make that wall. How you build your wall depends on who you are, what process you have, and what test chips have been made. The method is variable, but one easy way is to wrap a big ring of ground substrate contacts around the whole block. Remember, noise travels through substrate very easily.

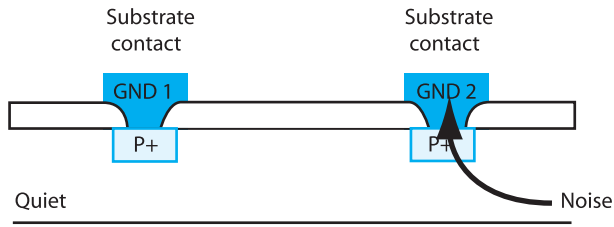


Figure 6–2. *Placing some slippery exits in the way helps eliminate traveling noise.*

As the noise tries to escape the block through the substrate, it first encounters a ground substrate contact. Hopefully, any noise voltages or noise currents will be attracted to this contact since it is grounded.

If we were to have a very small substrate contact as our ring, noise could work its way around or underneath it. A big contact means the noise can't get through. So make it a decent size.

Here's a bond pad in Figure 6–3, labeled GND. We put a voltage source on the pad, which measures zero volts. Then, as you can see, we have a thin, skinny resistive wire that connects to the substrate ring around our noisy area.

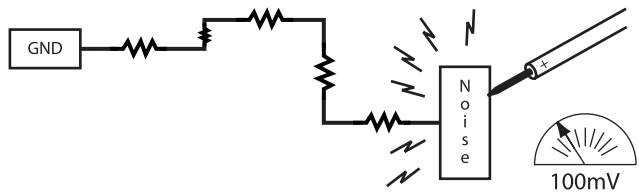


Figure 6–3. *You might think it's grounded, but with what size wire?*

This must be the work of someone new who hasn't read their books yet. They think they have grounded the substrate ring, their noisy source, adequately.

They think they should have no more noise. Their chip comes back out of spec, surprise, surprise.

There's a fair amount of resistance in each of these pieces of wire. So, if you were to measure the voltage at the noisy block with respect to ground, then it may not read zero volts, as hoped. It may read, say, 100 millivolts. All this extra wiring gets in the way, making escape easier. Noise will get past the wall.

If you are really worried about noise, do not use skinny wires. Use a nice, big, fat piece of wire.

Figure 6–4. *Now that's what I call grounded.*

Now when you measure the voltage on the noisy block, ideally it should be the same voltage as the ground pad. That's what we refer to as being **hard-tied** to ground—very well grounded.

With this large substrate contact that is hard-tied to ground, it is very difficult for any noise in the substrate to propagate through the ring. It all gets mopped up.

This particular ring is just an example to show you how we can ring a structure to build a Wall of Death. The rings you build will all depend on your process and your choices.

If you are lucky, your manufacturing people will have characterized some isolation structures for you. If not, you will end up with people guessing, thinking certain methods will help best.

In fact, you might consider involving yourself, if you have the interest, in determining what makes the best Wall of Death for your situations. I'm sure there will be quite a few advances between now and our next book revision—one may as well be yours.

These rings are referred to as **guard bands.** You could even have multiple guard bands around your noisy block to try to reduce the noise leakage even further.

Go Inside Your Own House

We have already placed guard bands around our noisy block. Now, not only might you guard band your noisy block, but if you have a quiet block somewhere in your circuit, you could guard band both of them. It's like walking into your own house as well as the band walking into theirs. Twice the isolation.

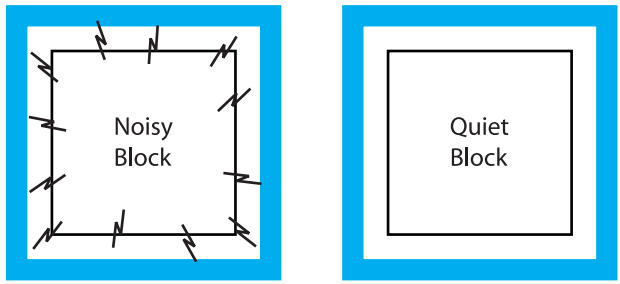


Figure 6–5. *Maybe both blocks should go inside their little houses.*

Close All Windows

Putting a solid guard band around the whole noisy block is effectively like putting the noise generating circuitry inside the house with all the windows closed.

If you have any gaps in the guard band, it's like having some windows open. Some noise can leak out. And believe me, a tiny opening in just one window can let a lot of noise escape. Keep the guard bands closed.

Call the Sheriff

The equivalent of the rock band agreeing to only practice during specified times of the week correlates to a design issue more than a layout issue. The circuit designers can say, "Ok, I'm going to architect my system such that the noisy circuits don't do anything while the quiet circuits are listening. And then, when the noisy circuits are doing their stuff, the quiet circuits are not listening."

When the noisy circuits are active, the quiet circuits have gone to the movies. They will alternate activity periods.

If you have the flexibility within your chip, having quiet and noisy circuits doing their business in separate time periods will eliminate that noise problem. Even though it is really a design issue, the mask designer should be aware of it as an option that circuit designers can employ. And who knows? With experience, the mask designer might be the one to make the suggestion in a design review meeting.

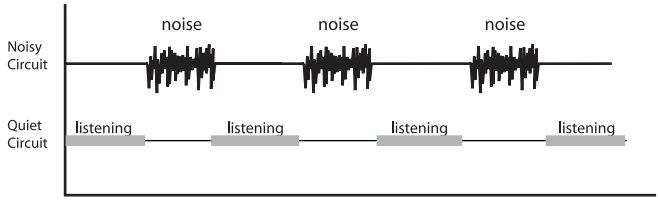


Figure 6–6. *Timing is everything.*

Move to a New Neighborhood

Next let's look at the concept of moving to a new neighborhood in our rock band example. This option is directly a mask designer's technique.

Here we have two floorplans of the same chip. (See Figure 6–7.) Our initial floorplan on the left shows the quiet stuff and the noisy stuff placed next door to each other. But, if you can, why not have the quiet and the noisy stuff as far away from each other as you possibly can, as in the floorplan on the right?

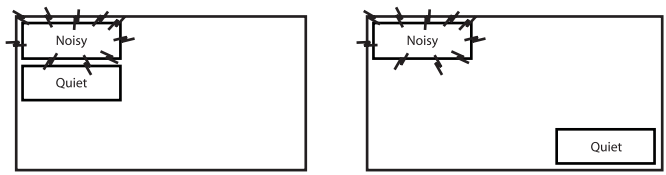


Figure 6–7. *"Hey, buddy, do you know how to sing 'Far Far Away'?"*

This placement technique should become daily habit for good mask designers. With time, you just know to automatically place noisy and quiet circuits away from each other, without thinking about it.

After doing these good habit techniques long enough, you begin to just see it in your head as already done when you look at a schematic. You know where your blocks will be placed without giving it any direct attention. Practicing good technique on a daily basis pays off big time. It makes your job a lot easier down the road.

To finish the Noisy Neighbor Example, the noise issues may be so dire that you need to make two separate chips. Across two or three chips is even better. The noisy people are now in the next county. The further the distance the better.

Wire Solutions

We have been discussing noise traveling through substrate. However, your noisy digital blocks will undoubtedly be talking directly to devices that are located in the quiet areas. Therefore, no matter how much work you do to quiet the substrate, you will be running noisy wires all over the chip. Or—and here's something to think about—you will be trying to run quiet wires all over the chip.

Let's look at some mask design techniques for wiring that can turn our schematic into a very noise-insensitive piece of layout.

Coaxial Shielding

Have you ever tried to hook a microphone up to an amplifier with the wrong kind of cable? You will hear this huge buzzy, noisy, nasty 50 or 60 Hz hum that comes out of the loudspeaker. Particularly if you try to use a loudspeaker cable for your microphone.

The reason for the hum is that the microphone cable should have been what we call a **shielded cable.** By that, I mean there is an outer wire that shields an inner wire. This type of cable is also called **coaxial cable,** or **coax** for short.

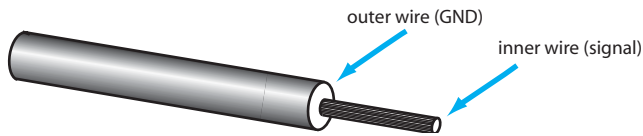


Figure 6–8. *Coaxial cable (coax) comes with built-in shielding all around the signal wire.*

Coax looks remarkably like a TV cable wire, or a networking wire. Our signal runs along the inner wire. The outer, shielded wire is connected to ground. If any noise happens outside, it will be picked up by the grounded signal, not the inner wire where our microphone signal is. That saves our inner signal wire from receiving unwanted noise from the outside.

You can do the same thing in layout as well. There are various ways to surround a signal wire with 360 degrees of shielding.

Let's have an area on Metal One that we connect to ground. Then we will run our signal on Metal Two. Effectively, you now have a grounded shield on one side. Any noise that comes from below is picked up by the ground, and not by the signal wire. (See Figure 6–9.)

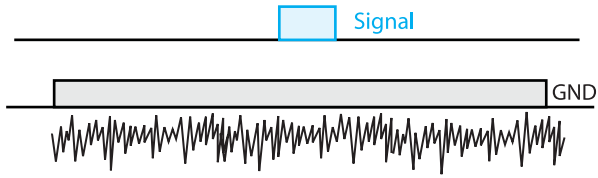


Figure 6–9. *Shielding the signal using Metal One.*

But, who is to say? Maybe there are other signals running over the top of our sensitive signal. So, some people extend this concept to include a grounded layer over the top, in Metal Three. (See Figure 6–10.)

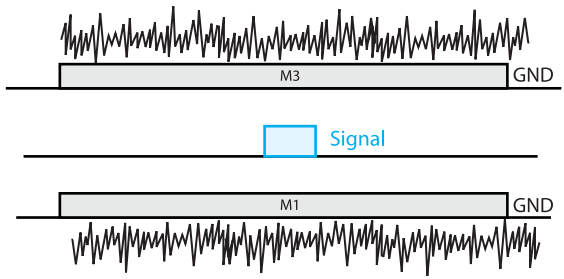


Figure 6–10. *Sandwiching the signal top and bottom is even better.*

That stops any noise trying to get in from either the top or the bottom. However, noise can still get in through the sides. So, we can extend our concept even further. Let's build some shielding along the sides, as in Figure 6–11.

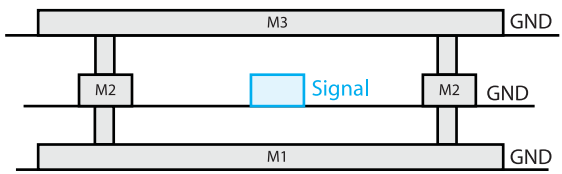


Figure 6–11. *Surrounded by shielding.*

Now we have our signal inside a box of metal that is completely connected to ground. This takes up a lot of chip real estate, as you can see.

If you have a whole bunch of wires running all over the place, shielding each wire in a box like this can be rather cumbersome. So, one final technique to improve our shielding is to run multiple signals in the same grounded box, as in Figure 6–12.

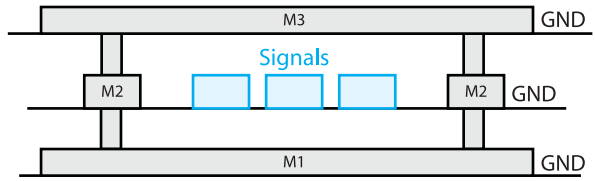


Figure 6–12. *Great place for shielding lots of quiet signals, or lots of noisy signals, from the outside world.*

That stops the noise from getting in. Or, if we have really noisy data running through these signal wires, then it stops the noise from escaping. You can use a shielding box for protection either way.[2]

Nevertheless, to build a shielding box structure, you have to know which circuits need the shielding before you begin your layout. You cannot begin your layout thinking you could add some of these fat boxes at some future point. You might not have the room. So again, you have to ask your circuit designer, *"What does it do?"*

Your circuit designer might say, "It's a low noise amplifier."

Ok, the phrase *low noise* should tell you to worry some. You would ask, "Are there any special shielding requirements? Any special isolation techniques you want me to do? Any shielding of signals going into it?"

Or, if your circuit designer says it's a mixed signal chip, you likewise ask about shielding. You ask, "Well, what kind of signal shielding do you want me to do on the wires running from one side of the chip to the other side of the chip?"

Knowing the circuit function leads you to these types of questions. Always ask about function. And ask early.

Differential Signals

Another design technique that directly affects your layout is using **differential signals.** We touched on differential signals in the Matching chapter, but at this point, we will discuss how to use differential signals as they relate to noise issues.

A differential circuit is a design technique that examines the difference between two specially run signals emanating from the same source. Two wires are laid out next to each other throughout their entire trail. Each wire carries the same information, but in an inverted state from the information in the other wire.

[2] Likewise, it's surprising how many people don't realize a blanket can also be used to keep cold things cold. They somehow think blankets only work in one direction.

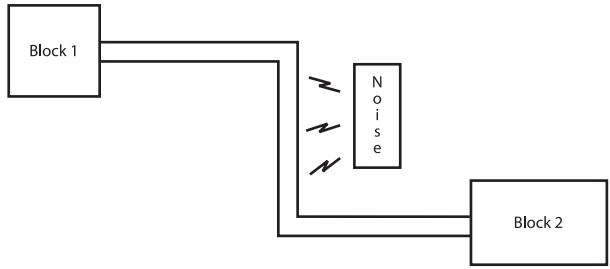


Figure 6–13. *Paired lines will be hit by the same noise spikes.*

You could have a pair of these wires running across the chip from one block to another block. They would run parallel and in close proximity to each other.

These wires might pass fairly close by a third, noisy block. Every once in a while, this noise source generates a spike of noise. A noise spike could originate from someone turning on a switch, or any of a hundred causes. So, just as we are trying to listen very, very carefully, Walter drops a shoe.

Because the two wires are very close to each other, it is very likely that the noise spike will occur at the same time and at the same amplitude on both wires. The spikes have been coupled onto both wires at the same time. In Figure 6–14, we see several such noise spikes.

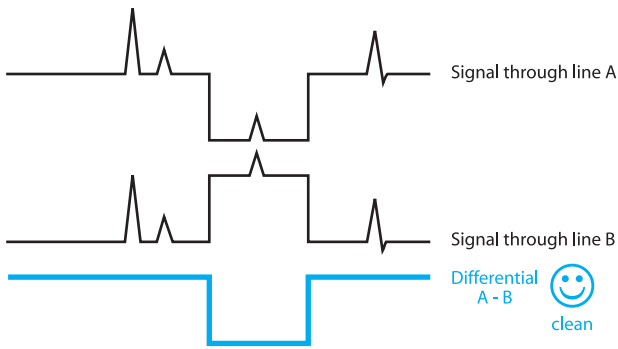


Figure 6–14. *The sound of the dropping shoe subtracts out, leaving only quiet, clean information. (You can see he dropped four shoes, just to annoy Felix.)*

If we look at the two wires independently, we would see the spikes in each wire. However, when we do our calculations, looking at the two wires differentially, we do not see any spikes in the result. We have eliminated the noise by subtracting one signal from the other. Spike minus spike equals zero. No spike.

Since the signals are inverted, subtraction yields very clear results. We see definite high states and definite low states. We have clean data with no spikes.[3]

The decision to use differential signals relates back to the question, *what does the circuit do?* As we mentioned before, if you were told that your circuit is a differential amplifier, you would hear the word *differential.* Then you think, "Ok, that means I need to run pairs of wires and I need to run them the same way." This clue is fairly obvious. With experience you will learn to pick up even more subtle clues that influence your layout decisions.

Differential logic, that is, a differential signal scheme, is very noise immune. Many people rely on a differential system in circuits where noise is a very important issue.

Decoupled Power Rails

There are times when you cannot avoid noise. It is just inherently there. So, sometimes people put big, chunky decoupling capacitors across their power rails. These are quite a large size.

The higher the frequency of a signal, the easier it will pass through a capacitor. That's a basic function of a capacitor. So, if you have a circuit block with one of these big capacitors across its power rails, then any noise that gets zapped into the power rail will preferentially be sucked down to ground. Very little will travel past the capacitor into the circuit.

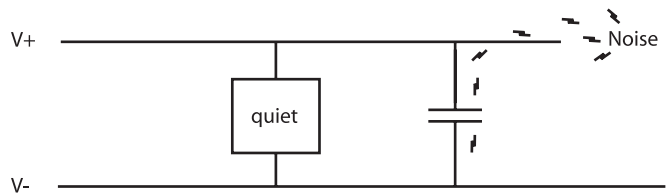


Figure 6–15. *Traveling down the capacitor to ground is much easier for a high frequency noise signal than trying to charge onward through the circuitry.*

Adding supply capacitors is a design issue, but it directly affects your layout. You need to know that circuit designers might pull this from their bag of tricks. You may be asked at the end of your mask design assignment to go around the layout looking for any white space, any holes, and fill them with capacitors across the rails. This technique is purely to reduce noise.

[3] If the signals were not inverted from each other, subtraction would eliminate the interesting data altogether.

If you would like to reduce the amount of new work proposed after your layout is nearly complete, you might want to ask up front about the possibility of eventually needing decoupling power rail capacitors. Your question could help save you a lot of work. Try to learn as much as you can as early in the design process as you can. Make suggestions. Ask questions.

Stacked Power Rails

Some people may even ask you to run power rails on top of each other. You might be able to alternate your power and ground rails like intertwining fingers, depending on the number of metals you have available in your process.

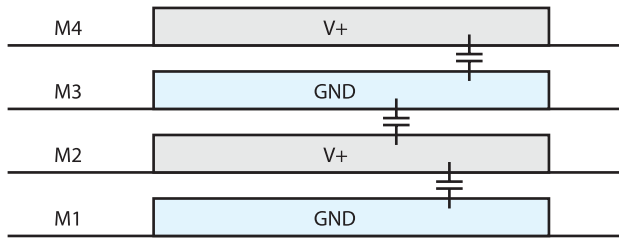


Figure 6–16. *Small capacitances form for free, if you stack your power rails.*

This technique builds-in little decoupling capacitors across the power rails for free. The capacitance factor is just the inherent capacitance that already exists between all the parallel metals.[4]

We can decouple our power rails in a much smaller space this way. We do not need to insert a large capacitor into the circuitry.

Each inherent capacitance may be small, but they do combine to give us a reasonable escape route for our high frequency noise. It might be enough for your power rails in some cases.

The advantage of putting in a large, real capacitor across the rails is that you get a lot of capacitance for the area used. But, you might have a very tightly packed chip, in which case you could use this kind of stacked power rail regime just to get some amount of free capacitance, though not as effective.

Again, you go to your circuit designer and ask the right questions. “Do you want me to run decoupling capacitors?” Or, “Do you want me to run the metals over the top of each other?”

[4] Using parasitics to your advantage. If we can do it once, surely there are more ways to use those throwaway parasitics to clever advantage. Give it some thought.

Mentioning what you can do in your layout to reduce noise might lead a new circuit designer to think to himself, "Yup, that's what I need to do. Thanks for bringing that up, that's a good point." It's nice to know your options. It's nice to be able to communicate.

Harmonic Interference

Designing the system well from a frequency point of view can help reduce noise as well. Of course, dealing with your circuitry frequencies is more a design issue, like decoupling the power rails, but these are all techniques a mask designer should be aware of.

Awareness of the circuitry issues builds a working vocabulary between you and your circuit designer. Besides, this one is kind of cool.

If you could tear apart a given signal, you would see the primary frequency plus many harmonics. A harmonic of a signal is another signal, usually weaker, located at predictable multiples of the original frequency.

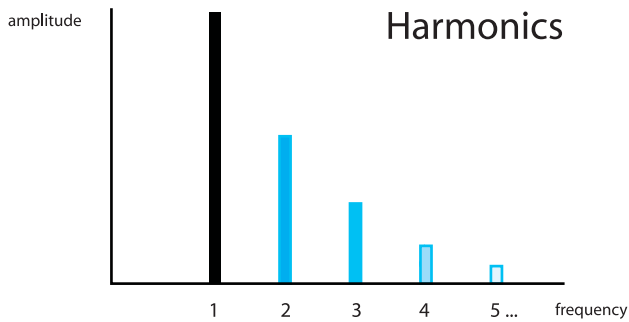


Figure 6–17. *Fundamental Frequency and The Harmonics. (Sounds like a good name for Brian Setzer's backup band.)*

If you hit middle A on the piano, you hear 440 cycles per second. (Count them, you'll see.) It sounds like a piano. Why doesn't it sound like a tone generator operating at 440 cycles per second? Or, why doesn't it sound like a trumpet playing the same middle A note?

A piano sounds like a piano because of the particulars of the signal. Besides the attack and decay and other factors, there are the harmonics. We truly do hear a faint 880 cycles per second at the same time, though the 880 Hz piano key was not struck.

We have all these quieter signals at regular intervals generated by the fundamental. They are all there. Our ears do pick them up. A piano does not treat the harmonics the same as a tone generator or a trumpet. Harmonics exist clearly enough to talk to our brains, to tell us which instrument we are hearing. Harmonics are real signals, even if we do not hear them by themselves.

Our square logic waveform is composed of sine waves just as the piano sound is composed of cyclical waves. Likewise, the logic waveform issues a lot of harmonic energy just like the piano note. Our supposedly pure digital signal is not so pure after all.

The harmonics just spread out like crazy, in predictable intervals from the fundamental frequency of our circuit signal.

The reason I mention harmonics is because one of the harmonics of your circuitry signal might just happen to occur at the same frequency as some other input signal you are trying to work with.

Let's say, for example, that this certain square wave just happens to be the clock to a bunch of digital logic, and the 19th harmonic falls directly on top of the frequency you are trying to pick up from your receiver system. Well, something certainly needs to be done to eliminate the noisy harmonic, or your receiver will pick up the clock tics.

Since the interference was caused by a harmonic of the clock frequency, then just by altering the frequency of the clock ever so slightly eliminates this potentially disastrous problem.

In this case, interference from the harmonic of an outside source was the same frequency as the signal we were trying to detect. Similarity of frequency, such as this, is as much a noise problem that we need to deal with as noisy neighbors.

It gets worse. You may have multiple frequencies all going on within a particular chip. Each one has a multitude of harmonics. Imagine sorting all that out.

And, it gets worse.

When you play two similar notes at the same time, as two strings plucked on a guitar, you will hear the two notes. However, if they are ever so slightly out of tune from each other, then you will gently hear this slight waa-waa-waa-waa modulation over the top of the sound of the two notes. They interact with each other, and your ear picks it up. You can hear the notes beating against each other.

That modulated beat frequency is the difference between the two original frequencies. (If the frequencies are close enough you actually can count them.)

Not only are you hearing the sum—that's the two notes sounding fairly loudly—but you are also hearing the difference—that's the waa-waa modulation.

So, for every two signals, you create a third signal, the waa-waa signal. Not only do we have all sorts of frequencies creating a multitude of harmonics, but also now we have created a new third signal from every two. And it will have harmonics.

If you have square waves all over the place on the same chip—which are each a Fourier sum of sine waves—they are all mixing with each other. Imagine how many extra signals this creates. You may end up with the third harmonic of the clock frequency of your digital circuit actually mixing with the 11th harmonic of the waa-waa from the combination of the input frequency and another frequency falling down smack on the frequency you are working with. That's going to cause you all sorts of sensitivity problems.

So, frequency planning is very important. And some of that translates directly into layout. Even if you are happy with one section of your chip making all the noise it wants, you should be aware of how harmonics and extra signals created out of nowhere can affect neighboring sections and other signal sensitivies all around the chip. Be aware that your circuit designer might be fighting these invisible tigers.

Again, ask questions of your circuit designer up front. If you are told there are worries about this particular frequency mixing in with some harmonics of something else, that immediately tells you to ask questions like, "Ok, which of my noise techniques do I need to use? Do I need to put down some guard bands around it? Do I need to isolate any of the shielded signals?" All your isolation training should jump out at you in your mind.

The problem with harmonics is one reason why you might have to change the way you lay things out. You might think you are dealing with such exclusive frequencies that interference is not an issue, but the harmonics could be conflicting. There might be harmonics happening that the circuit designer knows about that you don't.

Your job is not necessarily to adjust the frequencies of the circuits, but to understand that the harmonics could cause problems you will need to fix. You now have a better vocabulary awareness as you discuss the noise issues with your circuit designer.

Closure on Noise Issues

Good solid mask design techniques are not necessarily complex or highly sophisticated. As you can see from this chapter, good old common sense can work its way into your solutions. If explained well enough, you should feel as though you have known these solutions all your life. That's how clearly simple they really are.

But the field has only begun. We encourage you to continue using common sense, continue being creative. Have fun.

Here's What We've Learned

Here's what you saw in this chapter:

- Noise incompatibility of mixed signal chips
- Causes of noise
- Noisy Neighbors analogy leading to common sense solutions
- Using lower voltage swings
- Guard banding
- Time scheduling to alternate circuit operation
- Relocation of sensitive or culprit circuit blocks
- Coaxial shielding
- Differential signaling
- Decoupling the power rails
- Stacking your power rails
- Understanding harmonic frequency noise

page 141
BLANK

CHAPTER 7

Floorplanning

Chapter Preview

Here's what you're going to see in this chapter:

- Advantages of a well-prepared floorplan
- Floorplan driven by I/O relationships
- Floorplan driven by block layout
- Floorplan driven by critical nets
- Working with all these floorplan drivers at the same time
- Efficient block shapes
- Leaving enough room in your floorplan
- Communication samples regarding floorplanning
- Re-using existing layouts for size estimates
- Common mistakes people make
- Common frustrations and helpful solutions

Opening Thoughts on Floorplanning

Floorplanning can be the make-or-break of a chip. A good floorplan could make the chip very easy, very quick to lay out. A bad floorplan can make your life absolutely miserable.

The **floorplan** is the outline-only design that dictates how all the blocks are going to talk to each other and how the signals will flow between those blocks, as we saw in Chapter 1.

If you just hand the schematic to a team of people who go off, do their little bits of independent work, then come back again 6 weeks later with all their cells done, you are asking for trouble.

When you try to bolt all the individual cells together, you might find that they do not line up. The output of one cell might be on one side of the block, but the cell it's talking to has the input completely on the other side of the block. You can end up with lengthy signals going just everywhere.

You need to think about your chip floorplan even before you lay out any of the lower level cells. I strongly recommend that analog mask designers be involved in the pin-out and floorplanning activities from day one. If you understand what the chip is doing, if you are involved from day one, and if you are in the communication loop, you can fend off big nasties and save yourself grief further down the line.

Is all you're doing saving yourself some time by having a good floorplan?

It does more than just save time. It makes signal flow more efficient, particularly if you have a floorplan that is very convoluted, wires all over the place. Convoluted wiring can introduce coupling mechanisms, outputs will couple to inputs, parasitics come into play, you will introduce a whole bunch of unknowns that have not been simulated.

I have a saying, "I don't care if it works, as long as it looks good." It's a rather flippant saying, but if it looks good, 99% of the time it will work. The looking-good part of your work shows you have thought about signal flow, you have thought about floorplan, you have made the electron's life as it travels from one side to the other easier. It's just a philosophy.

Rule of Thumb: If it looks good, it will work.

Primary Drivers of Floorplanning

We have touched on floorplanning tools for digital chip place and route. You can use very similar tools for analog chips or for cell floorplanning. However, if you try to use a digital place and route tool for analog blocks, there is usually such a huge overhead with other files that the tool needs, it is almost impossible to use.

However, there are some tools now that are known as **connectivity-driven** layout tools. As you lay out a cell you place elements in the cell to tell the LVS

and the DRC tools where the inputs and outputs are. When you use this information in the next level up, you can get flylines. As you click and drag your blocks around the screen, the flylines move dynamically. That is a very useful tool for floorplanning. (See *Using Flylines,* Chapter 1.)

But, again, there is some overhead to be able to use those tools. So, whether this approach works for you depends on what tools you have available.

If you have a plain layout editor with none of this fancy capability, then it's all up to you. There is no replacement for good old, hand drawing the floorplan yourself. Get the schematics. Look at the flow from one block to another. Try to keep the flow and the floorplan smooth, logical and nice.

Whether you are able to utilize some automated tools or not, floorplanning is key. It is critical. It is probably the most critical element in both digital and analog (and of course, mixed signal) chips.

Next let's examine the three prominent concerns that will drive your floorplan: pin-out, block placement, and signal flow.

Pin-Driven Planning

The first part of your floorplan that you can be involved in as a mask designer is the **pin-out.** Some people call it the **pad-out.** This is the step that defines where the input and output pins are placed that will surround the chip in its package.

The quality of your pin-out directly affects how good your chip floorplan will be and how easy the chip will be to lay out. I strongly recommend that mask designers attend the discussion about the pin-out for any chip they will be laying out. They will learn more about the function and structure of the chip, and be able to participate in the decisions. The pin-out directly affects your work.

Mask designers should attend pin-out meetings.

This is true particularly for analog chips. Digital chips are auto-routed and auto-placed. Most of the floorplanning is done by software, so pin-out is not as much of a problem. In contrast, small, full custom analog chips are usually entirely hand-wired and hand-placed. Between these extremes is any number of variations requiring different levels of human intervention.

Effect of Pin Placement

Here's a simple example using a chip and its package. This will illustrate the effect that pin-out has on the floorplan.

The circuit designer sits down and decides what package the chip will go in. He says, "Ok, I want my input signals to be on one side and my output signals

on the other side. My powers will be on the top and my grounds will be on the bottom." (See Figure 7–1.)

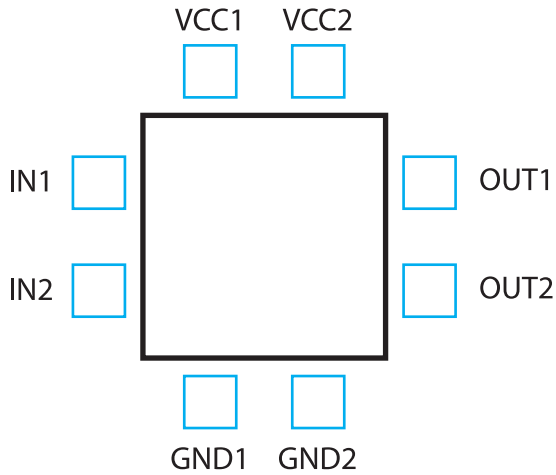


Figure 7–1. *Like items set side by side. One possible pin-out.*

Alternatively, you could use the same package and the same signals, but dictate the various signals to be in very different places. A power is now paired with a ground, and each input signal is paired with an output signal. This is also a totally valid pin-out. (See Figure 7–2.)

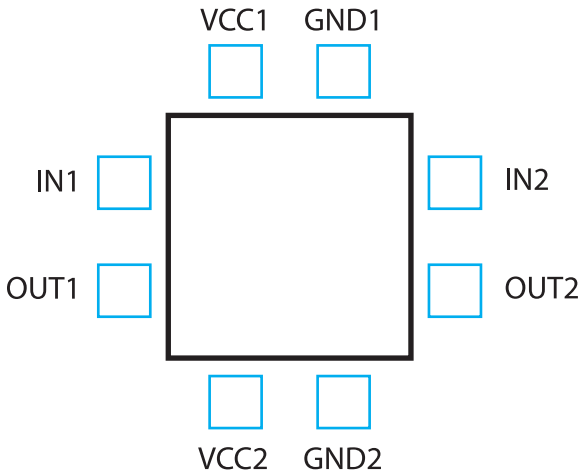


Figure 7–2. *Very different pin placement will produce a very different layout.*

Each one of these two drawings directly affects how you floorplan your chip. You can see that just this simple pin placement decision dictates how we need to run our power wires. The same is true for the other nets as well. A good choice of pin-out could reduce parasitics and help the mask designer produce a clean layout.

Of course, what is better for one example could be a worse pin-out for a second example. It all depends on the function and other objectives of your particular chip. Nonetheless, you can see that pin-out drives the complexity of the cross wiring of the blocks placed inside.

ESD Supply Strategies

One of the big killers in floorplanning is forgetting about the power supply strategy and the ESD protection for the chip before placing the rest of the signal pins. Usually ESD is the last thing on the circuit designer's mind. He just figures everyone will worry about that later.

But, as a mask designer, I would kick and scream and insist, "Hey, what about ESD? I want to know about it now. What are you planning to do? What's your strategy for this chip? I need to know this early, before I begin my layout."

Let's look at an example. Suppose we have a chip with four pins per side. In the pin-out discussion for this chip, we learn that pins 1, 5, 7, and 9 are protected to VCC 1 and GND 1. (See Figure 7–3.)

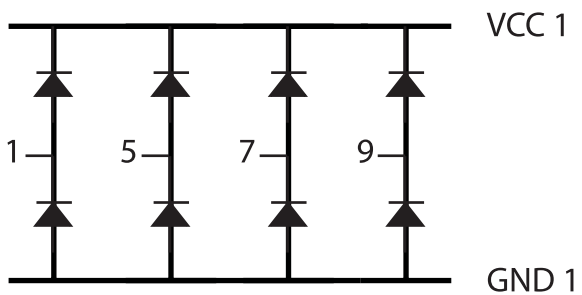


Figure 7–3. *Pins 1, 5, 7, and 9 protected to VCC1 and GND1.*

We learn the signals on pins 6, 8, 10, and 14 are protected to VCC2 and GND 2. (See Figure 7–4.)

And, finally, we learn that all the other pins have signals that are protected to VCC 3 and GND 3. (See Figure 7–5.)

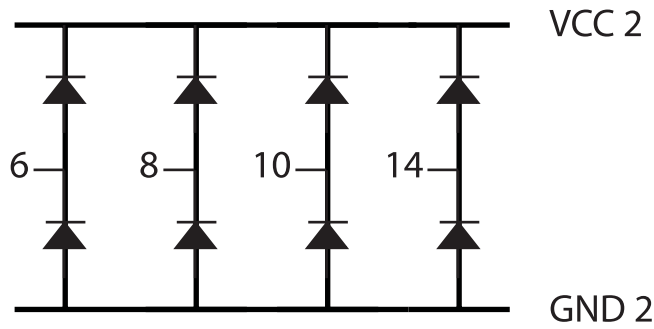


Figure 7–4. *Pins 6, 8, 10, and 14 protected to VCC2 and GND2.*

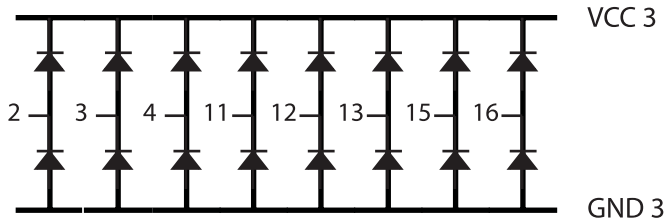


Figure 7–5. *The remaining pins protected to VCC3 and GND3.*

Figure 7–6, showing the final pin-out for the package, shows the coordinated colors for each of these protection schemes.

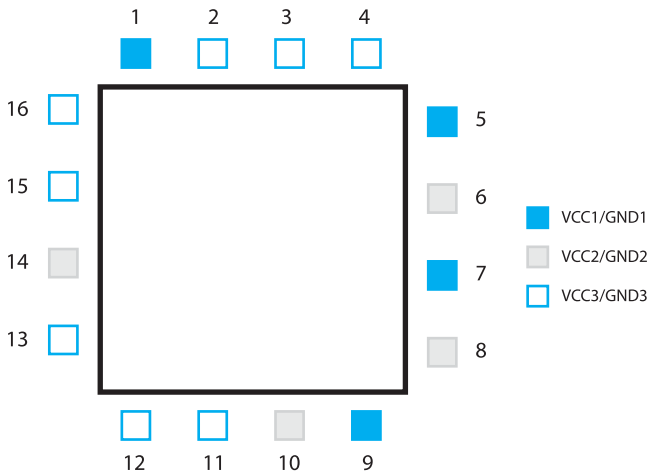


Figure 7–6. *Scattered pins on the same protect circuit.*

Notice that pins 1, 5, 7, and 9 are scattered all over the package. However, you need ESD diodes to protect them. That means you have to get VCC 1 and GND 1 to all four of those pins. But protecting pins 1, 5, 7, and 9 to a supply would run wiring past pins 6 and 8, which are protected on another supply. The supply wiring also would run past pins 2, 3, and 4 which are protected on a third supply. You create all these wires running past each other, all trying to get in the way of each other. This does not look good.

In addition, these wires must be fairly large, since they are all ESD protection. They must be able to take a reasonable chunk of current. This pin-out is going to place a lot of heavy supply wiring all around this chip, in, over and around all sorts of other heavy supply wiring. It will make your chip much bigger than it needs to be.

A nicer way to prepare for this chip's layout is to try to renegotiate the pin-out with the circuit designers. Let's ask our circuit designer to take the four signals that were on pins 1, 5, 7, and 9 and move them close to each other. With that change, we get all those signals that were protected to the same supply close to each other. We end up with this nice, small, short, self-contained area for the set of signals. We do the same for the other supply pin sets.

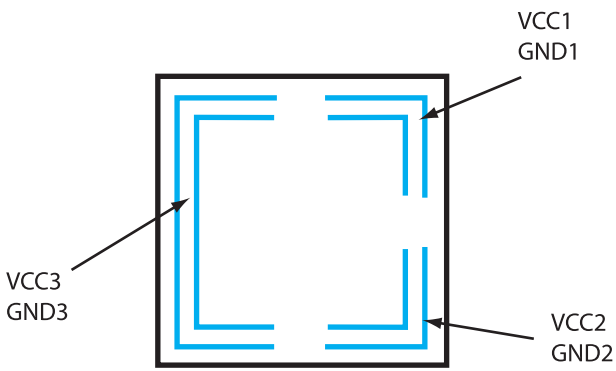


Figure 7–7. *Pin-out simplification showing like protects grouped together.*

We have all the pins that are protected to VCC 2 and GND 2 now located close to each other. It makes layout easier, makes the chip smaller, keeps parasitics down, and keeps cross talk down from one supply to another.

On the other hand, you might have no choice as to where those pins can be moved. Sometimes you have to keep certain signals on certain pins. It depends

on the package and the chip. For example, when the chip is bonded out to the package, there are certain pins that have lower parasitics than others. If you were building a really high frequency chip, for example, you might need the 4 gigahertz input to be on pin 13, let's say, because it's the lowest parasitic pin on the package.

Sometimes, some of the pin placement might be forced. In which case, you go back to the circuit designer and ask him to change how he protects his pins in order to get a better arrangement. (Don't give up. There could always be another way to solve the problem.)

You can see from the example that the pin-out decisions directly affect not only the layout of the chip, but the size and quality of the end product as well.

Floorplanning is key. It is critical. As we said before, it is probably the most critical element in both digital and analog, as well as mixed signal chips. And it begins with the very first stage: Pin-out. Be present as much as possible in the pin-out decisions.

How much of the floorplanning should a mask designer expect to do?

If you're real lucky, and you have a real experienced circuit designer, they'll almost give you the floorplan and the pin-out, everything all worked out. You might have to sit with them for an hour or two and go through it and ask about a whole bunch of stuff.

Or, you might do 100% of the floorplanning yourself. You may just be told, "Well, it's going in this size package. Do what you want. It's up to you."

How much will a mask designer know?

If you're involved from day one, you can get enough information. If you understand a bit about what the circuit is doing, and if you get involved with the floorplanning early, and you sit and work with the circuit designer, then that helps you know what you can do.

What about the mask designer who says, "They're supposed to do all this. They're the circuit designers."

It's everybody's job. Teams put these chips together, and unless everyone is communicating and fully involved throughout the entire process, the team will not function as well as it could.

We haven't even looked at the insides of the chip yet. We have only looked at the pin-out. Our next section discusses the individual blocks inside the chip.

Block-Driven Planning

Once we get a pin-out that we think will work well, and make our ESD wiring easy, then we can worry about the insides of the chip: Placing the blocks.

Planning your block placement is another issue to be addressed very early, just like the pin-out. Block placement will help you understand how you are going to perform top-level chip assembly and the kinds of problems you will encounter.

As always, try to keep the inter-block wires as short as possible. Try to avoid wires running around all over the chip.

Try to arrange your wiring by finding some symmetry, if you can. Besides helping a chip function better, creating symmetrical layout also reduces the amount of work you have to do. You lay out half the chip, flip it over, and everything is in place for the other half.

Figure 7–8 is an example of a poor preliminary block placement. We see that the upper left block needs to talk with the lower right block, directly across the bias block in the middle. We see a lot of wiring over the top of other devices, and even over the top of other wiring. This floorplan will be a mess to lay out.

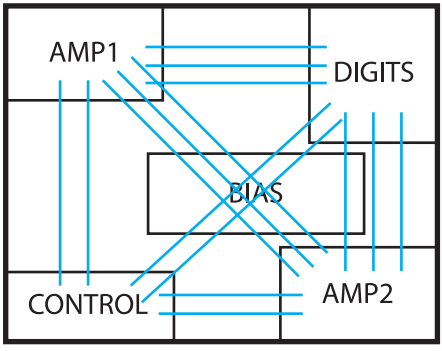


Figure 7–8. *Poor layout of blocks requires cross wiring all over the chip.*

In Figure 7–9 we see a much better floorplan. The block of control circuitry is now placed between the two amplifiers, so the wires connecting them are nice and short. The digital wires are off in one corner, and there's nothing wiring over the top of the biasing anymore.

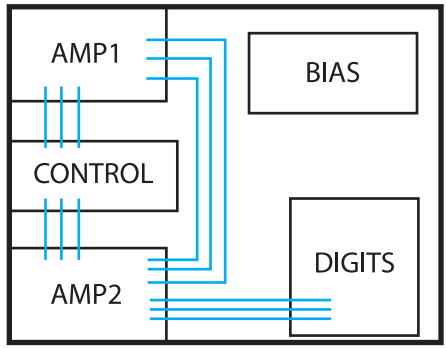


Figure 7–9. *Better layout avoids cross wiring, keeps wires shorter.*

Once you design a floorplan that you like, you have a rough idea as to where the signals need to enter and exit each one of these blocks. Now you can go off and start laying out the control block, for instance, knowing signals need to run north to Amp1, and signals need to run south to Amp2.

If you'd have gone blindly off and started laying out the control block before finding your best floorplan, how would you have known where the signals would go in or out?

However, let's look at our preferred floorplan again. We have rearranged our blocks to use the smallest amount of wiring, with the least cross-over. Yes, that is all well and good. We thought it was a great block plan, but now look at the pins that we thought were so wonderfully grouped. The pin groups do not align with our new block placement. (See Figure 7–10.)

You might realize, after playing with the blocks for a while, that turning things around, such as flipping the floorplan left for right, might work well with your existing pin-out. So you try that. (See Figure 7–11.)

In this case, it works for us rather easily. In other cases, you might have to go back to your pin-out for some additional changes, or just keep moving your blocks.[1]

Then the question is, well, where do you start? Which one comes first? Do you start your floorplanning with the pin-out, or do you start with the block placement? And the answer to that is . . . it depends.

[1] Chris recently told our nephew, Jason, to play Tetris as preparation for a career in mask design. Chris spent his youth doing jigsaw puzzles. You can see how these would help, can't you? I, of course, spent my youth doing my calculus homework. Didn't everyone?

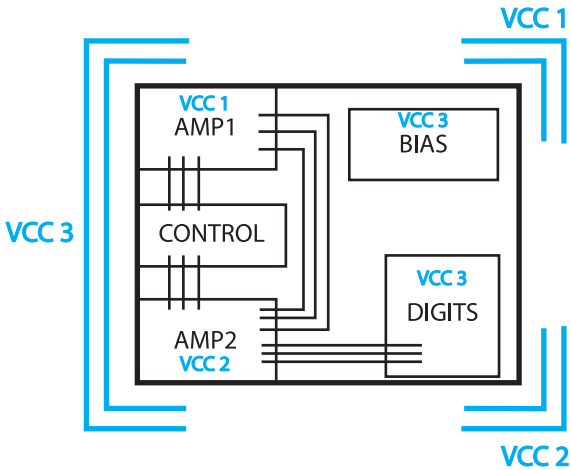


Figure 7–10. *We watch pin-out at the same time as we plan the inside blocks. Both levels need to be floorplanned together. Power supplies no longer align.*

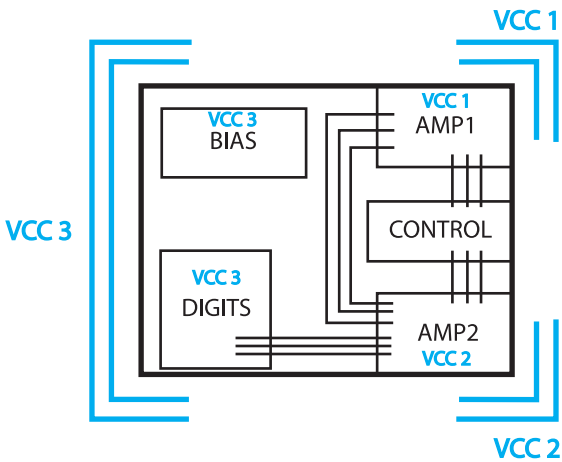


Figure 7–11. *Rearranged blocks now align with our premiere pin-out plan.*

It depends on which is more important. If you are more worried about how the internal blocks talk to each other, then the insides will drive the pin-out. If you are more worried with how the pins interact and connect with each other, then the pins will drive how the blocks are placed inside.

There may be times when you come up with a first stab at your pin-out, look at what's happening on the inside, change the insides, then go back and change

the pin-out. So, they can drive each other. You have to keep both in mind, sometimes you are not sure of either one until you are finished.

Developing a good pin-out and developing good block placement is an iterative process. You have to help your circuit designer buy into the idea. You might go around several times between the pins and the blocks until the two of you find the best solution.

Signal-Driven Planning

Our third floorplanning concern, certainly with high frequency or radio frequency circuits, is how your signals flow to each block.

On some chips, you won't care about signal flow. You will just be told to squeeze all the blocks in. As long as they fit, that's fine. You are told, "Just make it as small as possible." Well, there goes worrying about pins and wiring.

However, sometimes the signal flow has been meticulously planned and detailed by the circuit designer for a very important reason. You will be told, "I need to have a floorplan that looks like this. Don't move anything." In these cases, moving a block could make the whole chip worthless. I'll show you why.

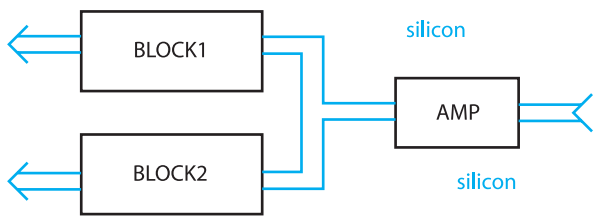


Figure 7–12. *Sometimes you are given fixed wiring layout.*

In Figure 7–12, the symmetry is the most important element of the circuitry. You have an amplifier that feeds two blocks, and there is a need to have a horizontal line of symmetry through the whole thing. You are told the blocks cannot be moved.

In this particular case, you notice there is some wasted silicon above and below the amplifier. You might complain, "But, I am trying to make my chip as small as I can, and if I can move the amplifier north or south then I can make the chip much smaller."

To which you might hear, "But then you would lose the symmetry. The circuit relies on the signal from the amplifier arriving at the two output blocks at

exactly the same time. If we don't get the signals exactly synchronized the circuit will not work. The symmetry is a requirement." So, again, here is another example where your mask design is influenced by the function of the chip, which you learn when you ask your three fundamental questions.

At times like this, the signal flow, not your block sizes or the pin-out, determines your floorplan. It might not be the best design for area efficiency, but it is the best you can do for circuit functionality. In which case, you just live with it and put filler in the cracks.

The question we have examined so far in this chapter is, "What is most important in driving the floorplan?" We saw three primary factors.

In our first example, it was the pin-out. We rearranged our pins to group like signals together. That affected our floorplan.

In our second example, it was how the blocks interconnected. We rearranged our circuit blocks to reduce the wiring and cross-over parasitics. That compelled our floorplan.

In our third example, it was how certain signals flowed. We looked at a high frequency signal with a fixed wiring arrangement based on symmetry. That drove a portion of our floorplan.

Any of these three factors can combine, of course. You could see all three of these as very important factors in some high performance, high precision radio frequency chips. Although other factors can contribute as well, these three primary drivers will typically determine your best floorplan.

Reshaping Blocks

Your floorplan and your signal flow can even dictate how you lay out a particular circuit block. Here's an example.

Your floorplan is done. Now you begin to look at the individual blocks. You sit down to take account of all you know about your first block. You know it talks to the outside world, so it needs bond pad connections. You also learned it is a differential block when you asked your circuit designer what it does. So, you begin with those two pieces of information: You build a differential signal to the outside world as an integral part of the bond pad area.

Your floorplan tells you to place your input and output pads to the south of the circuit. These I/O signals trail up into the top of the chip where the transistors live. (See Figure 7–13.)

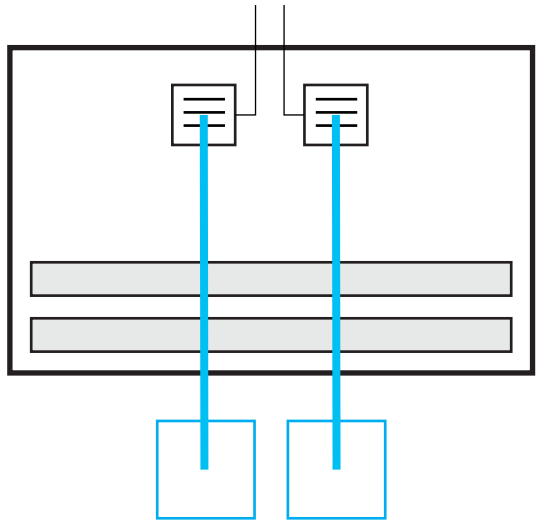


Figure 7–13. *Would you settle for this choice? Pads are wired to two transistors high in the block.*

Now, depending on what this cell does, this transistor arrangement might be just fine. However, it might be a very high frequency input cell, for instance. In that case, we can improve the design.

One change you might make is to place a couple of the very important input transistors between the pads, so your input traces are as short as possible. Since this extra effort is not always necessary, whether you chose this configuration or not depends on what your circuit designer tells you about the chip. If in doubt, bring it up. Talk with your circuit designer. Bring up some suggestions or ask more questions.

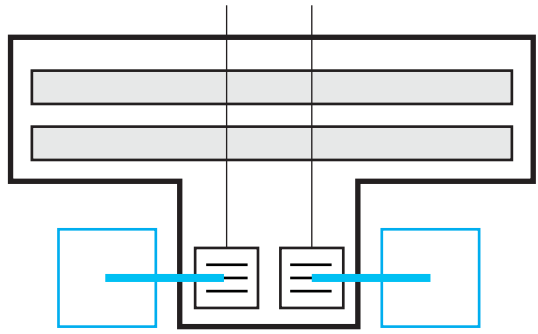


Figure 7–14. *Why not pull the I/O pads apart, and place the input transistors nearby? This keeps wires shorter.*

The more effort we put into our floorplan up front, the more it drives how the cell will look. In this last example, we've gone from a simple block to a block with some pads, and now a weird-shaped block with some pads. Finally, we have a good block plan with the pads in place for the actual cell.

Our block shape changed according to our knowledge about the chip, our floorplanning of other blocks, and the pin-out.

Sizing Estimates

Make your life easier by building a complete floorplan. Use all the information at your disposal. Don't rush. Don't sacrifice good practice.

One big mistake people make when they put a floorplan together is that they announce the estimated chip size before they consider enough information. Their plan turns out to be incomplete. This is usually the result of trying to proudly display a compact design rather than properly care for the structure of the chip.

Leaving Enough Room

Let's suppose Maskman Pete is given a chip with seven blocks of various dimensions. Pete puts the blocks together just as he packs frozen dinners into his freezer at home. He soon creates the smallest chip using all his blocks, and announces, "Ok, the chip is going to be x by y microns." He presents Figure 7–15.

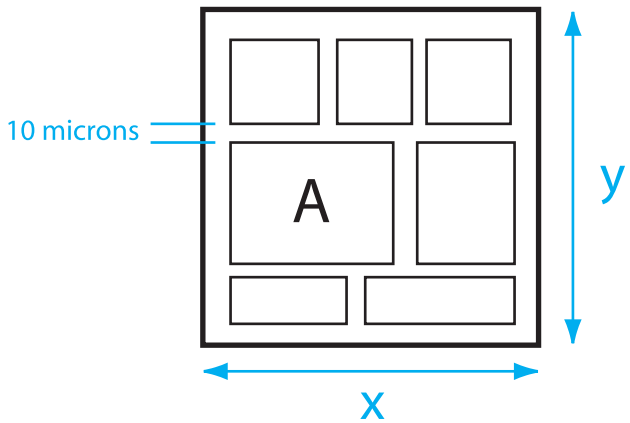


Figure 7–15. *We are so proud of cramming in our blocks tightly. However, we only left 10 microns to run all our wires. This will lead people to believe you can produce a smaller chip than you really can. You tease, you.*

One thing to remember when you are working on your chip floorplan is the wiring. It's fun to get just the blocks fitting nicely, but where are your signals going to go? Where is your power going to go? How can the clock tic get around? Do any wires need big, fat 360° protection metalization? All of these necessary concepts take room.

Once you get your rough frozen dinner placement, as Pete did, immediately sit down to begin a power supply strategy. How are you going to get power to these blocks? And that, naturally, depends on how many blocks there are and how many separate power supplies exist.

Then ask yourself if there is enough room for all the wiring that the chip needs. Base this on actual information you know about your chip. You may know that block A, for instance, has 200 signals coming out of it, and that you cannot wire those signals over the top of any other blocks. If you have only left yourself 10 microns to get 200 signals through, you're never going to do it.

Space the blocks accordingly. Allow for power rails and signals. Allow for special matching or noise concerns.

You will have spoken with your circuit designer before you begin, so you will know which area needs differential signals (requires more room). You will know which area needs special symmetry (requires more room). Or, perhaps you will learn that a certain area requires additional isolation techniques (requires more room). You might have had to re-floorplan your chip if you had not accounted for all this information in your initial dimensioning.

So, Pete returns to draw another floorplan. This time he leaves himself a whole bunch of room to get those signals out.

Now Pete's chip size is *a* by *b* microns, which happens to be larger than before. Good job. Accuracy is more important.

If you developed your floorplan based on bringing the blocks together real close, then your chip size grows as you do your final layout. Your wiring constraints and other concerns will force you to expand your chip size.

However, your business planning executives may have done their costing calculations based on your first chip size. Now, suddenly the chip is taking half again the area you initially claimed. It will cost an extra 50% to build. And all eyes look at you, as if you caused the chip to grow somehow.

Take wiring into account before you finish. And power as well. And clock signals. And shielding. And guard rings. All that kind of stuff that you discuss with your circuit designer over the schematic in the beginning.

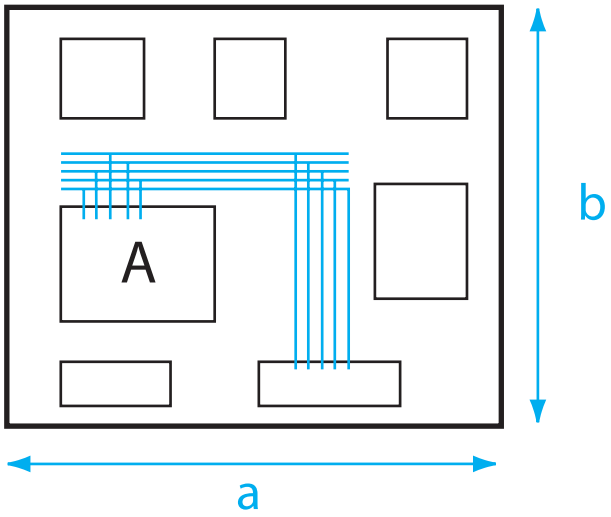


Figure 7–16. *That's more like it. Breathing room for our wiring. This is more realistic.*

You learn up front exactly what you'll be doing inside the chip. Then at some point you really commit to a chip size and a floorplan.

As each block in your floorplan starts to firm up, some will end up smaller than you estimated, and some much bigger than you estimated. You might have to go in and resize or even reshape some of these blocks. Naturally, changes occur. We can't control everything.

So, now you have done your homework. You have a well-thought-out, broad statement of intent. You feel that your floorplan is finalized.

However, no matter how good you think your final floorplan is, you are still not ready to begin your layout. Take that informed floorplan to the team and ask for input. You will say something like, "Ok, this is where I feel certain areas of the circuit will be." Then ask such questions as, "How do you like the way the blocks talk to each other?" Or, "Does it fit nicely with a good bond-out?"

Just get some buy-in from your design team. "From what I know at the moment, this is what the chip will look like. Does anyone have any comments?" Or ask, "Is there anything I need to worry about, based on this floorplan?"

Let all eyes and brains help you. Better now than later. And you might even find yourself invited to join the group for pizza, since you are obviously such a comfortable, easy-going part of the team.

Estimating with Existing Circuitry

One good technique to validate your floorplan is to use similar existing circuitry to give you an idea of whether your ideas will fit. That is, if your floorplan is going to work in the dimensions you have selected.

If you are lucky enough to be in a company that has already laid out amplifiers or mixers or biasing blocks, then you can ask your circuit designer if any of this has been done before. Ask if the schematic is a good approximation of any of the circuits that they have already built.

They just might respond, "Well, yeah, if you copy the mixer from that chip and the digital control block from this chip and add 20%, then fill in some of the blanks, it's pretty close." You can get a pretty good approximation of block sizes and chip sizes.

You might even want to add into your floorplan the examples you were told were fair approximations. You copy one amplifier from one circuit and two output blocks from another circuit. And, you know you need certain signals, so draw them all in your CAD tool. You end up with something that is a rough, preliminary sketch with all the actual wiring placed.

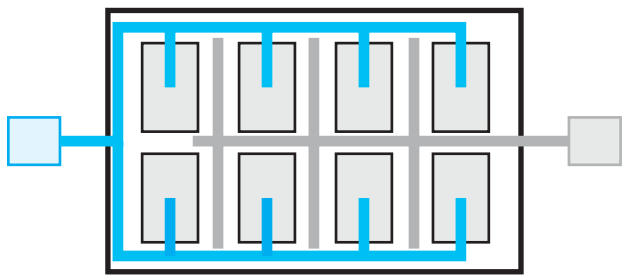


Figure 7–17. *If you know some of your nets, why not place them in your preliminary floorplan? More accurate sizing, better modeling, and it gets some of your work done in advance. Hey, can't beat that. Everybody's happy.*

Then at the very least, leave a place for power and your other considerations, as we mentioned in the last section. If you know where your pads are, then go right ahead and actually put in your power. Put in your clock. Put in place everything you know.

You might have a big power wire off to the right, for example, going to each of the individual blocks. You don't have to wire it up, just put in some polygons as placeholders. Then, later, when you do start to wire it up, you will know you have left room for the power wiring or the clock tree or individual signal nets.

It will save time later. Your circuit designer can then use this signal floorplan to calculate wiring parasitics based upon your estimated wire sizes.

Good investigating. Good allowances. Good planning. Once again you feel you have a finalized floorplan. You take your floorplan to show the team.

When you present this preliminary floorplan and wiring map back to your circuit design team, they will see very clearly where your thought processes are going. With so much information, they may be able to suggest better ways of doing something you have done, or they could show you a different technique for the entire design. Consider this good news. Team review of your work is probably one of the best opportunities you will ever get to hear new ideas.

The more work you do up front, the more exposure the design team has to your ideas, the less surprises there will be, and the less reworking you will do.

Once you have a floorplan to work with, you have a map to help you start working on the real chunks of layout. Now you can begin your lower-level work.

If you are the kind of person who doesn't feel that you can communicate, or who doesn't feel it's your position to communicate, then you will struggle. Be prepared to be at the wrong end of some major finger pointing.

Teach yourself to go talk with your circuit design team on a consistent and frequent basis. Sticking your head above the sand a few times a day is less painful than having it chewed off.

Closure on Floorplanning

You have spent the time to create a good floorplan for yourself. Now, as you start to look at the individual sub-blocks you're going to lay out, your floorplan is determining exactly what you draw—where, how far apart, and with what wiring.

If you have a very good floorplan that contains your cell placement, your pad placement, your ESD power strategy, and your regular power strategy, that basically gives you a map to work toward. There is a big payoff from taking the time to do a detailed floorplan, paying attention to all these concerns.

If you just close your eyes and hope, just start throwing things together without a detailed floorplan, who knows what's going to come up? Chances are you will have spent a lot of effort by the time changes are suggested.

If you spend a lot of floorplanning time up front, then you have already answered a lot of the questions that you would otherwise ask all throughout the middle of the layout process. You have done your investigating before you go anywhere near any of the circuitry.

Instead of waiting to say, "Where does the power run?", you have already asked. And the power routing has already affected your layout from the beginning. Less retrofitting. Fewer surprises for you as a mask designer. Good technique. You actually have a floorplan that flows well, and has the power worked out.

Someone is less likely to come along and say, "Oh, by the way, you need to put a 100-micron wire in there now." You would already know.

Now, of course, things change. Specifications change. Sizes change. Sometimes you will get changes an hour after you have finished your entire chip layout. It happens.

But aside from the problems we cannot control, keep getting early buy-in from your circuit designers. Let them see where you are going and what you are thinking, so you don't get to the design review and have your complete layout torn to pieces, ripped to shreds, and you have to start again. At least if that does happen, it is not because of your work. You will continue being the hero.

Floorplanning is key to good mask design.
Communication is key to good floorplanning.

Here's What We've Learned

Here's what you saw in this chapter:

- Floorplanning for faster tape out, smoother circuits
- Floorplan driven by pin-out
- Floorplan driven by the block diagramming
- Floorplan driven by mandated wiring nets
- Iterative floorplanning procedure
- Block reshaping options
- Wiring, clock, and other considerations
- Communication samples regarding floorplanning
- Sizing approximation techniques
- Common mistakes of floorplanning
- Common frustrations and helpful solutions

CHAPTER 8

General Techniques

Chapter Preview

Here's what you're going to see in this chapter:

- How to set yourself up for success
- How to do more work in less time and do it better
- How to lay out your chips to be more robust on the job
- How to save the day when someone needs extra stuff in the layout and there is no space but we only have 4 hours to tape out
- How to appear to have memorized two thousand rules
- How to pull last-second matching out of a hat
- How to turn three weeks' work into three days' work
- How to use your $2.99 Scooby Doo calculator to save your company millions
- How to leap over tall blocks in a single bound
- How to bend metal rules with your bare hands
- How to anticipate changes faster than a speeding billet
- How to sew a large, red S on your shirt (Collector Edition only)

General Techniques

This is it. You are ready to do your layout. At last, you have covered everything that you need to cover. I've given you a lot of information in this book. You should be ready to sit down and start some layout. But, it's a big, huge, daunting, scary task in front of you.

When looking for a place to begin, a lot of people get panicky and freeze, especially when they are first starting out as a mask designer. They are frightened of making a mistake.

Where do I start?
How do I possibly enact the thousands of design rules?
What obstacles could I have avoided if only I knew?
When are people giving me unnecessary work?
How can I minimize that inevitable last-minute change thrown at me?
What should I think about first?

Don't worry. We have some techniques to make your life a bit easier. These techniques will reduce the number of things you need to worry about. They are not wacky gimmicks, really, as you will see. They are ways of thinking, more than anything else.

Once you have been shown these techniques and incorporated them into your daily mask design routine, these techniques will begin to feel natural. Practice these techniques daily, and soon you will think you never did mask design without them. That is how down-to-earth practical they really are. Once shown, they seem simple.

Focus on these techniques and it should help you get started, help you get more work done in less time, and give you better results. (These techniques are not necessarily related, or in any order.)

By the way, we have gathered the technique titles to the back of the chapter for you, so you do not need to grab a pencil.

#1 Pick Five or Six Non-minimum Design Rules

Many new mask designers concentrate on every fine, minute detail of the process design rules. On an old CMOS process from 10 years ago, there were maybe 20 or 30 of these rules. You could easily grasp 20 or 30 rules.

However, in modern processes there are 1500 to 2000 design rules. There is no way anybody in their life could remember that quantity of rules.

Consequently, build a lookup table in your head of a small functional subset of design rules. Provided you know enough about processing in general, and how transistors are built, you can easily find a handful of generic rules that can see you through most of your work.

For instance, most processes have an N well. Usually the N well separation determines how far apart you can place transistors. So, why bother learning a bunch of other rules when you can just find the one rule that sets how far apart transis-

General Techniques

tors are? Typically for N well the design rule value is pretty big. Big enough to work for just about every device in the circuit. Find that rule, and that's the only one you will remember. Use this one separation number across the board.

Likewise, if you have five or six metals, with reasonably different minimum widths for each layer, just pick the largest minimum and only remember that one. Use it for every metal. After all, minimums are not mandatory sizes, they are only lower limits.

Minimums are not mandatory sizes, they are only lower limits.

Use this technique for the gaps between metals, as well—the metal spacings. If you have a Metal One spacing that is 0.8 microns, Metal Two that is 0.9 microns, and Metal Three that is 1 micron, use the 1-micron spacing for all of your metals. Why not?

Make just one rule to remember for yourself. Why bother remembering all the minimums for all these other metals and spacings unless you are totally crushed for space?

You can even choose your own design rule. For instance, let's say your minimum metal width is 0.6 for Metal One, 0.7 for Metal Two, and 0.8 for Metal Three. These are kind of weird numbers to have to remember. So, why not always wire your metals in 1-micron-wide metal?

Again, it's only one number that you remember. And it's an easy number.

This is how you can build a lookup table of about 5 or 6 rules that work as a catch-all, since each is bigger than your minimum processes. You can use those simplified rules for your routine work, in preference to the ones in the design manual.

Now, someone might say, "But we want our cells as small and tiny and as minute as we can." In which case, you must remember the individual, minute rules.

However, the people who say they want the cells as small and minute as they can are usually the digital standard cell people, the people who make cells that are designed for the place and route tools. In these cases, they are usually laid out using a grid approach. It's just the grid that you need to worry about. You can still get away using metal that is slightly bigger than normal, providing you maintain the grid nature of the cell.

There is another useful advantage to not using the minimum design rules in your routine work. (This is a good one.) If you find yourself really stuck for space at some point, having used larger widths is like having a Get Out of Jail Free card.

Let's say you have used a 1-micron width for metal spacing everywhere, and you suddenly need to get three more wires in your layout. Anyone who looks at the layout knows there appears to be no possibility of getting those three signals through. (See Figure 8–1.)

Figure 8–1. *What a surprise. Someone wants a change. It seems we now need to run three more signal wires where we currently have only two.*

But you are the Go-To Kid. You can add the extra signals because you set yourself up for emergencies such as this (just like stacking a deck of cards before performing a trick). Go find the design manual. Find the real minimums. Now is the time to use those.

Figure 8–2. *Everything is still DRC clean and we can still tape out today. Now you are earning the big bucks. If you're really good you might wangle a pizza bonus once in a while as well.*

By making gaps between metals bigger than they need to be, and using a larger than minimum metal width, you have built-in some slack. You have added what I call some Fresh Air.

Fresh Air is your friend.

Your layout still looks like it is really tight, compact, and dense. So you can fool people into thinking that you are laying out as compact and dense as possible. However, you know you have lots of little, tiny holes all hidden throughout your layout.

So when someone comes along and says they need to add some extra stuff, you can magically pull those holes out of nowhere and make them into one big hole to put their stuff in.

Suddenly you are a Superhero. It is sandbagging, a bit, but it's faster and more efficient in the end. They like you for that, too. What you are doing is finding an efficient way to turn out quality product quickly, while building in some emergency contingencies at the same time.

It works for more than just metals. Someone will come along and ask you to get more components in your layout. And if you have made all your compo-

nents at minimum spacing, then you have nowhere to go. So if you place them slightly further apart than the minimum spacing, your layout still looks good, it's still tightly packed, but you can pull a rabbit out of the hat by going to real minimums when you need to. It's a useful technique.

The advantages of having just five or six non-minimum design rules to work with are:

- It's a place to start. It gets you going.
- It makes your work a lot faster because you don't have to keep remembering too many design rules.
- It gives you better than minimum performance for your chip.
- It builds in some slack that you can reclaim later if you need it.

#2 Get Thee to the Lowest Parasitic Metal

If you have a really dense chip, you normally follow the alternating orientations for your metals, as we mentioned in an earlier chapter. For instance, Metal One will only run horizontally, Metal Two only vertically, and so on.

But high frequency signals need to be on the lowest parasitic metal. Full stop. So, this takes priority over the orientation rules. You may have to build in wiring channels just for these high frequency signals.

New mask designers have a tendency to wire just in Metal One and Metal Two. However, they should try to get any high frequency signals up to the lowest parasitic level as soon as possible, and keep them there as long as they can.

Be sure to get out your four-function Scooby Doo calculator to determine the metal with the lowest parasitic. It might not be the metal furthest from substrate, as you might guess. There are other factors, as we discussed earlier in this book.

If you ask your three golden questions and understand your circuit up front, it saves you time. You will do the right thing the first time out. If not, you might find yourself having done a lot of work, then saying, "Oh no, these high frequency signals need to be brought up to the lowest parasitic metal."

#3 Plenty of Wide Wiring and Vias

In general, a big mistake that new mask designers make is that they run all their wiring with minimums.

Using minimum width wiring is fine if you are trying to put together a very small, compact cell that will be used in a big digital chip. You almost have to use minimums in that case. But for analog-style layouts, you are typically more worried about having a low resistance path for a signal or a low resistance path for a power bus.

If you wire with minimum metals and vias all over the place, you will very often be asked to add more vias and widen the metal. You may as well make using wide wires with more vias your standard practice to begin with. You can always bring it down to minimum when you have to.

Also, you need to make sure the number of vias in a signal path is sufficient. In Figure 8–3, the trace on the left shows what most layout people do routinely—use skinny wire and a single via.

Figure 8–3. *Thicker wire allows additional vias between metals.*

However, if you have the room and the space, and the signal warrants it, try to put as many vias on that signal as you possibly can. You can see in Figure 8–3 the wider wire has room for four vias. Each via has a resistance, so if you have four vias on a particular wire, then you have a quarter of that resistance on each metal transition.

Aside from lowering resistance, consider that four vias are more reliable than one. So, even if you have to use minimum size wires, add extra vias where you have the room.

Adding extra vias is a good practice to use routinely. However, wiring a complete chip with four vias on each signal at every turn is going to waste space, so you have to be real choosy. Just chose the wires that need it. So you have to go to your circuit designer to find out which kinds of wires need the extra vias.

Be sure to ask, "Are there any low resistance wires or low parasitic wires you want me to put in this circuit?" If the answer comes back that a particular wire needs to have low resistance because it handles a lot of current, not only make sure you have enough vias to handle the current, but that the wire itself is big enough, as well.

This information should come to you as part of the answer to the second golden question, "How much current is there?" Or, you might know your circuit is running a high frequency signal, in which case it needs to be low loss (low resistance). Fat wires with lots of vias are good candidates for these two circuit requirements.

#4 Don't Believe Your Circuit Designer

One big mistake mask designers make is that they believe their circuit designer. Don't believe your circuit designer, particularly if your circuit designer is relatively new. Let me say that again,

Don't Believe Your Circuit Designer.

The circuit designer can make your life hell. Really. If you see anything that looks kind of weird or odd, or just doesn't make sense, go back and ask your circuit designer, "Why did you do that?" "Did you mean to do that?" "What's the reason for your doing that?"

Nine times out of ten you might be able to suggest a solution. Or, just by asking a question, you could make the circuit layout much easier for yourself and a lot more robust.

Don't Believe Your Circuit Designer.

For example, a circuit designer can forget about sizing the width of a resistor to handle the right amount of current. Look at the amount of current that is likely to be flowing through that resistor. Make sure the width is enough to handle the current.

For example, you have a 2-micron-wide resistor that is taking 5 milliamps. The resistor will take it for a short time, but it won't be very reliable. So, as you are laying out your cell, do a sanity check on what your circuit designer has told you to do. Go back and question the 2-micron-wide resistor.

Don't Believe Your Circuit Designer.

As another example, perhaps you have been told that a circuit needs to have a lot of good matching. You have been told that certain resistors need to match well. However, when you look at the width of the resistors in question, they are 1 and 2 microns wide. That's quite small as resistors go. Something in your brain should stand up and say, "Hey, that's not a good matching technique." Even if you can use the root resistor approach, there is some inherent nasty inconsistency in a resistor that is very tiny.

Go back and ask your designer, "Why 1 micron wide? Do you really need your resistors to be that size?" A lot of the time the circuit designer has just thrown something together. They forget about issues that could impact your layout if left to the last minute.

Oh, by the way,

Don't Believe Your Circuit Designer.

#5 Use a Consistent Orientation

One good way to make your life a lot easier is to pick an orientation for each of your devices, and never deviate from that orientation—ever.

For instance, have all your resistors, which are usually long and skinny, always running north-south. Always have the gate stripes of a CMOS transistor run east-west. Always have an emitter cut for a Bipolar transistor run east-west.

Orient your devices the same in every cell you layout. And later, when you place your cells in the chip, make sure that all of your cells are placed in the same orientation.

There is usually what is called a **processing differential.** Your devices will etch differently in different directions, due to the orientation of the crystal and the chemical process involved. For example, a CMOS transistor etched vertically can see a very different etch than a horizontally etched transistor. You will not get the same results. So, you cannot guarantee that the gate lengths will be the same. This was also discussed earlier.

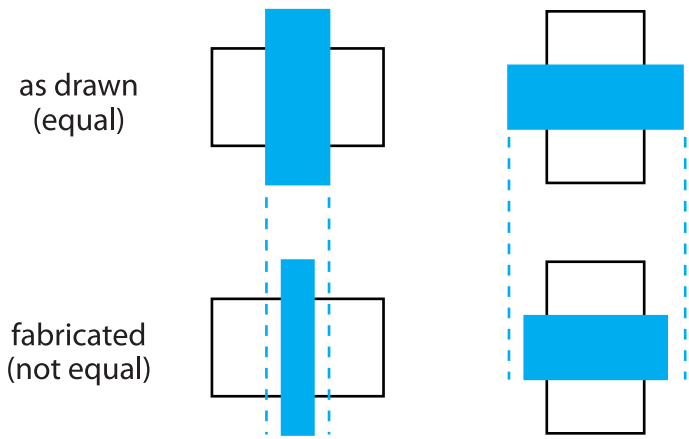


Figure 8–4. *Processing horizontally produces different results than processing vertically.*

If you do not care about matching, then either direction would get the basic job done. However, if you get into the habit of making sure everything runs in the same direction, then when someone comes along and says, "Oh, we forgot to tell you about the matching in this part of the circuit," it's nice to have the job already done. It's a good everyday technique.

Also, if you need to rotate a device 90 degrees in your layout, ask your circuit designer, "Is this component ok to rotate through 90 degrees?" Don't just

assume it would be all right. There may be some of those matching issues the circuit designer forgot to mention.

#6 Don't Go Overboard

If your circuit designer comes along and gives you 20 or 30 different criteria for matching in a particular cell, then you are in one of two situations. Either it is an extremely critical must-have cell, or the circuit designer is just going overboard.

Don't believe it. Question the heck out of that circuit designer. You can spend weeks and weeks and weeks laying out a schematic that has super-mondo-destructo matching when it may not need it. The zealousness of your circuit designer may not cost him a thing, but can cost you a load of work.

A great question to ask your circuit designer is, "What matching do you need?" Get the designer to qualify his decisions for such extreme matching techniques. If he says, "Well, I can really do with a 5% match," then what is the point of spending enormous amounts of your time doing this super-mondo-destructo technique that gets him a 0.5% match? There is no teacher to give you extra credit points. Why bother if it isn't needed?[1]

On the other hand, if he says he needs better than 0.5% match, then maybe we really do need all that matching effort. But, questioning the circuit designer makes him think twice about why he is asking for so much work. Your time is limited. You could be working on another project. So the advice is: Don't Let Your Circuit Designer Go Overboard.

Perhaps it isn't your circuit designer at all. Perhaps you are told to do *some* matching and you take it upon yourself to use super industrial strength matching technique. Perhaps you only need regular matching, which you could get by simply placing the two components next door to each other in the same orientation, but you waste your own time by doing three weeks' worth of work. Without your self-imposed work you could have gotten away with doing the layout in one day. It's all about being reasonable.

It's all about being reasonable.

#7 Keep Off the Blocks

This technique really only applies to analog circuits. Avoid running wiring signals over any of your blocks, or any of your other components. This technique will keep you from running into potential problems from a really noisy signal coupling its noise into some other part of the circuit.

[1] The purpose in math class is to get the exact answer. The purpose of engineering work is to make a dollar for the company. Two would be better.

A general rule of thumb is: Don't run sensitive or noisy signals over anything. That means over blocks or over anything else. You do not know what coupling mechanisms or parasitics you are introducing.

Don't run sensitive or noisy signals over anything.

In particular, don't run signals over capacitors. This would create a big parallel plate capacitor between the existing circuit capacitor and your wire running over the top. And who knows what would happen.

#8 Care for Your Sensitive and Noisy Signals Early

This is a particular killer. Most people forget about this.

Find out as soon as possible if you have any very sensitive or noisy signals. Again, we are talking about knowing what your circuit designer knows by communicating with him. Or her.

You would ask, "Have we got any really noisy signals in here?" "Do we have any really sensitive signals in here?" "What kind of shielding techniques do you want me to use?"

Communicate early. It is much easier to work shielding and isolation into your layout up front than to try to retrofit them in later. These can require an awful lot of room. Ask early. Plan early.

#9 If It Looks Nice, It Will Work

One encompassing piece of advice to any layout engineer is: If it looks nice, it will work. I can't stress that too much. If your layout looks nice it means that you have thought about it, planned it, spent a lot of time on it. If it looks really good, really well thought-out, you can be a chip saver.

If it looks nice, it will work.

I would rather people spend two weeks staring at the screen trying what-if's for a cell rather than just blindly jumping into the layout too soon.

I would rather they spend the time to come up with a really nice looking piece of layout. It makes such a difference. It really does.

It's kind of like the Zen approach to mask design: Make your electron's life easy. If you make it easy for an electron to seek its destiny, it will do it with a big, happy smile on its face.

General Techniques

If you make it bend, bounce around corners, make it go up and down and left and right, it will get angry and bored.

Be the electron. It pays off. I have put together circuits that looked highly symmetrical, like works of art, and they have worked really well. They typically exceed people's expectations.

Circuit designers have told me that one of the reasons certain chips work so well is the good job that was done on the layout. Simplicity, symmetry, good flow, no cross-overs... that all looks pretty to the human brain.

#10 Learn Your Process

If you are new to mask design, you will not understand your process particularly well—what the layers are, what they do, how they are built.[2]

After you become functional, meaning you can lay circuits out in your process, spend the time learning exactly what all those layers do. It doesn't have to be in great depth, but just the process steps in general.

At some point, your company will come along with their new, next generation process. You can bet your bottom dollar that it will be based on the one you have just been working with. If you can build a good understanding of the process you are using now, then the transition into the new process will be a lot easier.

Also, it increases your layout ability. When the chips are down (excuse the pun) and you have to squeeze in more circuitry, if you know your process well you can find your own innovative solutions. You can say, "Ok, how can I squeeze these components together? What tricks can I do that the design kit or the tool normally will not let me do, but because I know a lot about the process, and what I can and cannot get away with. . . ." It is a life-saver.

Now, that does contradict the rule about only remembering five or six rules, but that was to get you up and running. And, yes, in fact, for 90% of your layout time, you will be applying those five or six easy rules.

But be proactive. Go out of your way to learn what you can about the process. It could save your rear end, so to speak. Or very often, someone else's.

[2] Although there is a great book out there in which you could read about exactly these topics. We wrote it. *IC Layout Basics: A Practical Guide.*

If you happen to have the luxury of being in the same office space as the wafer fab, go find some of the process technicians or process engineers and ask them to give you an afternoon of their time. Go through the process with them. Try to absorb every ounce of that process.

If the wafer fab is not in your building, ask your manager to find someone to talk with you, or find more documentation about your processes. Or find who wrote the design rules and talk to them. Find someone. Go out of your way to find someone to teach you as much as possible about the process. The more you know, the more creative you can be.

#11 Don't Let Noise Find the Substrate

This bit of advice is somewhat esoteric. It's about substrate and noise.

Because the substrate of a chip is everywhere, once you start to get noise on the substrate it is delivered to every single part of your chip. So, try to use as many techniques as you can to keep that noise from getting into the substrate.

This can be achieved in any number of ways. You could place lots of substrate contacts around a particularly noisy device. You could shield wires. You could ask your circuit people to come up with a lower noise transistor, or a lower noise library.

You have to work with your circuit designers on each particular solution, but you will find at a layout review that people will be looking at how you have wired noisy signals and whether they interact with quiet pieces of circuitry through substrate.

#12 Spread Your Spinach around Your Dinner Plate

Remember when you were a kid and you didn't want to eat your spinach, so you just sort of spread it all over your plate so it would kind of disappear, and your mom would think you ate enough of it to make her happy? Then she came by and spooned it all back into a single pile, and she could tell you hadn't really eaten any? Well, we can learn from that experience.

Suppose your first attempt at a floorplan for a given block has a huge hole smack in the middle of it. Now, if you were to go off and lay out your block to that floorplan, someone will come along, point at that hole and shout, "What's that hole doing there? That's wasting space! We can put something in there." It's as annoyingly obvious as a pile of green spinach on white china. It just sticks out visually.

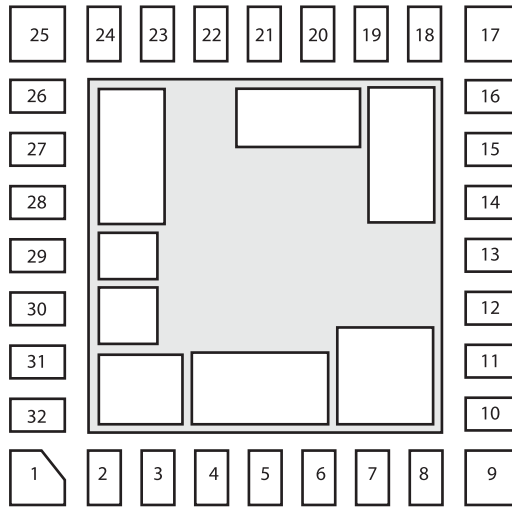


Figure 8–5. *Tempting pinup poster, popular with circuit designers.*[3]

If your first attempt at a floorplan liberates a big hole, rearrange that fresh air. Re-floorplan your cell so that the big gap is redistributed among all the blocks. You know, think spinach. Spread it around so no one will notice it. It sort of disappears.

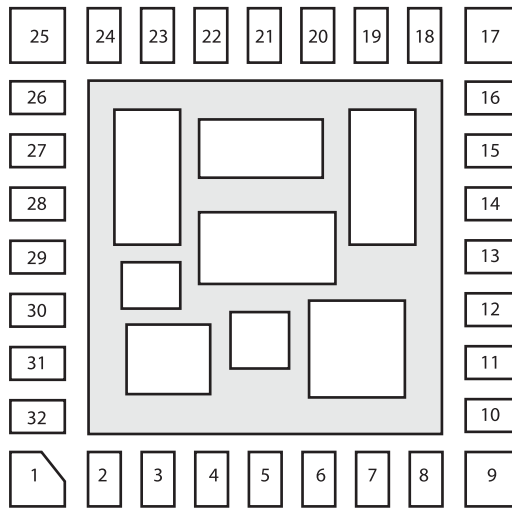


Figure 8–6. *Instead submit this. Keep the fresh air for yourself. Don't let them find it. You will need it for those last second additions.*

[3] Enlarge this and post on a wall near circuit designers. Put a pencil nearby. Take bets with coworkers on how long before the circuit designers fill up the hole. It will drive them nuts until they do. I get dibs on two days.

This redistribution technique is like building in slack with the wiring that we talked about earlier. Just a little nudging around gives us some sandbag slack later if we need it.

Suddenly you hear a cry, "Hay-ulp! .... Hay-ulp! .... I need to put a so and such in the chip! I need this today and just look at how nicely packed that chip is! There is no way to finish this chip in time! Oh, Dudley, what will I do?"

"Don't worry, Nell. *(Reversed gender roles available in Collector Edition.)* I know where I can get the space you need." And you do. You return to your original floorplan, with the large center space available. You somehow were able to save the day at the last second because you are magical. Only you knew the free space was there all the time.

Or, here's another option. Instead of leaving a hole in the middle, and instead of spreading it around, you can rearrange your blocks so that the hole is now moved to the outside. Now, you will have a weird shaped cell, but at least that real estate is now located where it possibly could be used instead of being wasted.

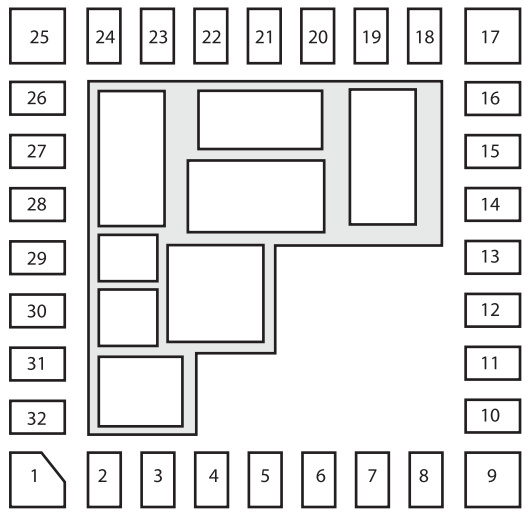


Figure 8–7. *It's difficult to combine this with other blocks, but it might just open some free space you need on the chip. It's an option.*

If I can, I go for the first option. Redistribution. Rectangular blocks are much easier to place with other blocks. Irregular blocks can be a pain to worry over. But either technique could work for you, depending on your chip.

The criterion for choosing the spinach redistribution floorplan or the free space in the corner floorplan is, *How confident is your circuit designer?*

If you have a circuit designer who waffles, "Well, I *think* this is most likely what I *probably* want. I'm pretty sure." Then that should ring some bells. Leave space hidden among the blocks because he is going to change the design. Bet on it. Spread out the airspace. You will need it.

But, he might come along confidently, saying, "It's finished. It's there. I don't care—I'm never going to change it again. It's dead."[4] In that case, you might want to go for the second, choppy design. It sounds like you will not be making any changes, so you may as well open up the free space and pack your cell in tight.

#13 Copy and Rename Cells before Making Changes

Most of the tools you will use in your layout are **hierarchical,** referring to the various functional levels of a layout scheme.

A big mistake people make is that they forget about the hierarchy of their design. They forget that the lower level cell they have just enlarged is also used 1500 other places. The change they wanted in one location has just changed every other use of that same cell.

For instance, in Figure 8–8, you see three separate blocks, each of which uses Cell A. In each block we actually are using pointers to Cell A, but they will be fully drawn for us every time we take a look at the block.

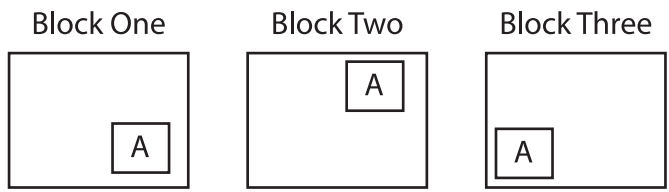


Figure 8–8. *Cell A is referenced more than once.*

Your designer comes along and says, "In Block One I want you to change Cell A to be a slightly bigger version." If you forget about the hierarchy, and you change Cell A in Block One, then it also changes Cell A every place where it is used, even in Block Two and Block Three.

[4] See section, "Don't Believe Your Circuit Designer."

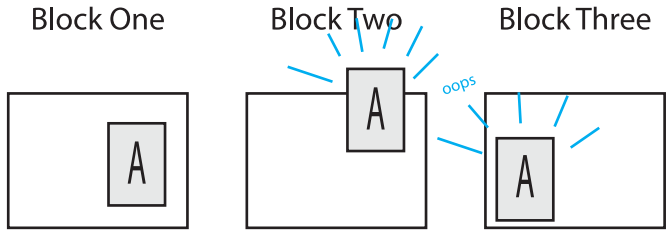


Figure 8–9. *But we only changed Cell A in Block One. How did this happen?*

Of course, you do not want that to happen because you were only asked to change Cell A in Block One. You get whistles and alarms going off all over Blocks Two and Three. Danger! Danger!

Here's a permanent solution for this problem. Get in the habit of making copies of cells *before* you alter them. Rename the copy something different, such as Cell A1. Place the copy, Cell A1, where you want the changes to occur, then make your changes to the renamed cell.

Only change renamed cells.

Now you have Cell A1 to work with—adjusting, stretching, shrinking, or whatever changes you have been asked to do.

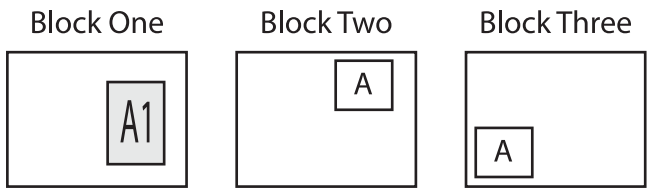


Figure 8–10. *Since we changed the name of our cell to Cell A1, our other cells that are still named Cell A are not affected.*

The renamed copy will be the only item changed. Blocks Two and Three remain linked back to the master Cell A, which has not changed. You'd be surprised how many people get caught on this oversight because they just forget. If you can make it a regular habit, you will soon be uncomfortable making changes to a cell unless it has been renamed.

#14 Remember Your Hierarchy Level

The other big mistake regarding hierarchy that can happen is that a mask designer might think he is working in one level of the hierarchy, but actually,

he is working in a different level. Strange things then seem to happen, as if ghosts took over the layout. Polygons appear out of nowhere shorting out all sorts of circuitry.

This problem occurs frequently when you push down into a lower level cell. The screen continues to display the higher level information for reference, but the work being performed is not happening in that higher level. With the higher level data still on the screen, it is easy for mask designers to forget where in the hierarchy they are working.

Then the inevitable happens: The mask designer spots a small item to correct, and without going back up to the higher level, he makes some changes that he absent-mindedly thinks he is doing at a higher level. However, he is adding data to a lower level cell.

And you remember from the last point how dangerous that can be. You add stuff into Cell A but you think you are adding stuff to Block One. It changes all the other Cell A's.

The mask designer thinks he is finished. He saves everything. But suddenly, he sees strange polygons floating around in free space. Now, who did that, he wonders.

Seeing the levels surrounding your cell can be confusing if you forget they really don't exist on the level of your workspace. Remember where you are. Come up to the surface to make your higher level changes.

#15 Build-in Easy Metal Revisions

This technique is more prevalent in the analog world. The basic idea is to build some contingency plans for yourself. Typically with high frequency or high precision analog circuits, you do not really know if the chip will work properly or not. You have a good idea, but you don't really know.

The mask designer will be asked to put in extra components in order to give the circuit designer the flexibility of changing his circuit once it has been made.

The circuit is not cast in stone. You have some extra components floating around that you can wire in. You can change component values or wire in a different circuit.

As the circuit is going through processing, the lab is typically building more than one wafer at a time. So, you may have 25 wafers of your circuit being built at the same time. Just before they get to the first metal processing step, they take maybe five or so of those wafers and hide them. They keep them

away on some shelf somewhere, then keep processing the remainder of the wafers.

Once the circuit is built and tested, some issues may arise that need to be addressed. Some values may need to change.

You pull out those stashed wafers you were hiding in the cupboard and re–lay out the metals on them. Only this time, you make them slightly different. You wire in extra components, or fix a mistake. You can change your metal layers on those unfinished wafers. You can wire in some of those little sloughy devices you put in.[5]

With this technique, you do not have to wait the whole wafer fab cycle in order to see your results because the metal processing will only take two or three weeks. If you were to fabricate entire chips from scratch with your new changes, you would likely wait 8 to 10 weeks while they build the whole wafer again.

But, if you are going to enjoy this advantage, you have to build-in your options in advance. There are several ways to prepare for metal revisions.

Be sure to leave yourself some slack, so you can run extra wiring if you need it. We discussed ways to do this earlier in this chapter.

Also, the circuit designer may give you several additional components that have no place in the circuitry. He says, "Well, I want to have a resistor from my transistor to ground, and my ideal value is 100 ohms. But I also want to have the options to run 80 ohms or 120 ohms."

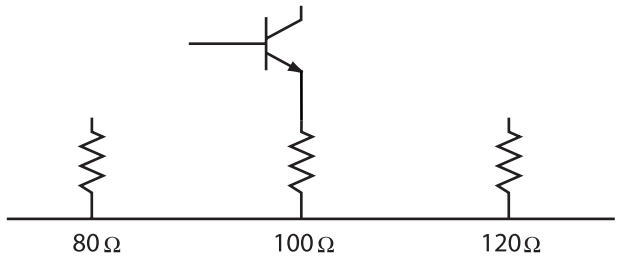


Figure 8–11. *Our main circuit is wired to the 100-ohm resistor, but the circuit designer has added extra resistors to the layout for possible metal revisions.*

[5] Stopping the processing just prior to metalization is similar to the quick fabrication idea behind using gate arrays. We need only take the time to work with the metals to obtain a new chip.

In this example you have been told you want extra components in the chip, two extra resistors. How do you best handle this in layout, anticipating a possible revision? You know these two extra resistors need to be wired in as a metal option. If you place your optional components such that you can make your wiring changes using only one level metal. Then there is only one mask you will have to revise.

Here are the three resistors and the connection to the transistor. You could just wire your main circuit from each optional resistor but stop just short of a connection. All metal wiring can be drawn in the same level.

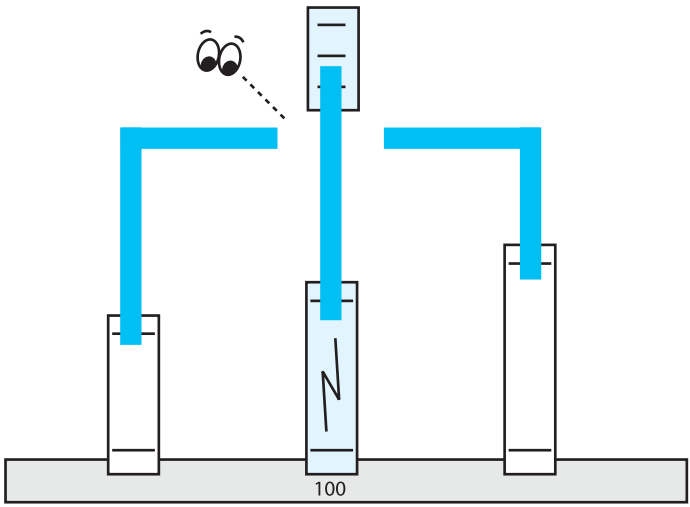


Figure 8–12. *We can see the optional components have been wired, but stop short of making contact. These should be easy to connect if we need a metal revision.*

If we need to add metal during a revision, there is a process called **focused ion beam.** This technique allows you to deposit new metal after the wafer has been processed. We can only use a focused ion beam to alter the exposed surface metal, however.

We can also remove metal during a revision. A laser can blast it away. Combining the two processes, you can cut existing traces and actually rewire your circuit without going through the wafer processing cycle—as long as your revisions are located on the top layer, of course.

Therefore, if you are going to build-in a contingency plan for possible metal spins, try to bring everything up to the top level metal. Make it obvious as to

where you want the joins. It makes your life easier if you have thought about these metal spins in advance.[6]

Leave an easy connection for the possible new circuits in case you want to use any of them.

Your new layout could look like Figure 8–13. We have laser-cut the original metal, and we have rewired to another resistor. Easy-peasy. Good layout planning keeps your life happy.

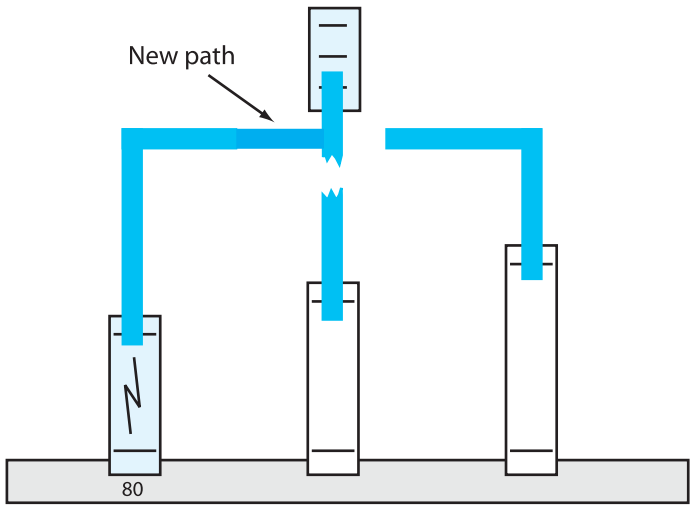


Figure 8–13. *It turns out we need the 80-ohm resistor. We easily cut the original connection and grow metal to create the new wiring path.*

One advantage of all this up-front planning is this: If it is easy to re-wire a circuit with focused ion beam techniques, then it will be easy for you to re-wire the metal masks. Once all the zapping and fibbing is complete and we finally know what is best for the circuit, the metal-only change is easy.

I've laid out circuits where there have been more spare components than real components used in the chip. They just didn't know what to expect. They used their best guess with the chip, but gave themselves hundreds of outs.

[6] Spin refers to the metal processing step.

In working your metal spin, it is very important that you do not move any of the lower layers. Don't move any of the diffusions or poly. Remember, the wafers are already processed and waiting on a shelf. Your new metal masks must line up with the old diffusion masks.

Once you have finished your metal revision layout, you can compare your new database with your old database and see quickly only what has changed. You will run a special check called **Exclusive Or (XOR)** to make sure that you have not moved any of the lower layers.

Let's examine an XOR truth table to see why this check works. Notice in the logic chart below that you only get an output (one) when the two inputs, A and B, are different. But, you get no output (zero) if inputs A and B are the same, whether they are both zeros or both ones.

XOR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Figure 8–14. *Inputs A and B produce Exclusive Or results, Z.*

We can use our original layout and our modified layout as inputs. If they are the same, we will see nothing. If there is anything changed, if the two are different in any way, we will see an output.

Let's use just a single polygon as an example. Input A comes from a polygon in your original database. Input B comes from the same polygon in your new database. These two polygons are laid over each other.

When we run the XOR check, any overlapping areas of polygons A and B that are identical have no output. However, any overlapping areas of polygons A and B that are different will produce an output. You will see an output wherever you have either added to or removed from your original polygon.

In Figure 8–15, your XOR output shows two small gray rectangles. In this case, the XOR reveals where you stretched upward (top gray skinny rectangle) and where you carved out a small block (lower gray rectangle). Your two changes have been spotted and reported. If either of these changes was unintentional, this is a good way to catch the error.

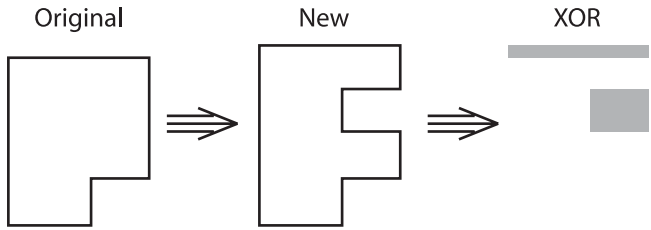


Figure 8–15. *The new polygon is taller and lost a chunk of material from the right-hand side. The XOR results show these changes.*

It is very unlikely you would actually replace an object with an identically shaped object, so running the XOR check detects every change made.

Since the XOR does not pick up the portions of your polygons that were left unchanged, you will see some really weird, spooky shapes coming out of this program. You will not necessarily recognize any of the XOR output shapes. The output polygons are just the changes, not whole devices. If you have made a small change along one edge of a device, you will see a long thin bar, which represents the amount of edge that was resized.

When you run this XOR database comparison you not only XOR the layers you know you have changed, you will XOR all of the layers. The only layers you should see weird shapes on are the layers you have changed. Right?

Rule of Thumb: XOR all of the layers—even the ones you don't think you changed.

What if you see something on a diffusion layer you know you did not change?

Well, you have apparently moved a transistor by mistake.

So, as you can see, it is very important that after doing metal revisions you run this XOR database comparison. Make it the last thing you do.

#16 Draw Big Power Buses

How do you know how wide to make your power rails? It may not be as simple as calculating the correct answer on your calculator. Really.

Let's assume that you are laying out a full custom chip. You have been told what all the currents are in each block in the circuit. You have diligently calculated the width for each power rail for every block. You have ensured that the power rails are on the lowest resistance metal. You have done everything you know to make sure that the power rails are adequate. But, at the layout review everybody says that your power rails are too narrow. What did you do wrong?

The answer is that you forgot the optical illusion factor. Even though a power rail is sized correctly, it can look weak and wimpy when it is laid among lots of circuitry. It appears too small. People get nervous.

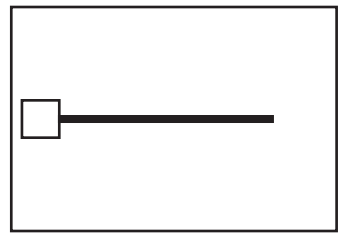


Figure 8–16. *A correctly calculated power rail can appear too small.*

There are two solutions to this problem. The first is to argue your case and prove that you have calculated the width of the rail correctly. This option can be a painful exercise in futility. The second option is to widen the rail. And, if you're really on the ball, you will have widened the rail anyway before the review.

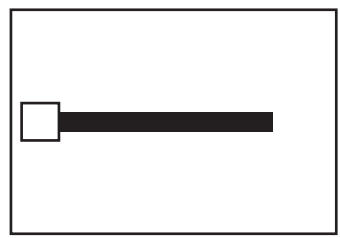


Figure 8–17. *You can bet someone will ask you to widen your power rail, until it seems right to them.*

The rail is now much wider than it needs to be, but psychologically looks just about right to people. Always make your power rails wider than they need to be. Remember, "If it looks nice, it will work." Make it look nice.

Another useful rule of thumb for power buses is the 10% rule. Once you have placed all your circuitry and you are ready to begin wiring, measure the height of the cell. Use 10% of that height as your power rail width. You still need to

verify the current handling capability of the wire, of course, but start with 10% as your minimum.

Rule of Thumb: Make power rails 10% of your cell height.

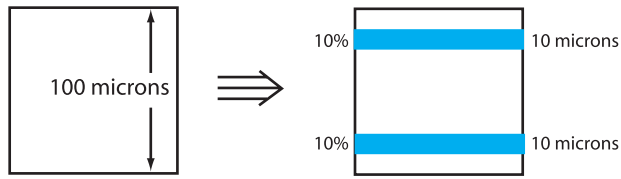


Figure 8–18. *Start with the 10% Rule for your power rails.*

Power rail widths are, in practicality, a false science. On one hand you can calculate the width you need, do the math, verify it to everyone, and file a report on Wednesday. On the other hand, you can almost guarantee that you will be asked to widen your rails, despite your best calculations. (Don't lose sleep over it.)

#17 Break Up Large Circuits

There will inevitably come a time when you are given a circuit block that is very large. Where do you start?

Usually there are sections of the circuit schematic that lend themselves to being laid out first. There might be small areas of your schematic that are easy to lay out. Concentrate on those areas first. Work with 5 to 10 components at any one time.

Rule of Thumb: Work with 5 to 10 components at a time.

Take a colored marker and highlight the areas on the schematic as you complete them. Eventually, all the small areas of the circuit with join up and you will be finished.

Closure on General Techniques

Rules of thumb are made to be broken. They aren't absolute ways to do things, but they are good daily practice habits. After awhile you won't realize you're using these techniques. They will become second nature. This is partly because they are common sense, and common sense is easy to remember.

So, there you go. Begin your layout. Save the day. Let people count on you for great designs that always lead to great performance.

But, above all these techniques of mask design, don't forget the Golden Rule: Communicate with your circuit designer from day one.

GOLDEN RULE OF MASK DESIGN
Communicate with your circuit designer.

Sit in the design reviews. Listen for cues as to what the circuit designers discuss as concerns. Go back after meetings for additional clarification of those concerns you heard that could apply to layout. Review your floorplan with your circuit designers regularly. Ask questions. Ask for help. Listen for new ideas. Suggest new ideas. Confirm you are on the right track. But above all, just keep communicating. Integrated circuit design is a team effort.

"Mommy, who was that mask man?"

Ancient Secrets of Mask Design

Pick Five or Six Non-minimum Design Rules
Get Thee to the Lowest Parasitic Metal
Plenty of Wide Wiring and Vias
Don't Believe Your Circuit Designer
Use a Consistent Orientation
Don't Go Overboard
Keep Off the Blocks
Care for Your Sensitive and Noisy Signals Early
If It Looks Nice, It Will Work
Learn Your Process
Don't Let Noise Find the Substrate
Spread Your Spinach around Your Dinner Plate
Copy and Rename Cells before Making Changes
Remember Your Hierarchy Level
Build-in Easy Metal Revisions
Draw Big Power Buses
Break Up Large Circuits
COMMUNICATE

page 186
BLANK

CHAPTER 9

Packaging

Chapter Preview

Here's what you're going to see in this chapter:

- Why you need to look at packaging before you start your layout
- Rules about putting chips in their little black boxes
- Different ways to connect the little wires
- Placement considerations for the input/output pads
- Three quick ways to know if your chip will fit
- How to smoosh the chip smaller by using empty pad space
- How to compute finished die size after the saw blade

Opening Thoughts on Packaging

Packaging considerations should be addressed even before you begin your chip layout. Packaging is really part of the floorplanning process, but it's worthy of having it's own separate chapter. As you will see, the choice of package for the chip drives your options for the floorplan. We will see some unique situations for chip size, block placement, and other issues which all begin with the selection of the final package.

The package you choose for your chip has a particularly large effect on placement of your I/O pads. You cannot just blindly put your input/output pads wherever you feel like it, because they interact with the package. There are some rules and guidelines you will follow in order for these signal connections to be bonded properly in the packaging choice.

There are hundreds of different kinds of packages. The assembly rules will be very different from one package to another, from one package vendor to

another, and from one assembly house to another. Nevertheless, there are some general rules that you follow in your floorplan to make sure that your chip will bond reasonably well.

Bonding Methods

When the chip is finished, packaged and sold, people see a small black box made of plastic or ceramic with rows of metal pins jutting out the sides. These metal pins connect to the circuit boards. However, what they see, the black box with its metal pins, is not the actual chip. It is a package, which houses your chip.

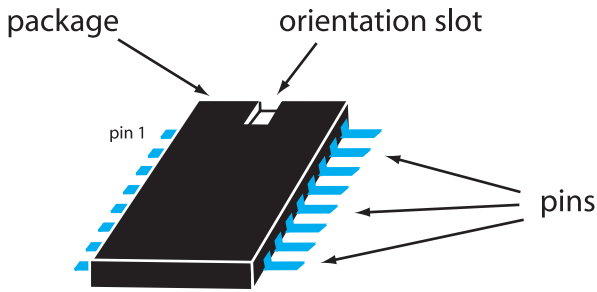


Figure 9–1. *The plastic or ceramic package is all you see when you install a chip.*

The sharp bits of metal are not part of the chip, either. They came with the wrap-around package. You would have to break the package apart to see the little silicon wafer chip with its microscopic circuits that look like city maps. It is not visible through the package. Sometimes, though, you can have packages with little windows, particularly erasable PROMs. You can actually see the chip inside.

The signals from your chip are accessible through bond pads we have placed on the surface of the chip. The **bond pads** are the locations where the packaging wires are bonded physically to the chip, hence the name.

Our chip has little bond pads all over it that will connect to the spidery metal legs of the package, called the **pins.**

To get the signals from the chip circuitry to the package pins, thin wires join the bond pads on the chip to the inside portion of the metal pins. All the wires are covered in plastic, which also covers the chip.

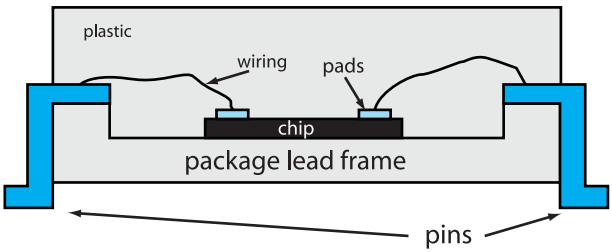


Figure 9–2. *The top plastic is molten when deposited. The wires are surrounded before it hardens.*

The wires are typically thin strands of either aluminium[1] or gold.[2] The thickness of the wires can vary depending on who is doing your packaging.

From a top view, with the top plastic removed, you can see all these little bond pads, placed around your chip.

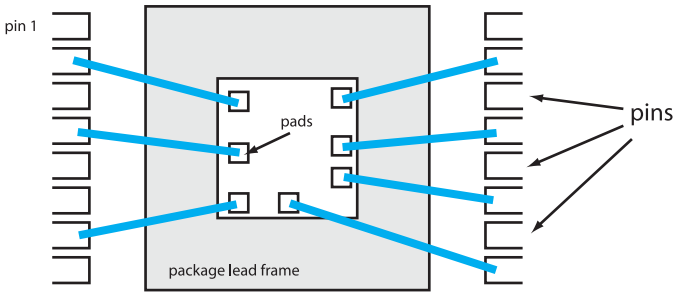


Figure 9–3. *The circuit is connected to the bond pad that is connected to the wire that is connected to the package pin that is connected to the circuit board that Jack built.*

There are several ways to attach bond pads to package pins. In the next few sections we will discuss three of these methods: wedge bonding, ball bonding, and flip chip technology.

Ultrasonic Wedge Bonding

Ultrasonic wedge bonding uses brute force and ultrasound. Your wire sits on top of the pad, carefully held in place. A small wedge-shaped piece of metal

[1] American: aluminum.

[2] Californian: river rocks.

that is vibrating at ultrasonic speed is pressed down onto the wire and bond pad.

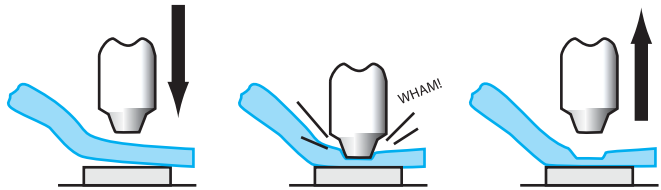


Figure 9–4. *WHAM! BAM! Great bonding under pressure.*

The intense pressure and vibration of the wedge generates heat that fuses the wire to the bond pad. The bond wire cools very quickly, firmly attaching the wire in place. The other end of the bond wire is then run out to the package pin where the wedge bonds the wire again. This time the wire is snipped off and the whole process starts again for the next pad.

Ultrasonic Ball Bonding

Another option is **ultrasonic ball bonding.** You use a machine similar to the one used in wedge bonding. However, with this method, the wire is given a high voltage zap that melts the tip even before it hits the pad. Once the tip is melted, it is placed in position on the bond pad using a tiny collar that is vibrating at ultrasonic speeds. The vibration of the collar provides additional energy to heat the wire when it is pressed onto the bond pad.

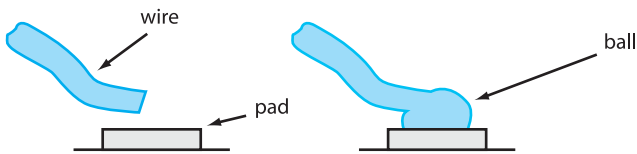


Figure 9–5. *The tip melts under ultrasonic power.*

When the bond wire cools you end up with this little ball of metal that hardens on the bond pad. Because you are not hammering away with a big wedge, there aren't as many mechanical problems. Plus, you don't have that little tail of wire that sticks out as you get with wedge bonding.

If you are using any of the wire bonding technologies, then you cannot have a pad in the middle of the chip. The wires would be too long. If you were to route a long bond wire from the center of the chip out to the metal pins, the wire

could droop down and short to something in the circuit. So, most bonding manufacturers insist that wire bond pads be located at the outer edges only.

Flip Chip Technology

There is also a third packaging option, which is what they call **flip chip.** As part of the final processing as the chip is completed, you place a large chunk of solder on each chip bond pad.

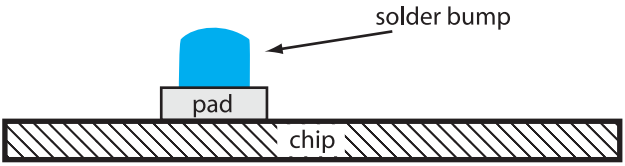


Figure 9–6. *Prepare to flip the chip by depositing a bump of solder.*

You then take your chip and flip it over. It sits upside down in the package. You then heat the whole thing in order to melt the small bumps of solder.

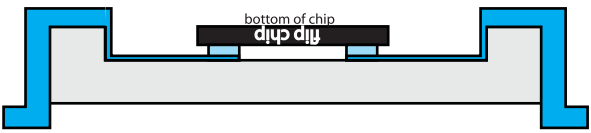


Figure 9–7. *We flipped our chip before melting the solder.*

Your pin metal traces are already in predetermined positions in the bottom of the package, just ready and waiting for these solder points to be flipped on top of them. You might have a full custom package for this one chip, just to make sure the connection to the pins is accurate.

Now, with flip chip technology, since there are no wires, you can have these flip chip pads anywhere you want on your chip. We are not constrained to using the edges for our pads as we were with wedge and ball bonding. The package pathways, which are metal strips, will not bend, droop, or otherwise interfere with each other or any other chip wiring.

This method is also used a lot in modules where there might be five or six chips in one big package. It is easier to pre-lay the metal traces and use flip chip technology than to allow wiring all over the place in a package with so many chips.

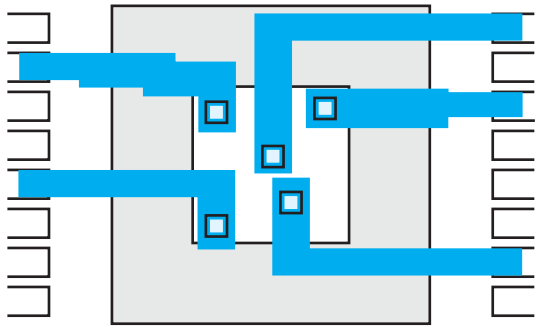


Figure 9–8. *Input/Output pads can be located anywhere on a flip chip.*

Multi-Tier Packaging

You can also have multi-tier packages that look like Figure 9–9. The package has more than one row of metal connection pins. The chip sits in the middle and you route one set of wires to each tier.

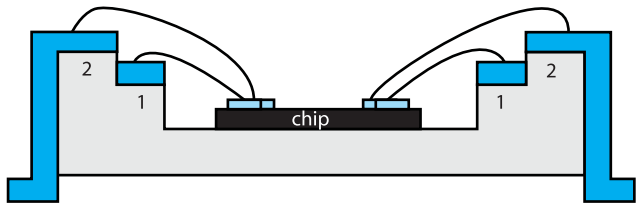


Figure 9–9. *A multi-tier package allows wires above other wires.*

The advantage of this type of package is that your shorter wires to the lower tier are quite taut and do not interfere with the wires far above. The separation between tiers is good enough to guarantee no wire shorts. Figure 9–10 shows you a top view.

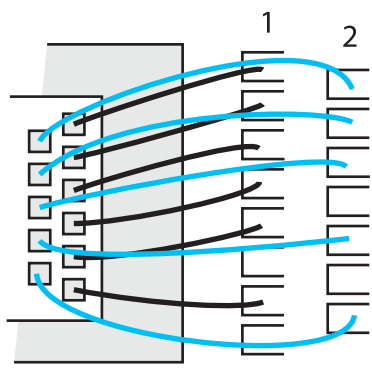


Figure 9–10. *Top view of the multi-tiered package with chip. The higher level wires (blue) travel above the lower level wires (black).*

Multi-tier packages allow you to bring some of your bond pads inboard toward the center of the chip a little bit, to make room for additional pads. But you need to leave yourself a gap in the outer ring of pads leading to the outside, so that the inner bond wire cannot short to any other wires. This technique is called offset bonding. The two rows are called **offset bonding pads.**

Issues in Packaging

Wire bonding is the most common method of attaching a chip to a package. These bond wires might tend to dangle all over the place in a poor bonding design. You have to be very careful lest you short those wires to other signals. You have to design your pad positions in conjunction with the selection of your package.

In these next sections we will look at some good bond-out habits to help minimize trouble.

Overall Appearance

The general rule is to make it look nice, as with the chip layout. Evenly distribute the pads around the chip if you can. Keep them to the outside. Keep them spread apart.

Make it look nice.

Figure 9–11 shows the metal pins from the package, the bond pads on the chip, and the likely paths of our bond wires. Everything is nicely centered in the package. Bond wires are even lengths. Everything looks nice. This will be easy to manufacture.

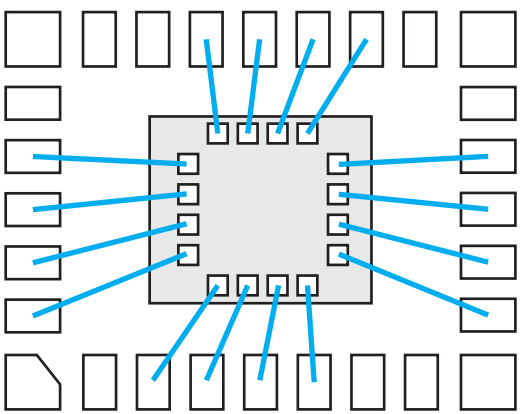


Figure 9–11. *Example of a good bond-out. It looks nice, so it will work well.*

Now, here's the same chip in the same package, but we can spot some troublesome issues. We have chosen a different bond-out, different pins for the wires.

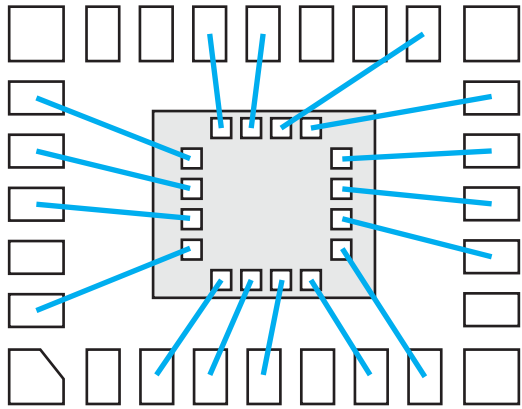


Figure 9–12. *Oops. How many problems do you spot here?*

Let's examine the various problems of this bond-out design in the following sections. And their solutions, of course.

45 Degree Rule

The first issue is the corner wire. You can see that the bond wire is starting to get pretty close to the pad next door to it.

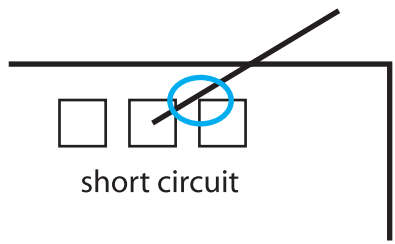


Figure 9–13. *A little too close for comfort. A likely short.*

This bond-out offers the potential of one bond wire shorting to the bond pad right next to it.

A good way to determine whether your bond wire could become too close to another pad is to look at the angle the wire makes with the edge of the chip. Look to see if your bond wire makes anywhere between a plus or minus 45-degree angle from the perpendicular position.

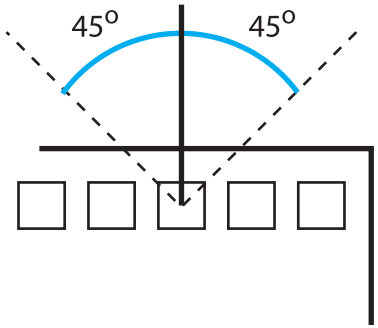


Figure 9–14. *A maximum deviation of 45 degrees in either direction helps keeps wires to themselves.*

If the angle starts to get greater than 45 degrees, there is a good chance you will start to short with the bond wire or the bond pad next door to it. We call this the **45 Degree Rule.** Keep your wires within 45 degrees of perpendicular.

Minimal Silicon Overlap

We have another issue in that corner. In Figure 9–15, the end pad is not in the corner of the chip, so the bond wire is overlapping too much of the silicon.

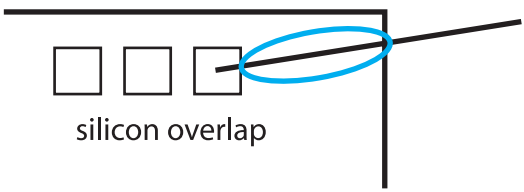


Figure 9–15. *Don't overlap any more silicon than you have to.*

Again, the bond wire could sag or droop, causing a short in the circuit. So, you have to keep the overlap of the bond wire over unused silicon as short as possible.

Wire Length

The third issue with our nasty bond diagram is that we have one wire that is quite long.

That bond wire will droop or flap around. There may be a rule you must follow for the length of the bond wire just to avoid this sort of problem.

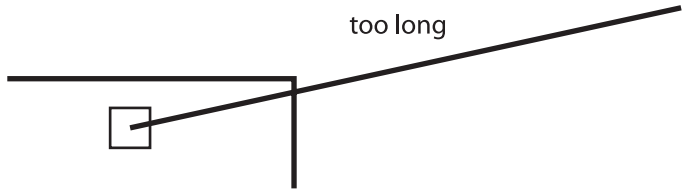


Figure 9–16. *Keep your wires as short as you can. This stretch is probably not necessary. Plan your bond out before you begin your chip layout.*

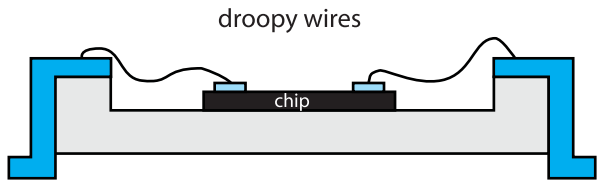


Figure 9–17. *Long wires droop.*

Now, all these rules—the 45 degree bond angles, the lengths, the overlap onto unused silicon—exist in all the assembly processes, but they are different depending on the various people and companies involved. Nevertheless, most follow these general guidelines. They are rather generic for most assembly purposes. And, as with most of this book, they are good common sense.

Pad Distribution

Some mask designers try to organize their pads by crowding all the bonding pads together at one end of the chip. Why do that when you can spread them out?

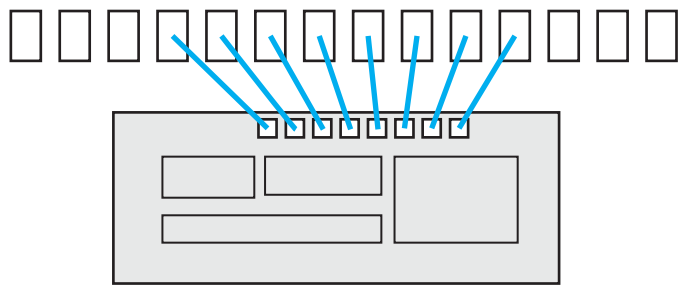


Figure 9–18. *Crowded pad placement.*

If you have the room, spread them out. Place your bond wires away from each other as much as you can.

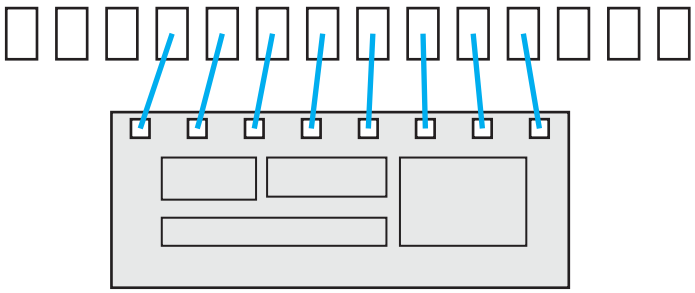


Figure 9–19. *Improved pad spacing.*

This design separates the signals more, reducing the risk of interference.

Sizing Estimates

Before you can even begin your preliminary floorplanning, you must know the requirements of the packaging. It's all well and good to have a great chip floorplan, but you have to be able to manufacture the thing.

Make sure your chip can be bonded out, that your I/O pads can work in the package selected. Selecting the package to a great extent is determined by the size of the chip.

Sometimes you may be given a package, told the number of pads, and you are asked to guess in advance if the chip will fit in that package. Or, at other times, the package choice may be driven by cost. The business people may be saying, "We want to make a chip we will sell for a dollar. If we can fit our chip in this particular package, we can make more money. If the chip is too big then we'll have to go to a bigger package and it will cost us more."

So, even before you have done any chip design at all, you may be asked to do some chip size estimates. "How big will it be?" "Will it fit in that package?"

Pad-Limited Design

There are several ways you can do your size estimating quickly. The first method is to sketch what we call a **pad-limited design.** This gives you a quick first-look, which may eliminate the package immediately before you spend a lot of time on more accurate sizing estimates.

First, identify your signal requirements. From that, you can determine the minimum size of the chip, based just on the number of signals, and hence bond pads, needed.

Find the minimum required spacing between the pads, and squeeze those pads as close together as you can in a rectangular shape. Then you say, "That's my chip. That's pad-limited."

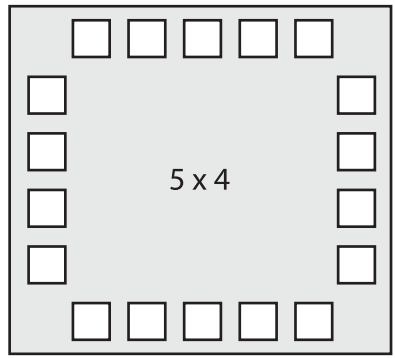


Figure 9–20. *The minimum size of an 18-pad chip, based on the number of pads.*

Irrespective of how much circuitry you have in the middle of this chip, this is the smallest that you can make it using your given number of pads.

You might wonder why we didn't use the corner spaces. This is possible if you are using larger than normal gaps between your pads. However, with minimum pad spacing, the two pads that would neighbor the corner pad are so close to each other that we have no room to run a wire between them to a corner pad. (But good thinking, if you were wondering.)

Now you can start to fill in the gaps with some circuitry. You go back to your circuit designers and say, "Ok, the pad-limited size for 18 pads in that aspect ratio is 4 by 5. Here's our minimum size. Is that going to fit in the package?"

Then the circuit designers go back to the packaging engineers with these dimensions. They will ask if that fits in the package. The packaging people will say either yes or no.

Of course, when you do the pad-limited design, you assume you can squeeze all your circuitry into this limited area. That might not actually work in the end. You might have too much circuitry.

However, you do the pad-limited design as your first check to see if using the specified package is even possible. It might eliminate that particular package immediately, in which case you might have your answer right away, and save yourself the time it would take to do a more detailed sizing estimate.

Core-Limited Design

If you knew you would need a very large number of I/O pads, you would expect the pad-limited design to be a good rough sizing estimate. But, sometimes you do not need quite as many pads, compared to the amount of circuitry you will need.

In this case, you would want to better examine the circuitry requirements to determine a rough sizing estimate. This is called a **core-limited design,** or **circuit-limited design.**

If you are lucky, you have been given a whole bunch of circuitry that has been laid out before. You are just reusing some of those same blocks in your new chip. In that case, you just take those blocks and put them as close as you can to each other.

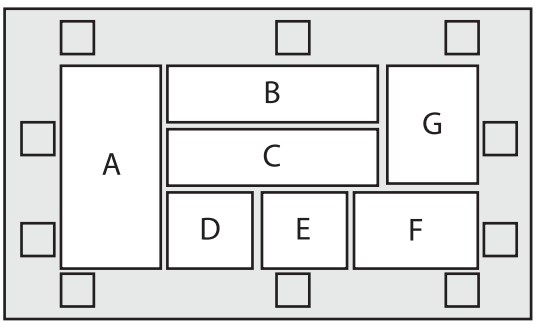


Figure 9–21. *The minimum size of the chip limited by the anticipated sizes of the blocks.*

The number of pads is no longer the most limiting issue. The rest of the chip is. The circuitry is driving how big the chip will be. Once you have an idea about the sizes of your required circuitry, place the pads around the outside. There is your estimate for your chip size.

You may be able to get some good guesses from your circuit designer. He might give you some previously used blocks and say, "I think this new mixer is going to be 20% bigger with a similar aspect ratio."

When you are doing some package choice what-ifs, you should do a pad-limited version first. Then you do some what-ifs based on guesses for the circuitry, using reference blocks if you have any. Then go back and say, "Well, I think it's going to be this particular size."

Package Maximum Check

The third way to quickly determine if a particular package will work or not is to examine the package dimensions. Find the maximum die size that can fit in the package, based on the rules and the size of the package. Usually you will see a setback requirement keeping the chip away from the metal pins by a certain distance. Determine from the package the largest size allowable for the chip.

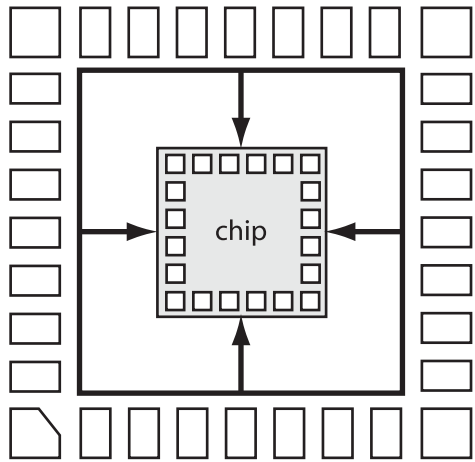


Figure 9–22. *The maximum size of the chip is given by the dimensions of the package design.*

In this case, you will say, "Well, the maximum die I can fit in that package is a four by four." And, if you have estimated a 3 × 3, then you are ok. If, however, you think you have a 5 × 5 mm chip, then it will be a push trying to squeeze everything into this package.

Final Die Size Calculations

You have finished your chip layout. You are happy with your bond angles. Your manufacturer has approved your bonding diagram. Now, your final procedure is to calculate what we call the **step and repeat** numbers.

If you remember, our die will be copied across our wafer by the hundreds. After the whole wafer is processed, we must separate all the hundreds of chips so that each one can be placed in its own little package.

To separate the chips, you put the completed wafer on a vacuum chuck or stick the wafer to some backing material. You then literally zip your separating mechanism up and down between the chips.

Typically, these days the separating mechanism is a saw, just literally a circular table saw. The saw removes a small width of material as it cuts the chips apart. It might cut a path 50, 60, or maybe even 100 microns wide. The thickness of the blade varies from company to company, but can be pretty big.

The real cool way of separating the chips is to press the wafer onto a sticky sheet of cling film before you saw it. As you run the saw between the chips, the cling film, being rather thick, remains uncut. Once the wafer is completely sawn apart, you can then stretch out the cling film. All the chips spread out so you can get to them.

They put these cling film wafers on a machine called a **spreader.** You end up with this really big, round sheet with individual die that are easy to get at with a pair of tweezers or a machine that picks them up.[3]

Now, if you were to lay out your chips directly alongside each other on the wafer, there would be no room for the saw blade to cut them apart. You would gouge or rip into the delicate circuitry when it comes time to separate them.

So, you need a gap between the chips. You have to leave some room for the saw blade path. These spacing structures we place between the die are called **dicing channels.** Sometimes they are called **scribe lanes,** or **saw channels,** or **streets.** They are all basically the same idea: A gap between the various chips on the wafer to allow the saw blade room to cut.

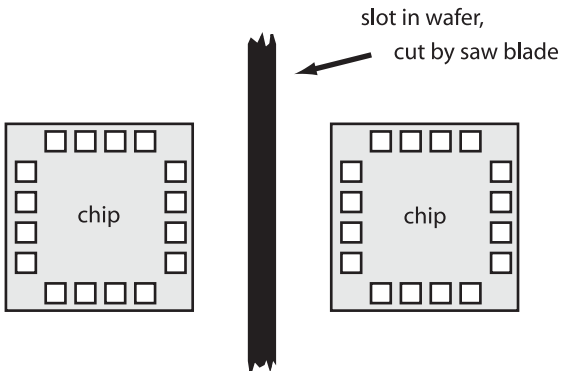


Figure 9–23. *We must consider room for the saw blade.*

[3] As an interesting side note, each chip has probably been probed while on the wafer for functionality, sometime before the wafers are diced into separate chips. The chips that did not work will have a black spot of ink dropped on them. So, the machine that picks them up off the cling film will look for a big black spot of ink, using a TV camera. It will only pick up the chips that are actually functional.

Typically, the mask designs for these dicing channels are openings in the passivation layer, and a couple other openings in the via layers. You do not want to run your saw through metal or other unnecessary layers of material, so you open a channel all the way down to bare silicon. This is where you run your saw blade. The saw blade can get clogged if you saw through too much metal.

There are usually one, two, or maybe more extra masking layers that are used to open the wafer clear to the base layer. These would be called the dicing channel layers, in your layout files.

Figure 9–24 depicts a typical area of a finished wafer, showing three chips. In this example, you will notice a big seal ring, which is essentially a huge substrate contact around the outside of each chip. Some people call it a **crack guard ring.** It is basically a chunk of metal. All the metals in the process are stacked on top of each other, in order to keep any cracks that occur out in the edge of the die from working their way in to the circuitry inside.

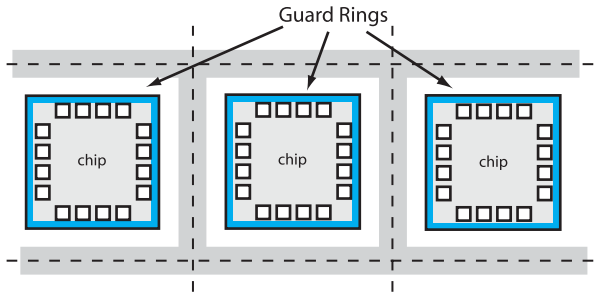


Figure 9–24. *Metal guard rings help protect the chips from the saw blades.*

Having a solid metal ring also gives you a well-defined edge of the chip. "There's the edge. That's what we've drawn. That's our nominal chip size."

You would not want to run your saw directly next to the edge of your chip, even with a guard ring, would you? If the process for the dicing channel over-etches you could destroy some of your circuitry. So, to further protect your chip, you build in a safety area, known as a **scribe margin,** between the edge of your active die and your dicing channel layers. The scribe margin is shown as dimension *B* in Figure 9–25.

Dimension *A*, in Figure 9–25, is the final drawn dimension of our chip as you see it in your CAD layout editor, including I/O pads and crack seal guard ring. Some people call dimension *A* the **active die size.** (These options, and their names, depend on which company you work for, of course.)

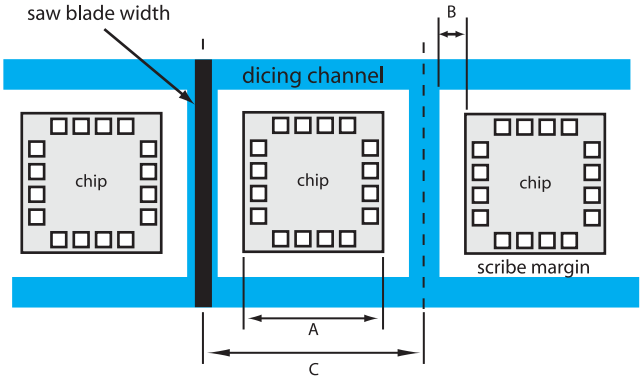


Figure 9–25. *Dicing channels are separated from the chips by a scribe margin.*

To find the final area on the wafer needed to build each chip, you measure from the center of one dicing channel to the center of the next dicing channel. This measurement would automatically include the scribe channels. This is denoted on Figure 9–25 as distance *C*. This is known as your **step and repeat** distance. There is a step and repeat distance for *x* (horizontal) and a step and repeat distance for *y* (vertical).

Here are three chips as they sit on the wafer. Notice the black area, the saw blade width. This path reduces the ultimate size of your chip by actually destroying some of the wafer as it cuts. The path of destruction is the width of the blade. The final size of the actual chip that will be placed in the package extends to the edge of the finished, sawn area.

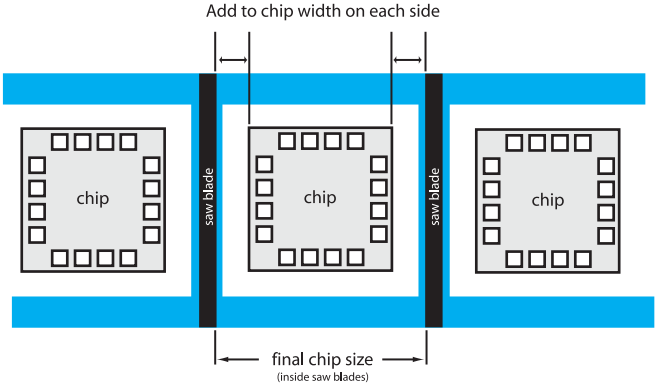


Figure 9–26. *Final chip size is just short of the step and repeat dimension, by the width of the saw blade.*

You can see that the chip you place in the package is neither the size of the chip you worked on in your layout editor, nor the step and repeat size that you determined for your wafer placement.

You can see the various items a little better in this close-up.

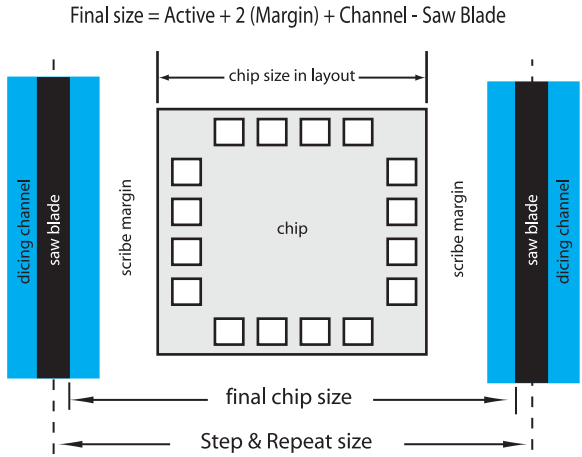


Figure 9–27. *Your final chip size is not the size you work with in layout. Don't forget that.*

To know the final die size of your chip, typically needed for packaging, follow these steps:

1. Begin with your active area width.
2. Add one scribe margin for each side (that's two scribe margin widths).
3. Add half of the dicing channel width for each side (that's one full dicing channel width).
4. Your final chip size then gets smaller by half the width of the saw blade for each side (that comes to one full saw blade width.)

$$\textbf{Final size} = \textbf{Active} + \textbf{2(Margin)} + \textbf{Channel Width} - \textbf{Blade Width}$$

And that's it, your final chip size. You can calculate either the horizontal or the vertical dimensions of your final chip size using this formula.

People often forget about all these margins and channels and sawing when they do their maximum die size calculations. So, be sure you understand how your dicing channels, saw blade width, scribe margins and your step and repeat issues affect the final die size.

This has happened to people I've worked with. There have been times when people want to get the biggest possible die in a package.

They say, "Ok, 2 millimeters by 2 millimeters is the maximum die." And that's what they dial in. They put their active dimension at 2 by 2, when it really should be their final, finished die size that is 2 by 2.

I've seen it happen, and it's a killer.

Try It

1. Given the following information, calculate the final die size that will be required for your chip.

   Active area = 2100 μm × 1970 μm
   Margin = 15 μm
   Channel width = 120 μm
   Blade width = 50 μm

   What is the final size?

   BONUS: What is the step and repeat distance on the wafer?

2. Given the following information, calculate the maximum active dimensions for your chip.

   Max die size = 3000 μm × 3000 μm
   Margin = 22 μm
   Channel width = 138 μm
   Blade width = 90 μm

   What is the maximum active dimension?

   BONUS: What is the step and repeat distance on the wafer?

ANSWERS

1. Use the formula:

   Final size = *Active* + 2(*Margin*) + *Channel width* − *Blade width*

   Find first dimension:

   $$\text{Final size} = 2100 + 2\,(15) + 120 - 50\ \mu\text{m}$$
   $$= 2300\ \mu\text{m}$$

Find second dimension:

$$\textit{Final size} = 1970 + 2\,(15) + 120 - 50\ \mu\text{m}$$
$$= 2070\ \mu\text{m}$$

Our final chip size is 2300 × 2070 microns.

BONUS: Same formula as above, but we do not subtract for the blade width. Step and repeat increments are 2350 × 2120 μm.

2. Rearrange the same formula to solve for active:

$$\textit{Active} = \textit{Final size} - 2(\textit{Margin}) - \textit{Channel width} + \textit{Blade width}$$

Both dimensions are the same, so we will calculate only once:

$$\textit{Active} = 3000 - 2\,(22) - 138 + 90\ \mu\text{m}$$
$$= 2908\ \mu\text{m}$$

Our chip Active will be 2908 × 2908 microns.

BONUS: This would be the given maximum die size plus one saw blade width. Step and repeat increments are 3090 × 3090 μm.

Now, personally, I like to round my chip sizes to a multiple of 10 microns. You can make the active final die size whatever you want, but if you stretch it or shrink it to the nearest 10 microns, it makes your stepping out easier.

It's usually only one or two microns extra, so it doesn't make a difference at all. (Of course, if I know the chip size is crucial, I use the actual sizes. But rounding the size up or down to the nearest 10 microns is usually no problem.)

Particularly Europeans used to do chip sizing to the thousandth of an inch, which is 25.4 microns. But it's just personal preference. It doesn't really matter.

Filling Pad Gaps

If you do have a core-limited circuit design, you will have wasted space around the edges between the pads. Maybe you can push some of your circuitry into this wasted space to try to use those gaps.

If you are reusing circuits, then you can re-lay parts of your circuit elements in and around these pads trying to get the chip as small as possible.

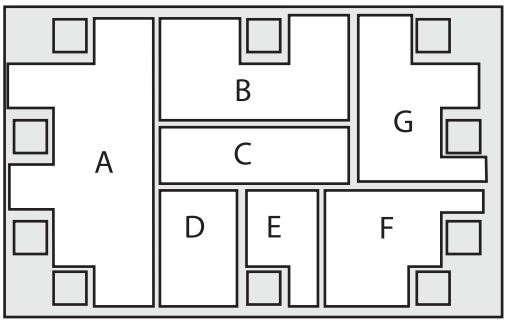


Figure 9–28. *We can use extra space surrounding our pads to shrink our chip size.*

Closure on Packaging

All these packaging decisions depend on many factors. How important is size for this application? How many pads do we need? How big is the circuitry itself? How large a chip will the package rules allow? Do we have the technology to manufacture chips in multi-tiers or with flipping? How important is it that we use a particular package? And so on.

Don't forget to consider the choice of packaging early—before you begin your layout. Know the size requirements and where your bond pads will go. These considerations will dictate your layout. Misjudging these requirements could make worthless many weeks of work in the wrong direction.

Here's a little summary:

- Learn your bonding requirements early.
- Make your bonding diagrams look pretty.
- Are the bond angles ok?
- Make sure you don't have too many wires in too short of a space.
- Don't go off and do a lot of layout before you know if your chip fits in the package.

Here's What We've Learned

Here's what you saw in this chapter:

- The importance of early packaging decisions
- Packaging rules
- Bonding methods
- I/O pad placement options
- Three quick package-fit estimation methods
- Using pad gaps
- How to compute finished die size

CHAPTER 10

Verification

Chapter Preview

Here's what you're going to see in this chapter:

- So, you think your layout is finished, do you?
- How to automate checking for thousands of design rules
- How to automate checking for a correct layout
- How layout-checking programs find your devices
- How to use the checking programs
- How to problem solve difficult errors
- How to read and write basic checking programs
- Why you should read and write checking programs
- How computers talk to you about your components

Opening Thoughts on Verification

Up until now we have just agreed that there are lots of design rules, maybe thousands. We have not told you how to make sure you have cleared each and every one. Even the most diligent person in the world is going to miss something somewhere with all those detailed rules, all those components and all those wires.

You only need one teeny, tiny, little mistake to completely kill your chip. With cycle times in a typical wafer fab taking 8 to 12 weeks, and a wafer costing thousands of dollars, you want to make sure that what you commit to silicon is correct.

To help us watch all our rules, most tools nowadays have computer-aided rule and layout checkers. You somehow enter all of the thousands of rules into the software. The computer goes away and checks your layout against these rules for you.

The computer makes sure that your metals are not too close to each other, that your transistors have the N well in the right place, that the P+ active of your source-drain region is the right number of microns wide, and so on.

Let's examine how computers are able to know so much about your layout and about the schematic, how they know whether the two match, and whether design rules have been broken.

Checking Software

The **DRC (Design Rule Check)** program knows everything there is to know about your process. It will go away and diligently check everything that you have laid out.

If the design rule control files have been written well, the DRC will find even the tiniest little mistake in your layout. You can be guaranteed that what you put out to silicon is correct and is likely to function.[1]

A DRC program typically will put back into your layout a bunch of error markers. These are highlights on the layout locating your errors.

You might repair all your errors in your first attempt. However, in the process of fixing all those errors, you may have introduced other errors elsewhere. Fixing DRC errors is another iterative process. You run the DRC, you fix the errors, run the DRC, fix the errors, run the DRC . . . keep running around this loop until finally no errors pop out.

The DRC was only the first level of checking. Just because you have a circuit that is DRC clean does not mean it is wired correctly. The wires may be connected to the wrong two components. You may have introduced some open circuits or short circuits, for example. So, we have another set of programs that check your layout even further. The second software check is called **LVS,** which stands for **Layout Versus Schematic.**

[1] I won't say *work,* I will say *function.* Particularly with high frequency circuits, just because it's wired correctly and design rule clean does not mean that it will work the way you wanted it to work, due to all the parasitics.

The LVS program is told which layer combinations make a transistor, which combinations make a resistor, and a capacitor, and so on. The LVS program can then find your devices by looking at the layers you have drawn. The LVS program will extract from your layout what it thinks you have built, then compares what it has extracted to the circuit designer's schematic.

The LVS tool not only checks the components and wiring, but it makes sure that you have the right values—the right sizes of resistors, the right sizes of transistors, the right sizes of capacitors, and, even further, the right types.

The DRC and LVS programs are very sophisticated and very accurate. For example, you may have been asked to place an N well resistor in your circuit. You might have misread the schematic and put in a poly resistor by mistake. This error will be highlighted.

Just like the DRC process, the LVS is also iterative. If you find a mistake in the wiring, you fix it, then rerun the LVS. This may cause you to find a new mistake, fix it, and run the LVS again. You keep reworking and rechecking until it finally pops out LVS clean.

However, in fixing the LVS problems, you may have introduced some more DRC errors. So, you have to go back and run the DRC again. Then, of course, once you are again DRC clean, you begin the LVS process again. You continue going around these iterative loops of DRC and LVS checking until finally everything pops out DRC and LVS clean.

Let's look at how a DRC program identifies components and checks your layout against those thousands of rules we mentioned.

Design Rule Check (DRC)

Both DRC and LVS programs begin by using the same basic set of operations to find circuit elements in your layout. They both use sequences of command lines that include Boolean operators. After that, the DRC and LVS programs diverge, completing different tasks. Let's look more closely at those Boolean command lines.

Boolean Command Lines

You may be familiar with the simple Boolean AND operation or the Boolean OR operation using single input items. For example, you might know that *true* AND *true* outputs *true.* You can perform the same operations on two-dimensional mask design layers. If you have two layers, you can output a third layer based on Boolean functions. I'll show you how.

Sometimes you may be the layout engineer who is new in the group, and suddenly here comes a brand new technology with no DRC rules files written for you. How do you check that what you are laying out is correct? You have been given a process manual and it's just up to you. You may have to write your own DRC rules.

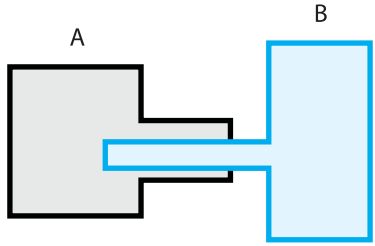
You might be certain your layout has been thoroughly checked and should be reading out clean, but the DRC and LVS programs won't give you clearance. This would be a nice time to know your DRC commands well enough to open these text files and possibly spot a problem. It happens.

It's important for layout engineers to understand how DRC control files are written. You might be asked to write them or need to use them one day.

Also, it's a career-expanding option you might enjoy if you find you have a liking for the logical side of things, if you understand layout and understand semiconductor processing. The people who write the control files are the people who understand the process very well.

AND Function

Let's play with these Boolean operations using one polygon, *A*, on one layer, and another polygon, *B*, on a different layer.



AND

A	B	Z
0	0	0
0	1	0
1	0	0
1	**1**	**1**

Figure 10–1. *AND function will trap all areas containing A AND B. This would be the overlap. Notice output Z exists only when both A and B exist.*

If you remember the Boolean AND function, you only get an output when both inputs are present. So, in this case, the output we get from our two layers is the overlap between the two polygons. The overlap represents the only area where both *A* and *B* are present. We can either keep this information in a temporary layer or just keep it in memory.

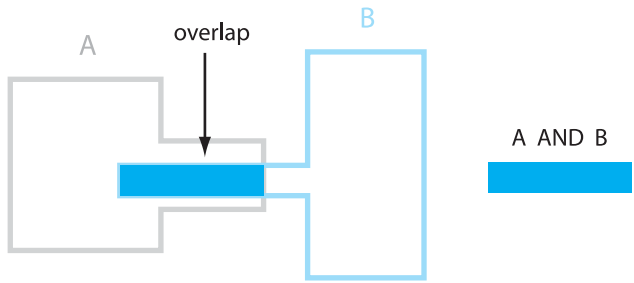


Figure 10–2. *The overlapping area is rather small, seen to the right. This will be stored in another file or in memory.*

The AND function is very good for finding CMOS transistors. For example, CMOS transistors have a poly layer for the gate and an active layer to define the source-drain regions. Wherever you have poly on top of active you will have a transistor.

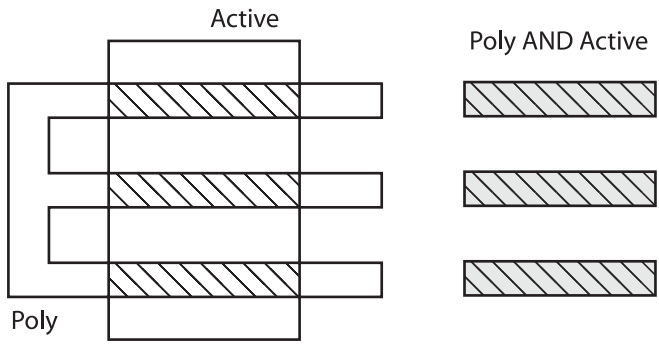


Figure 10–3. *Aha! We have found a transistor by looking for poly that overlaps active.*

You can assign the outputs of these Boolean functions to temporary layers. You can then reuse these new layers against each other or against other real layout layers. The layer we created by identifying the poly AND active is never a layer you will build with silicon. It is just a temporary file for you to work with later.

We could call that new layer by the name Temporary File Number One, using symbols TMP1, for example. We would write the line of code assigning this name as follows.

TMP1 = POLY AND ACTIVE

This is a typical line that you might write in a control file. You can then use TMP1 in a Boolean operation with another layer, and call that output Temporary File Number Two, or TMP2, as follows.

TMP2 = TMP1 AND NWELL

If you recall, TMP1 contains all our CMOS transistors. So, if we AND the CMOS transistors with N well, then we find all of our PMOS transistors.

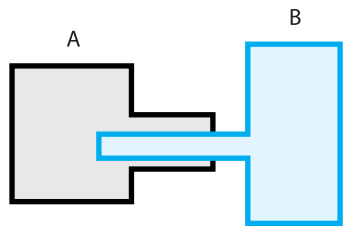
So now, TMP2 has been assigned as another derived layer. Like TMP1, TMP2 will never actually be built. It is merely a file containing the locations of all of the PMOS transistors. When we have finished our checking programs, we are free to discard TMP1 and TMP2.

Once we find our PMOS transistors, we can check them against the PMOS transistor rules. That's the juicy benefit of this program. The hard part is first telling the computer where our devices are located. A DRC control file has a lot of up-front work to identify the various devices that you want to check. We can build as many of these Boolean sequences as we like to find all of our various components. Once you have found them, then you check the devices against their appropriate rules.

OR Function

The next widely used function we will discuss is the OR function.

The OR function will give us an output when any one of the conditions is present. If we have *A* or *B*, or both, we will get an output. Therefore, the output of OR on our sample layers would look like a larger polygon than either of the originals.



OR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Figure 10–4. *OR function will trap areas containing A or B.*

The OR function merges the two polygons into one. We could go further, and name this merged area TMP3.

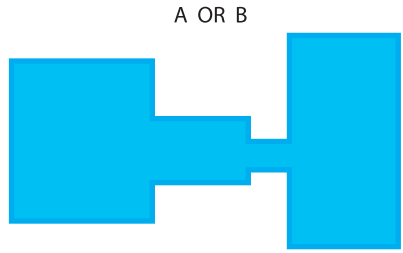


Figure 10–5. *The shape "A OR B" combines both areas.*

Here is the code that defines a merged layer called TMP3 from two layers called TMP2 and PPLUS.

TMP3 = TMP2 OR PPLUS

We have combined a real layer (PPLUS) with a previously defined temporary layer (TMP2). All we are doing at this point is building additional temporary layers that we will work with as we continue.

You might want to combine temporary files with other temporary files. For example, let's say you have found a bunch of transistors and you have found a bunch of resistors. There might be common design rules for both of them. You could put them both on the same layer and work with that defined layer, instead of working with each real layer individually.

NOT Function

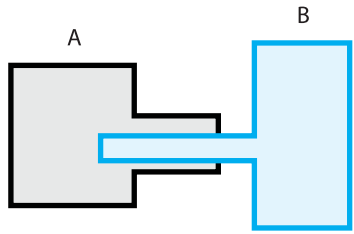
Sometimes people get confused with the NOT function. The NOT function, particularly in DRC's, can be described as an AND NOT function. This clarification helps some people understand more easily what is happening. Let's look at an example.

TMP4 = A NOT B

. . . is the same as saying . . .

TMP4 = A AND NOT B

Here is how you can use the AND NOT substitution to help. Look again at polygon B in Figure 10–6. Now consider everything that is NOT B. That would be everything outside the polygon. As you look at Figure 10–6, combine polygon A with the area you found for NOT B. Where do they overlap? That overlap is A AND NOT B. This is shown in Figure 10–7.



NOT

A	B	Z
0	0	0
0	1	0
1	**0**	**1**
1	1	0

Figure 10–6. *NOT function eliminates some sections from the first-named region. We get an output only when we have A, but we do not have B.*

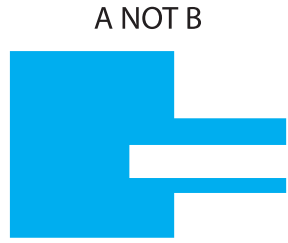


Figure 10–7. *Result of A NOT B, which can be thought of as A AND NOT B.*

The NOT function is useful, for example, if you have found a resistor layer. There could be another layer used in conjunction with that resistor to change its doping, that has different design rules. You can NOT the resistor layer with the specially doped layer. This effectively throws away the specially doped portions of the resistor layer. What remains are only the normal resistors. (See Figure 10–8.)

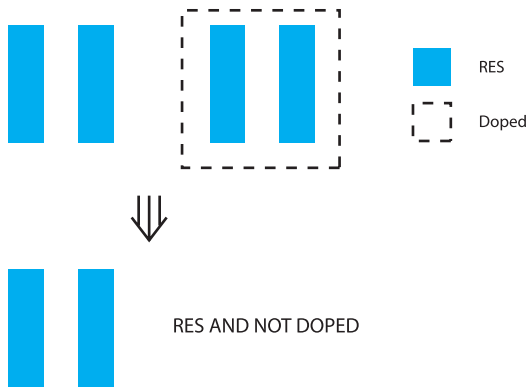


Figure 10–8. *We ran a NOT function to eliminate the specially treated areas.*

As another example, consider two rectangles of poly. In this case, one of the poly rectangles is completely inside P+, so the command line POLY AND NOT PPLUS will give me the top rectangle only. However, the command line POLY AND PPLUS will give me the bottom rectangle only. You should now be able to understand how we can select our components using only Boolean commands.

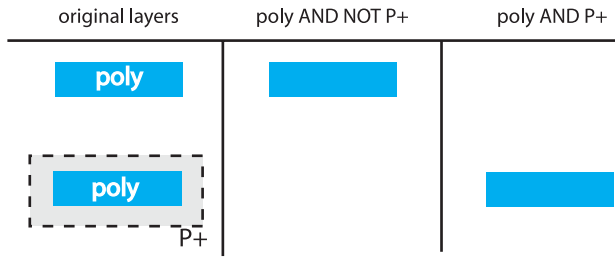


Figure 10–9. *Results you will see after running two Boolean commands on your original layers.*

The NOT function is order-sensitive. For example, A NOT B will give you different output than B NOT A.

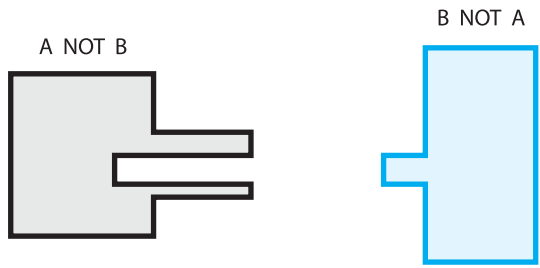


Figure 10–10. *The NOT function is order-sensitive. Notice the different results just by switching A and B in the order.*[2]

You can see that all these Boolean AND, OR, and NOT functions are very useful for finding little individual chunks of your circuitry. You can string all these commands together, creating temporary layers that hold those specific devices.

Let's follow all the command lines from the text above, to see what we have done so far.

[2] B NOT A reminds me—check out dilbert.com.

TMP1 = POLY AND ACTIVE
TMP2 = TMP1 AND NWELL
TMP3 = TMP2 OR PPLUS
TMP4 = A AND NOT B
R_NOT_DOPED = RES AND NOT_DOPED
TMP5 = POLY NOT PPLUS

So far, the DRC command lines have not checked devices against any rules. We have just derived layers containing devices. Now that we have found them, we can do some checking of the rules. That's what we talk about next.

Rule Checking Command Lines

There are usually several rules you want to check on any device layer you have found. Usually the first check is, "How far away are they from each other?"

This is a good check for metal, for instance. Specifically, you want to check that your metal to metal spacing is the right distance, that you do not have the metals too close to each other. You can do this kind of check on either the derived layers or your real layers.

The first check we will make is an *external* check. For instance, here is a line you could write in your file:

DISPLAY CHECK1 = EXTERNAL M1>= 2μm

This line of code checks the external edges of your polygons against each other. It checks that the distance from edge to edge is greater than or equal to two microns. In this example, the word *external* is recognized by the software as a pre-set command.

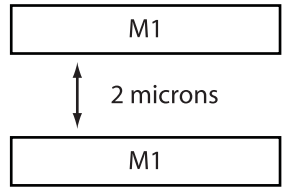


Figure 10–11. *Checking distance from edge to edge.*

Depending on the tool, if there is a violation of the rule you have specified, the software will usually place some kind of marker layer that lights up in certain places so you can see where your errors are.

The other common check is width.

Figure 10–12. *We can check internal distances as well.*

Again, metal is a good example. Below is a line of code to check Metal One (M1) width. Any widths that are less than 2 μm will light up. The errors are output to a layer called CHECK2 in the following line.

DISPLAY CHECK2 = WIDTH M1<2μm

Typically, these command lines are written as text files. The software runs each of the operations sequentially one after another. All these checks are usually built into the system tool that everyone in your office uses. Somebody wrote them all, just as we see here, line by line.

The lines defining temporary layers from Boolean operations are in the same DRC file as these rule commands. It's one long file. The temporary layers are defined first. Then the rule-checking command lines follow.

Each tool will have its own syntax, its own language. There is a huge book you can reference for each tool. The words EXTERNAL or WIDTH, for instance, could be words already built-in for you to use. You will have to check your manual for the syntax your software will recognize.

Use Boolean commands to locate.
Use tool syntax to check against rules.

Each check will usually have a list of options. You would be surprised at how much you can check this way. Most tools have some very sophisticated logic and technology built into them.

Here is our DRC rules control file that we have written so far:

TMP1 = POLY AND ACTIVE
TMP2 = TMP1 AND NWELL
TMP3 = TMP2 OR PPLUS
TMP4 = A NOT B
R_NOT_DOPED = RES AND NOT_DOPED
TMP5 = POLY NOT PPLUS
DISPLAY CHECK1 = EXTERNAL M1>= 2μm
DISPLAY CHECK2 = WIDTH M1< 2μm
DISPLAY CHECK3 = WIDTH TMP5 = 6μm

I have added another check as the last line in the program, called CHECK3. That checks the width of TMP5. TMP5 is defined as POLY NOT P+. So in the output file, TMP5, you will see highlighted any width of these objects less than 6 μm.

As you can see, with the number of layers involved and the number of checks required on every process, this rules control file could be huge. It could be thousands and thousands of lines.

Someone has to sit there and understand every little processing nuance and subtlety in order to write something this long. It can be a real challenge. It can be real fun. What they typically do is break the file into sections. For instance, they could have one person write the Metal One rules, and another person write the Metal Two rules, and so on.

You can usually open your DRC file and look through the lines. Try it sometime. With your manual to help interpret the syntax, you should be able to understand the commands. When a new tool comes along, you might even feel confident enough to translate the rules across.

Looking through the DRC rule file is how I solved a problem once. I was getting what I considered to be false errors. So I opened up the DRC rules files.

The person who codes the DRC can make one set of assumptions and think that the coding is ok. And, for his situation it probably is ok. But it can be very tough to code up a rules file for every single way of doing something. There is bound to be some rare condition not covered, that won't work.

I was getting a bunch of weird errors coming out of my DRC. So I opened up the DRC file, found the checks, stepped through them one at a time, found all the derivations, and finally found the mistake.

It was as simple as something like finding a less-than sign (<) where he should have put a greater-than sign (>). Sometimes there can be some real subtleties in there.

You feed it back to the developers, they fix it, and you move on.

Layout Versus Schematic (LVS)

LVS, which stands for Layout Versus Schematic, as you recall, uses the same commands as the DRC. You still use Boolean commands to find your P+ resistors, for instance. But, LVS is not as straightforward as a DRC. Extensions

to the design rule checking software that are found in LVS actually create real components and circuits.

Most people call the process LVS, but in reality, it is not just layout versus schematic. It is a two-step process. The first part of the LVS process is the extraction of the device information from the layout. Some people call it **connectivity extraction.** This procedure creates a file that says, "This is what I found. This is how I think it is hooked up." It is called connectivity extraction because it *extracts* the devices from your layout.

Even though you have used parameterizable cells, or pieces of layout that you know are correctly wired, the LVS program is blind to what you think. It says, "I don't care what I've been told before. I'm not even going to look at any other guesses. I'm just going to run with my own set of rules. If I find poly on top of active, then I say I have an FET of some sort."

The LVS is like a neutral third party, an independent arbitrator, another pair of eyes. Some LVS versions may actually give you a layout showing what devices it finds. Some versions may just give you a **netlist.** A netlist is usually a text file. Each line identifies one component that it has found, plus information about that component. We discussed these in Chapter 1, remember.

The second part of the LVS process is the comparison. Most modern tools use two netlists in the comparison process. The tool extracts a netlist of the devices that it finds from the layout, then generates a netlist from the schematic, then compares those two netlists.

As with the DRC rules files, the LVS files can be quite lengthy and complicated to write.

Typically, when you run your LVS on simpler circuits you get an output that says the files match. In that case, good job. However, on larger cells there will likely be some issues reported. These issues might range from wires you haven't completed, to wires you ran in the wrong place, to incorrect values of components, to not enough components, or perhaps the wrong type of components.

The LVS is trying to help report connection problems, but sometimes understanding outputs from an LVS program can be a tough battle. Let's look at a sample netlist generated by an LVS program.

Netlists

Here is a typical SPICE netlist, a sample extraction output.

```
R1   A   B   10K        PPLUS
Q1   A   D   E     F    NPN        A = 5
C1   E   J   5P         MOSCAP
```

Here is how to read the netlist:

LINE ONE: The LVS program found a resistor, which it called R1. This resistor is connected between nodes A and B. Its value is 10,000 ohms, and it is of type P+.

LINE TWO: The next device SPICE has found is a transistor called Q1. The transistor is connected between points A, D, and E. Its fourth terminal, which is usually the substrate, is connected to point F. It is an NPN transistor, and its area is five microns.

LINE THREE: The third line refers to a capacitor connected between nodes E and J. Its value is 5 picofarads and it is of type MOSCAP.

There are lines and lines and lines of this stuff in a netlist. These outputs can be quite long. We might have thousands of devices identified, each with its own line.

These lines form a text version of how the layout devices are connected. You can translate a netlist to a schematic. For the above three devices, our schematic would look like this.

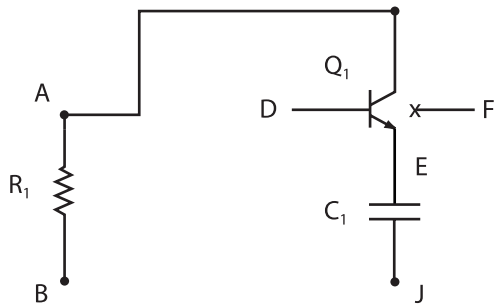


Figure 10–13. *Netlists contain enough information to draw a schematic.*

Problem Solving

Most LVS programs will give you a comparative listing of all items found in both the schematic and in the layout. These lists should match, shouldn't they? Well, yes, they should. Now let's look at some problem solving techniques, just in case they don't.

1. Check Number of Devices

If your LVS fails (finds problems), the first thing to check is that you have the same number of devices in the layout as in the schematic. Do you have the right

number of transistors? Did you forget to put in a resistor, perhaps? Who cares if they are even wired correctly at this point? First, just see if everything is there.

Let's say you get an output that looks like this.

LAYOUT		SCHEMATIC	
R	10	R	9
Q	3	Q	3
C	4	C	4

In this example, the LVS program tells us it has found 10 resistors, 3 NPN transistors, and 4 capacitors in the layout. And it has found 9 resistors, 3 NPN transistors, and 4 capacitors in the schematic. We apparently have an extra resistor in the layout. There is a good chance we placed one component too many, and that could be messing up our LVS.

A quick check of resistors should reveal the extra one. This problem alone may clear a long list of related errors.

2. Check Types of Devices

Here is another example of a problem that the LVS program might find. Although we have the correct number of resistors—we have seven resistors total—they are not the correct types. If you look at the LVS output below, we appear to have one resistor built from the wrong material.

LAYOUT			SCHEMATIC		
R	3	PPLUS	R	4	PPLUS
R	4	NPLUS	R	3	NPLUS
Q	3		Q	3	
C	2		C	2	

We are expecting to see four P+ resistors in the layout, but we have only three. Notice the reverse problem with the N+ resistors. We are expecting to see three N+ resistors in our layout, but we have four. We have apparently laid out an N+ that should be a P+.

Finding and fixing this error could erase many other related LVS errors, as is always the case.

Checking the number and types of devices before you even look at the wiring can give you a very good indicator of where the problem might be.

Rule of Thumb: Make sure the device counts are correct, by both number and type.

3. Check Number of Nets

If you know you have the right number and type of devices, the next problem to check is the number of nets.

Let's say our circuit is shown in Figure 10–14. We have five nets, numbered one to five. The top net, 1, includes the small wire that splits off toward the resistor. They are touching, or contiguous, so they are considered a single net. Each contiguous group is a net.

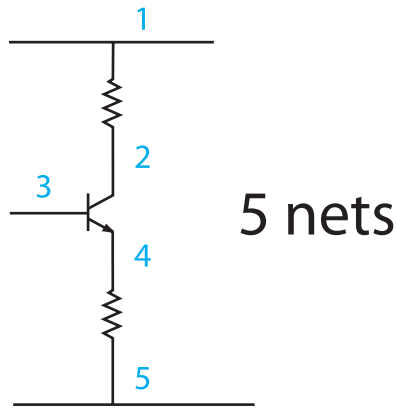


Figure 10–14. *Five separate trails of wire. Count 'em. Five.*

Check the net count on your LVS output. Let's say that the layout appears to have six nets, but we know the schematic has five. LVS reports the discrepancy. Look at Figure 10–15. The problem could be caused by something like this.

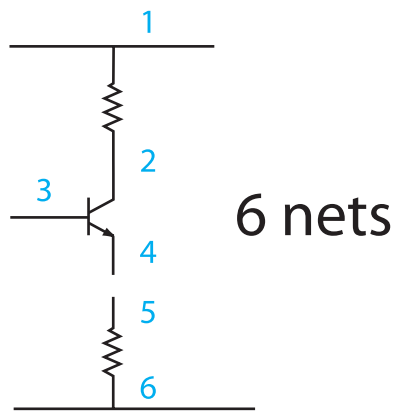


Figure 10–15. *There are indeed six nets here. Oops.*

Notice we truly have laid out six nets. LVS never lies.[3] We didn't mean to lay out six nets. Apparently, we made a mistake. If you have more nets in your layout than you have in your schematic, that usually indicates that you have an open circuit somewhere, as in this example.

An accidental open circuit could be caused by something as simple as forgetting to put a via between two metal layers. Or perhaps you just forgot a wire. Or you might suffer from a pull-apart caused by drawing off-grid. Whatever the cause, at least you now know what to look for, thanks to the LVS.

Here is another related condition. Sometimes the numbers go the other way. In Figure 10–16, our layout only has four nets, though our schematic has five.

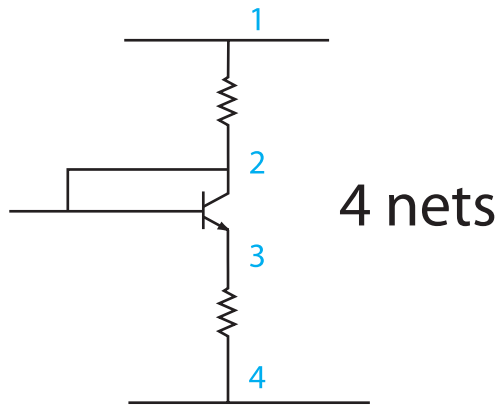


Figure 10–16. *Shorting will reduce number of nets.*

Notice the extra wire that shorts the two nets that were previously called 2 and 3. Since they are now one contiguous trail of metal, the two nets and the extra wire are all called net 2. We have taken two nets and made them into one net. Remove the short, and you will again have two nets.

Now, the LVS is not telling you where to find this short, but it is at least telling you what to look for. That helps.

Rule of Thumb: Check your net counts.

Again, checking your net counts can give you an indication of whether you have an open circuit or a short circuit. And, as with other problems, fixing one short or open circuit often clears a lot of other indicated problems at the same time.

[3] It exaggerates, but it doesn't lie.

It's quite common for companies to hire people just to problem-solve the DRC and LVS outputs. That is their whole job.

If you like the logic of searching for these errors based on the clues from rules checking files, you might want to look into turning your career toward this area. It is a great, rewarding challenge to write the files as well as problem-solve the outputs.

Give it some thought. Life is long. You can go anywhere with this. Work in the area you love the most, that's what's important.

4. Solving Complex Net Problems

LVS programs confuse easily. On a big circuit, you may end up with a never-ending LVS output that seems completely daunting. Pages and pages and pages of what look like gobbledygook. Certainly, you haven't done that much wrong.

These pages and pages may actually be only a few simple items. Where do you start? How do you find them in the midst of so much jumble?

You have looked at your component counts and types, and they are correct. No problem there. That was a good move, to check components first.

You have looked at your net counts, and they are kind of wacky, so you know there is something wrong. It may well be a mixture of shorts and opens, but nothing that seems as obvious as our simple example in the last section. With reams of moaning from the LVS output, where do you start sorting out complex net issues?

a. Power Supplies

The best place to start is to look at your positive and negative power supplies. If you don't have your power supplies connected correctly, it can cause errors. And a lot of them, believe me.

Rule of Thumb: Work on the power supplies first.

Moreover, power supplies are fairly obvious—they are big. You know where they are. That makes them easier to check.

Most LVS programs not only give you net counts, but give you the number of components it finds on each particular net. This will help your problem solving. For example, in Figure 10–17, the VCC net has three components on it, and the VCC2 net has two components.

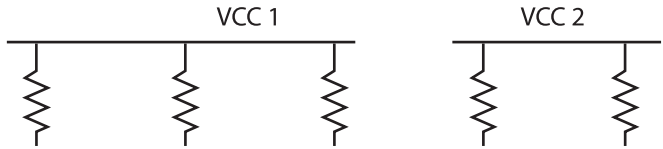


Figure 10–17. *Intended connection scheme.*

But, instead of wiring as indicated, Figure 10–18 shows a common mistake. We have the correct number of resistors and all connected to a VCC. No problem there, but LVS lights up with error remarks. Someone has hooked one of the resistors to the wrong VCC.

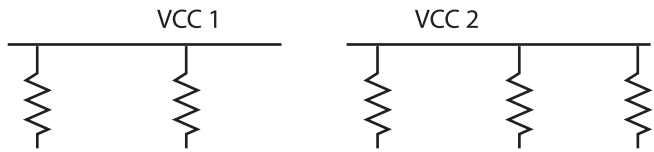


Figure 10–18. *Common mistake—component on the wrong power rail.*

A mask designer might think their randomly chosen VCC connection is fine, because, what the heck, they are both VCC's. "Ah, here's a VCC, I'll just connect my resistor here."

However, it's the wrong power net. VCC1 is supposed to have three resistors. But as laid out, it has only two. Start with the VCC's and the grounds. Make sure you have the right circuits on the right supplies. It's a backbone to straighten before going into other LVS net issues.

Another big mistake people make is connecting the power and ground backward. On large layouts especially, it is very easy to wire one block out of ten back to front. Make sure all your blocks are wired to VCC and GND correctly.

Concentrate on your supplies first. Get your supplies right. Look at the number of components that are on the supply nets. Check through them. As you repair little errors here and there, LVS items that you didn't even look at will start to disappear. You fix one problem, and fifteen problems suddenly go away.

The LVS program can get so confused by these types of mistakes that it starts to throw out just balderdash. Don't let it bother you.

When you get an unwieldy amount of errors, it's like climbing a mountain. Don't try to tackle the whole mountain in one day, just whittle away at the easy

things, one small bit at a time. Small goals. Start with the power structures and often that sorts out a bulk of the problems.[4]

b. Named Nets

Let's assume we have checked and repaired our device counts and types, and fixed all our power supply nets, but our net count and other LVS output still look wacky.

In a big chip, you may have thousands of nets and thousands of weird way-out-there components. Where do you next look for the problems? What would be the next easiest issue to resolve?

Let's keep looking at our nets. If you are lucky, some of those nets that have issues on them will be what I call a **named net.** By this I mean there might be some nets in your LVS report that are identified by name. Examples of named nets could be Bias 4, Output B, or so on.

In the schematic shown in Figure 10–19, someone has named the inputs, the outputs, and the bias node. So if your LVS output shows problems with any of these named nets, then you can be glad that you know exactly where they are.

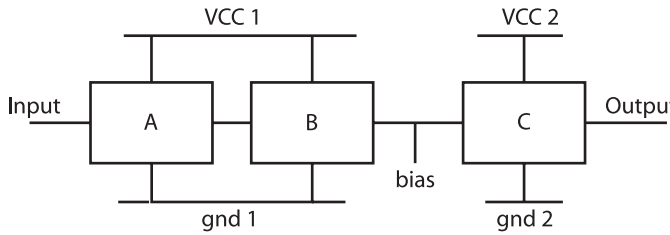


Figure 10–19. *If you have an LVS problem with a named net, you know exactly where to look.*

Work on the named nets first because you know where they are.

Rule of Thumb: Work on the named nets before other nets.

Of course, it's impossible to work on all the named nets at the same time. Often there are just too many. So, work on the smaller named nets first. Smaller nets are easier to trace.

[4] That reminds me of the instructions to untangle a marionette. The tangle can seem daunting, but it's amazing. If you untangle one string, you usually don't have to untangle any others. They sort out in the process. P.S. If you have never helped your kids build a marionette, there's your project for next weekend.

In fixing a few problems, you will likely knock out many of the other LVS issues, as we mentioned before. Based on previous misguided information the program might recognize a whole bunch of circuitry and say, "Ok, I think all that stuff is Block B," but you might actually have some signals swapped over and it isn't Block B at all. Here come a thousand LVS errors just from that one mistake. So, working on the smallest nets should bring you to your solutions more quickly. They may be the same solutions you would have found by working on the more complex nets anyway.

Rule of Thumb Corollary: If there are lots of named nets, find the smallest named net and work on that one.

Here is an example. Let's say your LVS printout is like the chart below. You have two nets, called Input 1 and Input 2. According to the schematic, Input 1 has 607 components, while Input 2 only has 4 components. Since we seem to have fewer components in our layout, it looks like we have failed to wire components in each net. Which net would you rather trace, Input 1 or Input 2?

LAYOUT		SCHEMATIC
INPUT1	599 components	607
INPUT2	3 components	4

Methinks that would be Input 2. It's much smaller. It would be much easier to find. Work on Input 2 first. Then see how many Input 1 issues resolved sort of magically by themselves as a result. Some of the problems in Input 1 circuit could be caused by the fact that you didn't have Input 2 wired correctly.

5. Don't Trust Your Circuit Designer

Let's assume we have fixed all our devices. We have fixed all our VCC's and ground. We have fixed all the problems on our named nets. However, we are still getting some weird stuff going on. We are still not LVS clean. It still looks like gobbledygook. (Are you sitting down? Good.) There is a good possibility that some of those nice circuit designer people may have changed something on you and not told you. (Gasp!)

Rule of Thumb: Ask about secret schematic changes.

If you are convinced that you have everything hooked up correctly, then go ask the circuit designer, "Did you change that circuit? I'm getting some LVS issues in the middle of a block that I had LVS clean. Did you change it?" They can change their schematic and not tell you. It's a sad fact.

You are using a live schematic, not a static, finished product sealed for your eyes only. You may have laid out a cell two weeks back. It's been finished all

this time. However, the circuit designer has just this morning found a problem in his design. He has gone back into the original schematic and made a change without telling you.

It can take two or three days of your time to find out that the circuit designer has hooked up something differently than you were told. Meanwhile, you're off checking all these other things, never even considering that your previously clean cell might not still be a clean cell. It's a frustratingly common occurrence.

Some companies have a lot of rigidity that helps control this problem. For example, once a circuit is released, circuit designers have to fill out an Engineering Change Form, have it signed, go through a Change Request Review Board, and so on, in order to update a schematic. At least you would be in the loop. But that all sounds cumbersome, doesn't it?

Design tools are starting to take care of this issue. They are getting smarter with every new version. Particularly tools that are driven by the schematic. When you fire up the layout, the tool goes off and checks the time stamps of the layout and schematic files. It will warn you, "Oy, mate. The schematic is changed, I'd look at this if I were you." But that's only a recent development in the last few years. Not everyone has that feature yet.

If you are getting some weird and wacky LVS output, you may have to go back and re-LVS some of your lower level subcells in order to check what's going on. Some cells may have changed on you.

6. Check for Possible Swapping Over

You may get a schematic like Figure 10–20. You have three blocks. Each block is LVS clean. However, all the lights flash and bells ring when you try to LVS the whole schematic. You have done the usual—device checks, power nets checks, named net checks. Hmm. Can't find the problem.

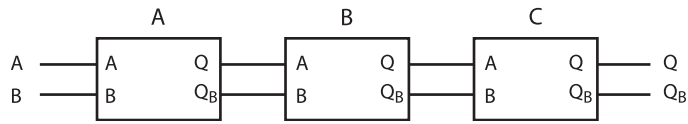


Figure 10–20. *But each block is LVS clean! Why won't the whole schematic come out clean?*

So, in the spirit of thinking a cell may have been changed on you, you re-LVS everything. It turns out all your subcells are LVS clean. Well, with the subcells clean, at least that tells you where the error in your layout is. It's not in the lower levels. It is in the current level of hierarchy.

Here is a common mistake for you to check next. Particularly in a differential circuit, because the paths are so similar, you may well have put two text identifiers back to front, swapped them. When you named the nodes in your layout, there is a possibility that even though your layout is wired correctly, you may have placed the text on the outputs backward.

All the LVS nets and output information point to something wrong in Block B. However, Block B checks OK. That's usually an indication that you have something swapped.

Two things could have created a swapping over problem. Either wires are crossed or labels are crossed.

For instance, in Figure 10–20, again, either the outputs of block A to the inputs of block B have been flipped somewhere, or the outputs of block B to inputs of block C were flipped somewhere. Those are the wire swapping possibilities.

Or, you could have swapped labels. That would be either the labels on the outputs of C or the labels on the inputs of A.

Figure 10–21 is what your layout might look like. Look closely.

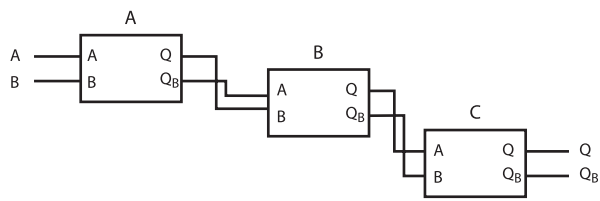


Figure 10–21. *Look closely.*

Between blocks A and B, you have connected Q output to B input, whereas our schematic says to connect from Q to A. It's very easy, particularly with differential wires, to swap something over. The error occurs between blocks A and B, though especially in large circuits, this can be very difficult to find.

You might even have a double whammy. You may have swapped wires between the first two blocks, then also swapped the wires between the next two blocks. Who knows what's going on in that case?

Here is another problem that could happen, and this happened to me. You may have wired it correctly, but the person who put the schematic together may have missed what was going on. They may have absent-mindedly looked at the pins and wanted to wire from Q to A, but actually swapped over one of their sets of symbols. A for B.

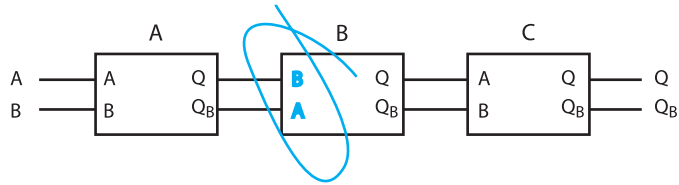


Figure 10–22. *Swapped labels are a common mistake on complex schematics.*

A good quick way to find out if you have swapped something over, particularly in a differential system, is just do some what-ifs. Exchange the A and B wires. See if that makes any difference. Take a gamble. Swap one set and see if anything clears up. If it gets worse, put it back and try swapping another set. If it gets better, you know something is swapped over incorrectly.

Rule of Thumb: If you can't find it, try some *what-if* swapping around.

7. Check for a Top Level Short

There is one other possibility. If you are getting LVS issues inside a clean block, it could be that you have a top level wire shorting to the layout.

If it's a differential circuit, my money is on the swapping problem mentioned in the previous section. But, a top level short is one more item you might look for.

8. Check for Ninja Invisibility

We will call this the ninja problem with schematics. You can't easily see it. It seems to be invisible. This problem can easily surface if you have a schematic that was drawn to look very, very nice and orderly. Schematics can look so nice that errors become difficult to spot.

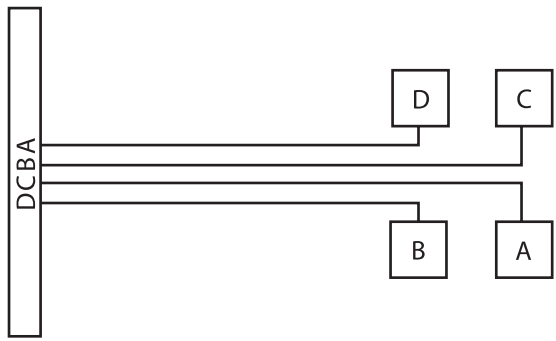


Figure 10–23. *Nice symmetry and organization can disguise an error. DCBA to DCBA, what's the problem?*

The schematic is drawn to look symmetrical and ordered. As a mask designer, you see that you need four blocks: A, B, C, D. And you see your four pins going to those blocks, labeled as expected: A, B, C, D. All appears orderly. You look elsewhere for the LVS problems, leaving this nicely ordered segment alone. But, too much order can create optical illusions.[5]

Signal A is supposed to go to pin A, B to B, C to C, and D to D. In this simple example, you can see the error without my telling you, I'm sure. D goes to A, C goes to B, B goes to C, and A goes to D. The pin order on the leftmost symbol is incorrect.

Now, of course, we are really talking about large, intricate schematics, where the input labels would be miles away from the output labels. This problem could go unnoticed for the longest time. Every time you check the input, you see A, B, C, D. Fine. Every time you check the output, you see A, B, C, D. Fine. It looks so symmetrical and orderly, that you may not catch the mistake in a large schematic for days.

9. Know Your Circuits

Sometimes your layout can find problems with the schematic. If you have laid out a particular circuit many times before, you will become very familiar with that kind of circuit. You may just make the components, glance at the schematic quickly, and think to yourself, "Sure, I know how to wire this us up. Done it a thousand times." And off you go. You wire it almost from memory.

However, when you run the LVS, you could find issues that result from a problem with the schematic. Your layout was correct, but the schematic was wrong. The error was discovered because you did not follow the schematic.

- **Rule of Thumb: Don't assume the schematic is correct, particularly if you understand the circuit quite well.**

- **Flipside Rule of Thumb: Understanding your circuit well can give you insight into LVS problems.**

Understanding the system, what the whole circuit is trying to do, can help you eliminate certain possibilities in your LVS work. This is our flipside rule of thumb to the previous rule. For example, if you see a whole bunch of similar net names such as bias 6, bias 7, bias 8 in your LVS output, that will tell you the problem is where all the biasing signals are, as opposed to somewhere out in Timbuktu. So you find your biasing circuits, and begin to work there. Learn about your circuits.

[5] I think it was Edgar Allen Poe who said the best place to hide something is in the obvious.

10. Let Others Help

Well, here we are. You have run through all these tips and you still can't find the problem. The only thought left in your head is the sound of the woodpecker outside. You have struggled for days and days and days and days trying to solve your LVS issues. After awhile, you just can't see the woods for the trees. It's time to get up out of your chair.

The final tip: Get a second pair of eyes. Have someone else look through your work.

It is surprising how many times someone comes along and says, "Oh, yeah, I see it here. You swapped those two wires over." You have been staring at the screen for a day and a half, and here some smart aleck just saunters along and ruins your afternoon. But, it gets you going.

So, talk to your peers. Have someone look through it. Other people's experience can point you in the right direction.

In fact, someone may have seen a particular LVS error reported in a certain way that you may not have encountered before. They look at your output from the LVS program and say, "Well, the last time I saw something that looked like that, it was because I hadn't hooked up a substrate contact correctly." So go look at your substrate contacts, see if they are hooked up correctly.

Rule of Thumb: Use all your resources—including other people.

Closure on Verification

Each LVS tool outputs its information in different ways. Some will output text files. Some will overlay graphical information directly on your layouts. Some will give you pointers as to where it thinks the issues are located. You might work with any of these output formats, but the information in this chapter will still direct you, in any case.

The only real way to become good at solving LVS problems is just using the tool a lot. Lots of experience. Lots of problems solved. Lots of frustrating hang-ups overcome. Lots of listening to others. Lots of learning.

The more you understand how the schematics are put together and how the tools work, the easier it all becomes. You will start to see patterns emerging. For instance, your LVS will always report a short circuit in a certain way. Soon you will be able to glance through pages and pages of LVS output and instantly spot what is pertinent. That ability takes time.

When you begin a new LVS task, it can seem very daunting. Just take it a small step at a time. Be logical. Be thorough. Take it slow. Make sure you know where you are, and eventually you will get there.

It's Sherlock Holmes. Once you have eliminated all the impossibilities, what remains is the answer. So you can ask yourself, "What can it not be? I've LVS'd that cell so it can't be in there. The LVS output is telling me all the things are hooked up right, so it's not that." Keep eliminating.

Besides being logical and using deduction, play with what-ifs. Just toy with it. For instance, "What happens if I swap this and that wire?" Do it. Swap them. See if things get worse or better.

Sometimes the errors can be very subtle. Just two wires out of 5000 might be swapped, or just one via missing in a vast sea of circuitry. Something even that simple could take you two or three days to track. Just be methodical.

If everything fails, have some pizza, play some loud rock and roll, go out for a laugh with your friends and family, and look at it again tomorrow. Relax. You'll find it. And when you are finally DRC and LVS clean, celebrate by having some pizza, play some loud rock and roll, and go out for a laugh with your friends and family.

Here's What We've Learned

Here's what you saw in this chapter:

- Common errors in layout
- Using the Design Rule Checking program (DRC)
- Using the Layout Versus Schematic program (LVS)
- Using Boolean functions to identify components
- Error correction
- How to read and write basic DRC and LVS code
- The advantages of knowing DRC and LVS coding
- Interpreting netlists
- Step-by-step procedure for difficult LVS problem solving

CHAPTER 11

Data Formats

Chapter Preview

Here's what you're going to see in this chapter:

- How to prepare your final data for the mask-makers
- Standard formats most people use
- How to communicate units along with the numbers
- Why you suddenly have a drawing 200 times too big
- Why your connections suddenly disconnect in translation
- What really determines your drawing resolution
- Some methods and new technology that help with these problems
- How your data points finally become lines and shapes
- Why you should understand data formats

Opening Thoughts on Data Formats

We have finally drawn every last piece of our layout. We have run our DRC's and our LVS's. We've done our chip size checks with bonds and everything. We spent months and months, burned the candles at several ends. We are now at the point where we can say we are finished. It's ready to go. It's done.

Depending on the tool we have been using, we now typically convert our layout database into a format that the mask-making house can read. We do not ship out our tool data as is.

Industry Standard Database Formats

There are two prevalent data formats you could use. One is called **CIF, CalTech Intermediate Format.** CIF is not used as much as the more popular format, called **GDSII Stream Format.** GDSII was originally invented back in the late 70's, early 80's by the Calma Corporation, now owned by the **Cadence Corporation.** GDSII is the industry standard database format for mask design information.

Almost every tool out there can read and write GDSII. A GDSII stream file is one, large, self-contained file. It contains everything—every library and every cell. It has all your layout information in it. It even maintains the hierarchy of your design, so when you read a stream file back into your tool you will see all the cells and layer information just as you designed them.

Header Information

The GDSII format includes a header at the start of the file. This section maintains information about the data in the file. For example, the header will state whether we are working in centimeters, inches, meters, or miles.

You could be combining files from multiple sources, for example. I might be working in microns and someone else is working in thousandths of inches, or mils. So this part of the header of the GDSII stream keeps everyone synchronized in the same units.[1] These references are known as user units. User units are typically specified in microns for mask design.

Coordinating Resolutions

The next very important piece of information the stream file contains is the number of database units per user unit. That piece of information effectively sets the resolution of the database. If you have a user unit of one micron, and a database unit of 1000, then you are allowed 1000 divisions of each micron. Therefore, the minimum dimension in that database is one thousandth of a micron.

Now, one thousandth of a micron is pretty small. Most semiconductor processes do not resolve to that fine of a resolution. The industry is starting to have device dimensions like 0.12 of a micron, and slightly smaller, so you certainly need resolutions in the tenths or hundredths of a micron, but not one thousandth of a micron. Nevertheless, the industry standard resolution is one thousandth of a micron.

[1] We wouldn't want rockets smashing into Mars, would we?

The problem with this resolution is that most technologies do not want you to draw to a thousandth of a micron resolution. They only want resolutions of maybe 0.1 of a micron. The e-beam gun that sketches the actual masks has a footprint that does not change size. The size of the beam is called the **spot size.** This size drives our resolutions, all the way back up to our layout level.

If you set the resolution of your tool incorrectly, you could digitize on a thousandth of a micron—i.e., the resolution of the database. However, if you are only allowed to digitize on a 0.2-micron grid, you will have a few lines on the digitization grid, then lines that are off-grid, then maybe it will go back on-grid again at some point. What you are drawing has a finer resolution that does not exist on the manufacturer's grid. It just cannot be beamed that small.

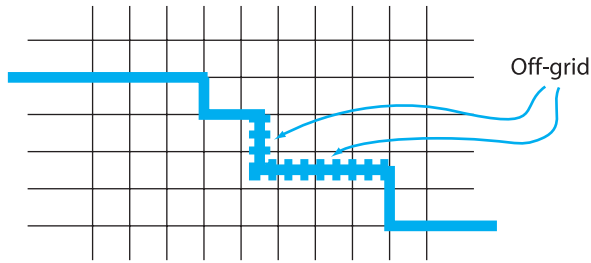


Figure 11–1. *Two segments of the fine-grid trace are not placed on the coarser grid.*

In Figure 11–1, we should only draw our polygons to the coarser 0.2 microns resolution, but our CAD resolution, being much finer than that, allows us to draw off our 0.2 resolution. When we output to GDSII, the resolution of the file maintains our finer numbers. However, the mask making machine must relocate our off-grid lines to the nearest manufacturing gridline. The electron beam only travels in increments of 0.2, in this example. It will not draw finer than that.

If we draw using finer detail, we will not know which 0.2-micron multiple the mask making equipment will choose. It will want to jump to the nearest 0.2-micron grid either before or after what we specified. We end up with a polygon that does not necessarily look like our drawing. We have a rounding error.

Although your database says you are 100% design rule clean, you could lose information on your mask if you are not careful about your units. You might end up with issues on the real silicon that the tool couldn't tell you about. The tool cannot know whether you are drawing to the final manufacturing grid size or how the data will later be converted. You sometimes see a grid-checking program as part of the DRC, but it's not standard.

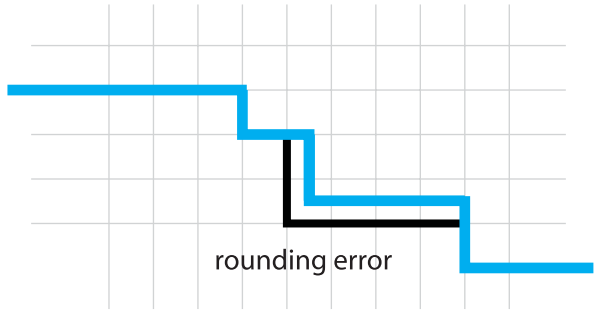


Figure 11–2. *Our off-grid segments are defaulted to the nearest grid-line, creating an error. Sometimes even these small errors can kill a chip.*

Force-fitting off-grid polygons into a coarser grid can cause what is called **shape pull-apart.** Our plan in Figure 11–3 is drawn to a finer grid scale than can be manufactured. Notice the join of these two metals is butted together off-grid.

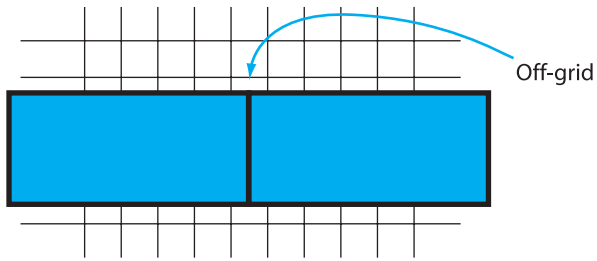


Figure 11–3. *Our shapes are butted together, but off-grid. They won't be able to go that far if the program rounds to the inside gridline.*

Now what's a mask making machine supposed to do with that? Each off-grid edge will have to find a gridline somewhere. Due to the rounding, the off-grid edge on the left might jump backward to find a gridline, and the off-grid edge on the right might jump backward to a nearby gridline as well. This leaves a gap between the shapes on our mask.

If this happens, it may cause an open circuit. Or, it could cause a thinning of a metal in that region. It's very dangerous in any case.

You should understand the relationship between the user units and the database units. Know the importance of what we call the **digitization grid.** Make sure you only digitize on the grid that is available for you at the manufacturing level.

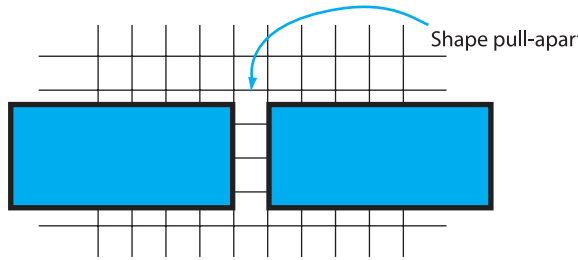


Figure 11–4. *Oops, just as we feared. The shapes have been rounded back to their previous gridlines, pulling our contiguous strip apart.*

The processing technology people will tell you the resolution for your manufacturing. It should be set up in the tool for you, if you're lucky.

However, with most tools you are given the option of going in and changing your drawing grid. If you change your drawing grid to zero, for instance, there is a good chance you will be drawing polygons that are completely off-grid. Most newer DRC's check for this.

Now, some people, because of the header in the GDSII file, say, "Ok, well, I don't want to run into problems, so instead of 1000 database units per micron, I'll only have 5 database units per micron." The minimum resolution in their database is now the same as their minimum manufacturing grid. This way, they can never place a polygon off their e-beam grid.

But, most tools expect 1000 database units per micron. If you convert a GDSII file that has a non-standard resolution, the conversion program might get confused. The conversion routine could be hard-coded for 1000 units, and will not even read the altered information in the header. So, you could create something that is 200 times too big when you convert it into another tool.

Pattern Generation

Once you have created your GDSII file, bearing in mind the digitization grid, the data usually goes off to a mask-making house somewhere. Sometimes the mask-making house will need a specific data format other than GDSII, so you have to convert into a format they can use.

You normally have to convert your GDSII files into a **PG Format,** a **Pattern Generation Format.** You might see some companies refer to data being "released to PG," or hear them say they are "PG'ing the data." That means you are pattern generating the data, getting it ready for the mask-making house.

Pattern generation converts your vector-based data, which contains coordinates of vectors, to a raster-based format, which contains horizontal stripes that make up the polygon.

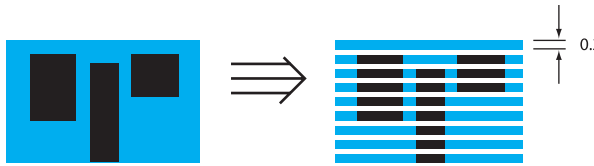


Figure 11–5. *Patterns are created using narrow strips stacked on top of each other. The strips are created one at a time by the traveling e-beam.*

Each of these stripes is the size of the electron beam that rasterizes your information. In our example, this would be 0.2 microns wide, one gridline. So the size of the electron beam determines the size of the stripes, which determines the digitization resolution that you need in your GDS file, which determines the grid in your tool at which you can draw.

A rasterscan operates like a television cathode ray tube. You turn the beam on and off as it passes by each horizontal line on the screen, to expose the pattern.

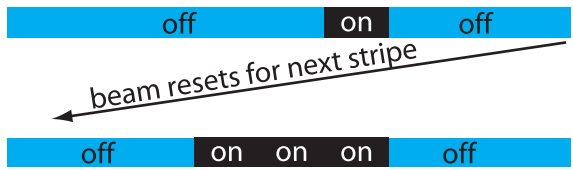


Figure 11–6. *Turning the electron beam on and off as it travels from left to right creates the entire pattern, one row at a time.*

Know Your Grids

Having off-grid data is a very common mistake for inexperienced mask designers. They often digitize off-grid because when they change the resolution of their tool to achieve a certain function, they often forget to reset the resolution back to their process grid. When they run their first DRC, everything lights up off-grid and they say, “oh no,” or something else that generally means oh no.

Even if you have cells that are pre-made for you, parameterized cells, there is still no guarantee. All the polygons might be on-grid within the grid of the cell, but not on-grid relative to the absolute grid. It is possible a mask designer

might have to go back and do another week's worth of work just to put everything back on grid.

Another time grid size pops up is when you are asked to convert from one technology to another. "Here's a piece of CMOS layout we did and we want to convert it from our 0.8-micron CMOS process to a 0.75-micron process." The new digitization grids might be very different, or even slightly different, so you cannot just nonchalantly copy the data straight across. You have to place every item back on grid. All the design rules might change. It takes a lot of time, a lot of care. It's a big deal.

This could also happen if the e-beam size changes. The change works its way back through all these data formats to drive a change in your layout. Grids would have to change all the way up the line.

Closure on Data Formats

Understand where your data is going in the end, and the format that will be used. Ask about manufacturing e-beam resolutions.

Verifying resolutions before you start your layout could save you a lot of grief. There are better things to do than relocate every piece of layout back onto grid.

Here's What We've Learned

Here's what you saw in this chapter:

- Data conversion protocol
- Industry standard database file formats
- GDSII file headers
- Hard-wired versus adjustable grid size options
- Prevention of shape pull-apart
- How the electronic manufacturing beam sets your grid size
- Options with your grid sizing issues
- Rasterization
- Why you should understand data formats

CASE STUDIES

Following are two layout assignments using different processes. The first Study uses a CMOS process. The second Study uses a Bipolar process. In addition to illustrating the unique characteristics of the two processes, each Case also emphasizes unique working skills.

In Case Study 1, we see story characters role modeling for us. They will learn as we learn. As you will see, the real learning in Case Study 1 happens when the circuit designer returns from vacation and looks at Bill's good intentions. We include dialogue and thought process, so you can see how a mask designer might communicate with the circuit designer, think through the task, listen to helpful criticism, and move on to design a spectacular layout. We hope the first Case Study particularly shows that layout is a team effort.

The second Case Study illustrates how your circuit requirements can alter your layout decisions, particularly RF requirements. We hope to show that there are many correct ways to complete a layout assignment, with trade-off decisions that must be made. We will show you how a Bipolar mixer layout changes as we consider increasingly demanding circuit requirements.

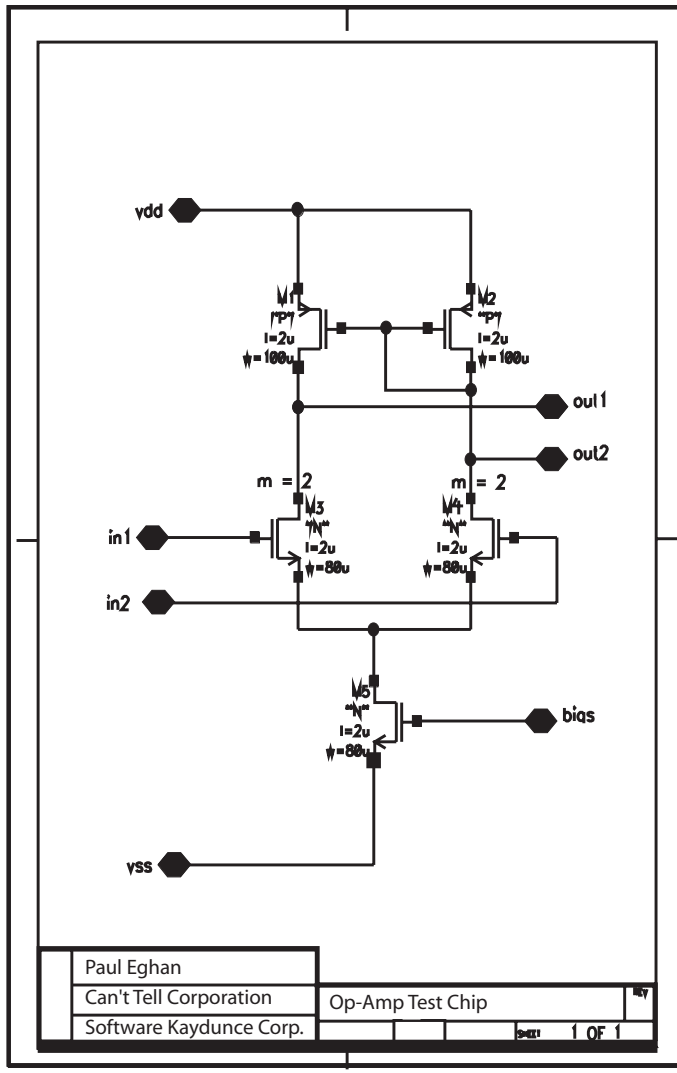
Read each Study once to see where it is going. Then, although each Study is written to be understood on the first reading, read the study again. Each Case Study will become more informative to you as you become increasingly familiar with it. The Cases are for your repeated study, not just one reading. You will be surprised how you pick up additional perspectives and information each time you go through them.

Instead of using colors to represent the unique layers, as in your CAD tool, we will use our blue highlight color to indicate new items in the drawing that are referred to in the surrounding text. Once we no longer are referring to the item, it will thereafter be shown as background gray, regardless of layer.

We provide a reference schematic at the front of each Case Study. We recommend you repeatedly refer to the schematic as you read through each Study.

Here's our first Case. Let's Study it.

Refer to the following schematic throughout Case Study 1.



CASE STUDY 1

CASE STUDY 1

CMOS Amplifier

Make a copy of the preceding schematic to place in front of you as you read along with this Case Study. Read through the Study once for general perspective. Read it again for analysis and better understanding. Read it again and again to increase your level of understanding and your internalization of these good practices.

Enjoy.

The New Job Assignment

Bill is a mask designer. He's been with the Can't Tell Corporation for five years generating CMOS input/output cell layouts. That's all he's been doing—just I/O cells for five years.

He is rather bored with generating I/O cells all the time. His manager refuses to put him on anything he finds interesting, so he begins to look through the corporate job postings. He sees that there is an opening for a mask designer in a new department being formed at the same company site, but three buildings away.

Bill wanders down to Building Four and learns more about the job. He realizes the new position will be very challenging for him. He applies for the job, goes through the interview process, and eventually is transferred into this new department.

Because the department is new, Bill is their first mask designer. The circuit designers are very pleased that they finally have a mask designer to work with. Up until now they have been doing their own layout and have not really been enjoying the work. Consequently, they are keen to get someone who knows how to run the tools, someone who knows all the subtleties and nuances of mask design.

Bill's first assignment is to work with Ted. Ted is a circuit designer with around 20 years of experience.

Rufus McNab, the manager of the new department, introduces Bill to all the department members. When Bill meets Ted, Ted says, "Hey, at last, a mask designer! We're real pleased to see you."

Bill says, "Thanks, it's good to be here. I'm excited to take on this new responsibility."

Ted adds, "Well, we've got good news and we've got bad news. The good news is we have a nice, simple chip for you to start with. The bad news is I'm going on vacation tomorrow. We'll get you started, but you'll be on your own for a week or so."

"Ok, that's no problem," says Bill. "We should get together and start talking about the design as soon as possible."

"Yes," says Ted, "How about 2:00 this afternoon?"

"Excellent," says Bill.

For the remainder of the morning, Bill looks through the manuals for the new process he will be working with. He is pleased to see that the Can't Tell Corporation is very consistent with their processing development. This process is very similar to what he has been using for the last five years. There are some subtle differences, but overall, it is a plain CMOS process like the one he is used to using.

Two o'clock comes around, and the meeting begins.

"Hi, Bill," greets Ted. "The project we have for you is a tiny test chip we're working on. We've already done some preliminary work for you. We know the package that it will be going in, and we have a rough idea of the bond-out. So, your job should be pretty straightforward.

"Here's the bond-out we have for you," Ted continues. "As you can see, it's in a 24-pin package. There are basically four op-amps on one chip, each with its own separate supply, but with a common ground, and a separate, but common, biasing pin. So, you should be able to just lay out one quarter of the chip and repeat that three more times."

"It looks pretty straightforward," says Bill.

"Yeah, it's a pretty straightforward chip."

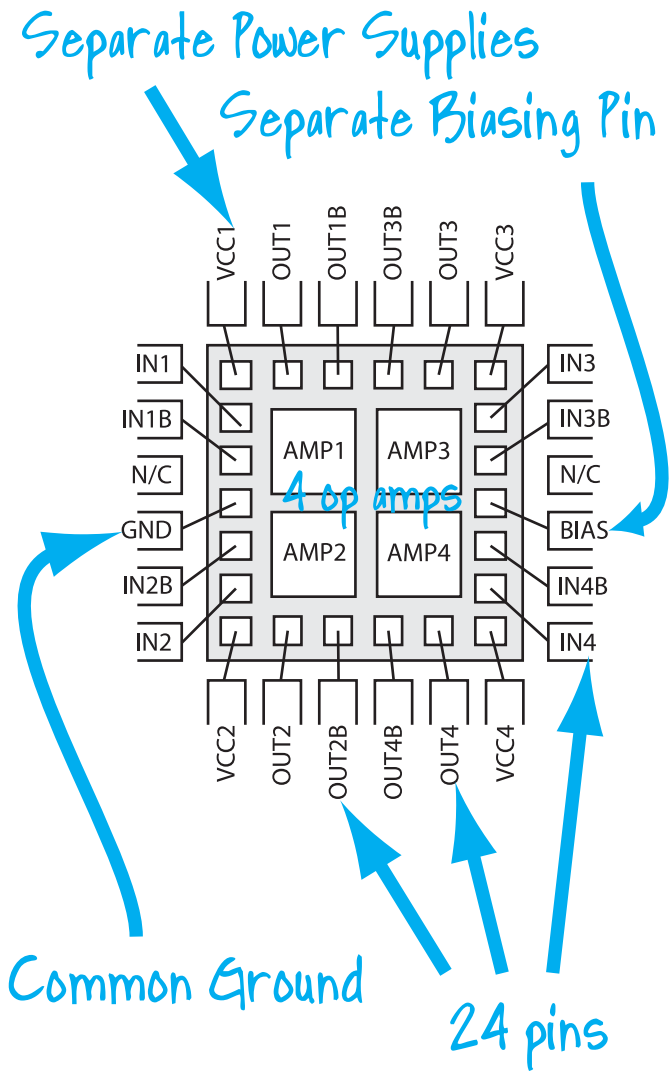


Figure CS1–1. *Bond-out design using a 24-pin package.*

"Do you have the schematics?" asks Bill.

"Yup, sure do," Ted replies.

"Oh, wow, that looks real easy," comments Bill. "What does it do?"[1]

[1] This is Golden Question #1.

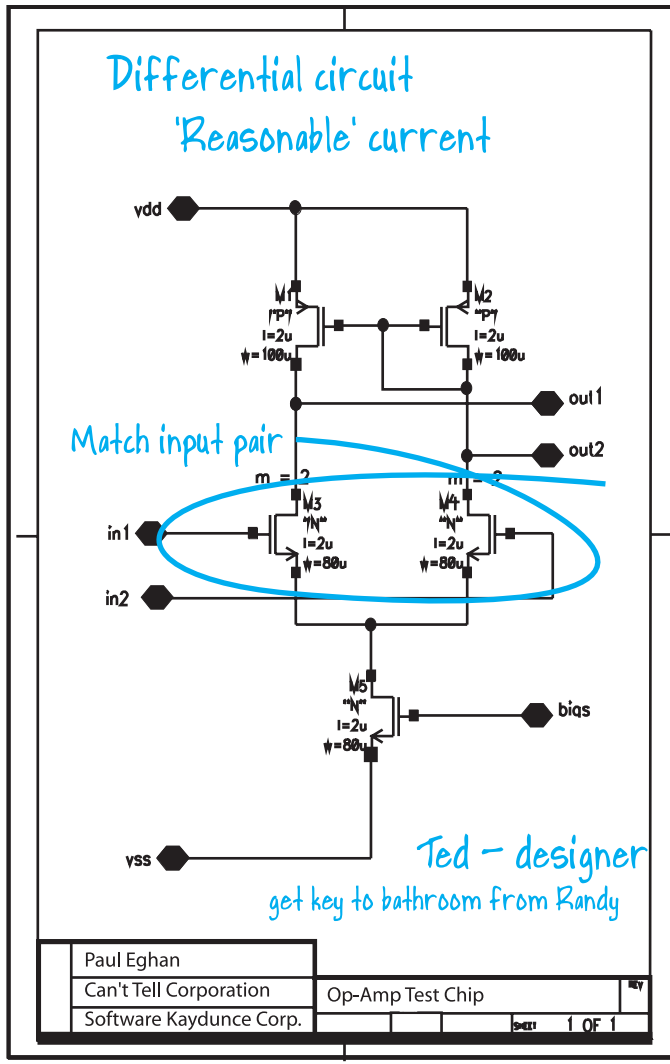


Figure CS1–2. *Schematic with Bill's notes.*

"Well, it's a differential op-amp, but we're not bothering to include on-chip biasing because we want to be able to tweak the biasing voltages externally. That's why we have a separate pin. We just want to see how well the process does and what it can do. You know, just check it out. It's a new process."

"Yes," interjects Bill, "I've read the manuals." Ted nods.

"How much current does this op-amp take?" asks Bill.[2] Bill is jotting notes directly on his copy of the schematic as he talks.

"Well, it's a reasonable amount of current. It's not wonderful, but there isn't anything you really need to worry about."

"Oh, good. Sounds simple. Is there anything with matching that I need to worry about?" asks Bill.[3]

"Well, yeah, the usual CMOS stuff. You know, just make sure the input pair match ok and you should be alright." Ted looks up to see if Bill has any more questions.

"Great. I'll get started on that now," concludes Bill.

Observation

Bill shows good work habits. He:

- *asks the three primary questions*
- *shows he is communicating with the circuit designer*
- *reads the manual*
- *notices the word "differential," and*
- *is taking notes*

These are all good qualities that will save him time and effort in any layout assignment.

Bill asks where the schematic file is located. Then he goes off to his desk and opens his software tool. Luckily, the Can't Tell Corporation has invested heavily in their tools, so they have the state-of-the-art tool suite from Kaydunce Corporation.

Bill is able to quickly fire up the toolset, find the schematic and automatically generate the transistors for his op-amp.

However, the first thing Bill sees are transistors that look unwieldy. Each transistor is a single-stripe transistor that is very long and skinny. Not only that, but the NMOS devices for the input pair are each split in two.

[2] This is Golden Question #2.

[3] This is Golden Question #3.

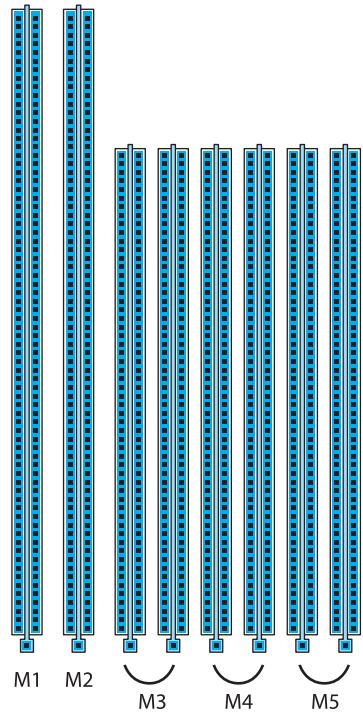


Figure CS1–3. *Long, skinny library devices.*

Now, luckily, Bill has had enough experience with I/O pad cells to realize that he needs to go find Ted. Bill will ask Ted to change his schematic in order to make this a nice, tight layout.

Before he speaks with Ted about the device sizes, Bill spends half an hour or so playing with different transistor sizing options, maintaining the effective gate width of the transistors, of course. Bill is trying to find a nice, easy way to split the transistors so they will fit in a compact layout design. Bill feels it would be nice to approach Ted with a suggested solution, instead of just appearing to complain.

Observation

- *Bill does not settle for the initial device sizes generated by the tool.*
- *He has confidence that there is a better way to make the devices.*
- *He works to find a solution before returning to his circuit designer.*
- *He is demonstrating his knowledge of basic layout technique by splitting his devices into smaller chunks.*

All of these actions are examples of good mask design practices. As a mask designer, these positive helpful actions make you a joy to work with. Develop these skills and practice them whenever you can.

Bill has missed a few pointers, however. The input devices were split in two. This is usually an indication of some matching requirement, which he will learn to regard more seriously with experience. Likewise, Bill did not ask any questions about chip size or packaging limitations. Some of these points could come back to bite Bill in the end.

In addition, Ted is distracted by his vacation plans, leading to vague and ambiguous answers for Bill. Ted needs to give his full attention to his mask designer.

After half an hour of playing with the tool and resizing the devices, Bill comes up with a set of transistor sizes that he feels will work well.

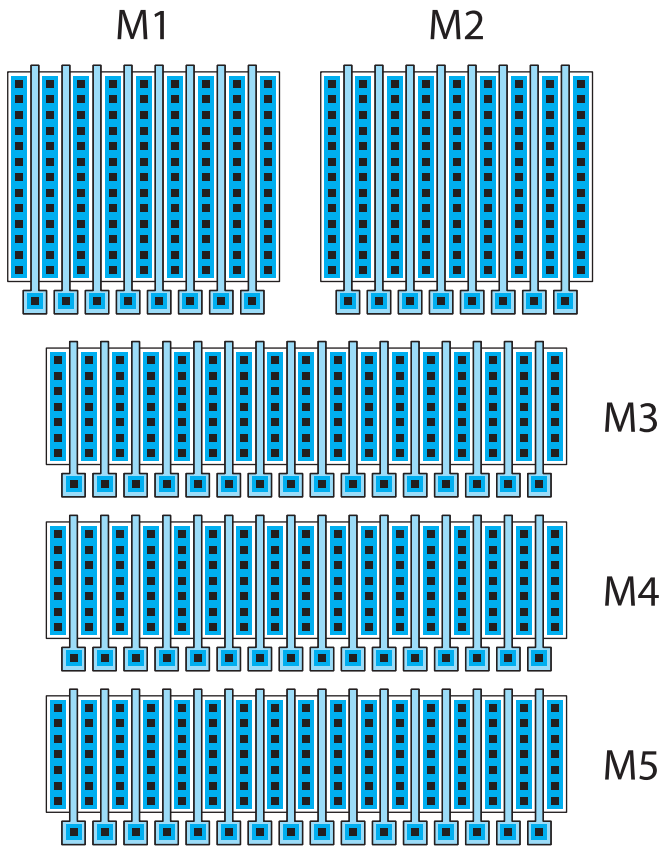


Figure CS1–4. *Library devices after reshaping.*

Confident that these device sizes will give him a nice, tight, compact layout, Bill goes back to Ted, the circuit designer. The conversation goes something like this.

"Hi, Ted. I've looked at your transistor sizes, and as far as I can tell you haven't really sized them right yet."

"Yeah, yeah," agrees Ted. "I thought you'd come back and complain about the sizes of the transistors. I haven't really finished the circuit design yet. I'm open for suggestions."

"Cool," says Bill. "I've already played with the sizes, and I have some suggestions I think you'll find acceptable. Can we fire up the tool and I'll show you?"

Bill shows Ted his proposal for the device sizes. Ted offers, "Yes, these are much better sizes than the ones I have. I see you have maintained the same effective gate width, so there shouldn't be any problem. Which ones are which?"

Bill quickly sketches out a drawing on a piece of paper and says, "Well, here's what they look like. The two big PMOS devices are roughly square at the top of the device. The input devices are fairly long and skinny, and the current source is the same size as the input pair."

"Good," says Ted. "I think that will work really well."

Then Bill adds, "Are you sure there is nothing else I should know?"

"Well, yeah," says Ted. "There is the usual current stuff and the usual matching stuff to worry about. Apart from that, there's not much, really. Everything you need is in the schematic, so you should be fine. You should be ready to go. I'll change the schematic to show the device sizes that you have here. I have to leave in about 10 minutes, but I'll do that before I go. I'll see you in a week."

"OK," says Bill.

Observation

Ted is not concentrating. He failed to notice that Bill has changed the input devices (M3 and M4) from double devices to single devices. Bill has maintained the effective combined gate width, so electrically everything is the same. However, Ted has forgotten how this will affect his matching requirements. We are starting to head into troubled waters.

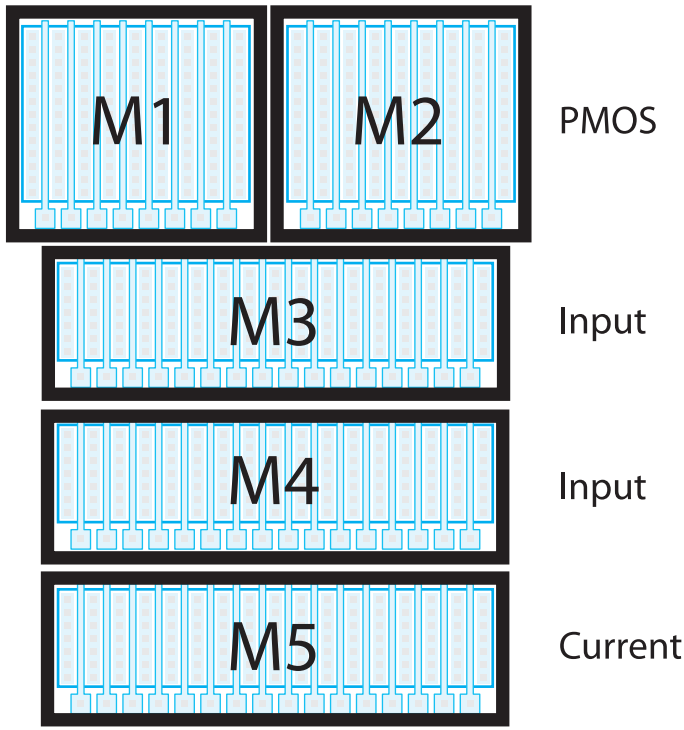


Figure CS1–5. *Bill's first floorplan idea.*

Bill Reasons His Floorplan

First thing Monday morning, Bill feels enthusiastic as he arrives at the office. He pulls up the schematic and sees that Ted has indeed changed the devices to be the right sizes. So, with great abandon he starts to work on the amplifier.

Bill first makes mental notes, remembering what Ted told him about the schematic. Ted said to think about matching and current issues. Bill also remembers the responses to his key questions, "What does it do?" "How much current does it take?" and "What are the matching requirements?" After taking mental stock of the concerns, he begins.

The first thing Bill notices is that there is no current annotated on the schematic. He has no idea how much current this amplifier is really taking.

Ted did say there was a *reasonable* amount of current in the cell, but what is reasonable? So, being the bright, enthusiastic lad that he is, Bill wanders around trying to find another circuit designer in the department who may know something about this. He knows better than to blindly throw his layout together on his own assumptions. Communicate, he was taught.

Eventually, he is put in touch with Joan, who is also working on this project. He asks her if she knows how much current is being put in the device.

Joan answers, "Well, I remember something from the design review, but I'm not really working on this project. My specialty is ESD structures. I'm making sure that components don't arc when you put too much voltage across them. But, I seem to remember it is a fairly low current circuit. Just one or two milliamps should be fine. Just size your metal to that."

"Ok," says Bill.

Bill returns to his desk and has a look through the process manual. He finds that his Metal One current density is 1 milliamp per micron. He also discovers from the manual that his minimum Metal One size is 2 microns.

He thinks to himself. "Hey, that'll do. If I can handle 1 milliamp per micron and my minimum Metal One wire width is 2 microns wide, then my wires can handle 2 milliamps. We should be fine if what Joan says is correct."

$$\frac{1\ milliamp}{1\ micron}\ (2\ microns) = 2\ milliamps$$

Bill continues thinking, "Ted didn't seem too worried about the matching of the devices, but I should probably keep them fairly close together. Let's look at the layout.

"I like the idea of having my PMOS devices nice and close to each other. Since the source-drains of M1 and M2 need to connect to the source-drains of M3 and M4, the positioning of the gates on those devices makes it tough to connect the amplifier outputs to them.

"If I rotate the PMOS devices through 90° I can have the gates connect to each other easily in the center and that will make the outputs easier to connect to M3 and M4.

"But!" Bill thinks, "Now my NMOS devices have their gates running perpendicular to the gates in the PMOS devices. I should rotate the NMOS devices also."

Bill rotates all his devices through 90° and makes sure that his PMOS gates are facing each other. Bill also realizes at this point that he should keep his input gates on his differential pair devices (M3 and M4) close to each other to maintain a good differential match. Bill's layout now looks like this.

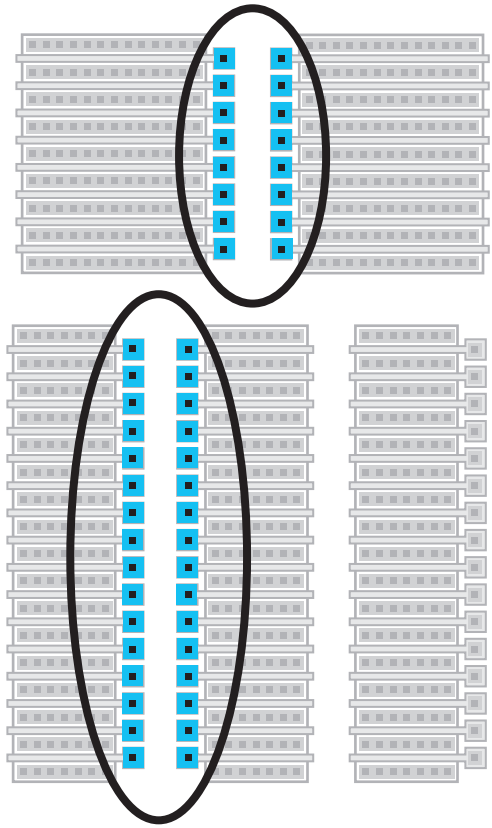


Figure CS1–6. *Rotating devices. Gate connections are close to each other to improve matching.*

Observation

Bill is reasoning well. By keeping his gate orientation consistent, he is ensuring good matching. Although there is no real matching requirement between the N and P devices, it is good practice to run all your gates the same direction. The added bonus with this approach is that you get a consistent metal direction. This is essential in very dense circuits.

The differential nature of this circuit is also being addressed by ensuring that the gate connections for the input devices are close together.

"Yeah, that looks much better," Bill reasons. "Now I have my gates for the PMOS devices all pointing to a common point in the middle, and looking at

my circuit diagram the gates need to join anyway. So that works really well." (*See schematic*.)

"My current source device (M5) can just go off to one side and connect down to ground.

"Yup. I think I like my floorplan like that. I think I'll start to wire things up."

But Bill stops for a quick thought, "Hmm. Let's have a look at the chip bond-out, to see if I can remember what Ted said about that. I should be sure my plans jibe with the pin-out."

Bill pulls up the bond-out again to have a look at it. He particularly notices where the input and output pins are located.

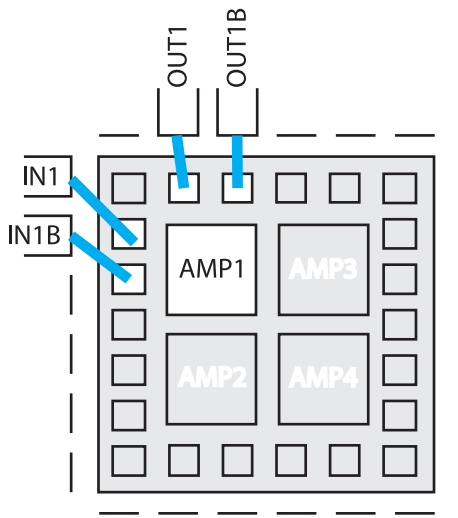


Figure CS1–7. *Inputs to the left, outputs out the top.*

"Ok, looking at just the top left corner of the bond-out Ted gave me, it looks like the inputs are coming in from the left-hand side, and the outputs are going out the top.

"So, how does that work with my floorplan?" he wonders.

Bill looks at his floorplan again.

"It could work reasonably well, provided I pull the outputs from a good common point in the center of the PMOS device, straight up to the output pads at

the top. And, I should be able to bring my diff pair[4] inputs from the pads on the left without a problem. I'll just pull them between M1 and M3, and down into the input pair gates," Bill concludes to himself.

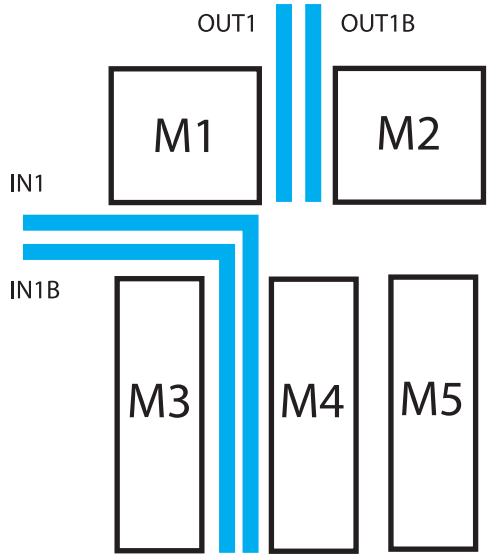


Figure CS1–8. *Bill makes sure the inputs and outputs are in the correct bond-out locations.*

"I think I have the matching Ted wants because my two input devices are real close to each other. That also takes care of the differential concerns.

"This seems to be a reasonable floorplan. Ok, I think I'm ready to start my layout."

Bill Thinks Through His Layout

Let's follow Bill's thinking as he creates his cell.

"I'll start with all the gate connections," Bill says to himself.

"I should probably hook all these gates together in metal instead of poly, just in case we want to do some revisions. Back when I was doing the I/O pad stuff, I know people liked me to hook gates up in metal. Knowing this is a test chip, we might want to do a metal spin sometime.

"Now, do I need to worry about NAC diodes on this thing? Let's have a look through the manual." Bill refers to the manual to answer his own question.

[4] diff pair—differential pair.

"Aw, yeah, ok, so I should probably plan to put those in now."

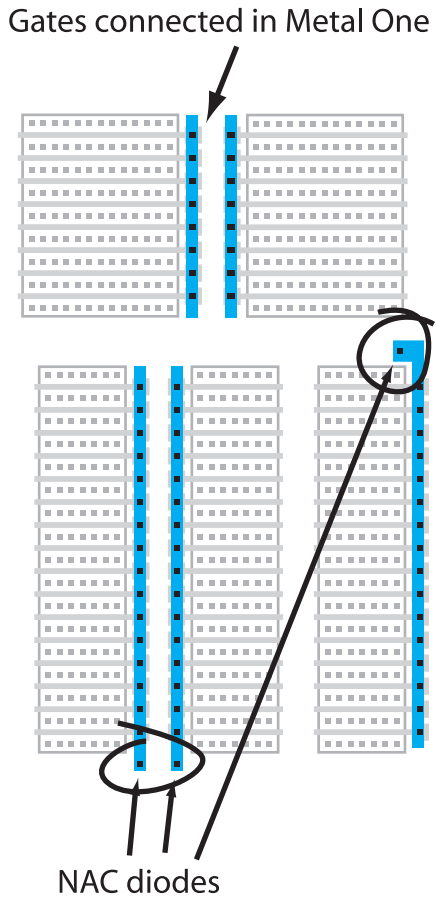


Figure CS1–9. *Gates connected in Metal One, and NAC diodes added.*

Observation

Bill is planning ahead, giving the circuit designer options that he has not asked for, but probably wants. Even though Bill has no direct experience in this type of layout, he is drawing on the experience that he does have. Planning-in the ability for a metal revision is always a good move. Likewise, wiring large poly gates in metal reduces the antenna effect and increases reliability.

Draw from what experience you do have and use it wherever possible. Don't wait to be asked to change something. Even better, ask your designer if he wants these extras. You will build a reputation of being one step ahead.

"I like to hook up the power early. So, let's add the power rails in," thinks Bill. After a thought, he decides, "Let's just have the power rails at the top and bottom."

Bill adds the power rails, top and bottom.

"Now, we need some well tie-downs, too. I'd better put those in now before I forget. Let's just tie those straight to VDD."

Then Bill thinks, "Hey, I can save some space here by combining the poly gate contacts of the PMOS devices. That lets me squeeze those transistors together a bit."

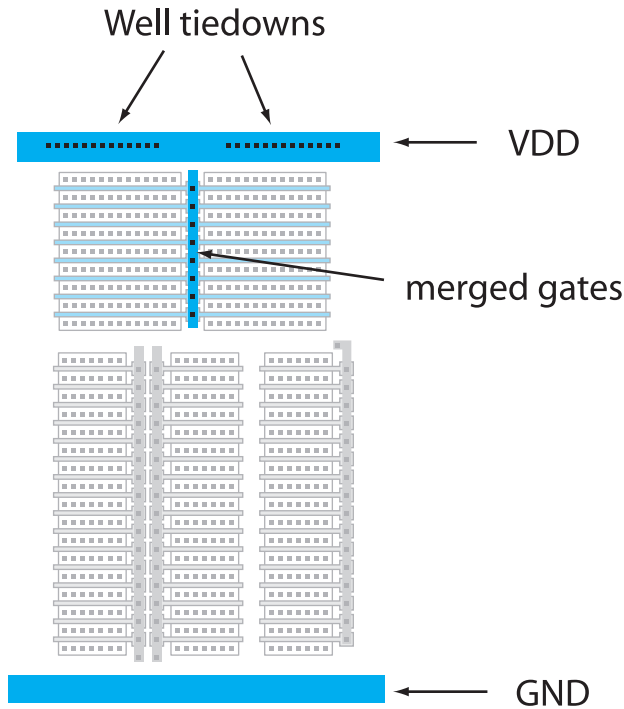


Figure CS1–10. *Power lines and well tie-downs added. Notice Bill has pushed his two PMOS transistors together (M1 and M2) so that their gates share a common connection.*

"OK, we can start wiring this cell now," Bill decides. "I'll start at the top with the PMOS devices, M1 and M2."

Bill examines his two PMOS devices as he thinks to himself, "I already have the gate connections pretty much taken care of. Now, which way around should I connect the source-drains on these devices? I will have to connect the source-drains to VDD and the output wires. I wonder which one I should start with at the top.

"Hmm. I'll draw a quick stick diagram to help me."

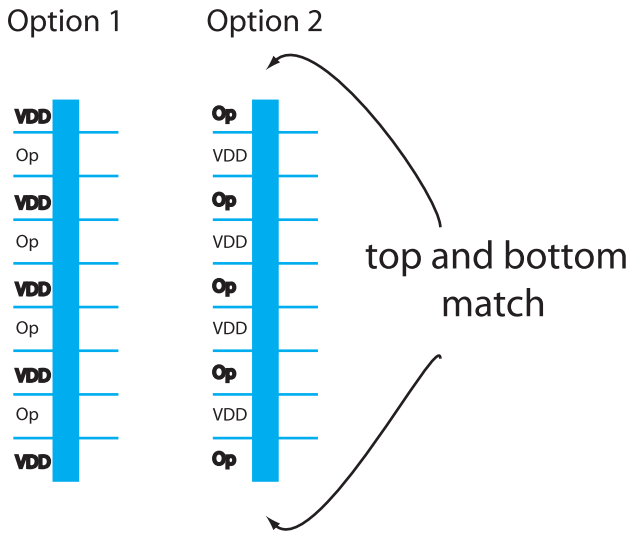


Figure CS1–11. *Stick diagram showing two options, depending on whether Bill starts with VDD or starts with output.*

Observation

Again, Bill is drawing from his experience. A stick diagram is a useful tool to aid in the mask design process.

Still unsure of which option to use, Bill quickly converts his stick diagram into a real layout so he can ponder his options. The first thing he notices is that he must increase the length of the poly on his gate connections to give himself some room for source-drain wiring.

As Bill stretches the gates on M1 and M2, he realizes he will encounter this problem on every transistor in the cell. Bill makes a note to himself to talk to the design team that is customizing the tool. Having to constantly stretch gates to allow the source-drains to be wired is very time consuming. Almost everyone will have the same issue, so why not change the transistor layout for everybody? In the meantime, he stretches all the gates in M3, M4, and M5, anticipating the problem in those areas.

He looks at his options again, this time as real layout (with gates stretched to make room for wiring).

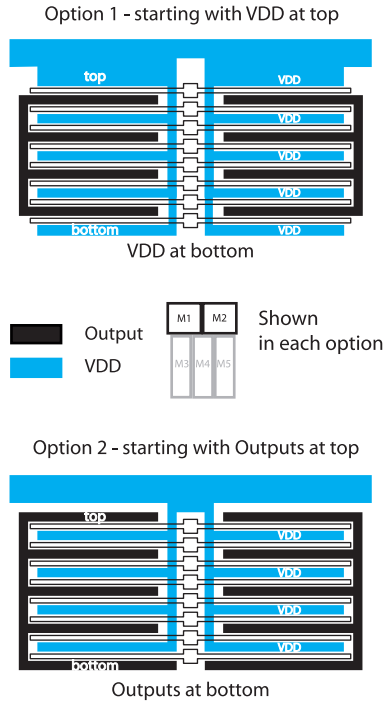


Figure CS1–12. *The previous stick diagrams were quick representations of these two wiring options. Bill selects the second option.*

"Well, if I pick the top source-drain connection as VDD, then my VDD wiring to these devices will be nice and easy. I can merge the VDD rail into the device. The drawback to this option is that I end up with VDD at the bottom of the device giving my output wires a long way to run. Not a good idea, as this is a differential amplifier and I should try to keep the output wires close to each other."

"I should chose option two," he concludes. "That makes my outputs easier to wire to the NMOS diff pair. My N well connections need to be connected to VDD also, so I can wire those at the same time.

"OK, that's my VDD hooked up."

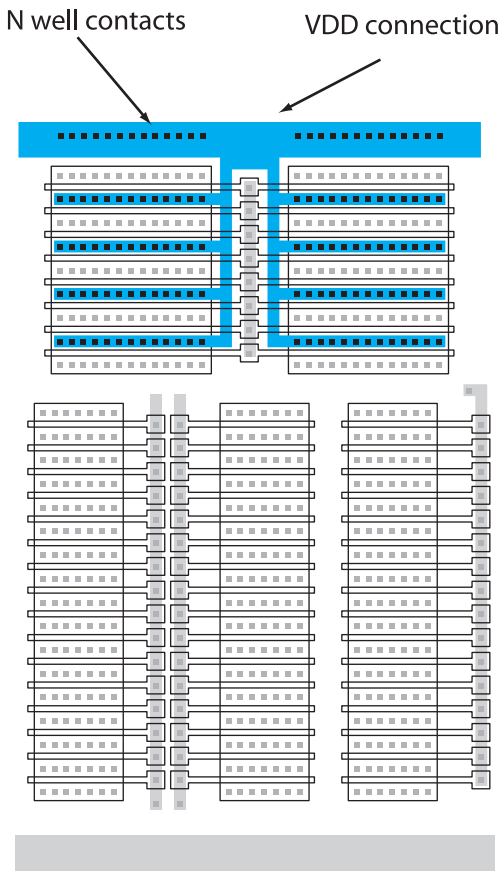


Figure CS1–13. *Connecting VDD's in devices M1 and M2.*

"Now that I have my VDD and gate connections wired, I can finish off these PMOS devices," thinks Bill. "The last wires in these devices are the output wires."

Bill examines the schematic and notices that the gates of the PMOS devices are connected to one of the output wires. Bill wires the final side of his PMOS devices and with a little extra metal connects to the central gate.

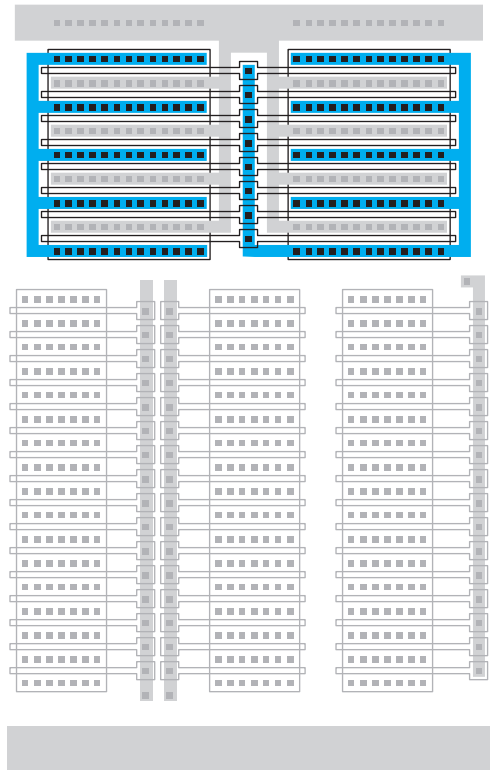


Figure CS1–14. *The outputs of our PMOS devices, M1 and M2 are connected in Metal One.*

"That's it," Bill thinks. "Apart from wiring the output wires to the input diff pair, I am finished with the PMOS devices."

Observation

Notice how Bill is breaking this layout into small, manageable sections. This technique is essential in mask design. It is impossible to consider every component in a circuit at once.

You will make lots of small decisions within a small section, then every once in awhile stop to examine the overall effect those decisions have on the rest of your layout. If everything still interacts well in the entire circuit, you continue making decisions from within small sections.

Sometimes when you stop to examine the entire layout, you may discover a top-level problem that was not obvious from the perspective of the small

section. In that case, some re-layout may be required before you resume your work.

Also, Bill rose above his job assignment when he decided to ask about changing the library design of the transistor gates. This sort of proactive interest outside his job assignment will help train Bill for larger responsibilities and help others see him as a valuable team member.

Bill thinks to himself, "Now I have the output wires on the PMOS devices easily accessible. I can run some nice short wires to the NMOS diff pair."

Bill chooses the top source-drain connection of the input transistors to be his output net. He wires the transistor accordingly.

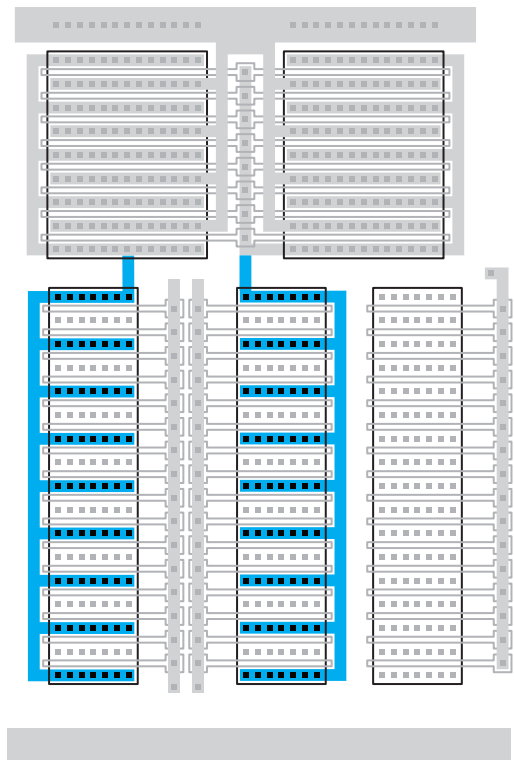


Figure CS1–15. *Output wires of differential pair connected to PMOS devices.*

"Now," Bill thinks, "it's time to join the diff pair transistors to each other." Bill joins the NMOS transistors to each other by wrapping the metal around the metal connections of his input gates.

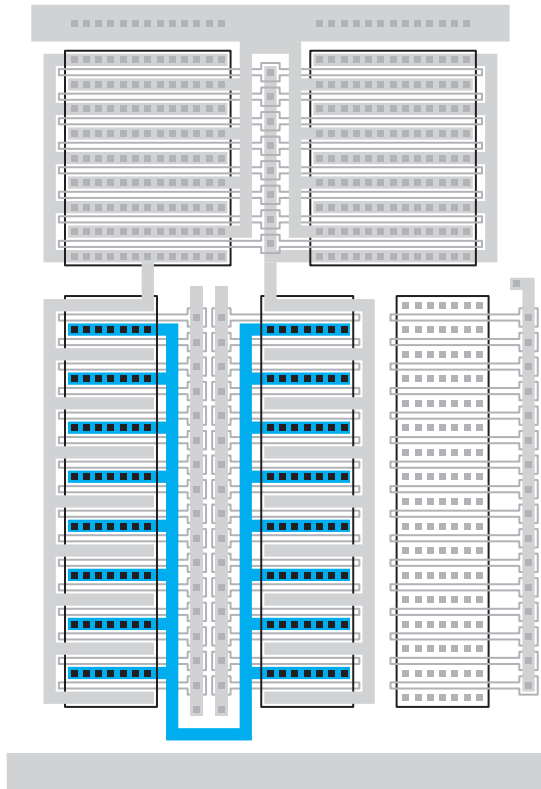


Figure CS1–16. *Common point of differential pair transistors wired to each other.*

"Ok, one transistor left," thinks Bill. Bill chooses one side of his final transistor and wires it accordingly. (See Figure CS1–17.)

"Just a second," thinks Bill. "I think I have spotted a way to save some room. If I make the common point wiring of the diff pair on the outside of the devices instead of the inside, then I can move my current source device to the left and share some metal. That will save me some space. It means I need to throw some work away, but saving space is always a good thing."

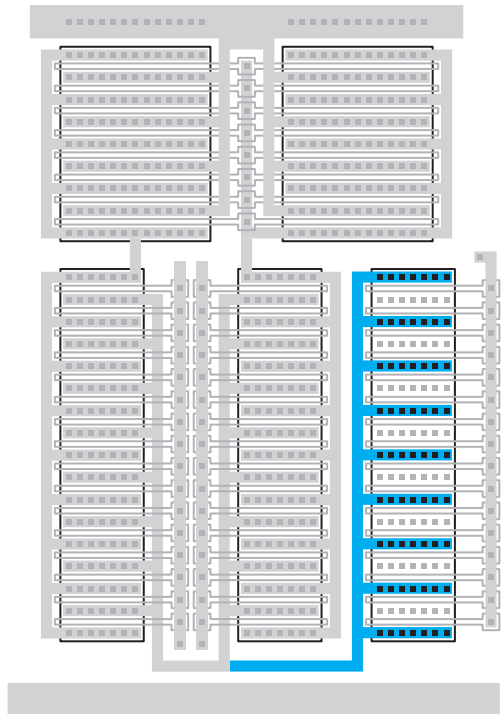


Figure CS1–17. *Differential pair transistors wired to current source.*

Observation

Bill has been working a lot today, and might be getting tired. He has missed an important point here.

By flipping the source-drains on his input devices to save space, he is now running the output wires immediately next door to his input wires. This is a potential source of instability that could cause the amplifier to oscillate wildly.

Space-saving is not always good. You must stop to examine the impact on your layout carefully.[5]

[5] Bill is providing more proof of Saint's Useful Thoughts Theorem.

Saint's Useful Thoughts Theorem: *There are a finite number of useful thoughts available in any one day. When you have used them up you may as well go home.*

Bill throws away the last few wires he has laid out and begins work on his diff pair again. This time, he runs his output wires along the inside of his diff pair. Bill reasons that his output wires are now closer to each other, which will help his differential matching.

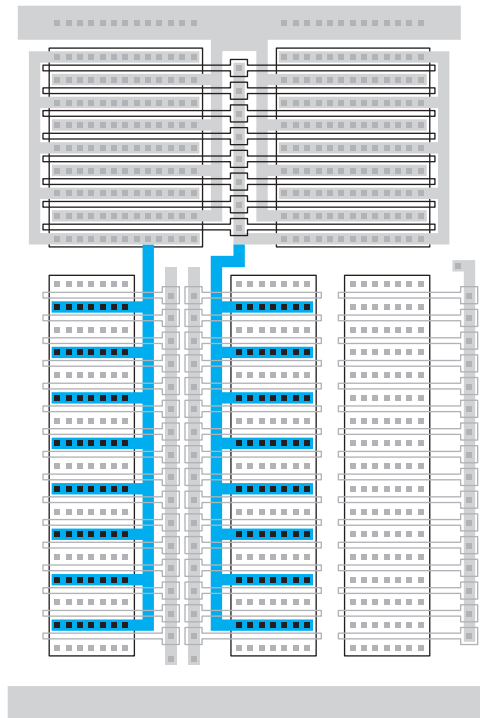


Figure CS1–18. *Output wires connected differently.*

Bill now wires his common point connections to the other side of his devices. (See Figure CS1–19.)

"Finally," thinks Bill. "I'm back to where I was about half an hour ago. I can now slide the current source device across to the left and connect it to the diff pair." He moves the transistor and makes the connection. (See Figure CS1–20.)

"All I need to do now is hook the ground, and I'm just about finished," thinks Bill. (See Figure CS1–21.)

"One last thing," thinks Bill. "I should add some substrate contacts around this cell to make sure I don't get any latchup problems." (See Figure CS1–22.)

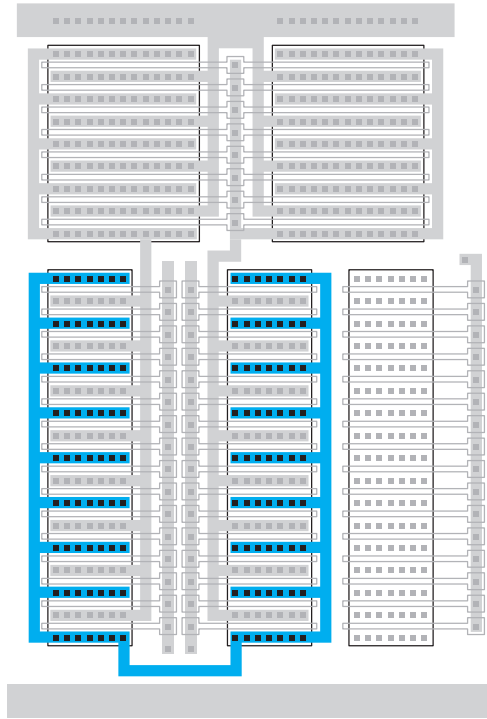


Figure CS1–19. *New common point connection.*

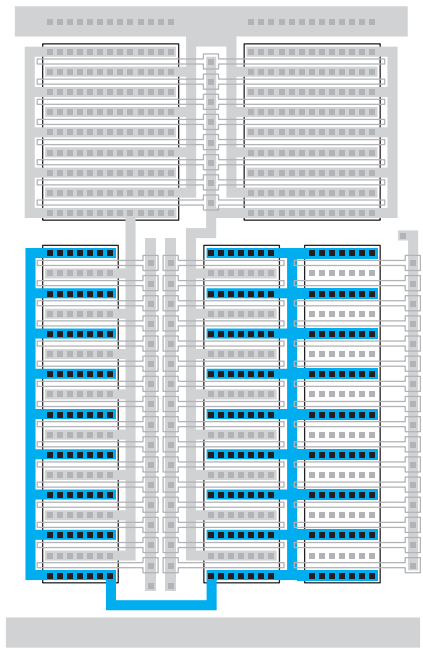


Figure CS1–20. *Merged metal connection of differential pair and current source device.*

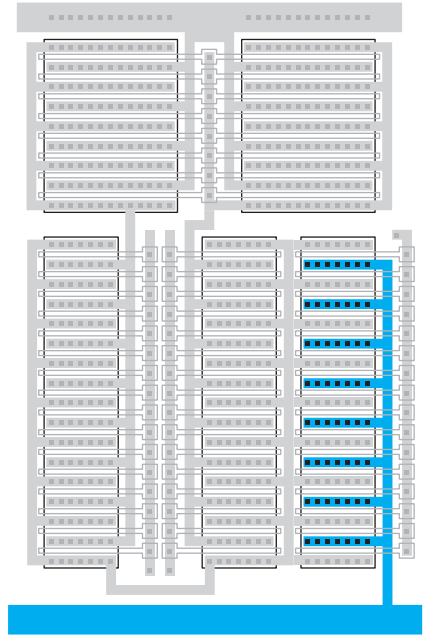


Figure CS1–21. *Final ground connection.*

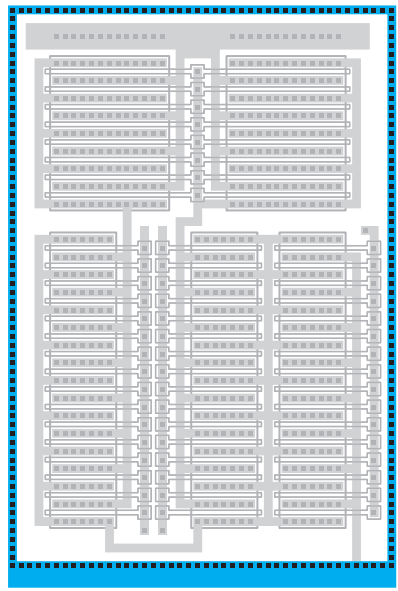


Figure CS1–22. *Addition of substrate contacts.*

Next, Bill places his two input wires on the left of the cell, and his two output wires through the top of the cell.

"So, I think I'm just about done. Let's run through what I have so far," Bill thinks to himself.

Bill begins running through the issues. "Ted was worried about matching. First of all, I have all my gates running the same direction in the whole cell, so there's some inherent matching there. I have my PMOS devices real close to each other. They are symmetrical along the middle line where the gates are. That's given me some matching. My input devices, again, are symmetrical, so I get good matching from the symmetry. I get good matching from the fact that the gates are close to each other. Likewise, my outputs are close to each other. Good matching.

"I have relatively short wires for the outputs. I have a relatively short wire for the common point of the differential pair—that's good. That's low capacitance. My current source device is nice and close to the differential pair, so again, that's low capacitance there.

"I don't have wiring wandering all over the place. And, it's a nice, compact layout, well put together.

"So, I'm pretty happy. I think Ted will be pleased with this. Let's see what he says when he comes back on Monday." (See Figure CS1–23.)

Ted Returns

Ted was fairly busy his first morning back after vacation. He had a bunch of voice mails and emails to answer. But, Bill and Ted met in the hallway.

Ted says, "Hi, Bill. How did you get on with the op-amp layout last week?"

"Oh, I think I got on really well," Bill says. "I've finished the layout. I'm really pleased with it. I think you'll like it."

"Great. I have a meeting at 11:00, so let's go through it together right after lunch."

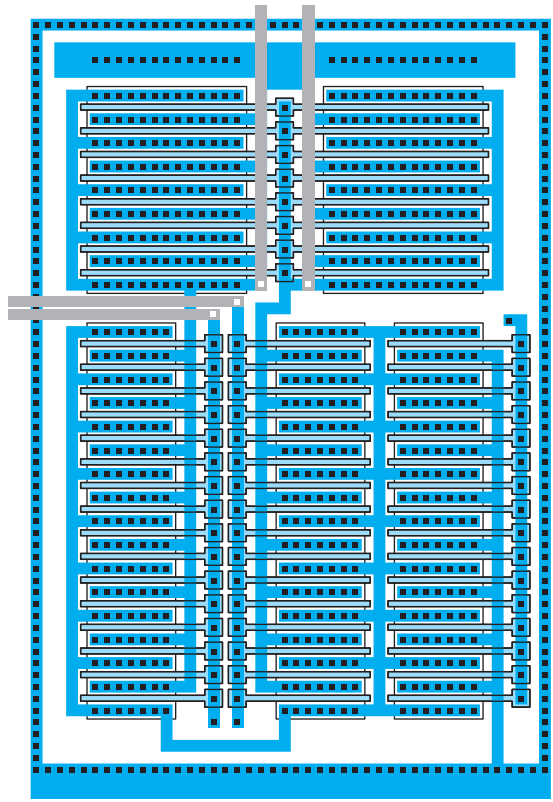


Figure CS1–23. *Bill's finished layout.*

After lunch they get together.

Bill says, "Ok, here's my layout. Let me quickly talk you through the floorplan." Bill begins pointing to all of the devices as he talks about each section.

"I've placed your two PMOS devices, M1 and M2, up at the top of the cell. I've tried to pay attention to all the matching requirements that you were talking about. The devices are both in the same orientation, and I have the gates common in the middle.

"The input differential pair M3 and M4 are placed south of the two PMOS devices on the left-hand side of the layout. I've tried to keep them close to the PMOS devices to try keep the parasitics down. And, also, I've kept the gates fairly close, because I know you were worried about the matching on this."

Bill continues, "Your current source device is on the far right-hand side of the layout. And, I've ringed the whole layout with substrate contacts with the power rails top and bottom. What do you think?"

Ted stares at the screen pensively for awhile. "What's that skinny wire in the bottom right-hand corner?"

"Oh, this one here, off the current source device? That's my ground connection."

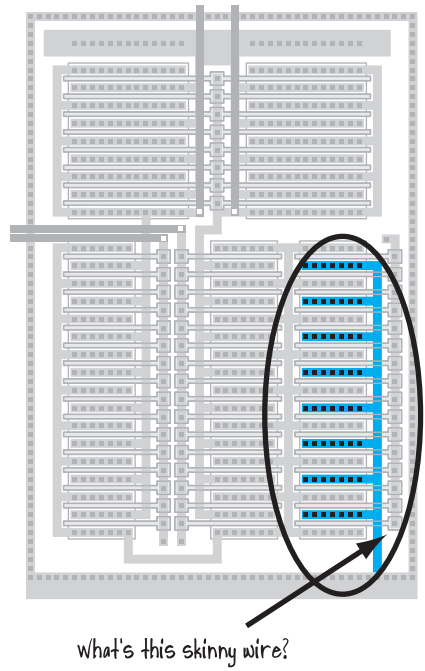


Figure CS1–24. *Ted spots a potential problem. The wire will not handle the current.*

"Hmm." says Ted. "What's the width on that?"

"Well, it's 1-grid wide," offers Bill. "I spoke with Joan and she told me that there were only one or two milliamps running in this. I went to the process manual, found the density numbers and I'm fine with 1-grid."

"Hmm," says Ted. "You know, what's low current for Joan isn't low current for me. She is used to working with currents in the 10's of amps with her ESD stuff."

"But Joan said she was at the design review."

"Yeah, she was at the design review. But, at that time I'd only done my preliminary simulations. When I simulated this chip over process and temperature I found the current can get as high as 5 milliamps."

"Really?" says Bill.

"Yeah, but I didn't present that at the preliminary design review because I hadn't gotten that far yet. It looks like we have a problem," says Ted.

"And the other thing I'm worried about . . ." Ted hesitates. "I like your layout. It's nice and compact. It's tight, well put-together, well-thought out. But I'm really trying to get a bit more matching on those two input devices.

"So, even though you have them placed really nice and close to each other," Ted adds, "I think I'd like to see them cross-quaded as well.

"Also, I notice the inputs and outputs of the two input devices are running directly alongside each other. That's dangerous. It's like holding a microphone up to the loudspeaker of a karaoke machine—the signal feeds off itself until it's out of control. Let's keep the gain instability problems to a minimum by spreading the input and output apart."

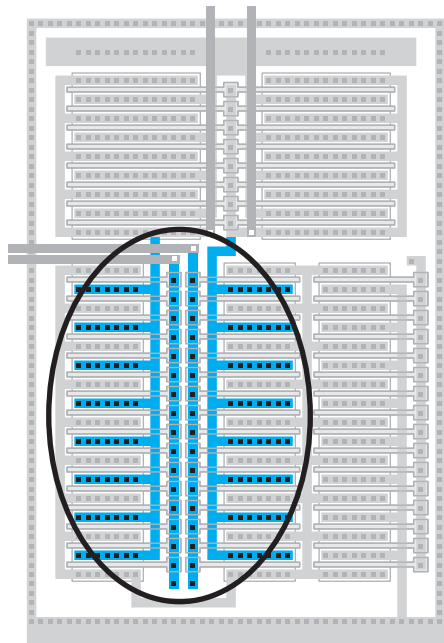


Figure CS1–25. *M3 and M4 are to be cross-quaded. How do you cross-quad two devices? Also notice the amplifier input and outputs are running alongside each other.*

Ted continues, "All the current falling out of the current source device in the bottom right corner is coming through one end of a single strip of metal. That concerns me.

"That device orientation would let the current leave the bottom of the transistor but hardly any current would be able to flow from the top. The current will crowd around that bottom right corner, even if we do widen it. So, you might think about rotating these devices through 90 degrees. You can still have your power rails top and bottom. But at least with the devices rotated, your current will have more room to escape," Ted finishes.

"Now, the more I think about it, the more worried I get," Ted concludes.

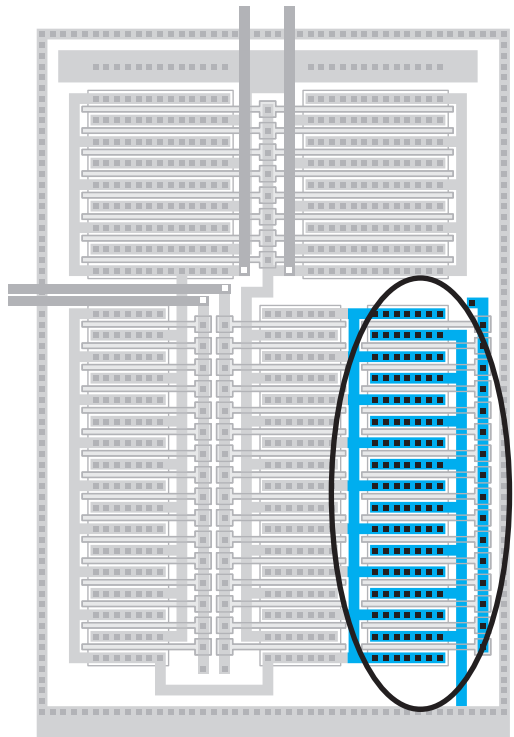


Figure CS1–26. *Rotating M5 by 90 degrees would allow better current flow from the far end of the device.*

"I don't quite see what you're getting at," says Bill.

Ted draws a diagram.

"At the moment," says Ted, "all of your current is coming out of the bottom right hand corner of the current source device, on this one thin strip of wire. All the current from the entire device is heading toward this one small area.

"But," Ted continues, "if you were to rotate the current source device through 90 degrees, then the current will have more metal to flow through. You can still have your nice big fat power bus down at the bottom. Your current densities will be a lot easier to handle," explains Ted.

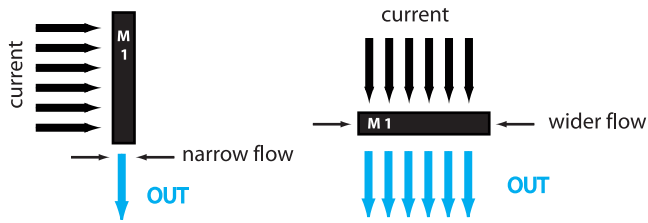


Figure CS1–27. *Rotating the device to allow more current to flow.*

Observation

Bill is not afraid to speak up when he needs clarification. Ted takes the time necessary to explain, even drawing a diagram. A lot of a mask designer's learning happens on the job in situations just like this.

However, Bill's layout is being ripped to pieces. What Bill thought was a good job becomes wasted effort. The moral of the story to date is that the circuit designer must communicate what he really wants. Good documentation and communication would have saved Bill a week of effort. Unfortunately this example is very common.

"Yeah, that's a good point," remarks Bill.

"Is there anything else I've missed?" Bill asks.

"Well, there's a nice, big set of N well contacts at the top of the PMOS devices, but I'm a bit concerned about them being so far away from the bottom edge. If we could get some N well contacts down at the bottom edge of the PMOS devices, and also increase some substrate contacts in there as well, that would help prevent any latchup problems."

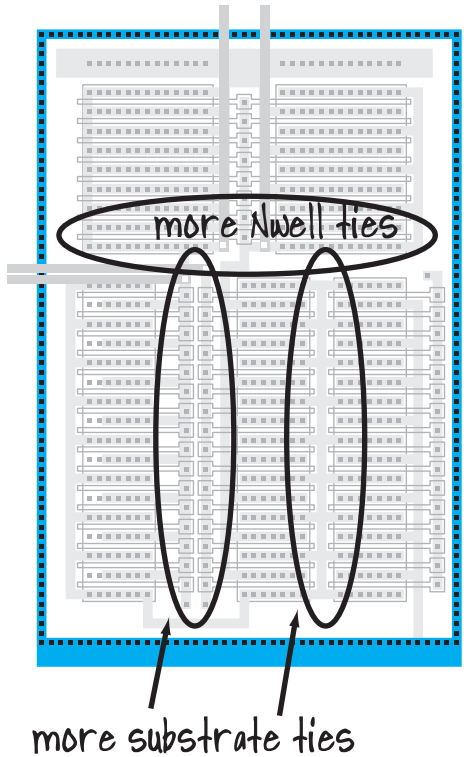


Figure CS1–28. *We need more ties. You see Bill added some around the edges, but we still need more at the extremes of the cell.*

"Oh," says Bill quietly. "It looks like I have a lot of work to do."

"Well, yes and no," encourages Ted. "You've done a good job. Have you worked on this type of cell before?"

"No," says Bill, "I've only done CMOS I/O cells."

"Wow," says Ted. "In that case you've done a *real* good job."

"Thanks," says Bill. "If you have a minute, I'd like to know how the circuit works."

"Sure, no problem," says Ted. "As I said the other day, the circuit is a CMOS amplifier.

"The current source device, M5, is configured to provide a constant current that gets switched between the two input transistors, M3 and M4, depending on the voltage coming in the two inputs.

"The two PMOS devices, M1 and M2, are configured as active loads, so effectively they are just acting as a resistor for this application."

"What do you mean by *active load*?" asks Bill.

"An active load is a term we use to describe the way those PMOS devices are wired. There is no signal going to the gates. Both the gates are connected to one side of one of the devices. This connection self-biases the PMOS device so that it looks like a resistor."

Bill wonders, "Why didn't you just put a resistor there instead?"

Ted says, "This process doesn't have any resistors. Yet. Using active loads is a common technique of biasing a transistor to look like a resistor. It works."

Ted continues, "The basic overview of this circuit is that the input devices switch the constant current generated by the current source transistor on and off. The current flows through the active loads where it gets converted into a voltage.

"The small voltages on the gates of the M3 and M4 control the large current coming through the circuit. The large current is converted by the loads into a large voltage. And that's your amplification. We went from small voltage to large voltage.

"So, as you can see, it's the current flowing in the circuit that is doing all the work," Ted summarizes.

"Ah," says Bill. "That's why you are so concerned about how the current flows through the transistors."

"Yes, that's where the most current is flowing." Ted motions toward the transistors in the layout, then stops to look at it for a moment.

"Now, I like the basic layout idea you have here," Ted continues. "I like the way you have the inputs close together on the left side of the cell. I like the way you've brought the outputs out symmetrically through the top. That works well for the overall chip floorplan and the bond-out that we have, so I see where you're aiming. But it's a shame I was on vacation. If I'd been around I could have saved you a lot of work."

"Ok," says Bill. "I tell you what. Why don't I go start work on some of these points that you've mentioned, and I'll keep in touch with you the rest of this week as things progress?"

"Sounds like a plan to me," says Ted.

Observation

Even though Bill is not a circuit designer, understanding some of the basics of how this op-amp circuit works is very beneficial. Understanding where current flows in a circuit can help him determine how wires need to be sized. Knowing what wires are only carrying a voltage can help him save space.

Learning more about circuit function helps Bill learn about signal flow, what affects device orientation, matching, when to switch to a different metal, and other choices he can make with his layout. He will learn to recognize common circuits, which not only makes his job easier, but will help him catch possible errors when something just doesn't look right.

As a mask designer, you do not want to operate from a cluttered wall of yellow sticky notes. As you learn more about circuit function your work becomes intuitive, almost instinctive. At that point it no longer feels like work, but play.

Bill Rethinks

"Ok," Bill thinks to himself. "Let's write these points down before I forget them."

1. Current can be as high as 5 milliamps.
2. Cross-quad the input pair.
3. Need extra well and substrate ties.
4. Rotate 90° to improve current flow.

Observation

Question: If you were Bill, where would you start?

Answer: You could start with any of these four issues. There is no right or wrong order to doing layout work.

"Ok, let's look at each of these issues one at a time," Bill thinks. "Cross-quading would mess up the diagram the most, and rotating 90° is a big issue, too.

"Well, the first item I want to tackle, maybe, are the little things. Just get them out of the way. Knock them off quickly.

"On the other hand, by the time I handle the large items, they could rearrange my whole floorplan. The layout might be totally different in the end, so all that work on the little things could be wasted.

"Well, on the other hand, I could just throw my substrate contacts down at the bottom and get it over with. I was taught to place them early to guarantee room for them.

"On the other hand, there might not be room at the bottom after I make the big changes."

Bill feels he has run out of hands, detects his gut feeling, and quickly knows his decision. "That's it. What I really want to do is start with the big stuff. I'll probably rotate my blocks first, because that's one of the biggest moves. Then I'll see about the cross-quading. Then I'll fatten my wires and make sure I have room for my ties last of all."

Bill realizes that with these major changes, all the work he's done basically has to be thrown away, so he just starts again from scratch. He goes back to his basic devices, and loads them onto a blank layout screen.

Bill gets to work.

Observation

The more a mask designer knows about a project from the beginning, the less rework there will be. Had Bill attended the design review, been on-hand throughout the project development, and worked with these circuit designers before on similar projects, he would have better understood some of the requirements of the layout, maybe even to the point of anticipating some of the changes and decisions that Ted later made. Participate in your projects as early as you can.

Certainly, lack of communication while developing the layout contributed to a large backlash feeling of wasted work. Early and constant communication with your circuit designer is key.

There are also times when requirements change as a project continues. There will be times when you do, in fact, throw away a certain percentage of your work, despite the best of up-front knowledge and excellent communication.

At those times, maintaining an appropriate attitude during adjustments often comes from realizing that all work done is of some benefit some-

where in the grand scheme of things. All layout work helps a mask designer become proficient. Any preliminary designs help the team visually to define their needed changes. Any involvement better educates the mask designer with the project. Subsequent reworking usually results in a much better final product.

Besides that, it's your job to go with the flow. Some changes just can't be avoided.

A good mask designer adjusts easily. This is another skill that makes you a joy to work with, and gives you a positive reputation.

Bill stops to think, "I have to rotate my current source device, since Ted was really concerned that all the current was crowding through the bottom of the wire. Well, then, in order to have that current source device match against all the other devices, I really should rotate all of them.

"Also, if I rotate all the devices, I can keep almost the same aspect ratio that I had before. Ok, I'll rotate them all." And he does.

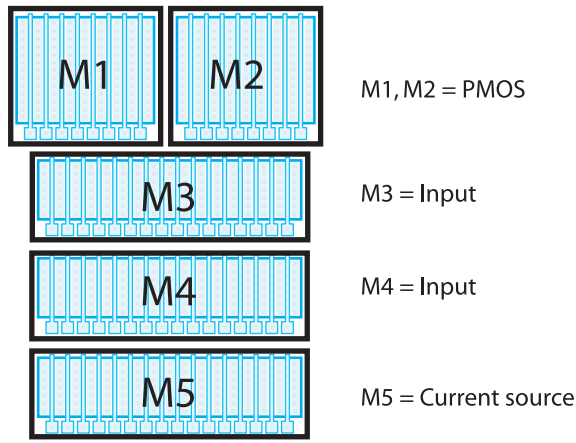


Figure CS1–29. *Rotated devices.*

"I kind of like this aspect ratio. I just have a good feeling that it's right. I'm not quite sure where I'm going with this yet, but I'll just see what happens."

Observation

Bill is not sure this orientation will work in the end, but he sees no problems with it at this time. He decides to continue in this direction, but he seems to realize that the possibility exists that he will come up against a wall at some point, and have to throw out his work and begin from scratch again.

Pursuing wrong directions happens sometimes. You never know until you try. Try different approaches, just to see where they lead. This free-spirited attitude of experimentation helps Bill get right to work.

He also has a good feeling about this direction. Sometimes our instincts help us, especially as we gain years of experience.

"Now I need to cross-quad the input pair devices. But, I can't now, because they are only two devices. To cross-quad I need four devices."

So, Bill goes back to Ted and asks, "You know how you wanted me to cross-quad the input pair?"

"Yes," answers Ted.

"Well, the way you have the devices built in the schematic at the moment I can't do it. There are only two devices. I need four devices for a cross-quad."

Ted lets out a sigh of recognition, "Oh yeah. I'll go change the schematic for you. I'll just split the devices into two and you can work with it that way."

"Great," says Bill.

So after 15 minutes Bill fires up the schematic, regenerates the devices, and builds his floorplan. (See Figure CS1–30.)

Bill places the two halves of the devices diagonally from each other. He marks his floorplan diagram to show the pairs.

Observation

Bill keeps the same name as the original device for one half, and adds the letter "b" to denote the remaining half. So, our original M3 is now

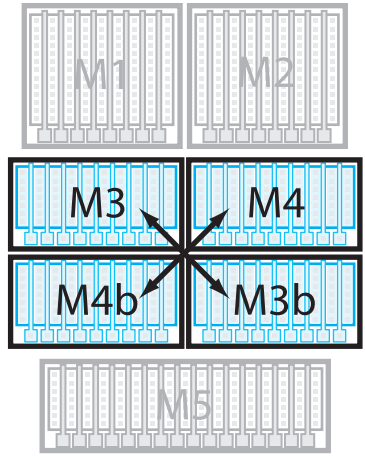


Figure CS1–30. *Input devices split to allow cross-quading.*

divided into two halves, called M3 (same name as original) and M3b.

Bill might have prevented some possible confusion had he totally renamed both new device halves. For instance, M3a and M3b.

It is good practice to remain unique and consistent when naming.[6]

The next thing Bill thinks about is current density. "It's time to figure out what size wire I need for this thing. Ted said the current could be as high as 5 milliamps. So, that means that my minimum wire width needs to be at least 5 microns wide.

"However, I'm working on a gridded process, so that means that I can either have a 2-micron wire or some other multiple of the grid spacing. So, what's the smallest number of grids that can get me the wire width that I need?"

"Ok," thinks Bill. "My 1-grid wire was too small. Let's see how big my other wires can be."

[6] I've seen some horrendous trouble caused because someone, or some series of people, have named the same device by five or six different names; or the opposite, where different devices have the same name! Yikes! Are we asking for trouble here?

Bill draws for himself a little diagram to investigate how much current each size of wire can handle. Bill remembers that the manual described the minimum width of wire for this process as 2 microns, and that the minimum spacing between wires is 2 microns as well.

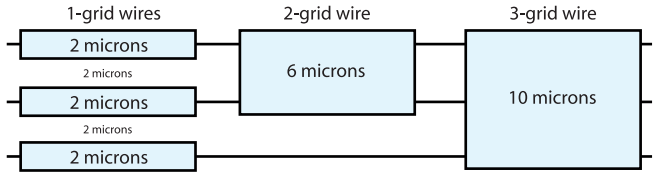


Figure CS1–31. *Bill is determining the wire sizing to handle 5 milliamps.*

"Let's look at the 2-grid wire. I have two single grid wires, plus a 2-micron space between them. That means a 2-grid wire in this process will be (2+2+2) 6 microns wide. That will handle 6 milliamps, which is more than the 5 milliamps I'm worried about.

"So, if I wire everything in the current path with this 6 micron, or 2-grid, wire, I'll be in fat city. I don't need to even worry about 3-grid wire.

"So, ok," he says to himself, "let's start work."

Bill looks at the four devices he will use for the cross-quad. Luckily for Bill the devices he needs to cross-quad have a wire that is common to all four devices.

He realizes that probably the easiest layout is to have the common wire run through the middle of all four transistors. So, he lays down a 2-grid wire horizontally through the middle of his four devices to act as the common wire, or as he puts it, the common *point.*

He thinks, "Cool. I have a 2-grid wire, so I should be ok for current density. Now, if I bring another 2-grid wire vertically down through the center, then I can get all the current out of those four devices to the current source device nicely, easily. So I think I'll start with that."

Bill draws the vertical 2-grid central wire. At the same time, he adds source-drain connections. He uses the same metal for his central wire as the source-drain connections.

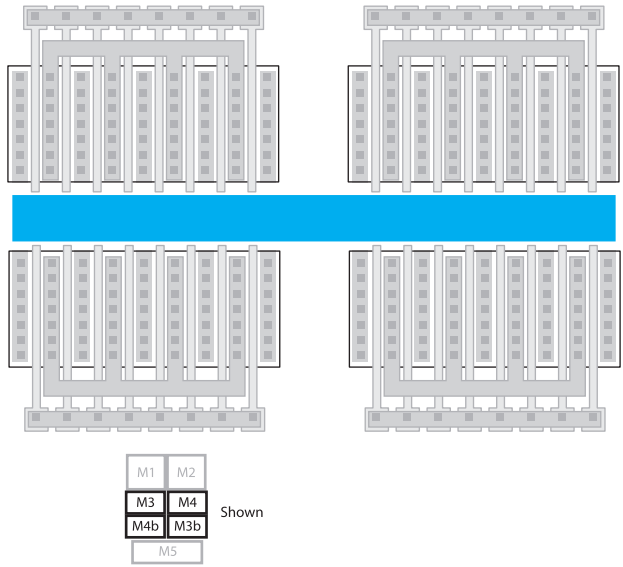


Figure CS1–32. *Bill lays a 2-grid wire, which handles up to 6 milliamps in his process.*

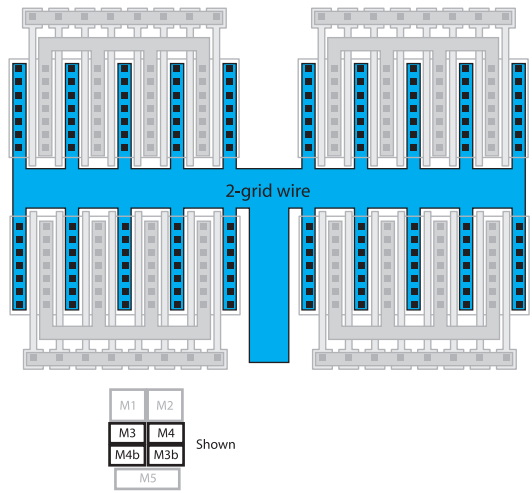


Figure CS1–33. *Connecting all source-drains, with a large 2-grid wire dumping current out the bottom middle area. This looks like a nice, easy flow for the current.*

"Now wait. This is a differential amp," Bill thinks. "I'd rather have my inputs running through the middle of the devices because it improves my differential matching. But, if I have a 2-grid wire running through the middle, then how can I get my inputs through there as well, without messing up the symmetry?

It would be real tough to get the inputs in the center symmetrically. So, hmm. How am I going to do this?"

Bill sits and thinks for a while, staring at the screen. He feels he is not coming up with a ready solution, so he finally says, "Ok, well, I have to have my inputs running through the middle. I have no choice about that. So, I'll start there, with what I know I absolutely must have."

Bill draws two lines, which will be the inputs. He looks at that awhile.

"Let's start again," Bill thinks, "beginning with these two lines." He decides to just play around with the devices. He is not feeling stressed about any of his attempts. He is just sort of horsing around, trying this and that.

Bill flips the transistors. This puts the gates closer to the two input wires he just drew. He is not sure where he is going yet.

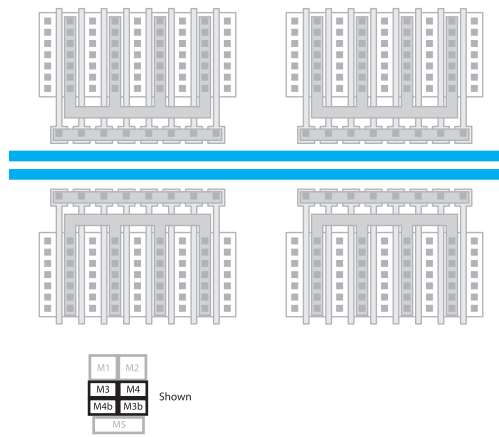


Figure CS1–34. *Bill begins again, this time starting with the input wires.*

Observation

Bill's experience allows him to be able to ignore his previous work to come up with novel approaches. That helps him break through his road-block.

Bill also takes enough time staring at the screen. He is not worried about whether his coworkers think he is busy or not. He knows he needs to sit and think. Thinking is working. Bill is showing confidence.

"Ok, there are my two inputs. Now, I still need to get the common wire to each transistor, so if I place the devices far enough away from my inputs, then I can run my 2-grid wire on either side of the inputs," he reasons.

Bill moves his transistors away from his inputs and places a 2-grid wire between the input wires and the transistors. He also runs that same fat wire out the bottom toward the current source device, still using some of his original thinking.

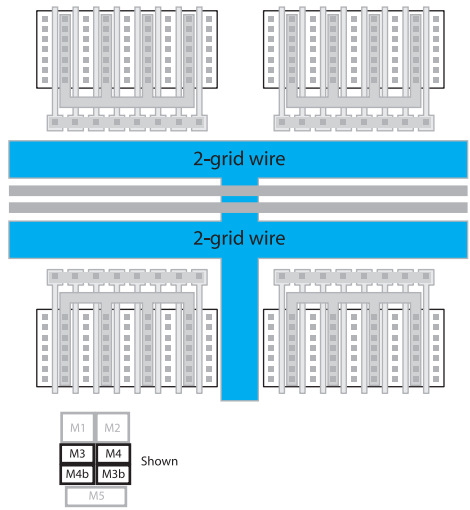


Figure CS1–35. *Moving the transistors back to allow room for the 2-grid wire.*

Then Bill thinks to himself, "All I'm really worried about is total current. Since I split a 2-grid wire into two paths, why can't I just make each half into a 1-grid wire? That should save some room."

He talks to Ted. "Hey, how do you like this as an idea for current flow on the floorplan? I think I can get away with a single grid wire either side of the inputs."

He shows Ted his layout with 1-grid wire.

"Well," comments Ted, "I like the idea of running the inputs through the middle. But one thing to think about is this: This circuit is differential. The full amount of current is flowing through each of those input transistors.

Ted continues, "For example, there are times when the entire circuit current will be in M3 completely. As the differential signal transitions, M3 turns off

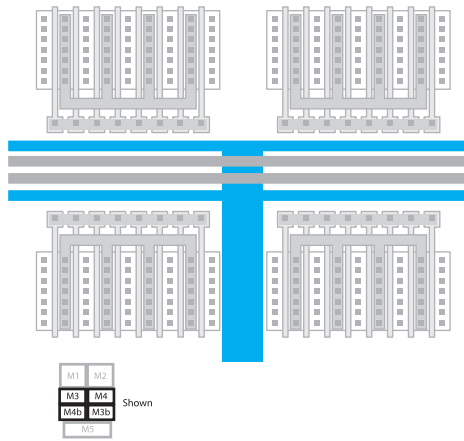


Figure CS1–36. *Bill thinks of space-saving options. How about just using 1-grid wire, since we have two of them?*

and M4 will turn on. So all the current is either all in one device or all in the other device. It just transitions each time the signal swings, so you should make sure that all 5 milliamps could be handled by your wiring at any point in time. So, let's see if a 1-grid wire can handle it.

"If M3 is on," Ted theorizes, "the whole M3 device carries 5 milliamps. That means that each half of M3 carries 2.5 milliamps. A 1-grid wire can only handle 2 milliamps."

They nod to each other as Ted concludes, "We will have to go with the 2-grid wire, even though it is slight overkill. 1-grid is just not enough to handle half the current."

Convinced that the 2-grid wire on each side is the way to go, Bill sketches a final drawing. He asks Ted, "Well, is this ok if I have the output coming through the middle and it's a 2-grid wire?"

"Yeah, that's fine," Ted says. "When device M3 is conducting, then the current will flow through the top left and bottom right transistors. When M4 is conducting, then it flows through the other two transistors. Either way, we have a max of 5 milliamps, so the central wire can handle it."

Bill and Ted are now in agreement. They are happy with the floorplan of the cross-quad device, so Bill goes off to start his layout.

First, Bill makes a mental note: He has to bear in mind that wherever signal current will flow he should have a 2-grid wire.

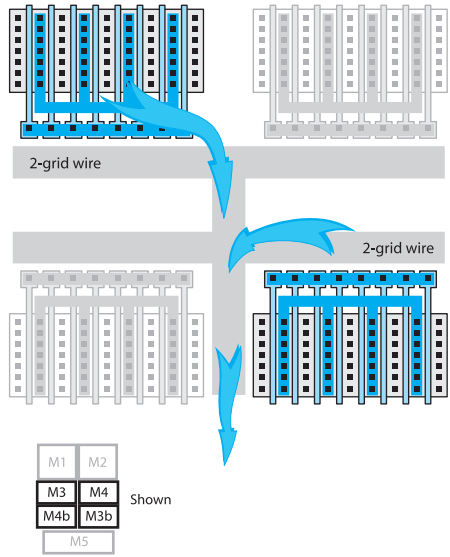


Figure CS1–37. *When device M3 is turned on, its two halves will send current down the center wire.*

Now it's time to wire it up.

Looking at his layout, Bill realizes that it is impossible to connect the source-drains of his transistors to the 2-grid common wire. The gate poly heads are in the way.

Bill wonders about this for a minute, then thinks, "I know. I can pull the gate poly heads underneath the 2-grid wire, into the center. The poly will connect my gates to my input wires. That leaves the 2-grid wire standing directly next to the source-drain regions just as I want." He makes the adjustment to his layout.

However, when Bill makes his adjustment to the poly gates, he creates a conflict in the middle of the cross-quad. In the very center of his cross-quad he is using two different signals, both in Metal One. One is the pair of input wires, and the other is the 2-grid common wire. That's a short between the input wires and the common wire. He needs to separate the two signals. Both cannot use Metal One.

Bill decides to use Metal Two to jump the first set input wire over the 2-grid common wire. There is that problem solved.

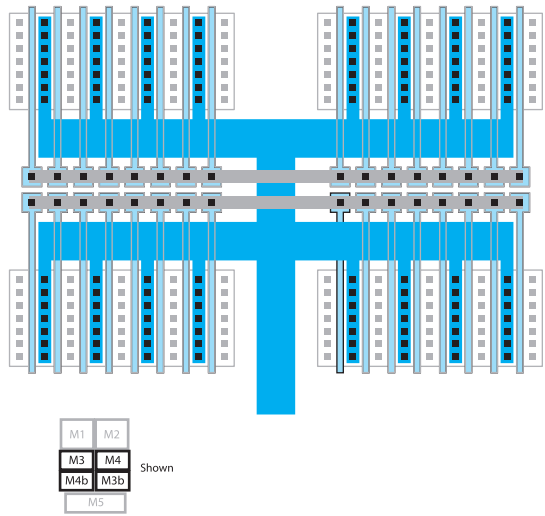


Figure CS1–38. *Gates brought across the 2-grid common wire.*

"But," Bill thinks, "I need to cross the other input wire over the common wire also, and I've already used Metal One and Metal Two. Is there something I can do besides wiring in Metal Three?" He wants to keep his number of metal layers to a minimum.

So, Bill sits again quietly for a few minutes. Bill finally decides to use poly to cross *under* the 2-grid wire for the other input.

Observation

Using poly for wiring purposes is only acceptable in certain circumstances. In this instance, we are connecting gates to each other. These input gates are not carrying any substantial current; they are only carrying a small voltage.

Any extra resistance in the lengthier amounts of poly will not cause any voltage drops to occur, since very little current is flowing. However, the input capacitance will increase due to the extra poly. So, you need to be careful when using poly for wiring. Make sure that you understand your circuit functionality before using poly to make a cross-under.

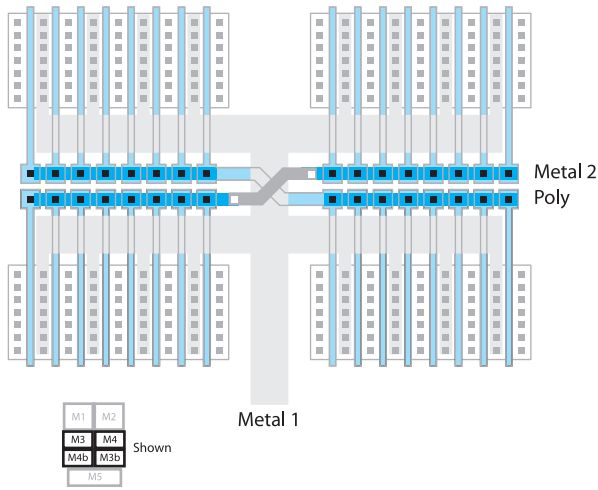


Figure CS1–39. *Poly layer crosses under Metal One, and Metal Two crosses over.*

Let's look a bit closer at that center crossing.

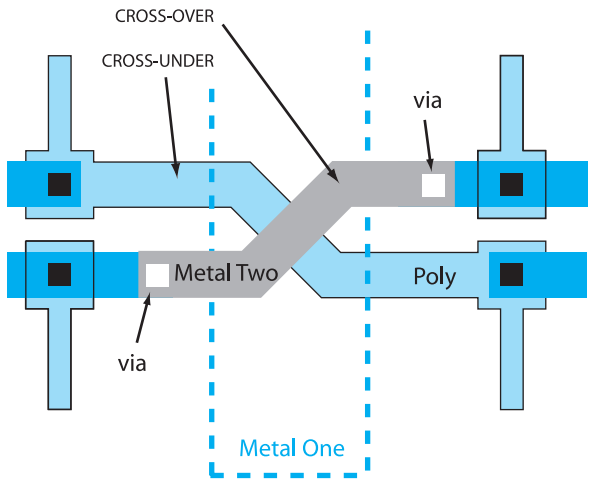


Figure CS1–40. *Close-up of the center cross-over/cross-under.*

Next Bill looks at his wiring. He has the common node[7] wired as a 2-grid wire. He makes sure these connections are nice and wide because they have to handle the entire 5 milliamps that this device is carrying.

[7] There are many phrases for *common node,* which refers to the common wire, also called the common point. It also could be called the common trace. We could call it King Tuten Common if we wanted.

He has the inputs wired sufficiently in single-grid wires, close to each other. Bill adds some NAC diodes.

He now needs to connect the source-drains of the device outputs to each other.

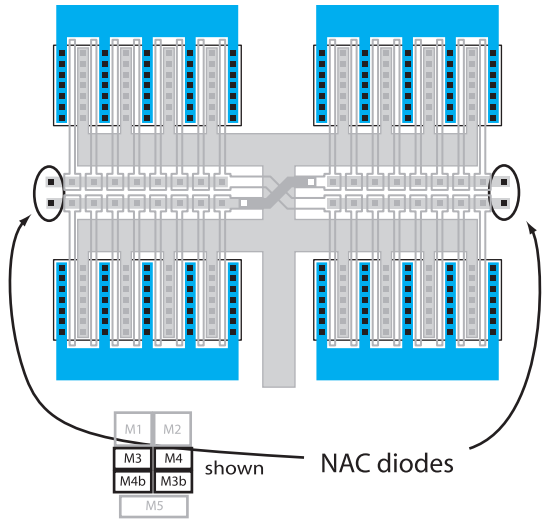


Figure CS1–41. *Source-drains that need to be connected.*

Bill considers the output wiring. Again, Bill stares at the screen and at first contemplates running the final connections from corner to corner in Metal One.

As he looks at the screen he realizes that since Metal One runs all over these devices, probably the easiest way to hook up the outputs is to use Metal Two. He thinks he sees some nice paths that should not short any other Metal Two or cause problems in lower layers.

Bill thinks about other wiring issues before he commits to an output wiring scheme.

Remembering what Ted said about current flow, Bill runs the current out of the centers of his devices wherever he can. He also knows that the current that goes into these devices is the same as the current that flows out of these devices, so the outputs have to be a 2-grid wire as well.

The wiring scheme he comes up with to go from corner to corner looks like this.

Figure CS1–42. *Cross-quad area finished. Device outputs are connected in Metal Two with plenty of vias and wire width to allow good current flow.*

The wiring is not quite as symmetrical as Bill would like, but it's good. Also he has looked at the current density numbers for the vias and knows he has to place a lot of vias in the current path to handle 5 milliamps.

Finally, Bill thinks he is finished wiring the cross-quad device, so he takes the final cross-quad layout to Ted.

"What do you think about this?" he asks Ted.

Ted looks at it and exclaims, "Wow. Excellent." Ted looks it over a bit longer, then adds, "Good job. That's exactly right. It does everything I want."

As Bill heads for the door, Ted remarks, "Don't forget about the substrate contacts." Bill wonders what Ted meant by that, so he takes another look at his layout before he leaves.

Bill wants to confirm what he thinks Ted means. Bill clarifies, "I think I see what you mean. I need to get in some substrate contacts fairly close to the cross-quad. So, I should probably pull the devices apart slightly and put some substrate contacts in."

Ted agrees.

Observation

Bill was about to leave the room when Ted made a casual comment. Bill stopped to clarify before he left. This is good practice to take the time to clarify, never to feel rushed.

So, back in his office, Bill stretches the layout in the x-axis direction and inserts some extra substrate contacts between the devices.

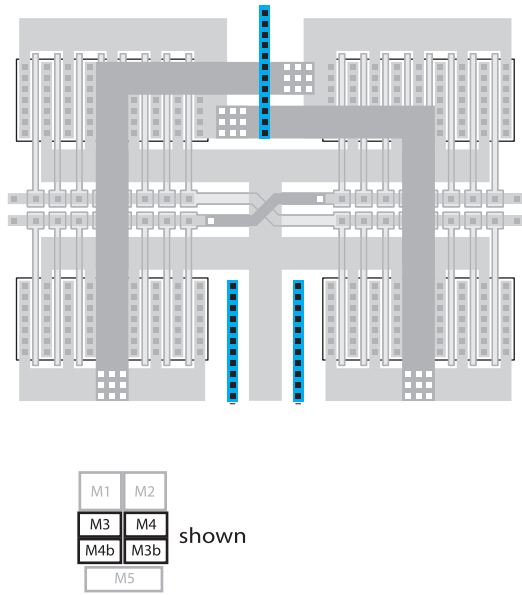


Figure CS1–43. *Devices pulled apart and substrate contacts added.*

"There is something bothering me about that current source device (M5)," thinks Bill. "I don't know why, but I have this feeling that if I split the device into two halves, that I will end up with a better layout that is easier to wire."

Observation

Sometimes during the layout process the work you have done to date starts to dictate where you will end up. In this case, the symmetry of the

layout that Bill has already done is sending Bill cues as to what to do next.

His present floorplan shows the current source device as a single, solid piece of layout. This does not lend itself well to the symmetry of the layout created so far. This kind of gut feeling comes with experience. It can be uncanny how well this kind of feeling can guide you.

"I think I'll go back and ask Ted if he's ok with me splitting the current source device as well." Bill draws a quick floorplan to show Ted what he is thinking.

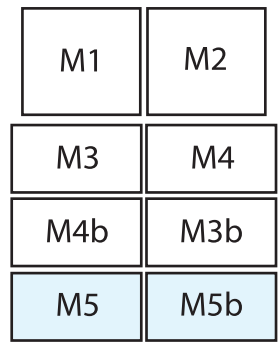


Figure CS1–44. *New floorplan. Notice M5 has been split.*

Bill goes back to Ted, explains his idea, and Ted agrees, "Yeah, that's a reasonable option. Also, then, all your NFET's line up in a nice row against each other."

Back at his layout screen, Bill reloads the schematic, and places the split current source devices and stretches his common node through the center in preparation for future wiring.

Trying to think ahead, Bill decides to wire his common node to the top of the M5 devices such that he can easily hook his ground wire to the bottom of the device. In order to connect to the top of the device Bill must stretch his poly gate fingers to give him enough room to get a 2-grid wire into the source-drain fingers.

Bill also needs to reconnect all the gates of this current source device since he split the device in two. Bill makes a small cross-over in Metal Two to join the gates to each other.

As Bill starts to wire the ground connection to the current source device, he notices a problem. "Darn it," says Bill. "I've trapped the substrate contacts on

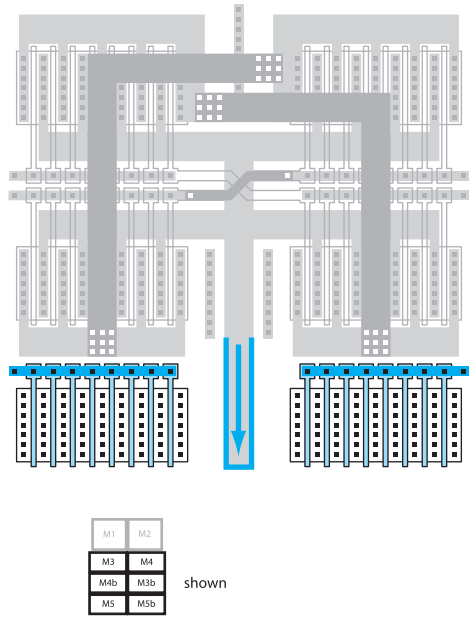


Figure CS1–45. *M5 split and separated.*

Figure CS1–46. *Common wire complete and gates joined in M2.*

either side of my common wire. I need to connect those substrate contacts to ground, and it's going to be really tough and nasty to get them past the Metal One of my common wire that I just connected."

Bill thinks for awhile, and finally hits a good solution. "I know," says Bill. "If I hook my common wire in Metal Two, then I can run my substrate contacts underneath the gate and common wire connections in Metal One."

Bill modifies his layout to connect the common wire in Metal Two. Being conscious of the amount of metal and vias on this wire, he makes the Metal Two connection the full width of the current source devices and places as many vias as he can.

Figure CS1–47. *Common node reconnected on Metal Two to allow Metal One substrate contacts to pass underneath.*

Happy with the ways things have progressed, Bill connects the other side of the current source device to a large Metal One power bus and places as many substrate contacts as he can for good measure.

Afraid of having to repeat a great deal of work, he stops at this point. He takes this new layout to Ted, particularly to show him the current source and the input pair configuration. He asks what Ted thinks of it.

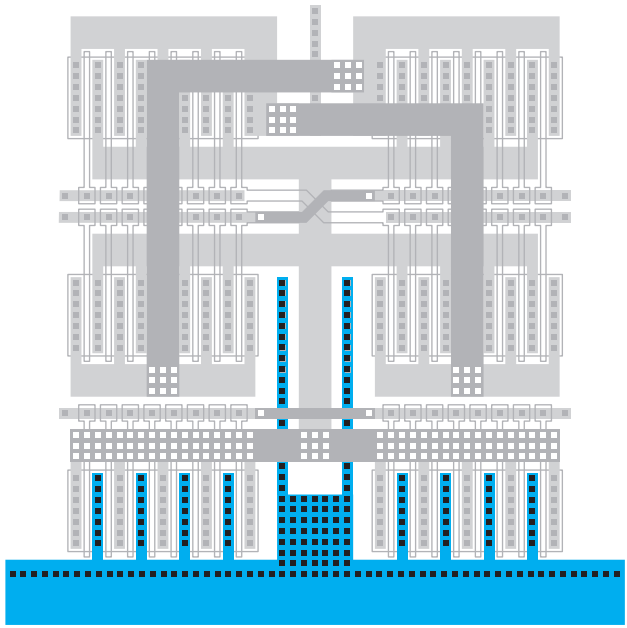


Figure CS1–48. *Ground of current source device connected.*

"Hey, that's finally starting to get somewhere," says Ted. "I like the symmetry. I see what you're doing. I really like it. It looks like you have almost finished this."

"Great," says Bill.

Back at his layout screen, Bill realizes he is on the final home stretch. All he has to do is build and place his final two PMOS devices, M1 and M2, and he will be done.

Drawing on the symmetry he already has in his existing layout, Bill decides to work on a single device and then copy the layout to save some time.

Bill uses the layout tool to create his first PMOS device.

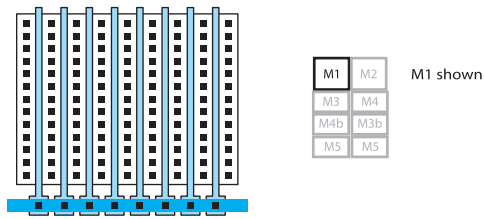


Figure CS1–49. *PMOS device as created by the layout tool.*

Referring back to his notes, and remembering what Ted had asked him to do with his N well contacts, Bill decides to completely enclose his PMOS device with N well contacts to make sure his N well is adequately tied down.

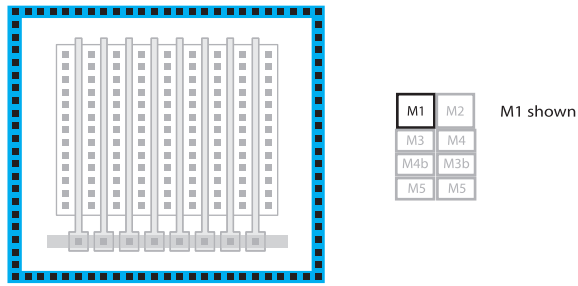


Figure CS1–50. *N well contacts added.*

Having placed his N well contacts, Bill now places his power rail, which not only needs to connect to the PMOS device, but also needs to connect to the N well contacts. Bill combines his power rail and N well contacts into one piece of metal.

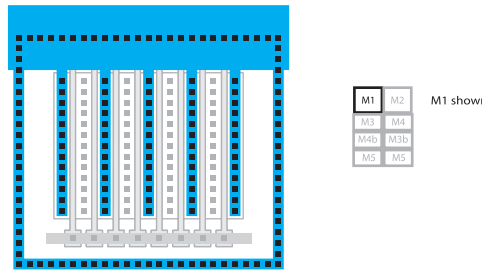


Figure CS1–51. *Positive power rail connected to device and N well contacts.*

Referring back to the schematic diagram, Bill realizes that the other side of this PMOS device is also connected to the output wires. Not only do the output wires interface with the outside world, they also carry the same current that drives the input diff pair. Bill realizes he must make this connection a 2-grid wire also.

Unfortunately, Bill does not have room to connect the other side of this device with a 2-grid wire. So, he stretches his poly gates and N well contacts in order to make room for a 2-grid wire.

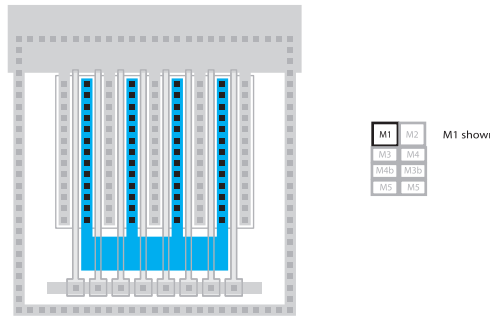


Figure CS1–52. *Output of PMOS device connected with a 2-grid wire.*

Bill is almost finished.

He knows Ted is going to want to get substrate contacts close to this device. So, Bill completely encircles the PMOS device with substrate contacts.

Feeling confident, he copies the layout of this device to create the final transistor, M2. Trying to save space, he merges the substrate contacts between the two devices. At this point, Bill adds two small NAC diodes to the gates of each device. He has been caught out by this before.

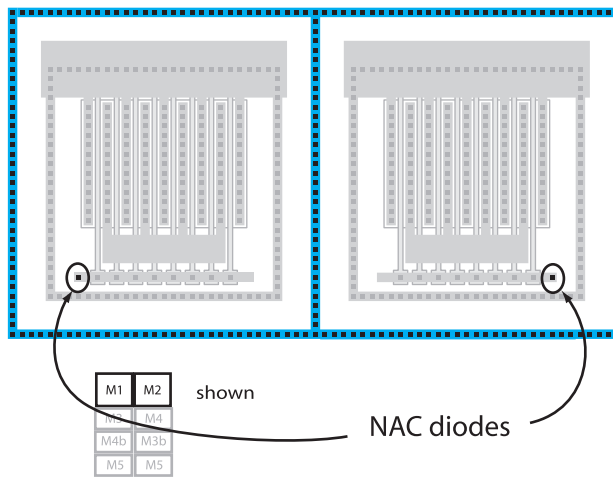


Figure CS1–53. *Final PMOS layouts with all NAC diodes.*

Unfortunately, having the substrate contacts between the two devices means that Bill has to create a cross-over in Metal Two to join the two halves of the power rail.

Similarly, the gate connections of the two devices also need to be joined in Metal Two.

Bill makes these two connections using Metal Two and adds a small piece of Metal One to join the output wire of device M2 to its gate connection as specified in the circuit diagram.

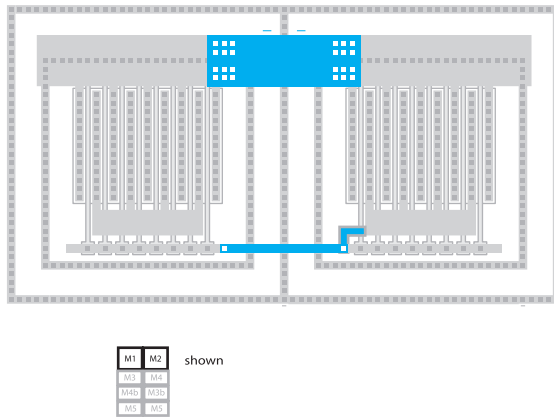


Figure CS1–54. *Power rails and gate connections made in Metal Two.*

Bill places his two PMOS devices, along with their substrate contacts, into the layout that he has already created. He makes sure that he carefully aligns the centerline of these two PMOS devices with the centerline of his input diff pair and current source.

Bill connects his PMOS devices to the output metal of his input diff pair with a 2-grid wide piece of Metal Two and places more substrate contacts to encircle the entire cell. Bill has finished. However, he decides to save himself some work at the top level of his chip by wiring his input and output wires inside his main op-amp cell. Referring back to the bond diagram, Bill adds some extra wiring to his output traces to enable him to easily connect to his output bond pads.

Similarly, Bill adds some extra metal to his input traces to save himself some work at the chip assembly phase of the project.

Observation

Bill is beginning to anticipate Ted's concerns from having worked with him for two weeks on this cell. Imagine how smoothly a team can operate after working together for years.

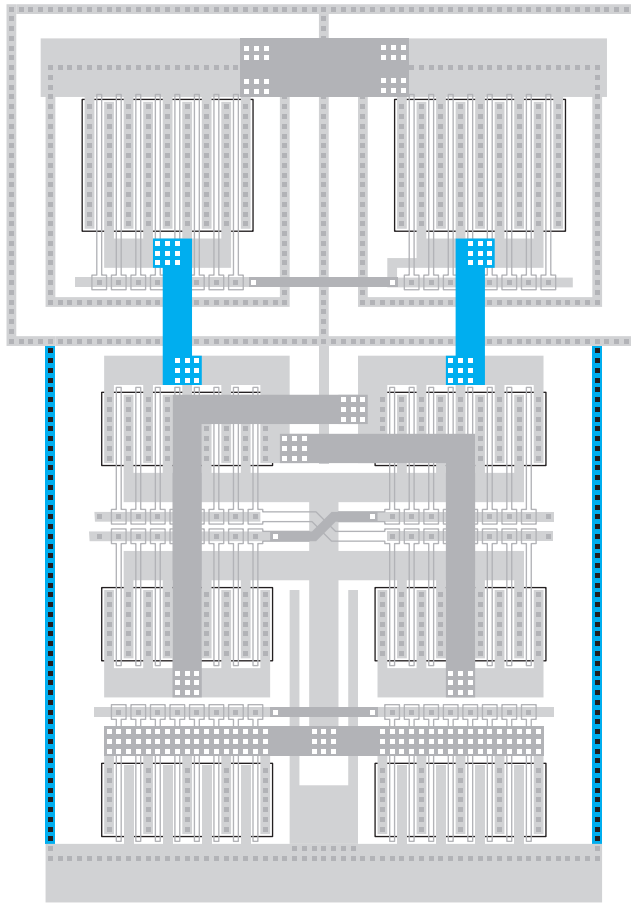


Figure CS1–55. *Connecting PMOS to NMOS in Metal Two.*

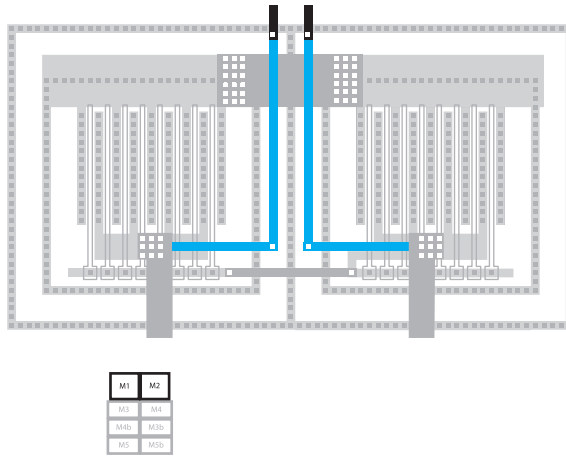


Figure CS1–56. *Symmetrical output traces heading toward bond pads.*

Finally, Bill is happy. He thinks he's done everything Ted wants.

Ted looks at it, and says "Great. That's wonderful. That's a much better layout than the first one you did. You can start working on the full chip now. I approve the layout of the op-amp."

Here is Bill's final layout of one op-amp. He will copy this layout three more times, since there were four op-amps required for the chip. He will flip or rotate each layout as required, to fit the bond-out plan.

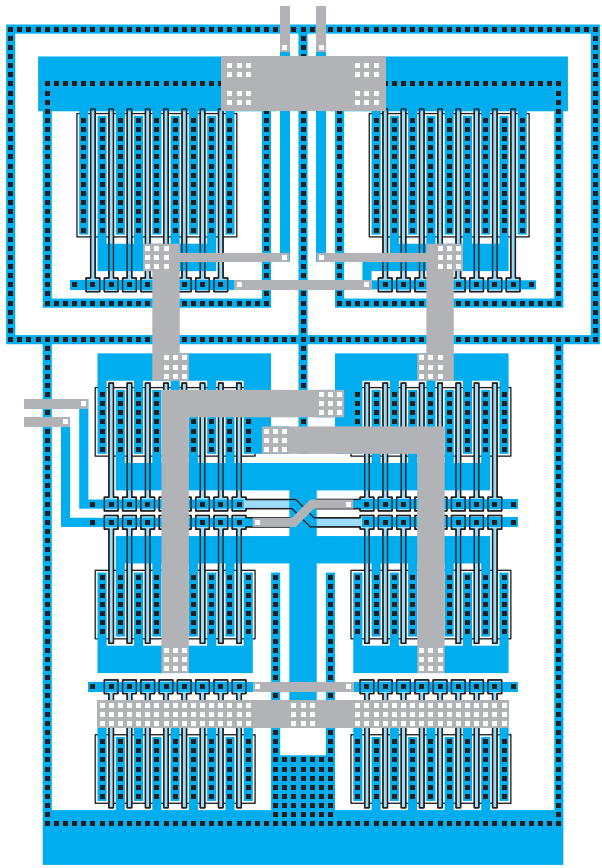


Figure CS1–57. *One finished op-amp. Most excellent layout.*

The Chip Is Assembled

Bill asks Ted, "Do you have a final schematic of the final chip, because that's what I need to work on now."

Ted says, "Not quite yet, but I do have a schematic of one of the corners with all the pads and the ESD devices on it. But you'll probably want that as a schematic to work to anyway because there are four of these and we want them all to be identical. So having a schematic of just one corner makes your life easy."

"Yes," says Bill. "That's really useful."

Bill is given the following schematic.

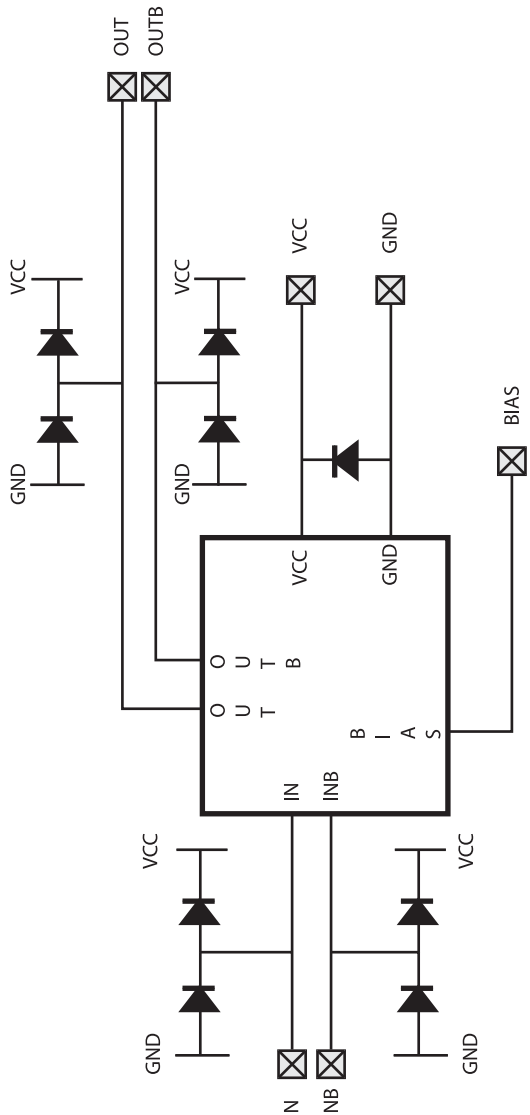


Figure CS1–58. *Schematic of corner.*

Ted begins to explain the schematic to Bill. “I have added ESD diodes to the inputs and the outputs. There are also ESD diodes across the power and ground rails.”

Bill says, “Um. I’m not really familiar with the analog ESD protection scheme that you’re talking about there. Can you explain it a bit more?”

“Sure,” says Ted.

Ted draws a box to represent his circuit and diode symbols to represent the ESD diodes.

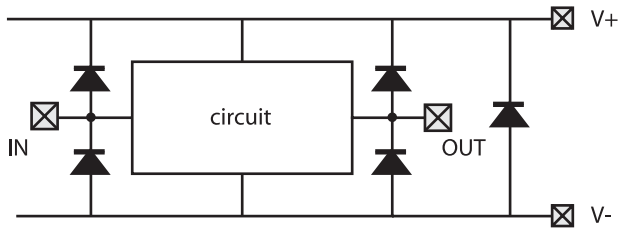


Figure CS1–59. *ESD diode protection, in both directions, on every pin.*

He continues, “What we’re trying to do here is protect every pin to every other pin. When someone picks up this chip with their hand, they could cause a static zap that could destroy everything. So, we want to give the zap an easy path to run through instead of running through our circuit.

“We have no idea what pins will be touched when the chip is handled, so we have to cover every option. There has to be extra protection circuitry to protect our real circuit from any ESD zaps.

“Remember that ESD voltages can be up in the thousands of volts range. So, any currents that flow can also be very high, even if it’s just for a short period of time.

“For instance,” says Ted, “if I was to pick up the chip and accidentally zap one of the input pins to one of the output pins, then in this case I would have a discharge path that looks like this. (See Figure CS1–60.)

Let’s assume that we have a positive ESD voltage on the input pin. Because the input pin voltage is positive with respect to the output pin, one of the diodes on the input pin becomes forward biased. And, one of the diodes on the output pin becomes reverse biased.

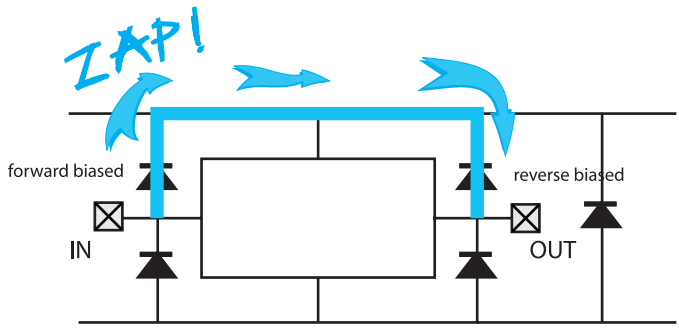


Figure CS1–60. *IN electrical zap is positive with respect to OUT.*

"The forward biased input diode provides an impedance that is much lower than the impedance of our real circuit, so any ESD current will flow through the forward biased diode instead of into our real circuit.

"The ESD voltage is quite high, so the reverse bias diode on the output pin goes into breakdown and gives us another low impedance path. With these two diodes conducting, all the energy of the ESD event is bypassed around our real circuitry. We are protected.

"That's one example. Another example is if I zap the input pin negatively with respect to the output, then I get the current flow the other direction. And one of the output diodes becomes forward biased. The idea is the same, the direction is reversed."

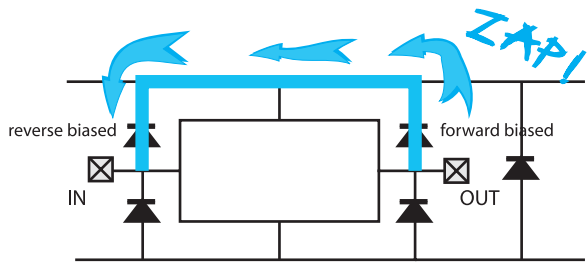


Figure CS1–61. *IN electrical zap is negative with respect to OUT.*

"My third example is that I zap the input pin positive with respect to the positive power pin. In this case, the input pin becomes more positive than the power pin, and I get one of the input diodes forward biased."

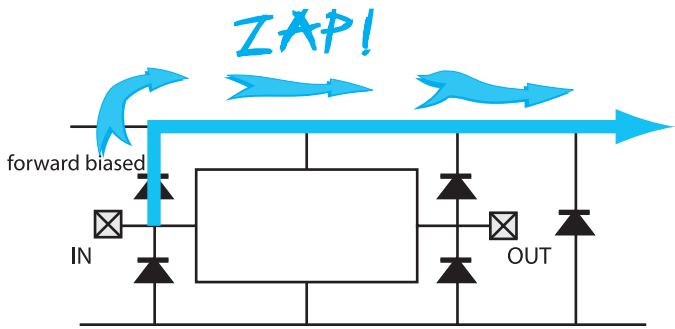


Figure CS1–62. *IN electrical zap is positive with respect to circuit positive.*

"With this type of protection scheme, we can protect every pin against every other pin and make sure our circuit doesn't die," Ted concludes.

"Ah. I understand now," says Bill. He goes off with his schematic of the corner to start work.

The first thing Bill notices is that all of the inputs and the outputs have identical ESD structures. Most of his signals have a pad and two diodes. Bill thinks to himself, "If this is going to used repeatedly all over the chip, I may as well make myself a cell that I can just build once and place as many times as I need it."

Bill decides to make a small piece of layout that corresponds to the repeated protection circuitry, namely a pad and two diodes.

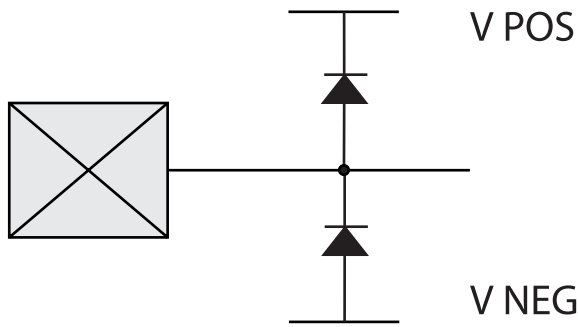


Figure CS1–63. *Bill builds a cell to be used repeatedly.*

Bill first of all gets his CAD tool to create one of the diodes. An individual diode appears with an outer ring and an inner block.

Bill places two of the diodes and his bond pad randomly at first, so he can look at them to work out how he wants to place them permanently.

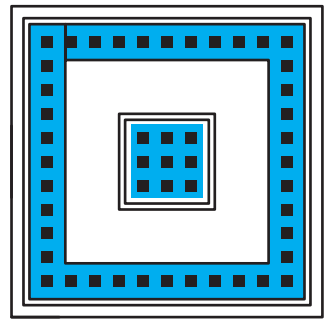


Figure CS1–64. *ESD diode layout.*

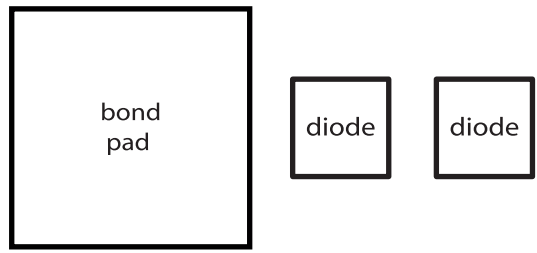


Figure CS1–65. *Initial placement to see what we are working with.*

Bill notices that the two diodes, side by side, are roughly the same width as the pad. He tries to minimize the area of his cell by placing the diodes as close as he can to the pad. He looks up all the design rules to make sure he is not placing components too close to the pad. He ends up with a rough placement with the two diodes side by side over the pad.

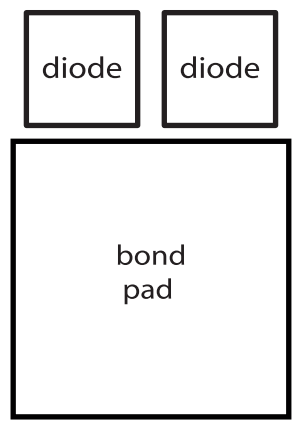


Figure CS1–66. *Space-saving placement of diodes and pad.*

Bill looks at the schematic and notes that the positive rail and the negative rail go to all of the ESD diodes on his inputs and outputs. Bill figures that he will have to wire the positive and negative supplies almost completely around the outside of his chip.

In order to make his task slightly easier, Bill decides to build the power rail into his ESD cell.

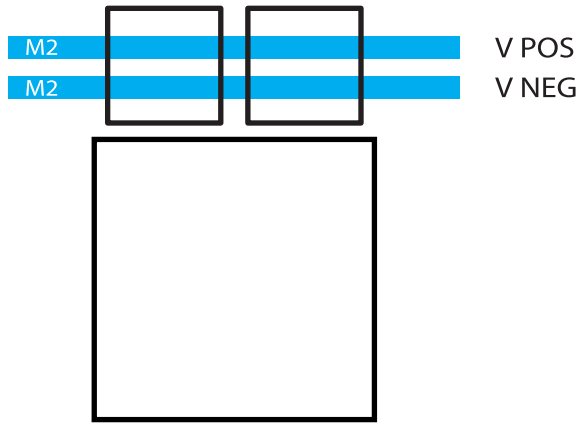


Figure CS1–67. *ESD supply rails in Metal Two, built into the chip.*

If Bill places his power rail in Metal Two, then he can run the Metal Two over the top of his ESD diodes. Not only does Bill have to get the signal from the pad to the diodes, he has to get the signal past the diodes and into the circuit, so Bill pulls his diodes apart a touch.

Bill creates an underpass in Metal One to get his signal from the pad, which passes between the two protection diodes and underneath the two power rails. The pad is Metal Two, so he needs a small piece of Metal Two to connect to the underpass he has just made. (See Figure CS1–68.)

All Bill has to do now is find out which way around the diodes are, and connect them.

Bill examines the diode layout, but he is not sure if the cathode of the diode is the chunk in the middle or the outside. If he is not careful he might connect the diodes back to front. They must be in the proper direction.

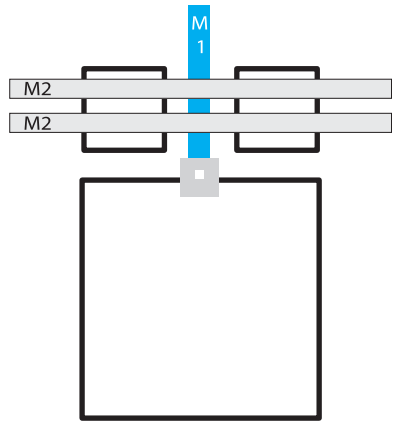


Figure CS1–68. *Metal One underpass to get signal from bond pad past the diodes.*

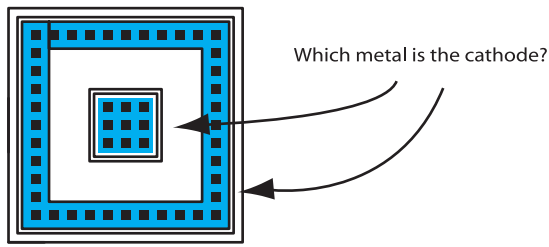


Figure CS1–69. *Diodes can easily be connected backward.*

Observation

An easy way to find out which terminal is the anode or cathode is to examine the layers that make up the diffusions of the diode. Diodes are usually PN junctions. So, the terminal that is connected to the N type diffusion is always the cathode.

Bill checks his design manual and finds a drawing that answers his question.

From his schematic, Bill realizes that the signal wire from the bond pad needs to connect to the outer ring of the diode that connects to ground, and the inner square of the diode connects to the positive supply.

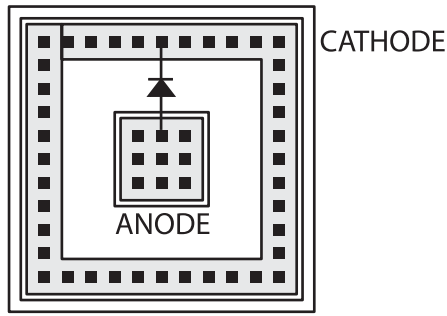


Figure CS1–70. *N type diffusion connection is always the cathode.*

Bill decides to start his connections with his VCC power rail. For no good reason other than it's Wednesday, he decides to have his VCC rail closest to the bond pad. Again, because it's Wednesday, he selects the left-hand diode outer ring to connect with the VCC rail being careful to check the polarity of the diode.

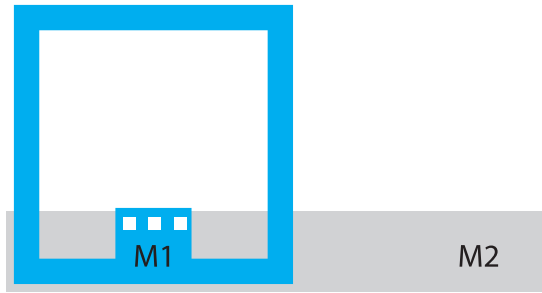


Figure CS1–71. *VCC Metal Two rail connection to outer ring of ESD diode.*

Observation

Very often when you attend a layout review, you will be asked questions such as, "What made you decide to hook those components like that?"

Typically, the piece of layout in question is what I call Wednesday afternoon layout. There might be no logical reason for the choices you made.

In this case, there are two answers to their question. The first answer requires you to babble lots of techno-geek words for five minutes trying to explain all the technical reasons why you made the choice. The second answer is to shrug your shoulders and say, "It was Wednesday. It was

windy. A particularly strong cosmic ray passed through my brain and triggered certain neurons. You figure it out." That's a Wednesday decision.

Count on Murphy's Law: If anyone in the layout review asks you about any choice made in your layout, it will always be the Wednesday afternoon decision they ask about.

Bill now needs to connect the other side of this diode to his signal trace. He needs to use Metal Two to jump over the Metal One ring of this diode in order to connect to the signal underpass he created earlier.

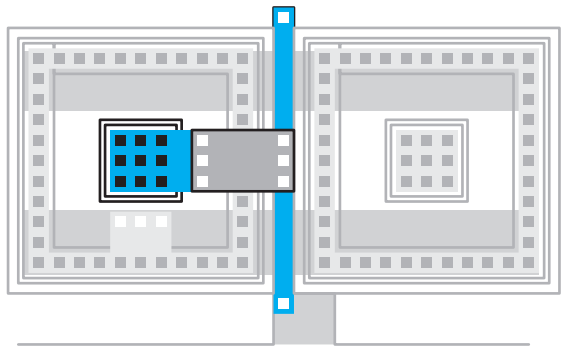


Figure CS1–72. *Signal connection to positive rail ESD diode.*

The other diode's outer ring also needs to connect to the signal wire. Because the signal underpass is in Metal One, it is a simple matter to connect the signal to the outer ring of the second diode in Metal One.

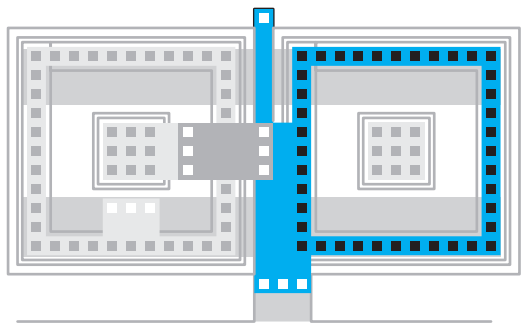


Figure CS1–73. *Signal connection to outer ring of negative rail ESD diode.*

The final connection of this ESD structure is the ground rail connection. The ground rail needs to connect to the inner terminal of the second diode.

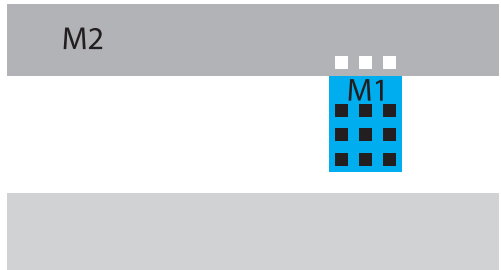


Figure CS1–74. *Ground rail connection in Metal Two, to inner terminal of ESD diode.*

Finally, Bill places a few substrate contacts along the top of his ESD structure.

His cell is complete.

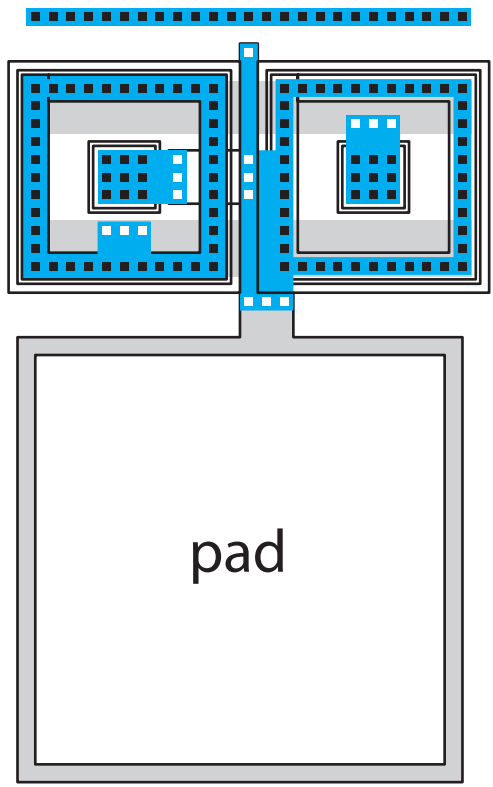


Figure CS1–75. *Completed ESD protection cell.*

Observation

Even though the schematic does not call for it, Bill is creating some cell hierarchy to make his life easier. However, Bill does not have a schematic to verify his new layout. Bill will need to rely on the verification of the full corner layout to make sure he has wired this new cell correctly.

This extra hierarchy is a good idea, since it saves lots of time.

An option for Bill is to ask Ted to create a schematic for him to verify this new layout. Maybe Bill could even ask Ted to modify his corner schematic to include this extra level of hierarchy.

Work with your circuit designer. Try to match the hierarchy of your layout to the hierarchy of your schematic. With large chip databases, verification (LVS) of each level of hierarchy can help you identify where wiring errors occur.

With his ESD pad finished, Bill opens a new cell. He places his amplifier cell, two ESD pads for his inputs and two ESD pads for his outputs.

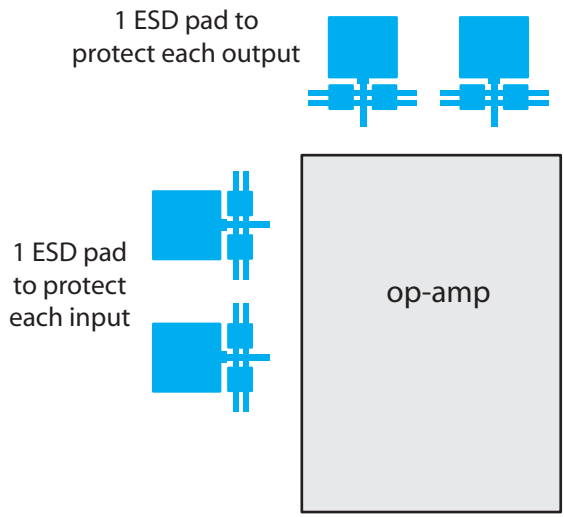


Figure CS1–76. *Initial placement of op-amp and protection devices.*

Bill arranges his ESD pads around his op-amp such that his inputs are nicely centered around the ESD pads and his outputs are nicely centered around his ESD pads. Bill then wires all of the ESD power rails.

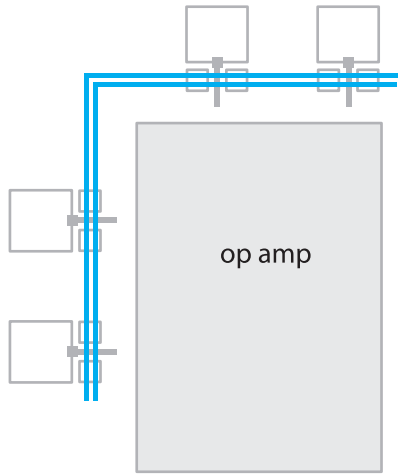


Figure CS1–77. *ESD supplies wired to protection devices.*

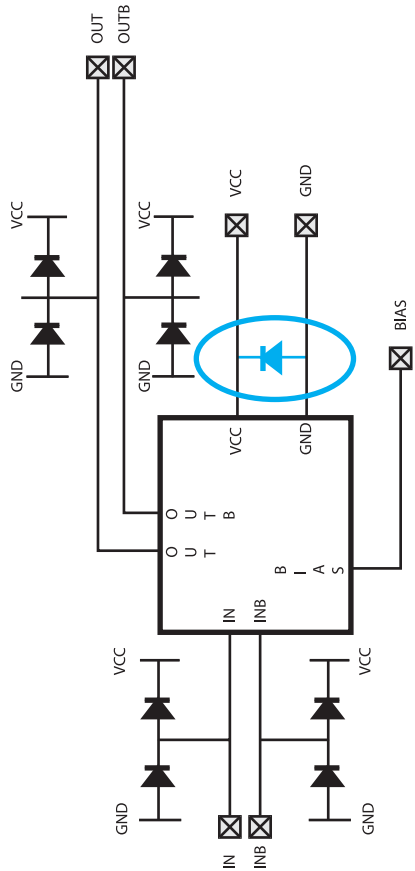


Figure CS1–78. *One more diode.*

Bill is glad that he has this nice, unique power rail just for his ESD, wrapped around the outside of all of his op-amp circuitry.

As Bill refers again to his schematic, he realizes there is an extra ESD diode that needs to fit across the supply.

Bill decides to place this ESD diode underneath the Metal Two power rail, in the top left-hand corner of his layout. It will be easy to connect since both VCC and ground are easily accessible there.

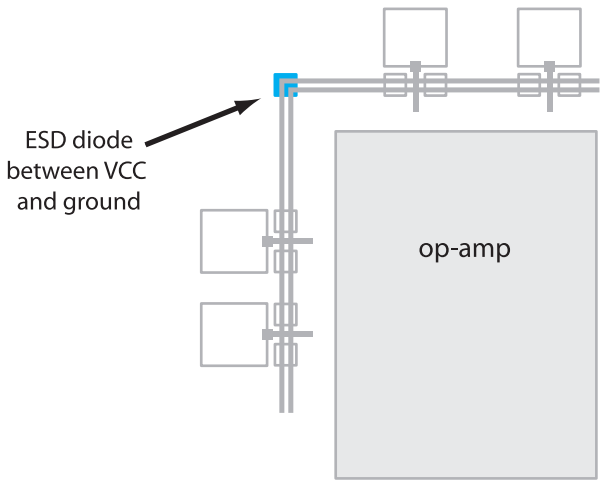


Figure CS1–79. *Placement of supply protection diode.*

Bill now has his input and output bond pads taken care of, and all his ESD diodes are in place. All he needs to do now is place and connect the bond pads for his power and ground wires, and hook the inputs and outputs of his op-amp to the ESD cells. At that point he will be done with the corner.

Bill looks at the preliminary bonding diagram he was given. He notices his VCC needs to go in the far left hand corner. His ground pad needs to be common to all four amplifiers. So, Bill decides to place the ground pad at the next level of hierarchy once he knows how much room he has to play with. Bill then places his VCC pad.

Luckily, for Bill, his outside edge power supply for his ESD diodes is also VCC. If he had made the outside ESD rail his ground, then he would have had to bring his VCC wire in and underneath the ground rail.

Bill puts a big chunk of Metal Two from his pad to his outer ESD rail.

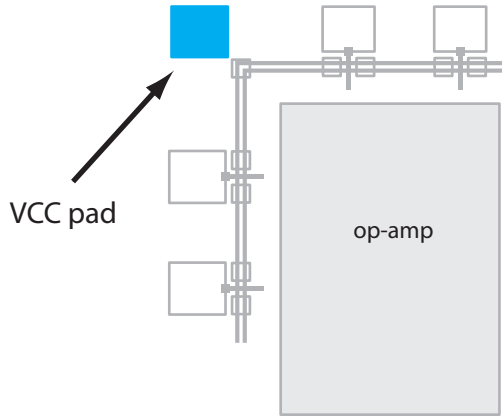


Figure CS1–80. *Placement of VCC pad.*

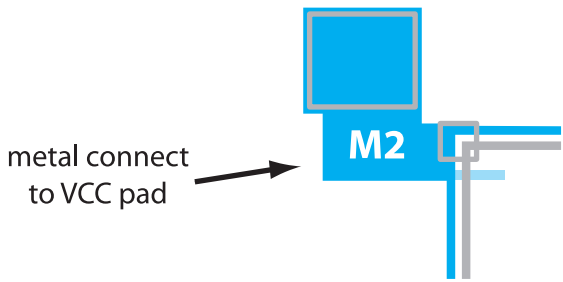


Figure CS1–81. *Connection of VCC pad.*

He also has to hook up the VCC pad to the VCC rail for the op-amp. He has already had to do some jumping up and down in metals inside his op-amp anyway, so Bill is not too worried about having to continue that trend.

Bill extends his Metal Two in the op-amp to connect to a Metal One underpass joining his op-amp VCC rail to the bond pad.

Bill now wires his input and output pad cells to the op-amp, making sure his wires have identical lengths.

Bill tries to keep his I/O wiring the same length because they are differential wires. That's why he does some odd things with the inputs in particular, to try to maintain the differential nature of the wire.

At this point, Bill declares his corner finished.

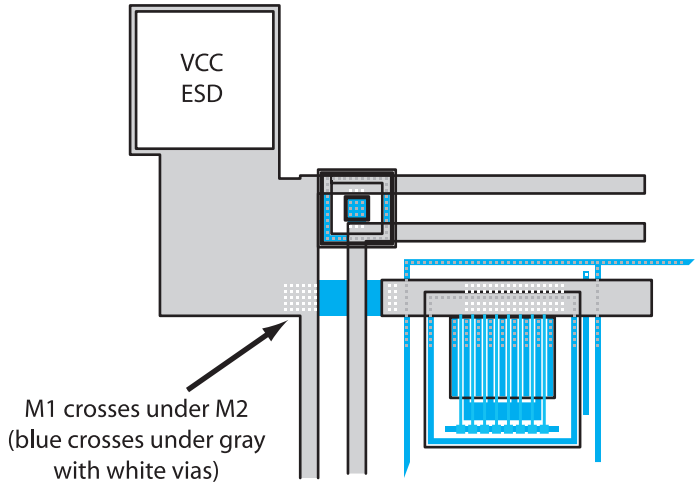


Figure CS1–82. *Op-amp VCC connection to bond pad.*

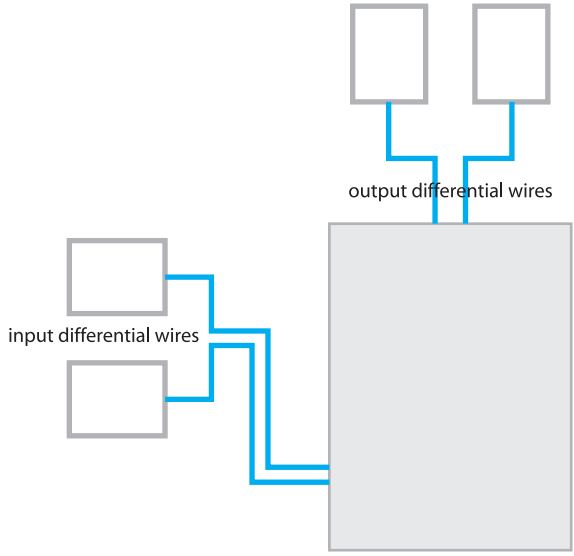


Figure CS1–83. *Input and output wiring.*

Bill goes back to Ted and says, "Hey, Ted. I've got my corner mainly finished. Do you have the top level circuit diagram yet? I think I can have this chip finished in the next couple hours."

Ted says, "Actually, I've just finished drawing it." He gives Bill the second schematic. The top level diagram shows Bill the four op-amps. Each of the

four bias wires from each op-amp is now connected and pulled out to a bond pad with wires. All four of the grounds are commoned.

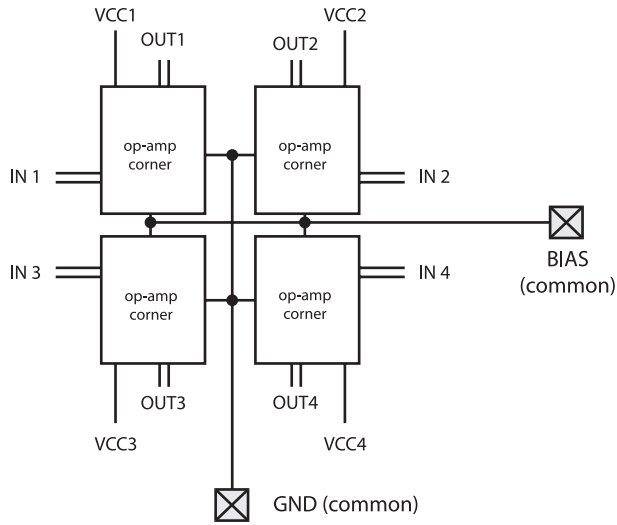


Figure CS1–84. *Top level schematic.*

Bill is now able to finally connect all of his circuits to each other.

Bill places his corner cell four times.

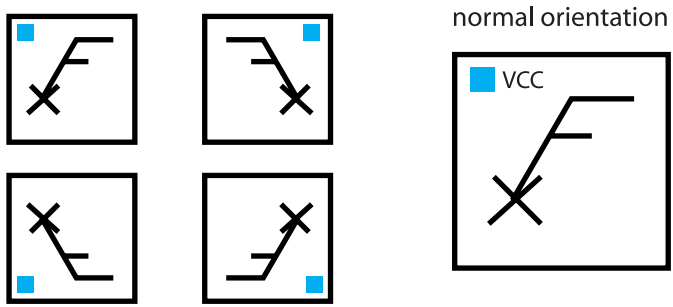


Figure CS1–85. *Op-amp placement and orientation.*

He has to flip some of his op-amps across horizontal and vertical axes in order to ensure that the VCC pins are in each corner. He knows to put the VCC pads in the corners because he has been looking at the top level bonding diagram.

Bill now has to add a bond pad for the ground and one for the bias, according to the bonding diagram he's been given. His ground pad has to go to the left side of the chip; the bias pad to the right side of the chip.

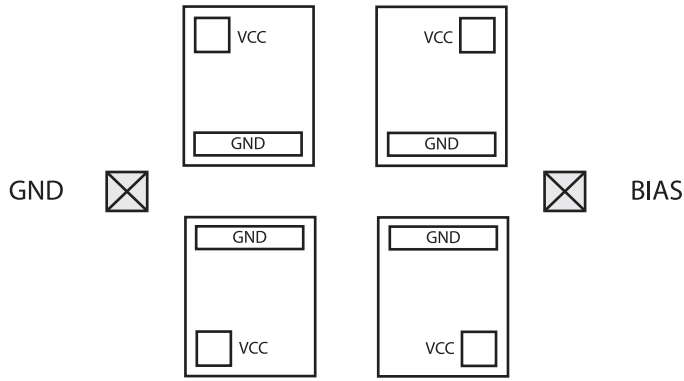


Figure CS1–86. *Ground and bias pad placement.*

Bill now concentrates on his ground wiring. He notices from the schematic that all of his input and output ESD diodes are connected to the ground wire. So, Bill hooks up all of his ESD grounds to each other and joins them at the pad.

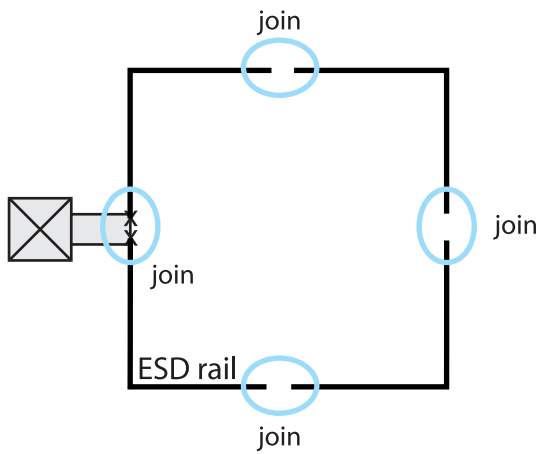


Figure CS1–87. *ESD ground wiring connections.*

The four op-amps each have their ground rails pointing toward the center of the chip. Bill is able to create a nice, big, fat bus that runs through the middle

of the chip. He fills up every space he can to reduce the resistance of the ground wire.

At this point, Bill has completed his ground wire.

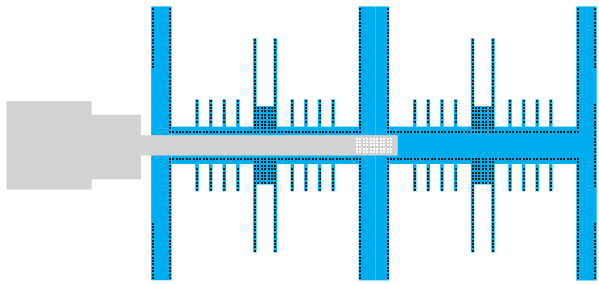


Figure CS1–88. *Nice fat ground connection.*

Bill's foresight and planning has made his chip assembly quite easy.

He now needs to wire the common bias node.

Bill looks at his circuit diagram and thinks to himself, "Well, hang on a second. Doesn't this bias signal also need ESD protection? I don't see any on the schematic. I thought we had to protect every pin against every other pin."

He goes back to Ted and says, "Looking at your schematic, I don't see any ESD protection for the bias pin."

Ted replies, "Oh yeah, you're right. Well spotted, Bill. What should we do about that?"

Ted and Bill think for a while. Then Ted says, "Well, . . . This is a four supply chip with a single ground. You need to protect this bias pin not only to a negative rail, but also a positive rail. You have four positive rails to chose from, which one do you want to choose?" They think about it, and Ted pipes up, "I don't really think it matters."

So Ted fires up the software and starts to put ESD diodes inside the corner schematic.

"Hey," says Bill. " If you put ESD diodes on the bias signal in the corner schematic, that means I'll have four sets of ESD diodes. Don't you really need to put your diode on the top level?"

"Yeah, good point," says Ted. So Ted closes the corner schematic, opens up the top level schematic, and adds two ESD diodes at that level instead.

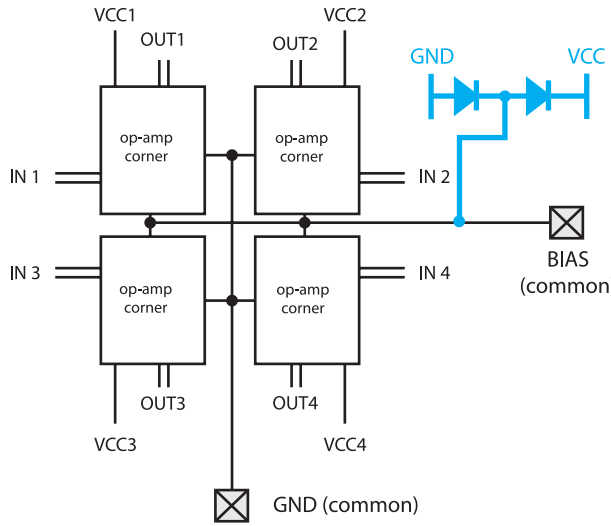


Figure CS1–89. *Addition of ESD diodes to bias pin.*

Bill returns to his desk and places one of the ESD pad structures for the bias signal. Ted has already told Bill he doesn't care which power he uses. "Just choose one," Ted said.

Bill decides to use the positive supply immediately above the bias pin as the supply for his protection diode. Bill connects this supply node and connects the ground side of his protection diodes to the ESD supply ring that runs around the whole chip.

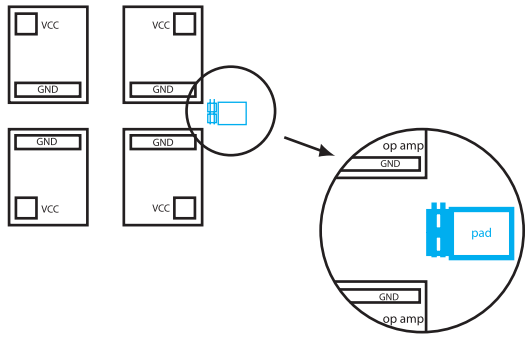


Figure CS1–90. *ESD protection for bias pin.*

Bill now wires his bias pad to the four bias pins of his op-amps.

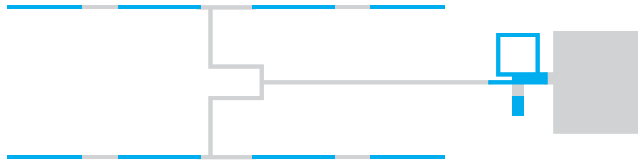


Figure CS1–91. *Wiring for chip bias pin.*

Bill thinks to himself, "I think I have everything. I have all the grounds hooked up to each other. I have all the bias wiring hooked up to each other. I've hooked up all the ESD grounds to the main ground. I think I'm finished. All I have to do now is put in the final copyright symbols, mask identifiers, and I'm done."

Bill spends the next 15 minutes putting his final touches on the chip layout and he is finished.

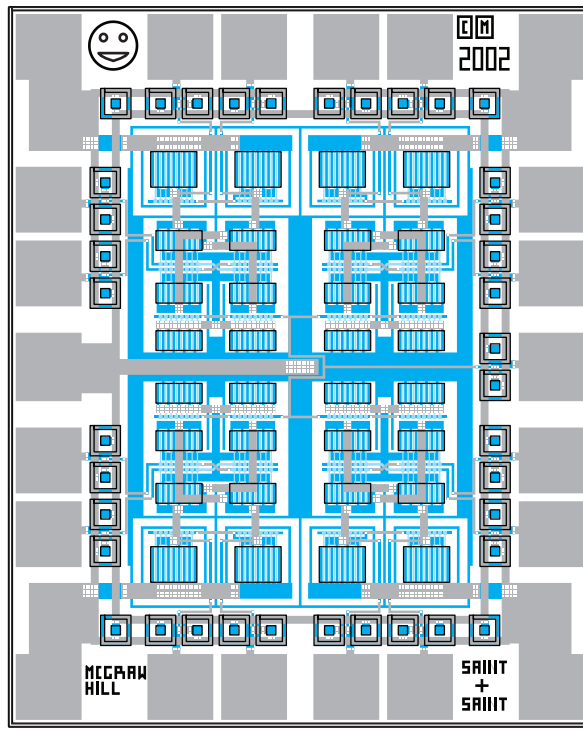


Figure CS1–92. *Final layout, suitable for framing.*

Packaging

The final thing Bill has to do is provide the packaging department with a final bonding diagram, based on the real die and the real package.

Bill takes his final layout, gets an electronic version of the package drawing from his packaging people, places his chip in the package and draws an accurate final bond drawing. Up until now they have only had a hand-sketched version. The assembly house has to make sure that all the bond angles are ok and the wires are not too long.

Because of all the good up-front work that Ted has done there are no problems with the bond-out and bond angles. The rough bond-out diagram he gave to Bill helped.

The final chip bonding diagram is shown in Figure CS1–93.

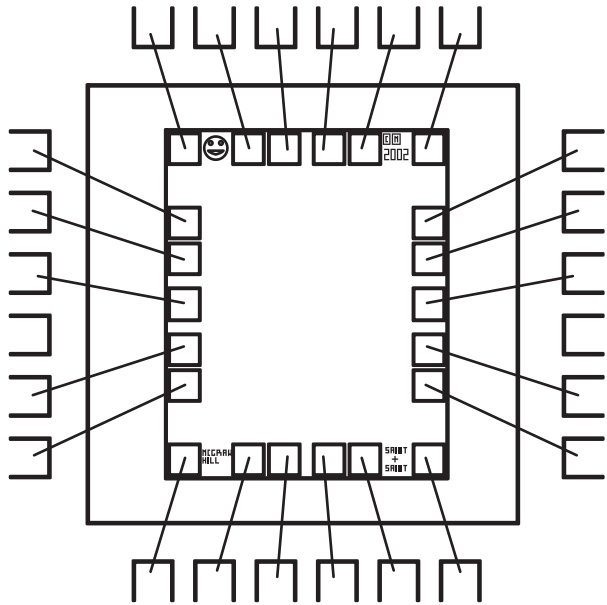


Figure CS1–93. *Final chip bonding diagram.*

Most bond diagrams don't need to show detail of the insides of the chip. They are only interested in the bond pads and some unique identifiers to show which way around the chip should be. The little happy face and other symbols are not necessarily just for fun—they are also a good reference for position and rotation.

Observation

Too many projects have been spoiled by having a completely symmetrical chip that bonds out to the wrong pins. Someone placed it the wrong way around.

Put some unique identifiers on your chip so that when you stare down through the microscope you can see quickly which way around the chip should be.

What kind of unique identifiers have you used, Chris?

Company logos.

And?

Oh, the name of the chip in one corner, the odd guitar here and there.

And?

The odd train here and there.

Get to the good one, Chris.

The odd Buzzby hanging from a telephone pole.

That's the one I was after. The Brits will get the joke. And you said you had people requesting different train engines?

Yes. Customers would come back asking for more chips and they would specifically request certain types of train engines on their chips. It can get pretty fun.

END OF CASE STUDY 1

Appendix

This appendix contains full chip individual net plots for:

- Outputs
- Inputs
- Ground
- Substrate contacts
- Powers
- ESD ring
- Well contacts
- Bias

Output Traces

Notice that the source-drain regions for the PMOS devices and the cross-quaded NMOS differential pair are also included, along with the anode and cathode metalization for the ESD diodes.

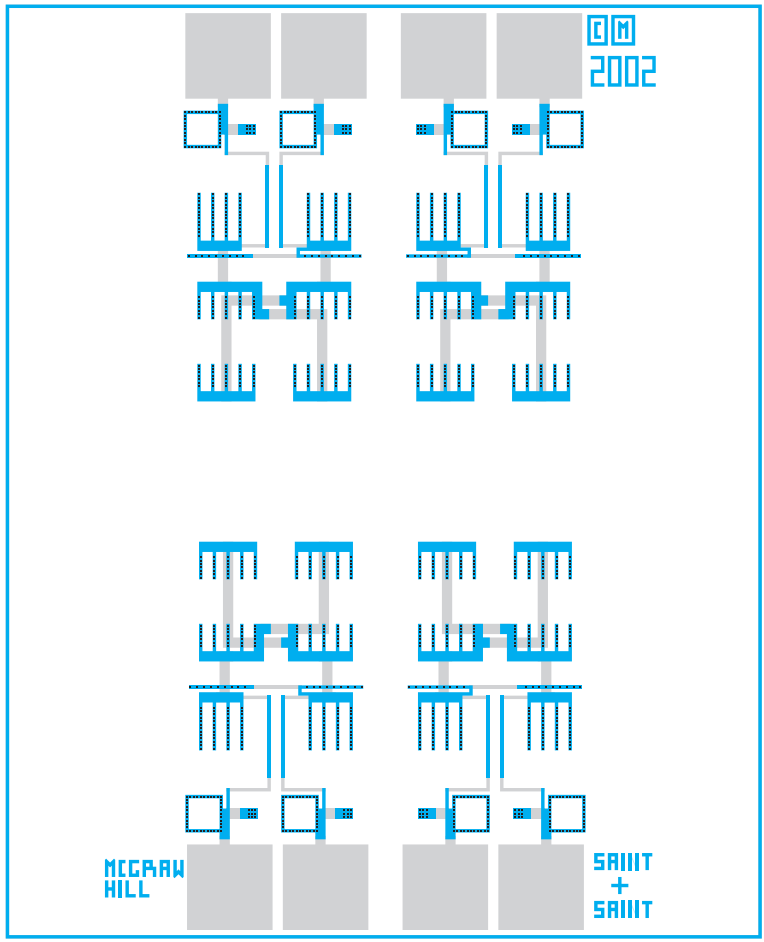


Figure CS1–94

Simplified Output Traces

This plot shows just the wiring that is associated with the output pads. Notice how the wiring is run symmetrically from the center of each pad to the PMOS devices ensuring even path lengths and identical parasitic capacitances.

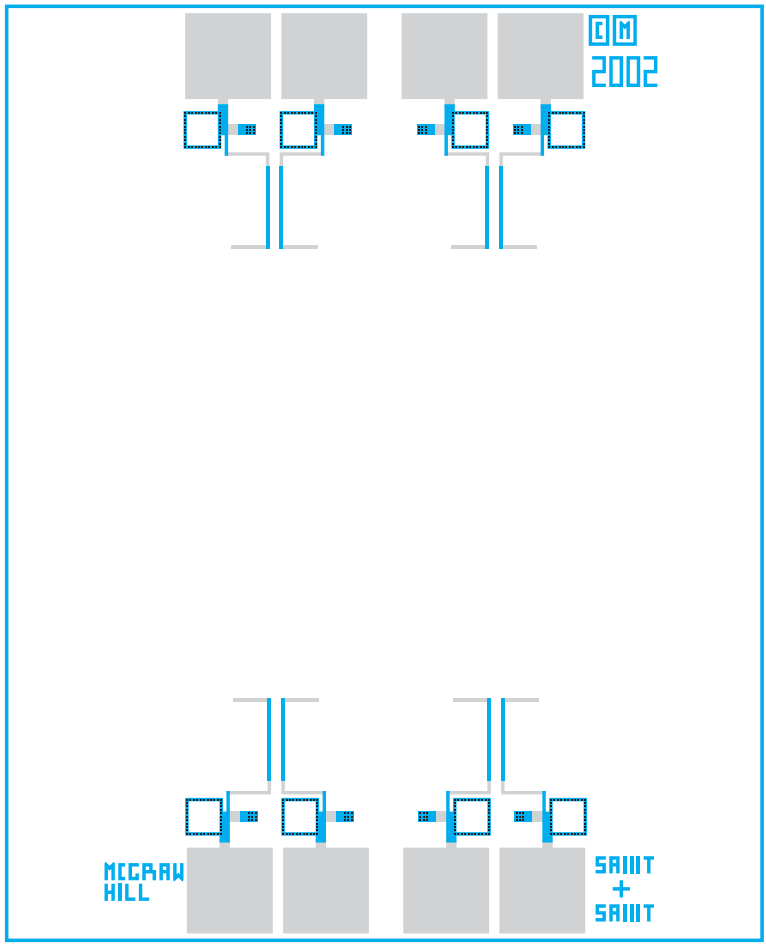


Figure CS1–95

Input Traces

This plot shows the symmetry of the amplifier input traces and includes the gate fingers of the input devices. As with the output traces, the inputs are wired symmetrically from the center of the bond pad to ensure balanced trace lengths and identical parasitic capacitance.

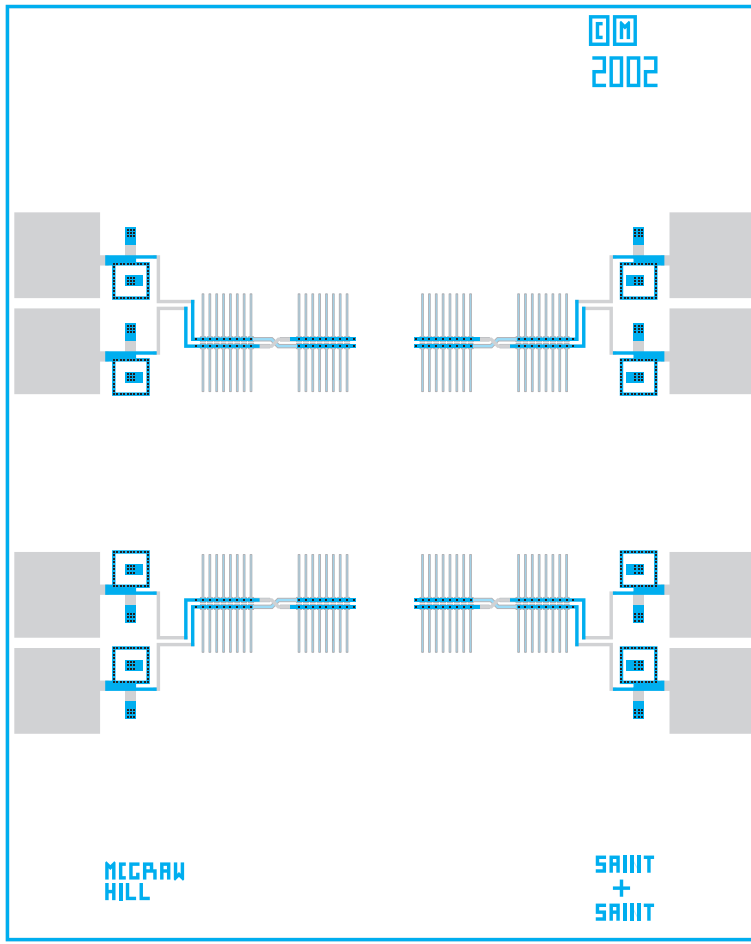


Figure CS1–96

Ground Wiring

This plot shows the entire ground wire.

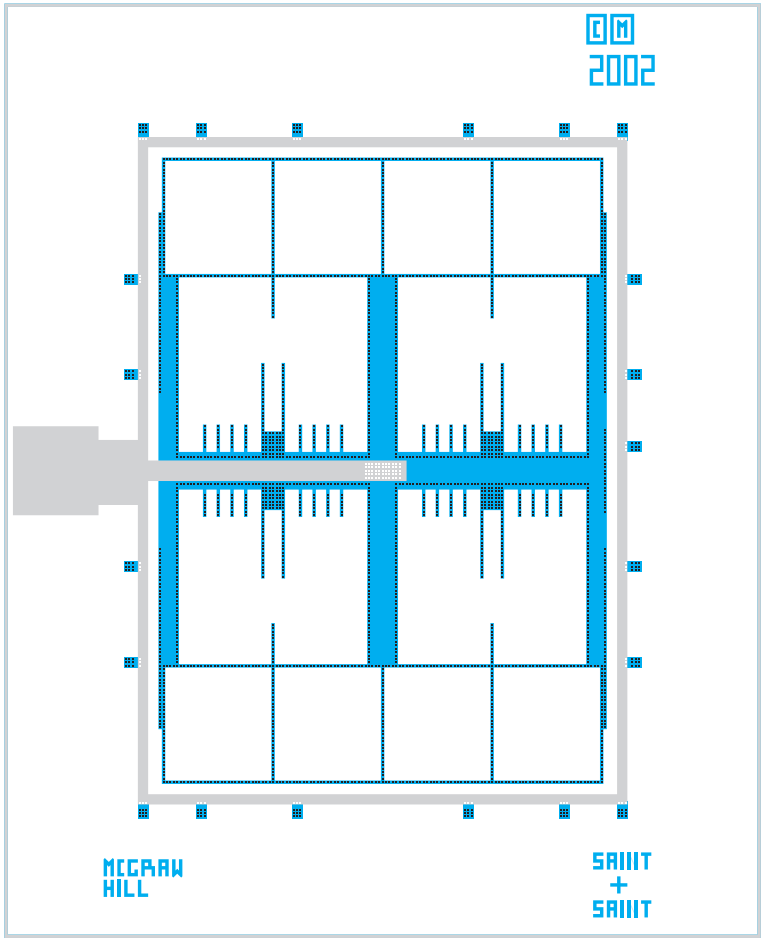


Figure CS1–97

Circuit Ground Wiring

This is a simplified plot of the ground wiring showing only the metal that is used to connect to circuit elements. Notice that the Metal Two wire from bond pad connects in the center of the chip to ensure that the current flowing from each amplifier is evenly distributed.

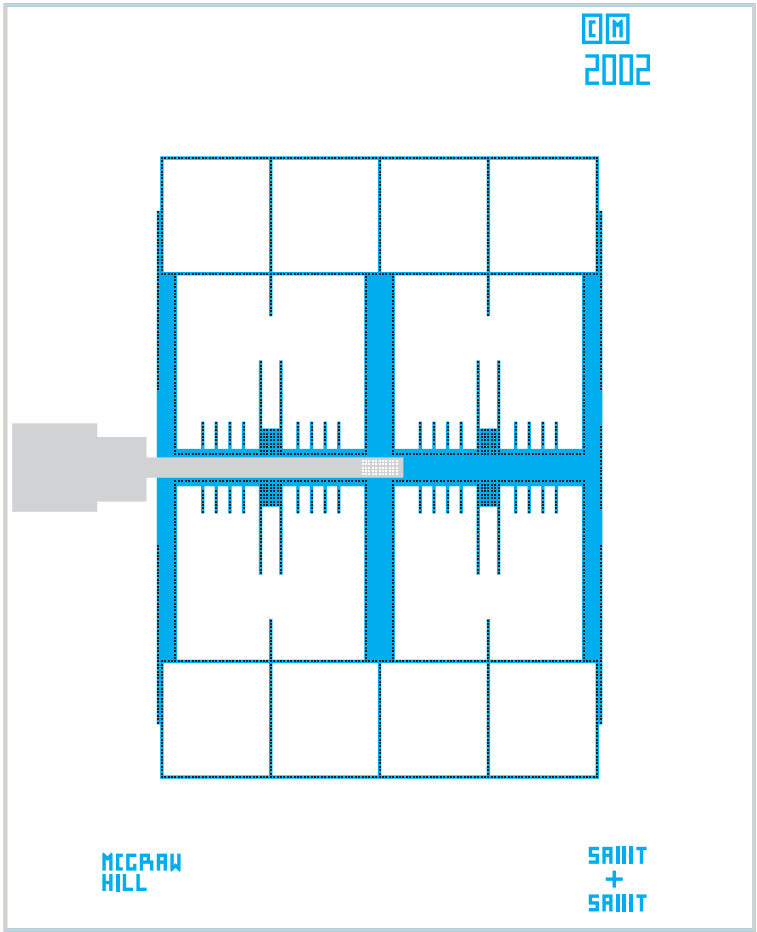


Figure CS1–98

ESD Ground Wiring

This simplified plot shows the ground wires associated with the ESD protection system. Notice that this wire creates a continuous ring around the active circuitry and that the ESD diode inner terminals (anodes) are connected.

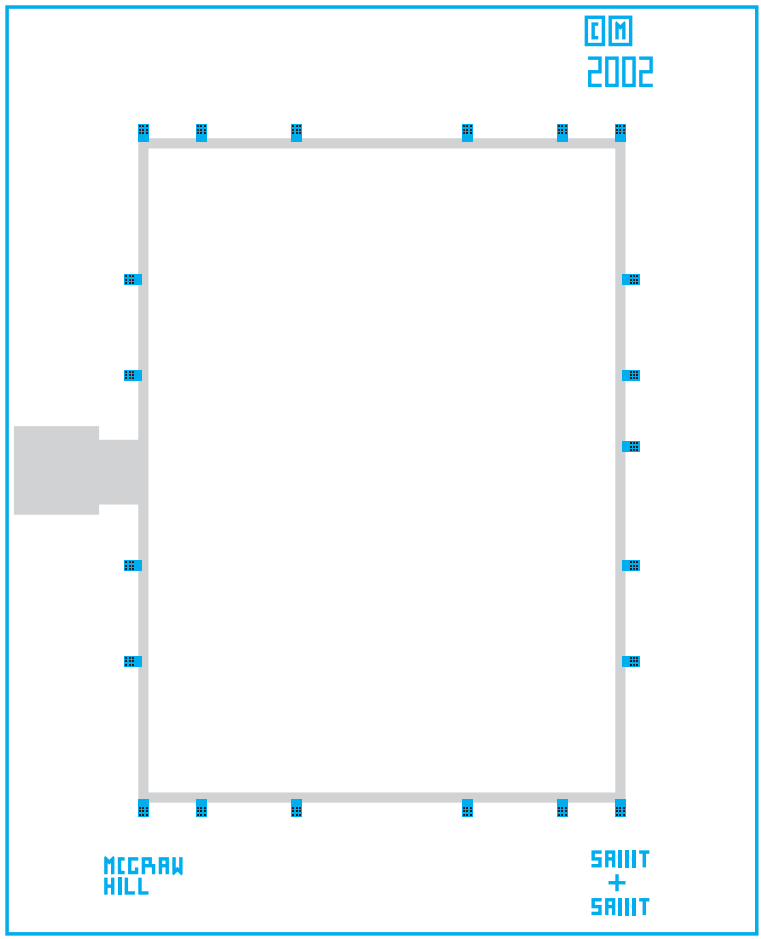


Figure CS1–99

Substrate Connections

This plot shows only the Metal One associated with the substrate connections. Notice how similar it is to the plot of the circuit ground connections. It is very common to combine ground wiring and substrate connections.

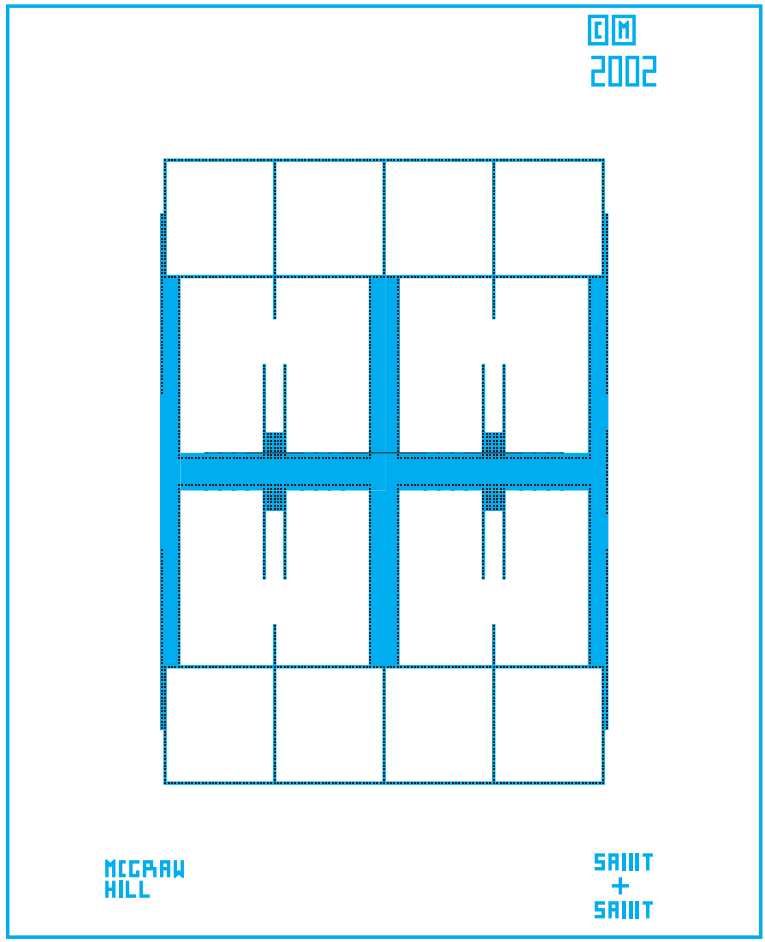


Figure CS1–100

Positive Supplies Wiring

This plot shows the wiring associated with the four circuit positive supply wires.

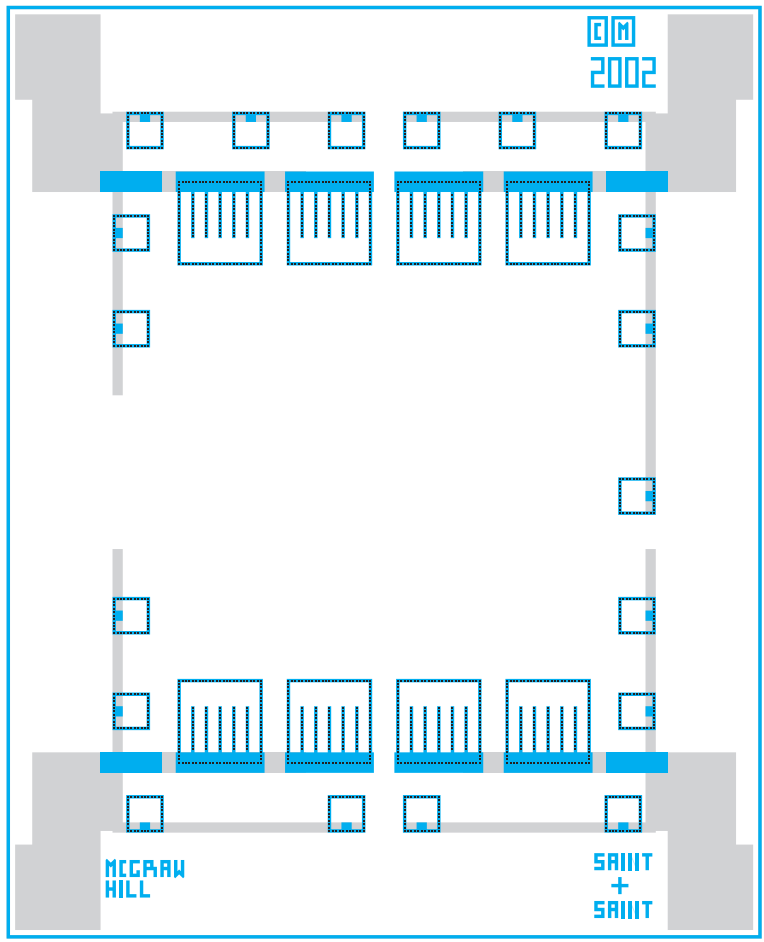


Figure CS1–101

ESD Positive Supplies

This simplified plot shows the positive supply wires associated with the ESD protection system. Notice that these wires do not create a continuous ring around the active circuitry and that the ESD diode outer terminals (cathodes) are connected. Also notice that one supply has an extra ESD diode for the bias pad.

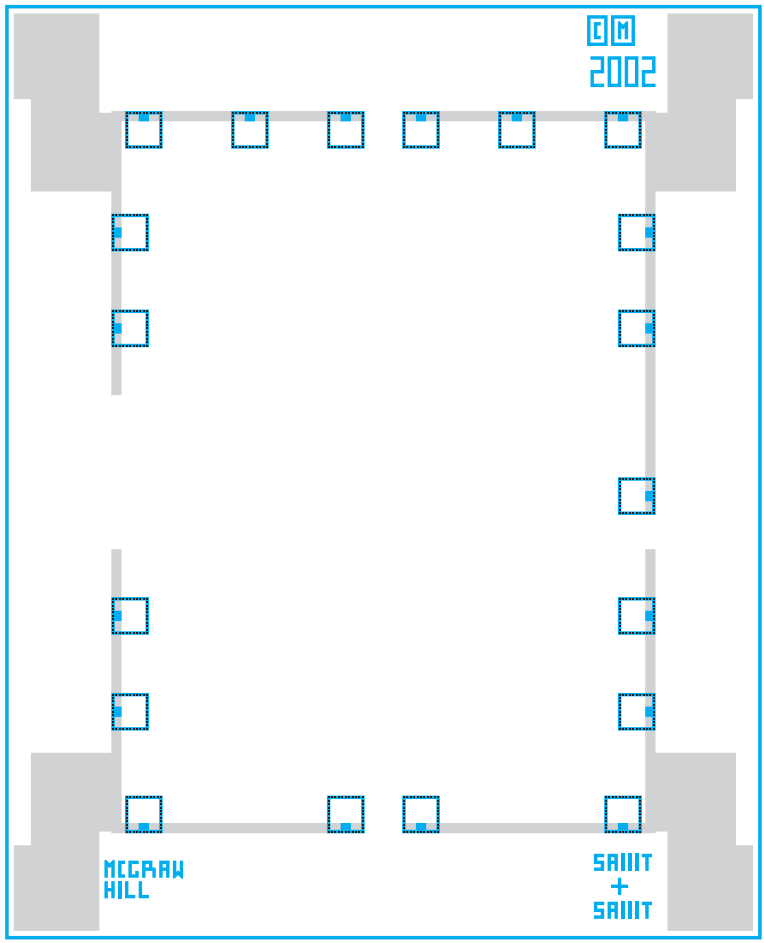


Figure CS1–102

Circuit Positive Supply Wiring

This is a simplified plot of the positive supply wiring showing only the metal that is used to connect to circuit elements.

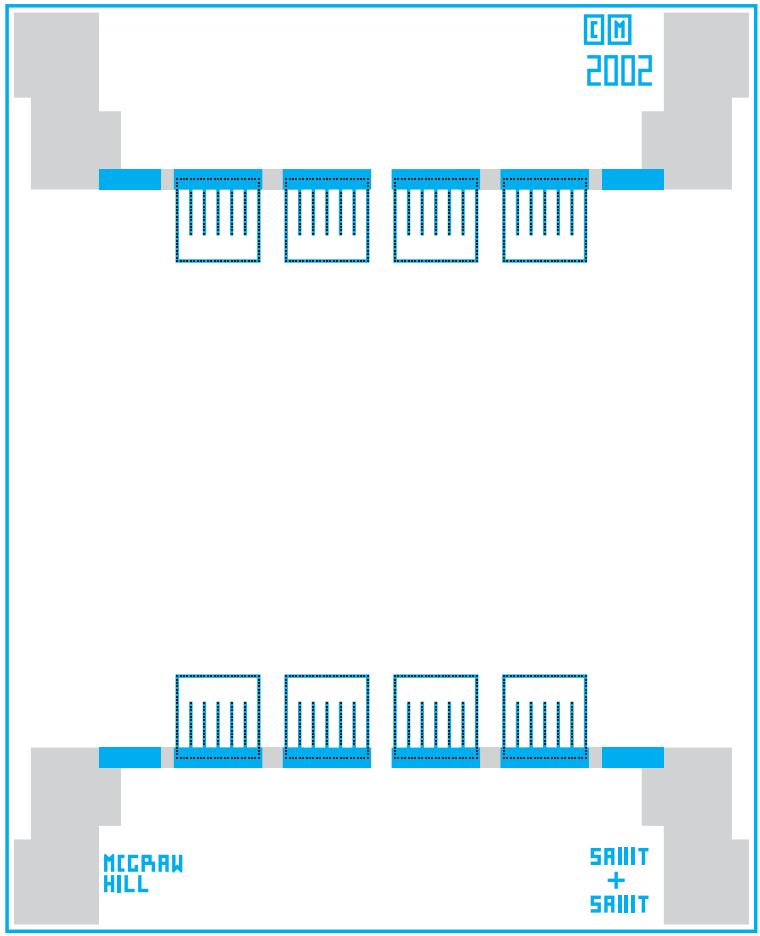


Figure CS1–103

N Well Connections

This plot shows only the Metal One associated with the N well connections. Notice how similar it is to the plot of the circuit supply connections. It is very common to combine supply wiring and N well connections.

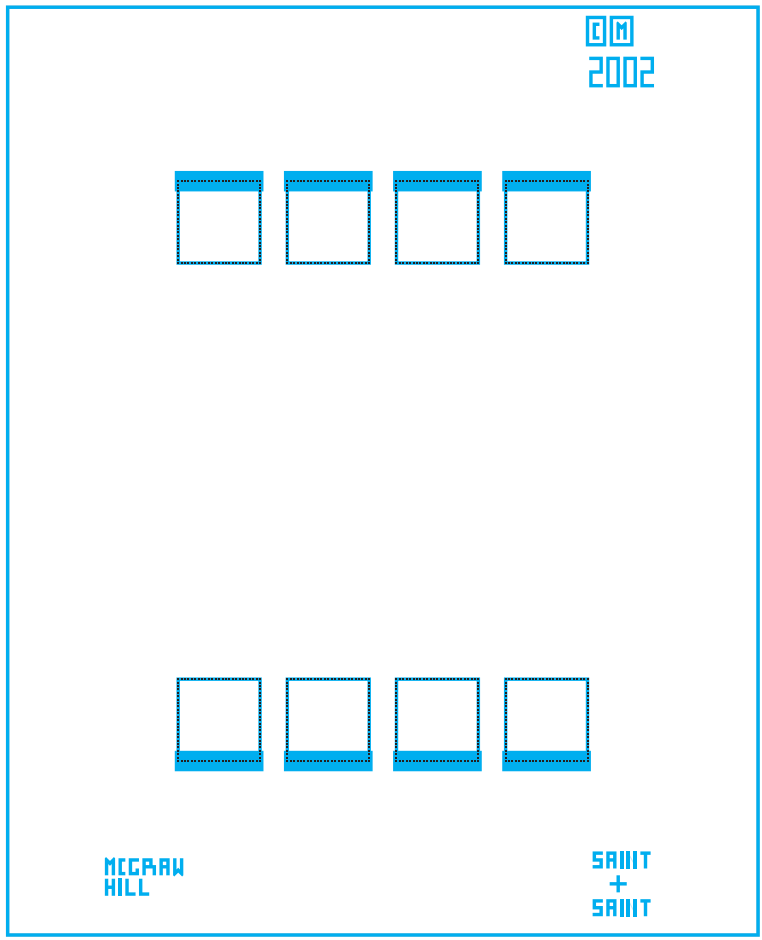


Figure CS1–104

Bias Wiring

This plot shows the wiring associated with the bias pad. Notice how the four amplifiers have their bias wires centrally connected and fed to the pad from a common center point. The gate fingers of the current source device are also shown, along with the ESD diode metalization.

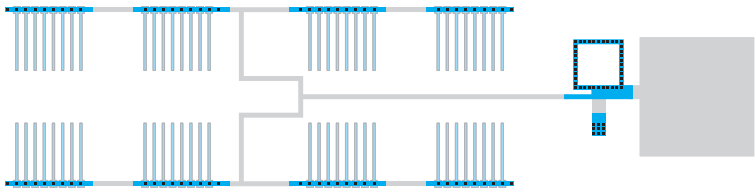
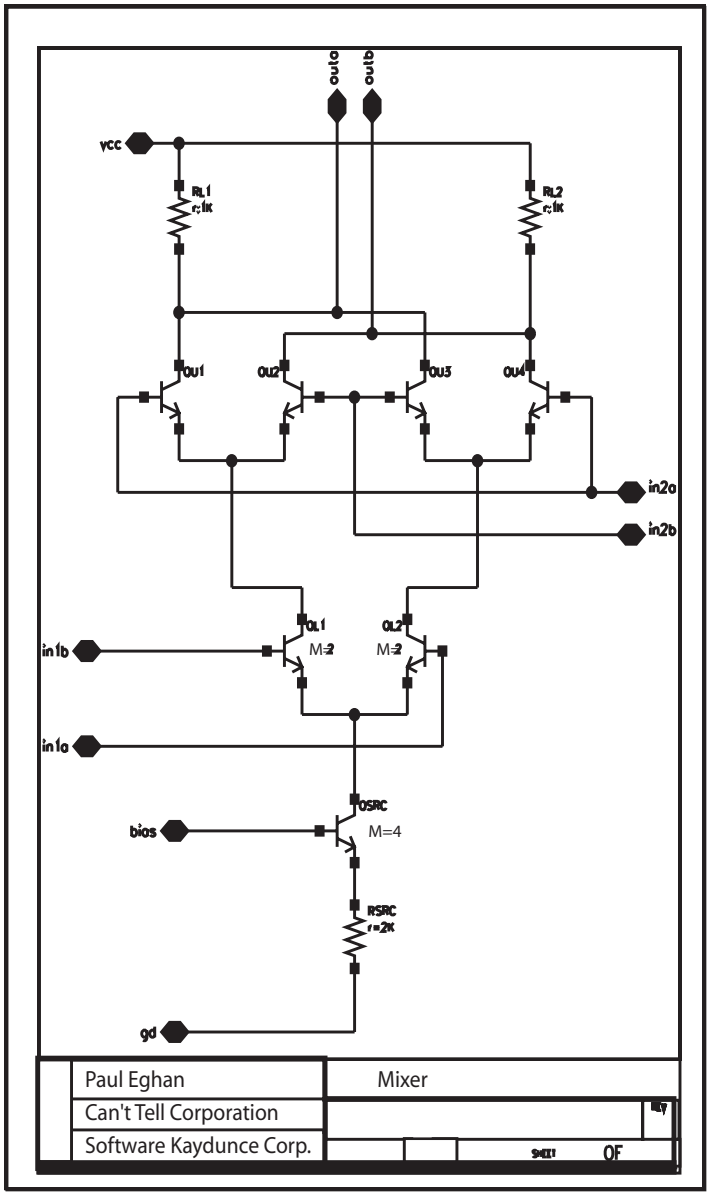


Figure CS1–105

Refer to the following schematic throughout Case Study 2.



CASE STUDY 2

CASE STUDY 2

Bipolar Mixer

Make a copy of the preceding schematic to place in front of you as you read along with this Case Study. Read through the Study once for general perspective. Read it again for analysis and better understanding. Read it again and again to increase your level of understanding and your internalization of these good practices.[1]

Enjoy.

Introduction to Case Study 2

Unlike Case Study 1, where we followed each step of an engineer's thought process as he builds his layout, in Case Study 2 we will dissect an already-completed layout. We will look at the various layout techniques used and discuss the ways this cell is put together.

We will illustrate how this layout can be improved, and improved again, depending on the level of demand of the circuit. This study will particularly help readers lay out RF circuits that have stringent requirements for matching and low parasitics.

Whereas Case Study 1 was entirely CMOS layout, this is a Bipolar layout. It is a high frequency Bipolar layout, in particular. This study is designed to give mask designers with primarily CMOS experience some exposure to the kinds of techniques you must use in higher frequency RFIC[2] problems.

[1] Just like Case Study 1. Imagine that.

[2] Radio Frequency Integrated Circuit.

We will discuss three examples that are all good, valuable, workable solutions for the layout. Depending on the requirements of the circuit, any one of these three layouts might be the best, most efficient design.

First we will dissect what I call a *reasonable* version. The reasonable version might be all you need. In some situations, you would not want to invest additional time and money to make further changes, so in those cases our reasonable version is the best solution. Then we will get to dissect a slightly better version of the same layout for higher performance requirements. Lastly, we will dissect an even more critically engineered layout that satisfies very demanding circuit requirements.

Again, all three layouts are good layouts. Some would work better at higher frequencies than others. Some are more suitable for lower frequencies that do not require extra efforts and involvement.

The purpose of this Case Study is to give readers exposure to high frequency considerations, and some good general techniques.

The Assignment

Our circuit for Case Study 2 is a very simple, straightforward Gilbert cell mixer.[3] A Gilbert cell mixer is a circuit that mixes two signals together. Mixers are very typically used in communication devices. In this Case Study, we will examine a Bipolar Gilbert cell mixer that uses NPN transistors and poly resistors.

"What Does the Circuit Do?"

The version of mixer we will look at is called a down-convert mixer. This mixer combines two high frequency signals. One signal is usually a fixed frequency. The other signal is usually frequency-modulated. The frequency-modulated signal contains the interesting information.

Suppose we call the fixed frequency *sin a,* and we call the modulated frequency *sin b.* If you do the math involved in a mixer, you will see that the two inputs are essentially multiplied together. Multiplying two sinusoidal waveforms generates the sine wave of $a + b$ as well as the sine wave of $a - b$. In other words, you will be able to see both the sum and difference frequencies

[3] Named after circuit designer Barrie Gilbert.

as a result of the mix.[4] It is the difference frequency that we are really interested in.

The fixed frequency could be 900 MHz, for example. The varying frequency could be centered at 900 MHz, but could vary by plus or minus 20 MHz. If you were to feed those two frequencies into a mixer, you would be able to grab two results. You could obtain the difference between the two signals, at plus or minus 20 MHz. Or, you could obtain the sum of the signals, ranging from 1780 MHz to 1820 MHz.

If we place a low pass filter on the output on the mixer, then we can filter out the high frequency sum signal, leaving the low frequency 20-MHz signal that we are interested in. This is called a down-convert mixer. It is usually used in receiver applications where a low frequency signal needs to be stripped away from its high frequency carrier wave.

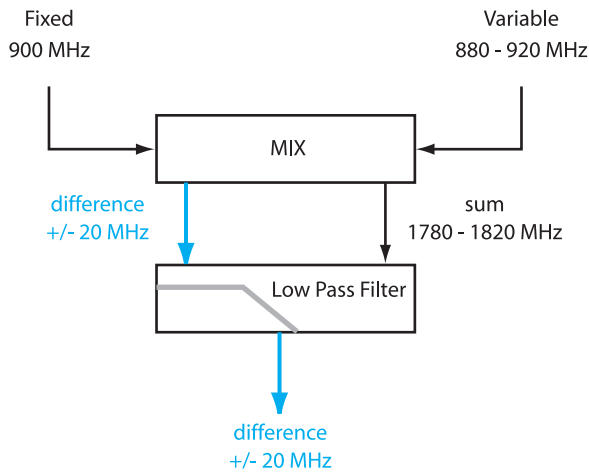


Figure CS2–1. *Low pass filter allows us to hear the difference between two signals. A down-convert mixer can be used to strip away a carrier wave, useful for receivers.*

If we were to place a high pass filter on the output then we could filter out the low frequency difference signal and be left with the much higher frequency sum signal. A mixer used in this configuration is called an up-convert mixer.

[4] Shift sin (b) by 90° to make it cos (b). It's the same signal, just shifted. Then you can see from this formula how we get both sum and difference of our two signals:

$$\sin(a)\cos(b) = .5[\sin(a+b) + \sin(a-b)]$$

It is usually used in transmitter applications where a low frequency signal needs to be modulated onto a high frequency carrier wave.

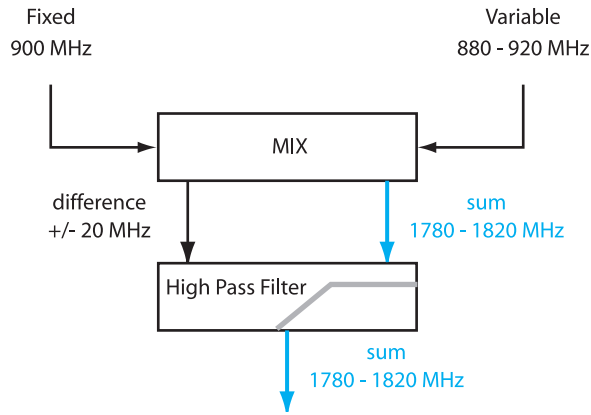


Figure CS2–2. *High pass filter lets us listen to the sum of the two signals. Up-convert mixer can place information onto a carrier wave, useful for transmitters.*

Our Case Study uses just a simple version of the Gilbert cell mixer. We will only see the main guts of the mixer. We have no internal biasing or low pass filtering, for example.

We will be referring to four primary sections of the circuit during this Case Study—the Current Source, the Lower Pair, the Upper Quad, and the Loads. Each of these refers to a section of the schematic, as you can see in Figure CS2–3.

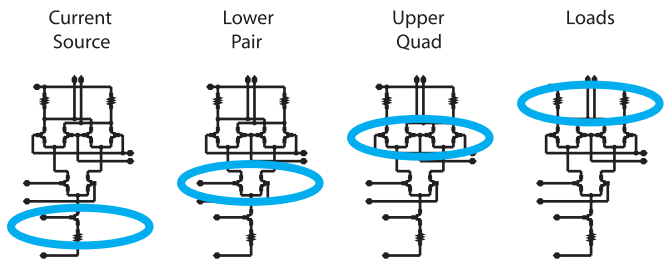


Figure CS2–3. *Our mixer is made of four sections.*

The Current Source is used to provide a fixed current. It is this current that will be used as it travels through our switches, and is finally read as a voltage at the outputs. The current first travels to the Lower Pair transistors.

The bases of the Lower Pair transistors switch the current as it passes through. The current will either pass through the left transistor, or through the right transistor, or partially through both, according to the differential input signals received at the transistor bases. We call this differential signal *Input 1*. The switched current then travels to the Upper Quad.

The transistors in the Upper Quad also switch the current, according to the differential input signal they receive at their transistor bases. This second input signal is called *Input 2*. It is the combination of switching the current first through the Lower Pair and then switching those currents through the Upper Quad that mixes the two signals together. After the current has received its second switching, we now have mixed two signals. This mixed signal, in the form of switched current, travels to the Loads.

The Loads are just resistors. Current flowing through resistors develop a voltage, $V = IR$. We can read the voltage at the outputs. This voltage contains our desired information. We might then want to run this information through a low pass or a high pass filter, but the filter will not be included as part of this Case Study.

By understanding this circuit, you can see that the heavy current usage travels through the collectors and emitters or the transistors, so we will be concerned with current densities on those wires. The bases only carry a small switching signal, so we will not bother with big fat wires for those wires.

You can also see differential pairs of wires on the inputs and outputs. This tells us that somebody is concerned about noise in this circuit. Immediately we think to incorporate some matching techniques to reduce parasitics.

"What Are the Circuit Requirements?"

The salient points of a circuit like this are:

- It's a differential circuit
- It needs reasonably good matching
- It needs even, low parasitics
- It needs good symmetry
- It needs good, smooth current flow
- It needs good, smooth signal flow

Bipolar Transistor Review

Before we dive into the mixer layout in detail, we will take a moment to remind ourselves of the layout of a single Bipolar transistor, which will be used extensively in this assignment.

In CMOS layouts, you can stretch and change the number of fingers on an FET. However, typically in Bipolar layouts, you are just given a fixed layout of the transistor to work with. There may be two or three different sizes you are allowed to use, but you are limited to this small selection. If you need a bigger device you wire these transistors in parallel. You don't stretch them.

You can see in Figure CS2–4 a very simplified, typical layout of an NPN transistor, and its symbol. Notice that the layout contacts do not correspond to the order of the symbol contacts. (Collector-Emitter-Base versus Collector-Base-Emitter)

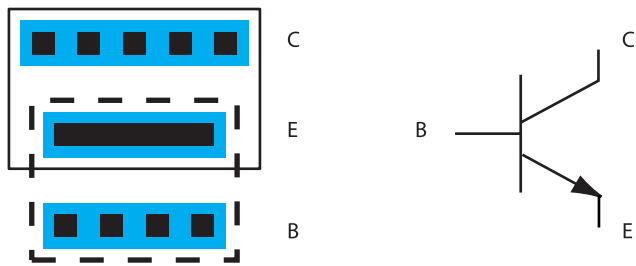


Figure CS2–4. *Common NPN Bipolar layout and schematic symbol. No stretching allowed with these components.*

You can see the collector is at the top with five contacts, the emitter is in the center, and the base is at the bottom with four contacts. If this size is not suitable, you will look in the library for another version, or you will string several of these in parallel.

In this Case Study we use several of these in parallel.

First Layout

Here is the first layout. (See Figure CS2–5.)

Initial Overview

Our first layout resembles the structure of the circuit diagram. The Current Source is at the bottom; then we see the Lower Pair, the Upper Quad, and finally at the top are the Loads.

Let's make note of the connections to the outside world. Since we are mixing two signals, we see an input pair on the left, and an input pair on the right.

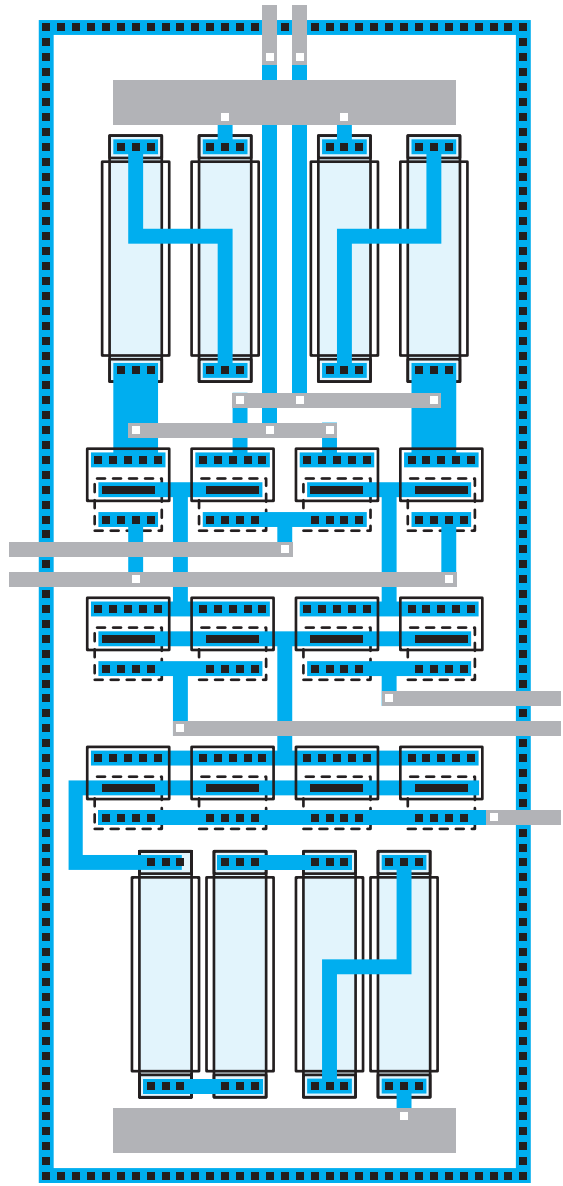


Figure CS2–5. *Complete view of first layout.*

Each input is a differential pair, for better noise reduction. You can see the diff pair output coming up through the top of the device. The wire going to the bias pin is located off of the Current Source device, to the right.

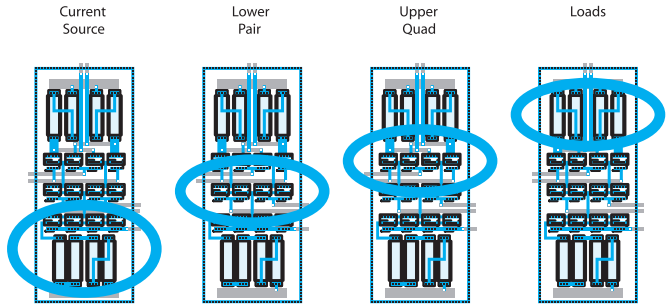


Figure CS2–6. *We see the four sections of the schematic reflected in the four sections of the layout.*

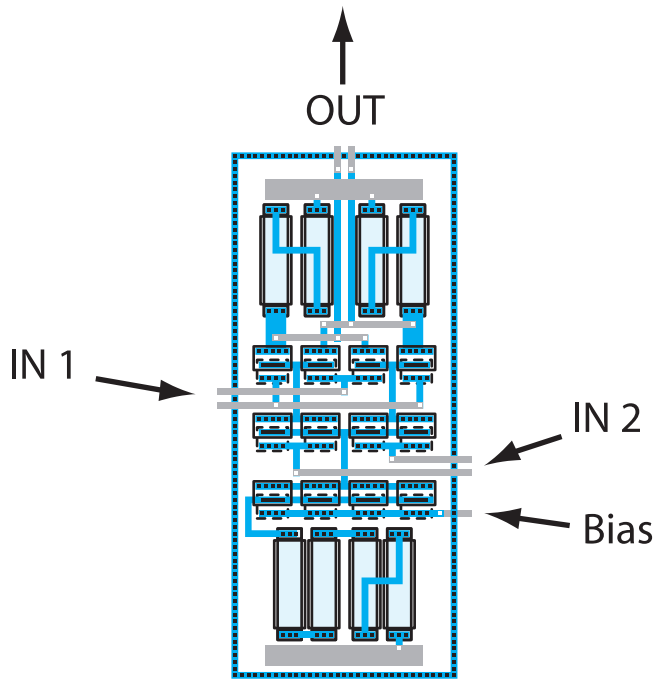


Figure CS2–7. *Connections to the outside world.*

Notice in the circuit diagram, the Lower Pair consists of two devices in parallel. If you look at the schematic for the Lower Pair more closely, you will notice the remark $M = 2$ attached to each of the transistors. The remark $M = 2$ means there are really two transistors in parallel for each device, even though they are not shown. Therefore, we will have four transistors total, which will be what we call the Lower Pair.

Likewise, the Current Source transistor, which is on the diagram as QSRC contains the remark $M = 4$, which means it is four devices in parallel.

Current Source

Let's examine our Current Source device first. The schematic tells us the device QSRC is $M = 4$. Translation: *The Current Source transistor is four devices.*

Transistors

You can see in Figure CS2–8 the four devices that make up the Current Source transistor. Connections are highlighted in blue.[5] Notice each device is a Bipolar transistor as we just examined in the last section. With Bipolar transistors, you just place them at the minimum device spacing and wire them up. These are connected in parallel.

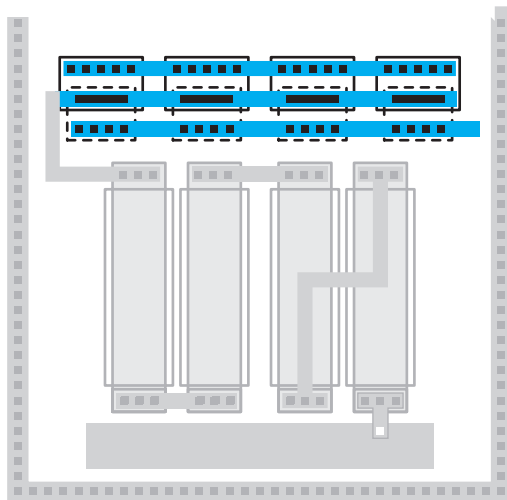


Figure CS2–8. *Four devices in parallel make up the Current Source transistor.*

Resistors

The emitter, the middle wire, needs to connect to the resistor, RSRC, which goes to ground. RSRC is really four resistors in series, so the emitter connection is at one end of the four resistors, and the connection to ground is at the other end of the four resistors. You can see in Figure CS2–9 that we have wired the four resistors in a snaking pattern between the emitter of the transistor and ground.

[5] As mentioned for the first Case Study, we will use blue as our highlight color to illustrate the relevant discussion. All other layout items will be shaded gray if not being currently discussed, regardless of layer.

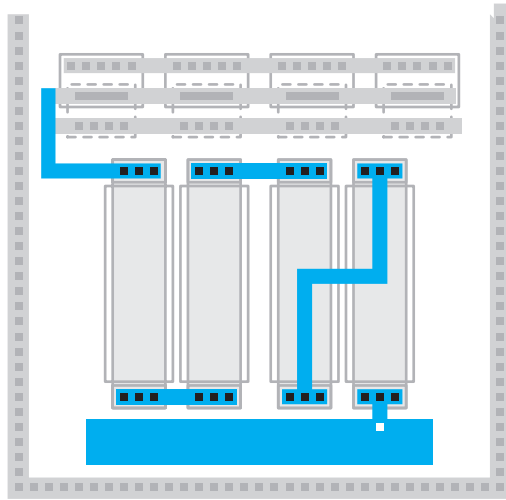


Figure CS2–9. *Connecting the emitter to the resistor that leads to ground. Since we cannot stretch, we use four transistors in parallel, and we use four resistors in series. However, this is the same as one large transistor connecting to one large resistor.*

The wiring for the Current Source emitters themselves is easy to connect. It makes one nice, easy path horizontally. However, the current that comes out of that emitter comes all out of one end, which could be a problem. Whether it is a problem or not depends on how much current is flowing out of the emitter. If there is sufficient current, it could be a bottleneck, causing current density problems.

Good layout is more than lining up polygons accurately. Spotting potential difficulties for current flow or signal flow are also critical to the success of your entire project. The layout engineer can make or break the effort of the entire team.

An easier way to run the current through the emitters would be to pull the current out of the middle. We will look at this alternative in the next version of this layout. On the other hand, as mentioned before, if your circuit has minimal requirements, perhaps this layout is more than sufficient for the job.

Notes on First Layout

One of the criteria for this circuit, if you recall, was to create the signal path very symmetrically. Let's stop to see if we should make any changes in our Current Source devices regarding symmetry.

Since the Current Source device is just providing a DC current to run the whole circuit, it's not really in the signal path, it's just powering up the circuit. So, it doesn't really matter what we do with that device regarding symmetry.

However, the symmetry in the Lower Pair and the Upper Quad is much more important. They carry the information we do care about. So we make a mental note to examine symmetry when we look at those areas.

One thing we do need to worry about in high frequency circuits is keeping the various signals away from each other. In this particular piece of layout we have two differential inputs, IN1 and IN2. We have pulled in IN1 off the left side of the cell and IN2 off the right side of the cell. Keeping these input signals apart from each other for as long as possible reduces any interactions that might occur between the two signals.

The third part of this equation is, where is the output? In very high frequency layout, we try to keep the output as far away from the inputs as we can. Otherwise we create the potential for feedback interference. In this layout, the output comes out the top, rather far from the inputs. So, this layout is quite good for these high frequency criteria.

Now let's examine the Lower Pair.

Lower Pair

Both Lower Pair devices in the schematic, namely QL1 and QL2, show $M = 2$ attached to their symbol. So, each transistor is really two devices in parallel.

Two transistors in parallel are really one device as far as I'm concerned. Both the collectors are in parallel. Both the emitters are in parallel. Both the bases are in parallel. It's the same as if we had one long device that we decided to split into two parts.

That is why this is called the Lower Pair. We are really looking at two devices. However, we will see four transistors. Confused? Four transistors, two devices. Each device is a pair. (See Figure CS2–10.)

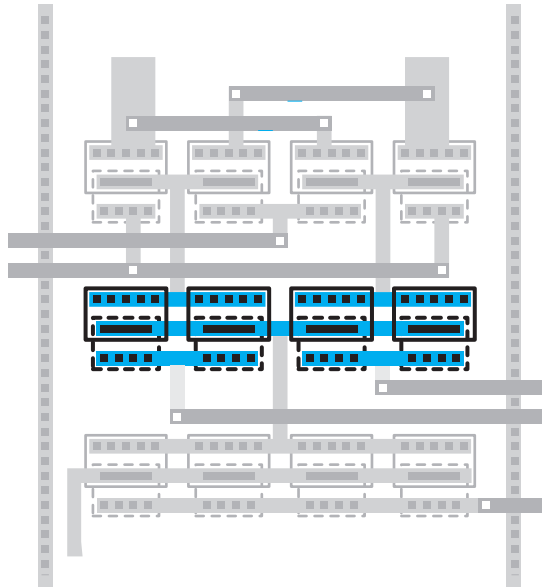


Figure CS2–10. *Lower Pair. We see two devices. Each device is split.*

Emitters

By the way the devices are connected, devices QL1 and QL2 form what we call a differential pair. You can see in Figure CS2–11 that our common emitter connection runs straight through the middle.

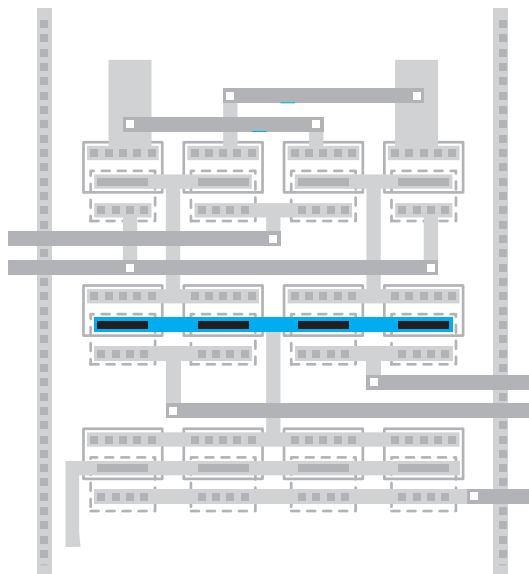


Figure CS2–11. *Common emitter connection through our diff pair.*

As you recall, we made some notes about trying to form some symmetry in the signal paths. For symmetry, the best place to pull the current is from the middle of the pair, so that each pair sees an identical current path. So our common emitter connection to the Current Source device will be straight through the center of each device.

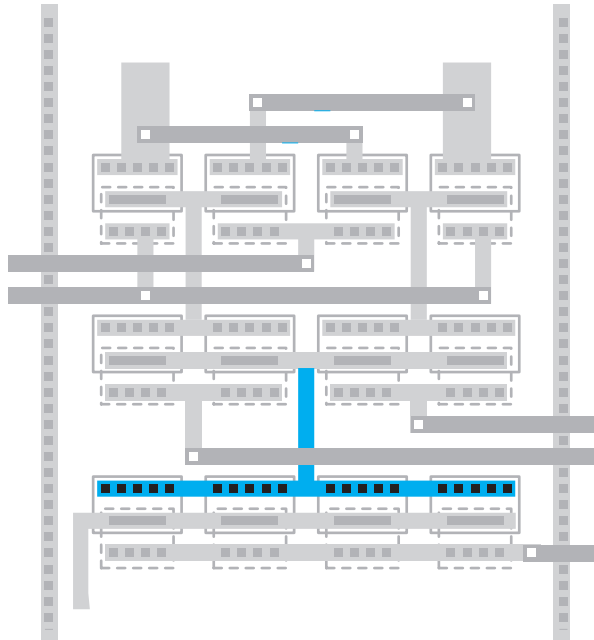


Figure CS2–12. *Connecting down to the Current Source device from the center for symmetry purposes.*

Note how the current is being pulled out of the center of the whole cell. The current from the two halves will flow through exactly the same impedances. If we had pulled the current from the left end of the Lower Pair, for example, the current from device QL2 would have had to run through all four emitters before it popped out. So, device QL2 would have seen more metal resistance than device QL1. It is important to maintain symmetry.

Bases

Our inputs are connected to the centers of the bases to maintain symmetry in the signal path. (See Figure CS2–13.) Both input wires feed directly to the centers of the devices, which is good for signal flow.

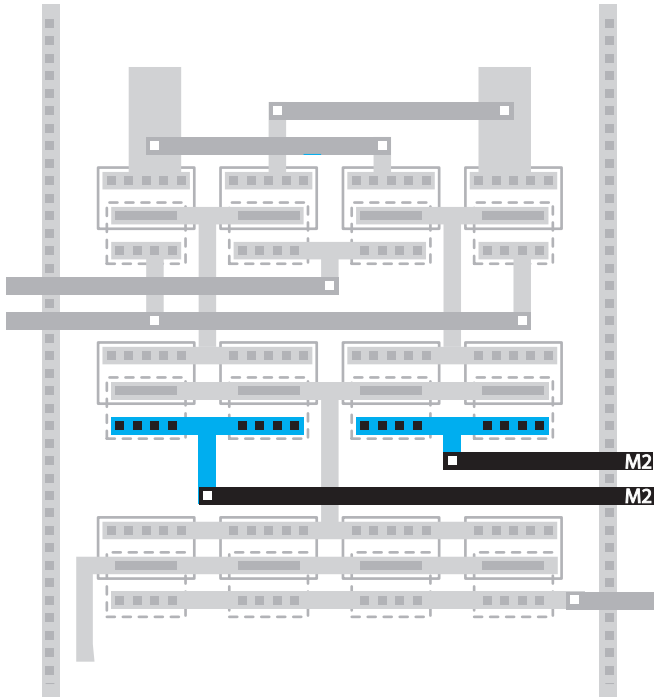


Figure CS2–13. *Pulling from the centers of the devices is good for symmetrical matching, but notice the Metal Two input wires are different lengths.*

However, we actually could improve the layout for these input wires. One input wire is slightly longer than the other, so we might consider making them of more equal length.

We have to make a trade-off here. Would we rather feed our inputs to the center of our two devices, as we just saw in Figure CS2–13? Or would we rather have our input connections closer to each other, as we could do in Figure CS2–14, trying to make the path lengths more equal? One method helps the transistors switch more evenly with each other, and the other method helps the parasitics to be more equal. Your circuit designer can help you determine which is better for each application.

Collectors

The collectors of QL1 and QL2, in the Lower Pair, connect to the common emitter points of two diff pairs, which are in the Upper Quad. Here is another central connection that helps our symmetry. (See Figure CS2–15.)

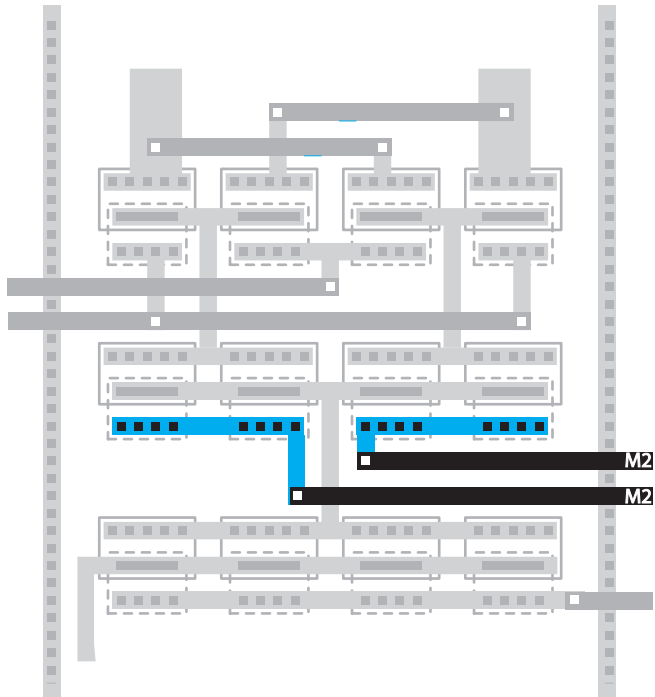


Figure CS2–14. *Consider this option, which matches wire lengths better, but connects to the ends of the devices.*

Current flowing into our Lower Pair is brought in symmetrically to each of the two devices. It's a nice central feed.

Now let's look at the Upper Quad.

Upper Quad

The Upper Quad is effectively two differential pairs. We have a diff pair on the left, and a diff pair on the right. (See Figure CS2–16.)

QU1, QU2, QU3, and QU4, are each single devices. Even though the appearance is similar to the halved devices in the Lower Pair, each transistor is a complete and separate device.

Emitters

In each case, we want our feed to the Upper Quad devices to come from a central point. (See Figure CS2–17.) Again, we are trying to maintain our symmetry for better matching.

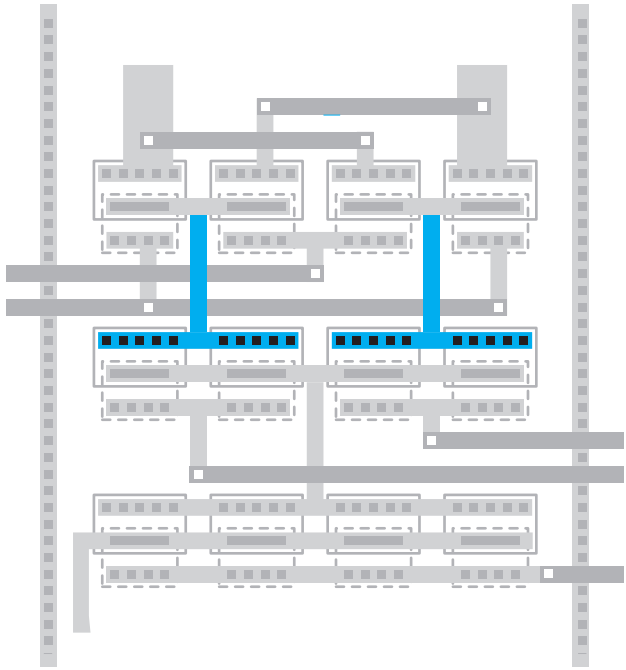


Figure CS2–15. *Collectors of QL1 and QL2, the Lower Pair, connect to the common emitter points of two diff pairs, which are the Upper Quad.*

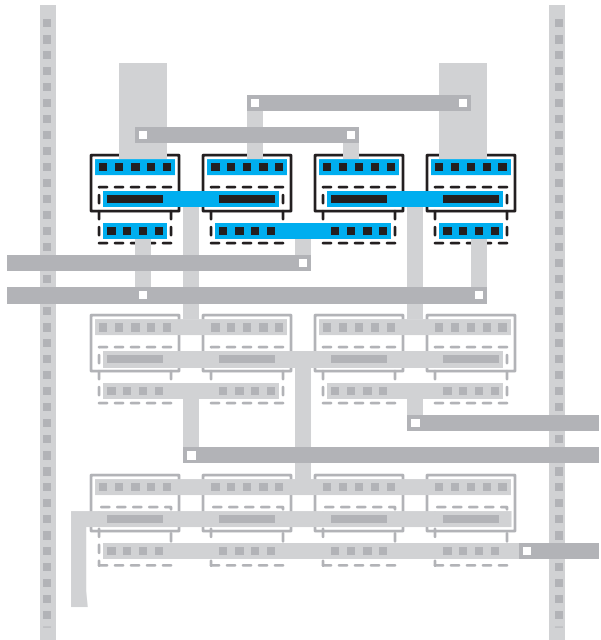


Figure CS2–16. *Upper Quad. Four devices set as two differential pairs.*

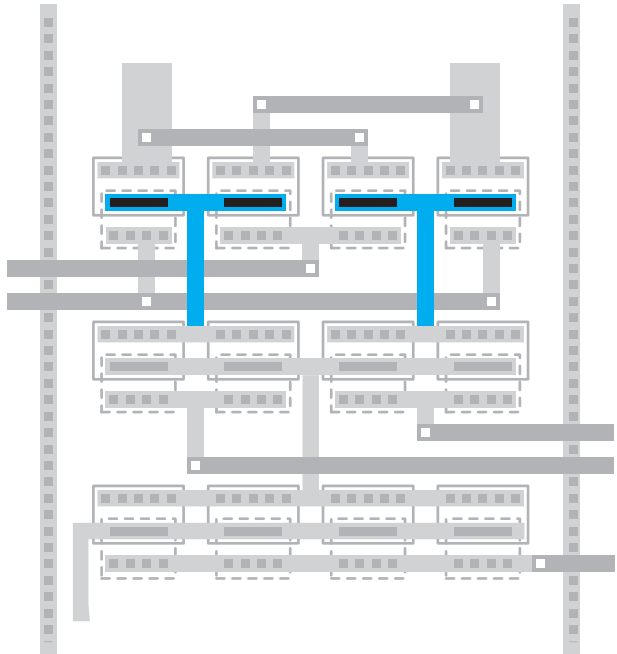


Figure CS2–17. *Emitters of each pair of devices in Upper Quad fed centrally.*

Bases

We decided to make our lives easy on the emitter wiring. We decided to have the two left-hand transistors as one diff pair, and the two right-hand transistors as the other diff pair. However, that configuration forces our hand as to where the input wires need to go.

Because of the configuration we have selected, the two outside devices need to receive half of the differential input, and the two inside devices need to receive the other differential input.

IN1 comes in from the left. We feed one half of the differential signal into the two bases on the outside. The other half of the signal goes to the two bases on the inside.

That's pretty much how the circuit diagram is drawn.

This is just one example. There are many different ways to do this wiring. We have tried to keep our input traces as balanced as possible, but there are always going to be some issues with the way this works.

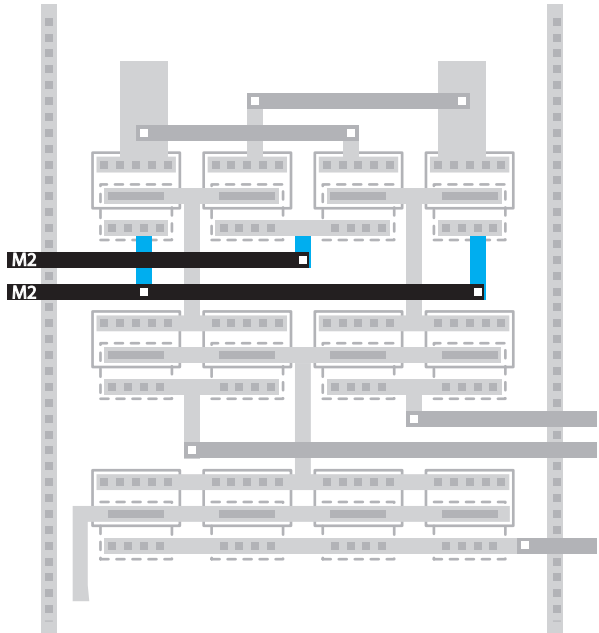


Figure CS2–18. *Running input signals to two diff pairs.*

Collectors

The collectors of the Upper Quad devices also become the output wires. We want to keep our output traces as symmetrical as possible. You can see we have connected alternating devices, passing into Metal Two when needed.

The collector of the left transistor, QU1, needs to connect to the left transistor of the other diff pair, QU3. (See Figure CS2–19.)

Likewise, the collector of QU2 needs to get over to the collector of QU4. (See Figure CS2–20.)

Loads

Our finished Load area is shown in Figure CS2–21.

Output

We want to try to bring our differential output out from a nice central point. So we move our resistors apart to allow room to run our output traces down the middle.

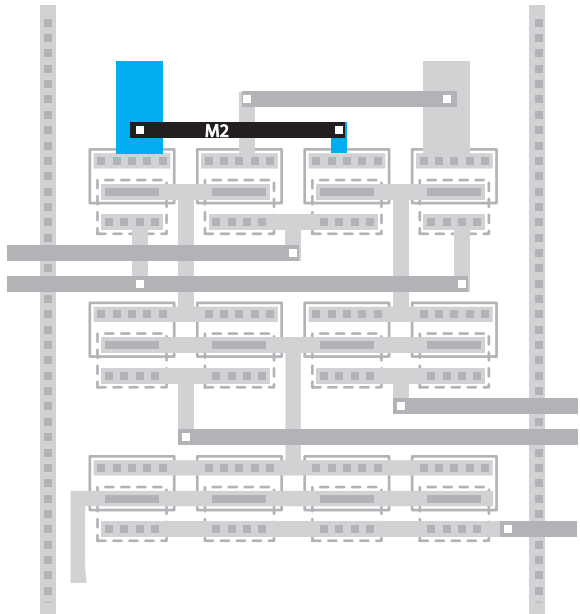


Figure CS2–19. *Collector of the left transistor, QU1, connects to the left hand transistor of the other diff pair, QU3.*

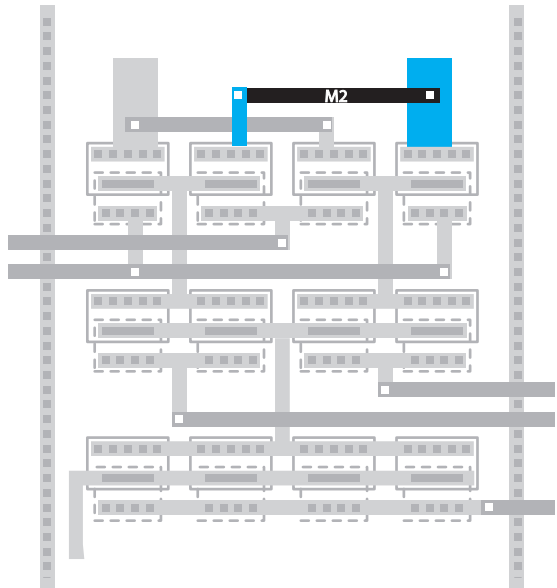


Figure CS2–20. *Collector of QU2 connects with the collector of QU4.*

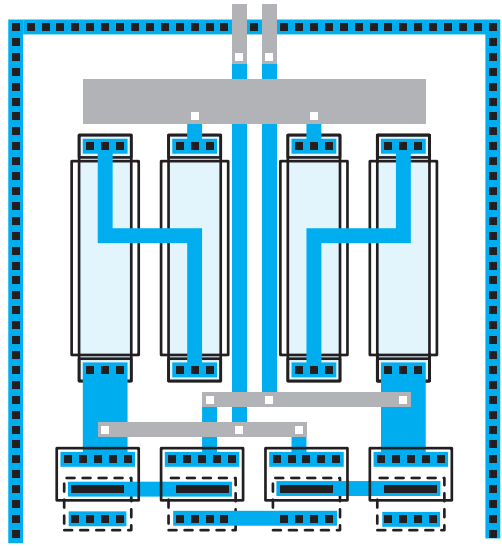


Figure CS2–21. *Load resistor area at the top of the layout.*

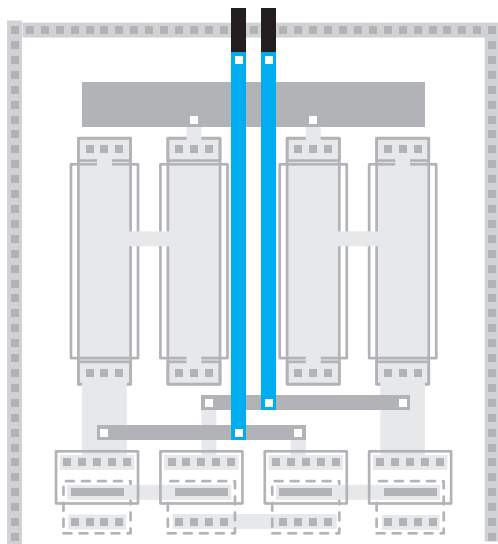


Figure CS2–22. *Output traces through the middle.*

Notice how the outputs tap into the horizontal wires that join the Upper Quad collectors to each other. Now, our Loads really can hook in anywhere because even though they have the output signal on them, they are mainly there for DC biasing. Our Loads also connect to the positive rail at the top.

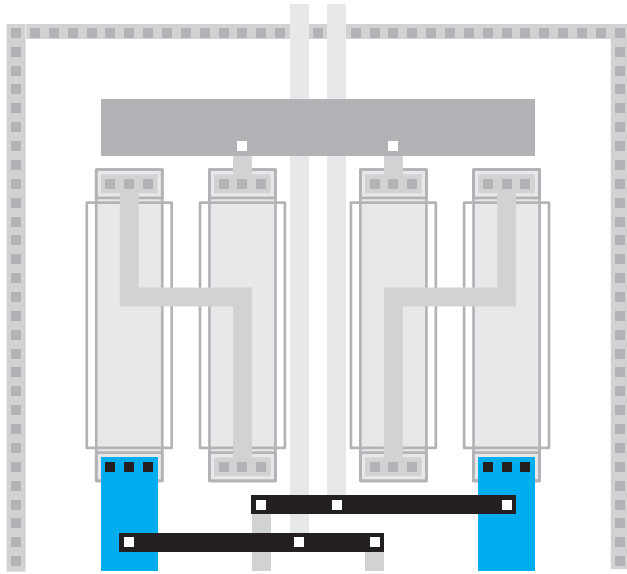


Figure CS2–23. *Connecting Loads to the collectors of Upper Quad.*

Resistors

We wire the Loads in series, as in Figure CS2–24, using the positive rail to connect our two Load resistors.

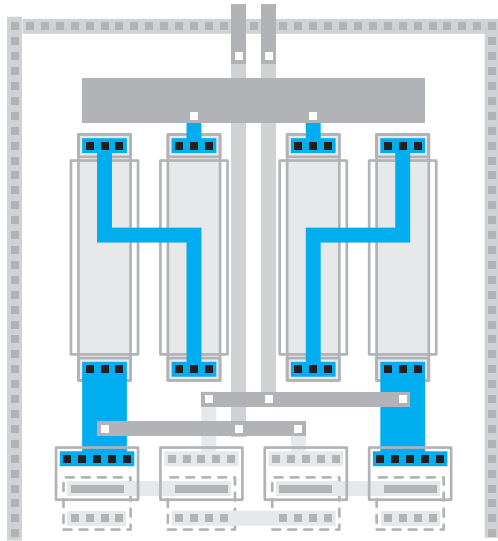


Figure CS2–24. *Connecting the Load resistors.*

Remember to continue looking at the original schematic from the front of this Case Study as we go along. These resistors, of course, are shown at the top.

Analysis of First Layout

This layout has good symmetry, and it has reasonable matching. However, there are some potential problems with this layout. Especially if the circuit is a demanding, very high frequency circuit, there are some improvements we can make.

Our concerns are as follows:

- We want to improve the current flow on the Current Source device. All of the current is coming out of one end of a long skinny piece of metal.
- On the Lower Pair, the current of the left half of device QL1 has to flow through and above the right hand half of QL1 before it gets to the common point. The same applies to transistor QL2 also.
- Resistor matching needs attention.
- The Current Source resistor numbers 2 and 3 see a different amount of etch than resistor numbers 1 and 4. The difference in etch is due to the fact that the inner devices are surrounded by devices on all sides, but the outer devices are not. (See Chapter 5.)
- Likewise, on the Loads, the two resistors in the center that connect to VCC see a different etch than the resistors that connect to the collectors.
- The Lower Pair could really do with some improved matching.

In the next section, we will look at a layout technique called the wrap-around transistor. Then we will move on to investigate the second layout, which uses these wrap-around transistors.

Bipolar Transistor Layout—Wrap-Around Technique

One of the problems with the first layout was current flow in the emitter. We want to avoid having the current crowd out of just one side.

If you pull the current out of one edge of a long thin emitter, then the current from the opposite end has an issue trying to get past all the other material in the way.

Current is being spat out from the emitter all the time. It all ends up coming out this thin, skinny edge. If we could find a way of laying out the transistor

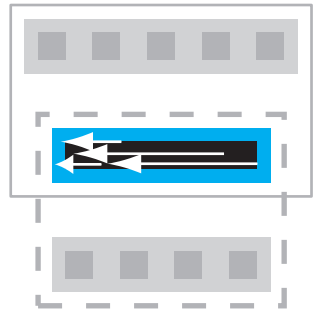


Figure CS2–25. *Taking current from one end of our emitter causes current density problems.*

so that the current could escape through a wider edge, that would be much nicer for the current.[6]

We are stuck with our basic transistor layout, so we have to be a bit creative to get current out of this device more evenly. What we do is bring the current up and around both sides of the emitter. Connecting on each end gives us twice the escape area as before, with half the distance traveled.

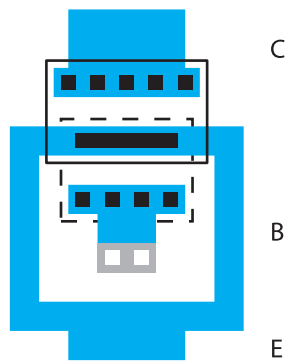


Figure CS2–26. *Creative method pulls from the emitter on both ends.*

Allowing current to pass out both sides of the emitter reduces the amount of distance the current has to flow by half. It also increases the current handling capability of this transistor by a factor of two, because now we have twice the width of metal coming out. The parasitic resistance on the emitter has been reduced.

[6] Zen art of layout: *Be* the current. Make the current happy.

Notice that the terminals of our NPN transistor are now in different places. The collector is at the top, the base is in the middle, and the emitter is at the bottom. This placement is exactly as it is in the circuit diagram. This makes the layout easier to understand.

This technique improves our parasitic control and improves our current handling at the same time. It's just the sort of clever thinking people are doing all the time in this ever-changing field.

There are surely hundreds more of these obvious little tricks we have not yet discovered. Keep an eye out for any simple opportunity to be clever. It's the simple ones that elude us, yet seem so obvious in the end. Then put your name on the method and let us know about it.

Wrapping metal around the transistor like this is a fairly typical way of doing Bipolar transistor layout. However, one very important caveat about using this kind of layout is to make sure you connect to it symmetrically. Otherwise, you have wasted all the work you did pulling the current out evenly on both sides of the emitter.

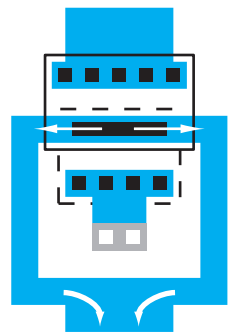


Figure CS2–27. *Wrapping the transistor improves parasitics and current handling.*

The problem with wrapping metal around the transistor is now having to connect somehow to the base. Notice the base is fully enclosed in Metal One.

The solution we use in our Case Study is to extend the base in Metal One and put a Metal Two connection on the end of it. The Metal Two will then cross over the surrounding Metal One, giving us a connection outside the emitter's ring. You can see we have included vias (white squares) between Metal One

and Metal Two to form a connection. Jumping into another metal layer is an unfortunate drawback of the wrap-around technique.

Notice there are trade-offs everywhere you turn in Bipolar layout. We improve one thing, just to make one other thing more difficult.

Only you will know which way to turn, where to apply a technique, where not to apply a technique. It all depends on your particular circuit requirements. And every circuit assignment you get is different, requiring different conclusions, different trade-off decisions. What worked best last time may not be the best trade-off next time. What Sally down the hall is doing with her circuit may not be the best solution for your circuit. And two completely different styles of layout could yield the same results in the end. Bipolar layout can be done correctly a number of ways.

Decisions. Thinking. Weighing trade-offs. Creative solutions.

Let's use this transistor-wrapping technique in our next mixer layout.

Second Layout

The second layout has been wired with the assumption that there is a reasonably high current flowing through the mixer. Consequently, all emitter and collector connections have been widened to handle a higher current. We have used the transistor wrapping technique to reduce the current densities in our emitter connections. Since the current gain of a Bipolar transistor is usually quite high (greater than 100), the currents flowing in the base wiring are very small in comparison. So, all our base connections can be wired using minimum wire.

The second layout as shown in Figure CS2–28.

The second layout is built entirely using the transistor connection method we just discussed—the wrap-around method. Each transistor's emitter is pulled out both sides and brought around to meet at the bottom.

Current Source

Figure CS2–29 is a complete view of the Current Source. Remember that this is the bottom portion of the schematic shown at the beginning of this Case Study.

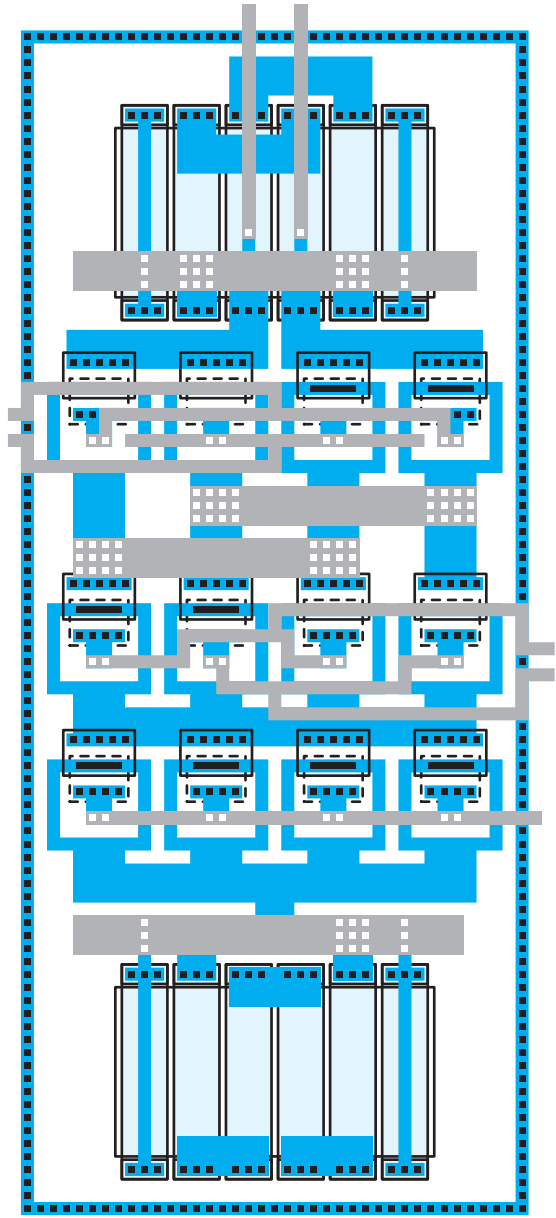


Figure CS2–28. *Complete view of second layout.*

The transistor sizes are identical to those in the first layout. Again, we have four transistors in parallel. In exactly the same way as in the first layout, all the emitters need to connect together, all the bases need to connect together, and all the collectors need to connect together.

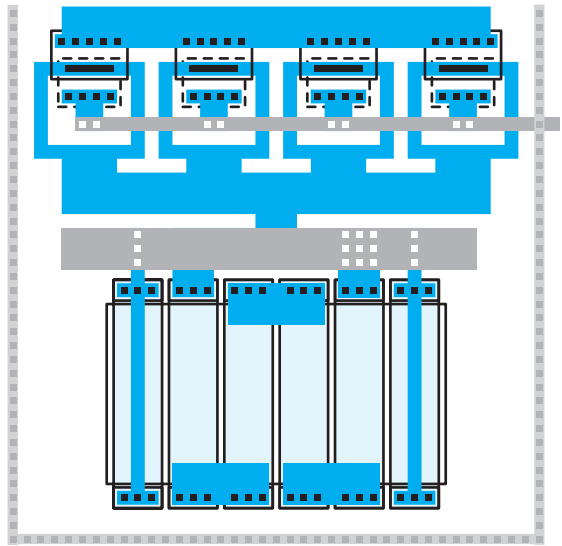


Figure CS2–29. *Entire Current Source shown. Notice the wrapping method is used on all transistors.*

Emitters

Since we went to so much trouble getting the emitter current out, it's nice that the emitter connections are now a nice big, fat, wide wire. Our complete connection scheme gives each emitter a well-defined and happy current path.

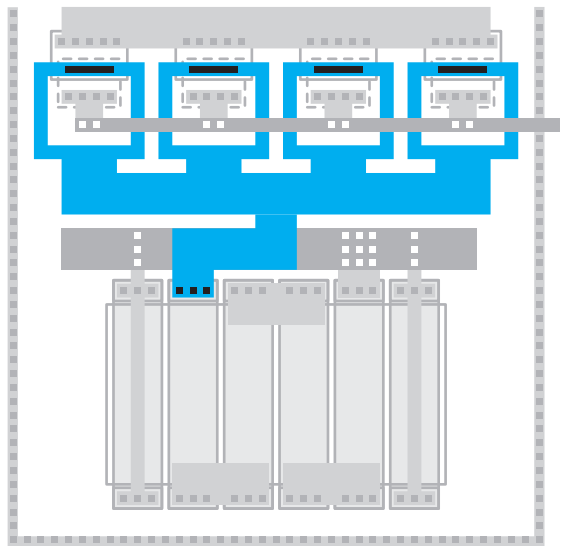


Figure CS2–30. *Big, fat emitter connection.*

We are pulling the final current out the big emitter wire bus from the center. Consequently, the transistors on the ends of the bus see slightly more impedance than the two transistors in the middle, but it's a fairly wide wire, so the extra resistance shouldn't be of much concern since it will be very low.

Bases

All of our bases are connected in Metal Two, shown highlighted in Figure CS2–31.

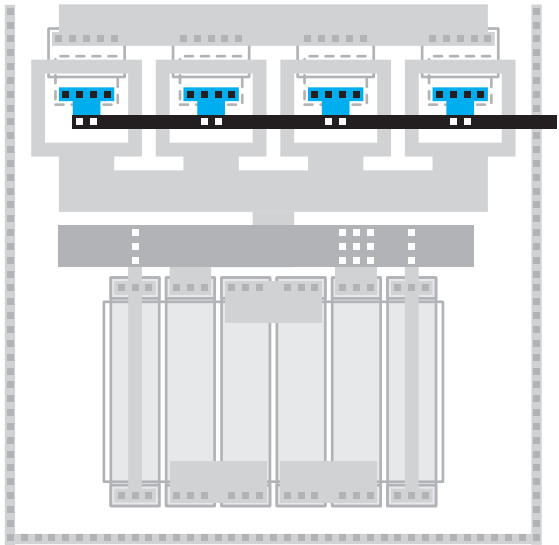


Figure CS2–31. *Bases connected in Metal Two.*

Collectors

And finally, all of our collectors are connected together in Metal One, shown highlighted in Figure CS2–32. This is an easy straight-across connection.

Resistors

The wire that connects to the resistor of the Current Source is now pulled out from the center of the emitter wire. We have also widened the wiring between the resistors so that the internal metal wiring does not add too much parasitic resistance. This is a fairly typical technique you might want to use when running a lot of current. (See Figure CS2–33.)

If this circuit was only running a small amount of current, then you probably don't need to go to these lengths. But, let's assume we have issues with current densities, just to see an example of the kind of wire widths we would need. You can see we also have a nice, big chunk of vias to connect to the ground wire.

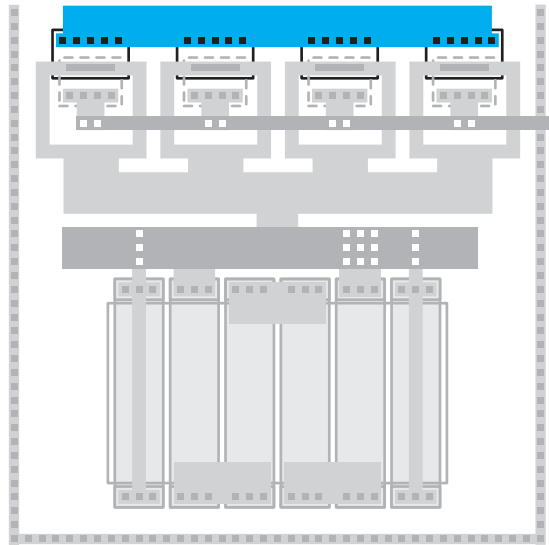


Figure CS2–32. *Collectors connected.*

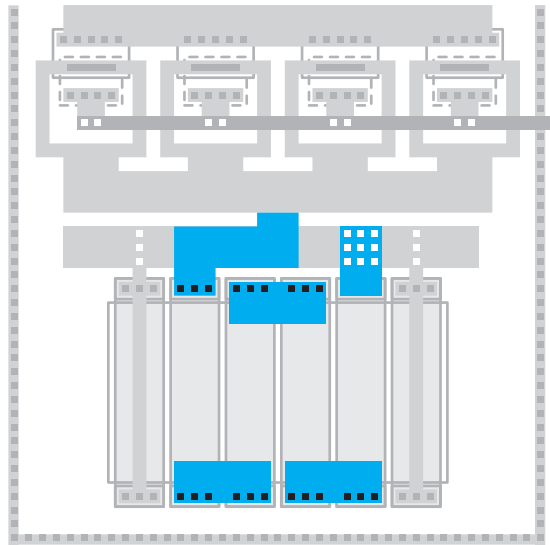


Figure CS2–33. *Connecting to the Current Source.*

We cannot interdigitate these four resistors because they are all connected in series. Essentially this is one big resistor. Interdigitization is really only useful for two or more devices.

You might notice that we've also added a couple of dummy resistors on either side of our Current Source to help the resistors match during etching. (See Figure CS2–34.)

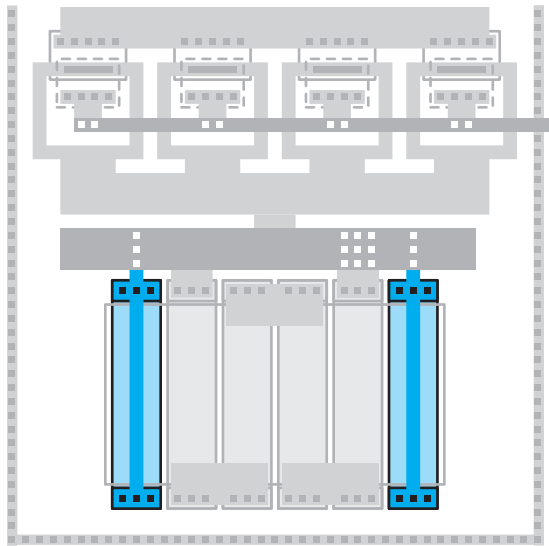


Figure CS2–34. *Two dummies to help the four circuit resistors etch identically.*

Notice we shorted the dummy resistors together and tied them to ground. They are not part of the circuit. They are only there to give the outer two circuit resistors similar conditions for etching as the inner two circuit resistors. (See Chapter 5.)

Let's move on to the Lower Pair.

Lower Pair

Figure CS2–35 is a complete view of the middle sections from our second layout. Refer to this view for both the Lower Pair and Upper Quad sections.

Interdigitation Plan

As we stated in the analysis of the first layout, we want to improve the matching on this Lower Pair. So, in this second layout, the two halves of the diff pair have been interdigitated with each other. Notice that QL1 halves and QL2 halves alternate, so that they see more equal conditions. We will connect the halves as shown in our plan, starting with the emitters in the next section. (See Figure CS2–36.)

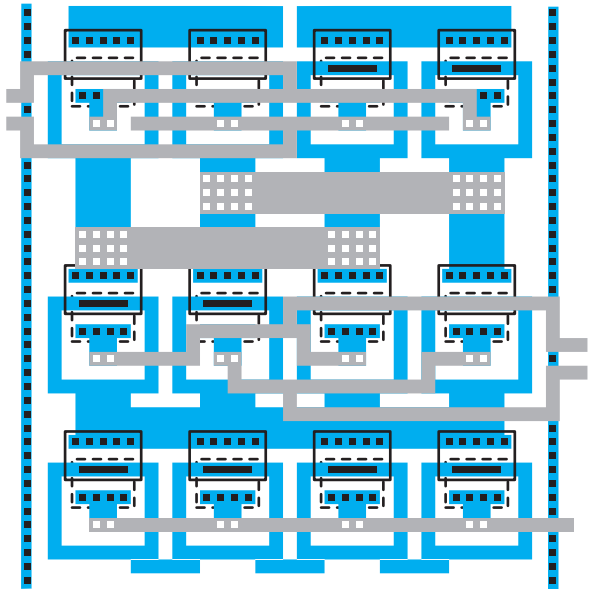


Figure CS2–35. *Middle section of second layout.*

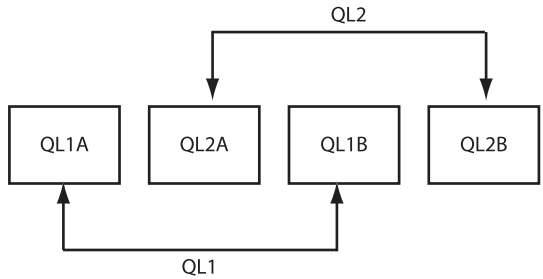


Figure CS2–36. *Lower Pair halves interdigitated.*

Emitters

All of the emitters are connected to each other and to the collector of the Current Source device using the same piece of fat Metal One. This is a good use of metal—two functions for the price, and real estate, of one.

Collectors

Due to our interdigitation, we need to connect the collectors of the two QL1's to each other, and the collectors of the two QL2's to each other.

Because QL1 is split into two halves, with one of the halves of QL2 smack in the middle of it, we have to get the metalization out. That forces us to use Metal Two, as highlighted in Figure CS2–38.

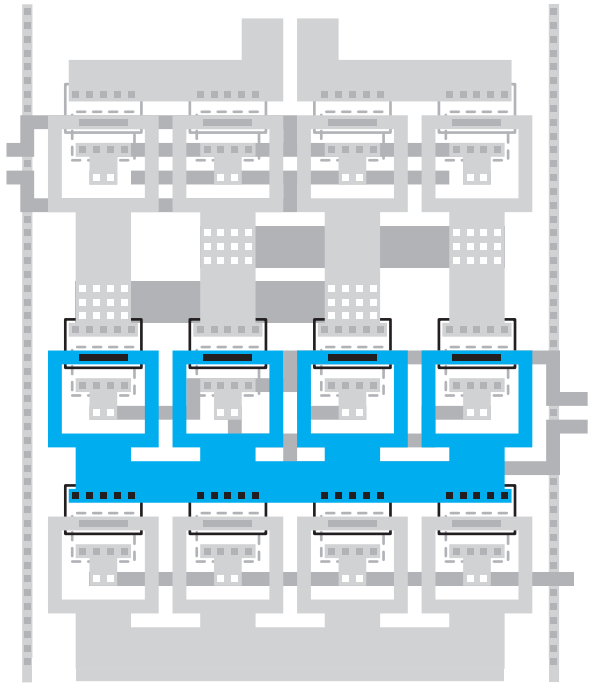


Figure CS2–37. *Emitter connections to each other double as connections to Current Source device.*

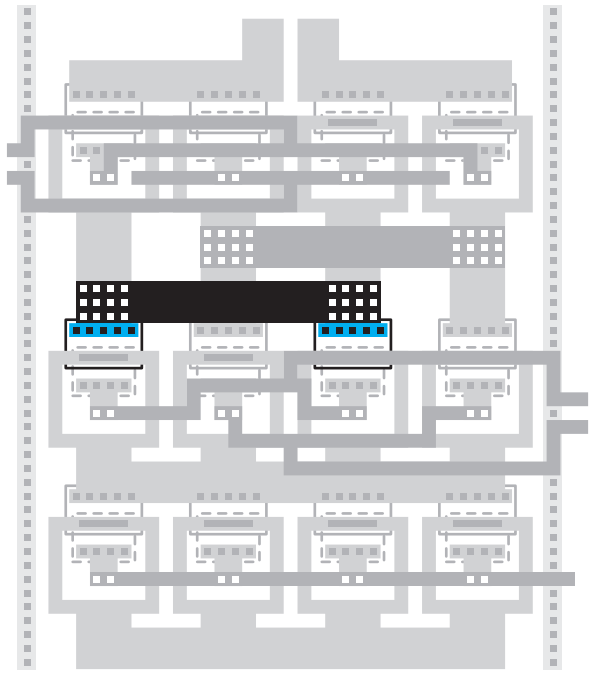


Figure CS2–38. *Forced to jump up into Metal Two (highlighted) to connect two halves of the device QL1.*

Likewise, we must jump up into Metal Two to connect the other pair, QL2, as seen highlighted in Figure CS2–39.

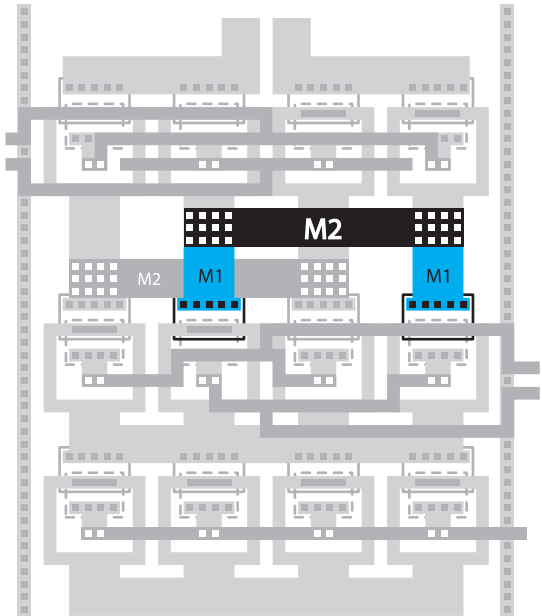


Figure CS2–39. *Forced to jump up into Metal Two to connect across to the two halves of the device QL2.*

Bases

First, we need to connect our bases to each other in the same way we connected our collectors to each other. The first two halves of the device are in positions 1 and 3. We need to jump up over other metal structures, so we use Metal Two, as shown highlighted in Figure CS2–40.

The other two halves to be connected are in positions 2 and 4. This forces us to jump up and over other metal again using Metal Two. This is shown highlighted in Figure CS2–41.

We have connected our bases to each other, but now we need to get our signals into the center of these devices.

Inputs

We want to centrally feed the input wires to the existing base wiring to ensure we have balanced parasitics between all transistors. So, we bring our inputs into the middle.

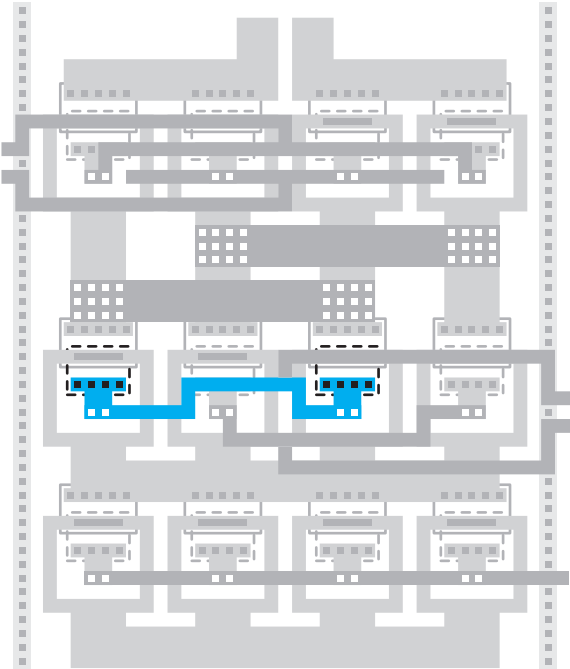


Figure CS2–40. *Connecting bases of two device halves in Metal Two.*

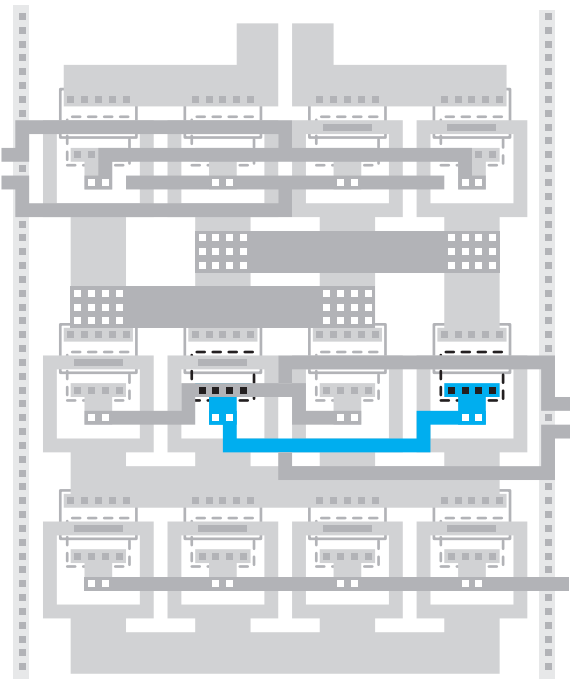


Figure CS2–41. *Connecting the halves of the other interdigitated device, also using Metal Two.*

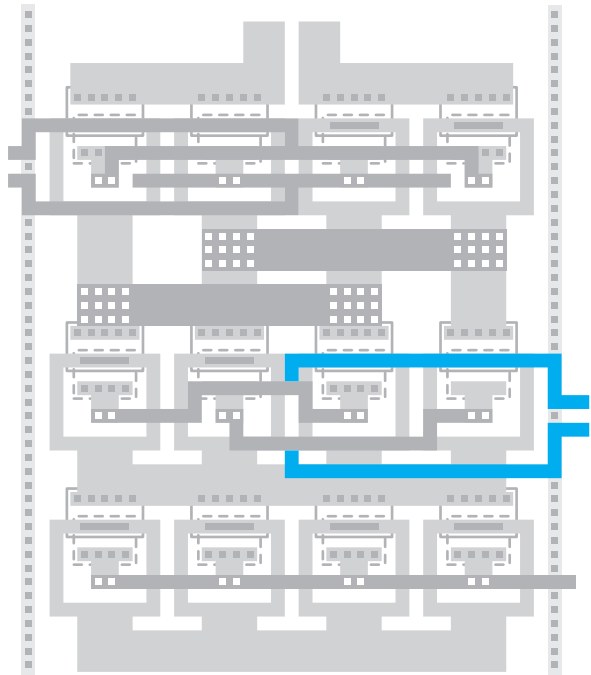


Figure CS2–42. *Inputs fairly evenly matched this time.*

We have balanced our parasitics on our two input wires. You might recall that in the first layout, our two inputs were of unequal lengths.

The parasitics of the two wires are still not perfectly matched due to structures underneath and nearby, but they are a lot better than they were in the first layout. (See Figure CS2–43.)

We have now connected our Lower Pair. We have interdigitated them to improve the matching, and we have connected them to our Current Source. Now let's turn to the Upper Quad.

Upper Quad

The Upper Quad area is shown at the top of Figure CS2–35, several pages back.

Interdigitation Plan

Again, we want to improve the matching, so we will use the same interdigitating trick. It will look very similar to what we have for the Lower Pair. We have drawn our plan in Figure CS2–44.

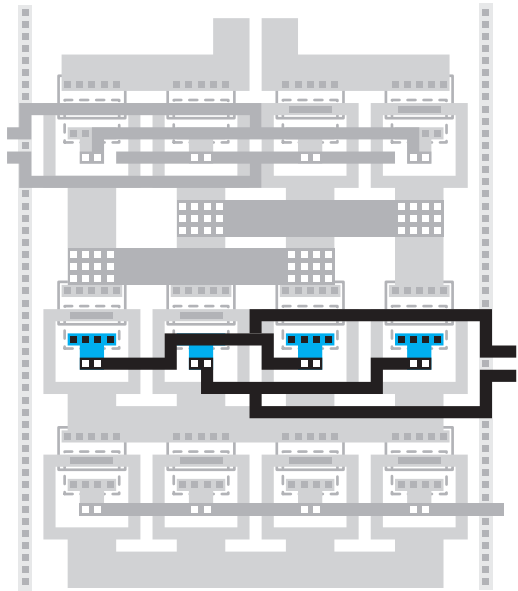


Figure CS2–43. *Complete view of all input traces for Lower Pair.*

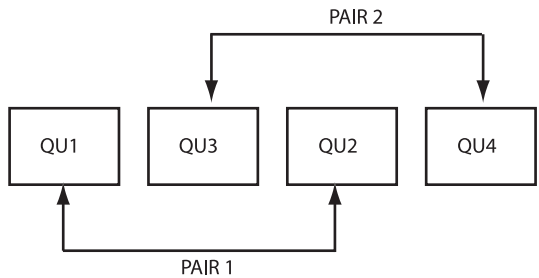


Figure CS2–44. *Planning to interdigitate our four Upper Quad devices.*

In this case, there are two diff pairs that are interdigitated—four actual devices. When we interdigitated the Lower Quad we had a single diff pair—two devices split into four pieces. Each device in the Upper Quad is an individual transistor. It's the same trick, regardless.

Emitters

Transistors QU1 and QU2 (which form an interdigitated pair) need to connect to one of the Lower Pair collectors. We will use the common emitter connection from our QU devices to connect with one of the Lower Pair collector connections. This is shown in Figure CS2–45.

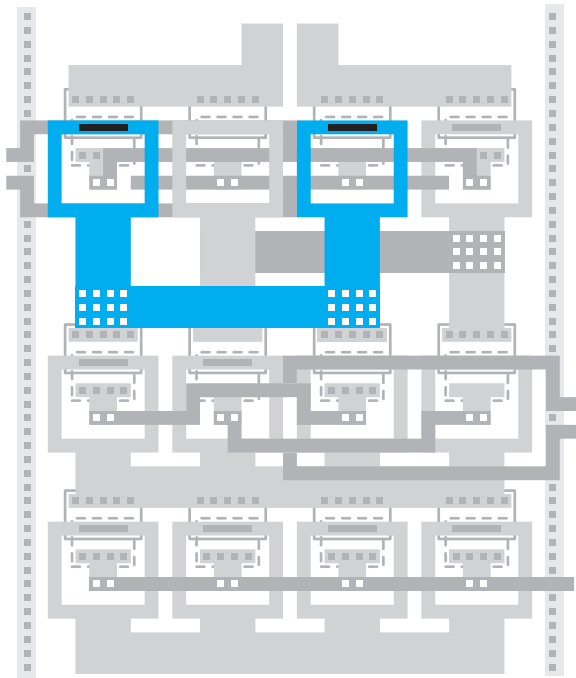


Figure CS2–45. *Connecting QU1 and QU2 to the Lower Pair.*

We do the same type of connection jump for transistors QU3 and QU4, as shown in Figure CS2–46. Again, we connect the emitters of the QU devices to the collectors of the Lower Pair.

You can see we are sharing some of the metal. Some of the metal that is connecting the Lower Pair collectors to each other is also connecting the common emitter points for the Upper Quad.

So it's a two-for-one special. We're again saving ourselves some space by reusing existing wiring, and lowering the parasitics as well. Good everyday practice.

Collectors

As with the Lower Pair, we will connect our Upper Quad collectors. Due to the interdigitation of the Upper Quad transistors, we will not need to jump across devices. We can use Metal One. (See Figure CS2–47.)

Bases

Because we interdigitated our diff pair, device QU1 is on the far left side. However, its base needs to connect to QU4, which is on the far right side. So,

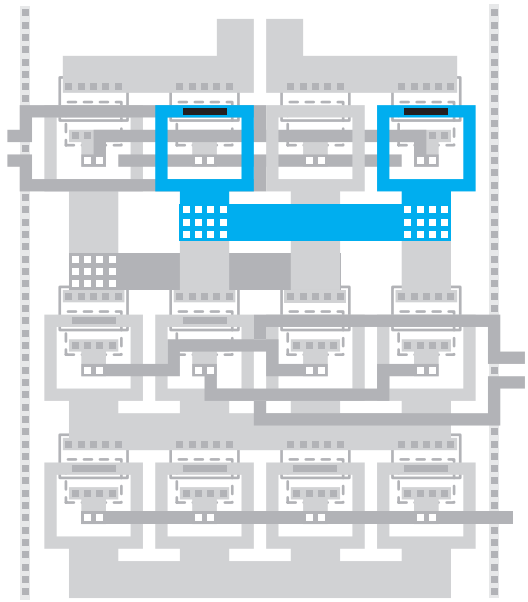


Figure CS2–46. *Connecting QU3 and QU4 to the Lower Pair. Jumping across horizontally we will use Metal Two.*

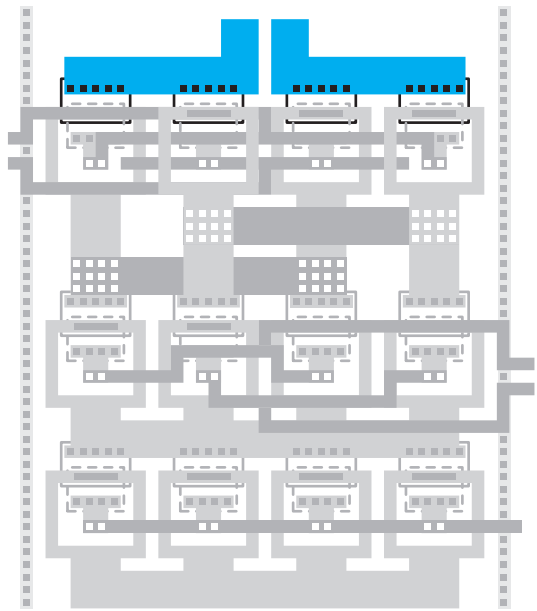


Figure CS2–47. *Connecting collectors using widened wires.*

we again use Metal Two to jump horizontally across to connect our bases. (See Figure CS2–48. *You can also see the input wire highlighted.*)

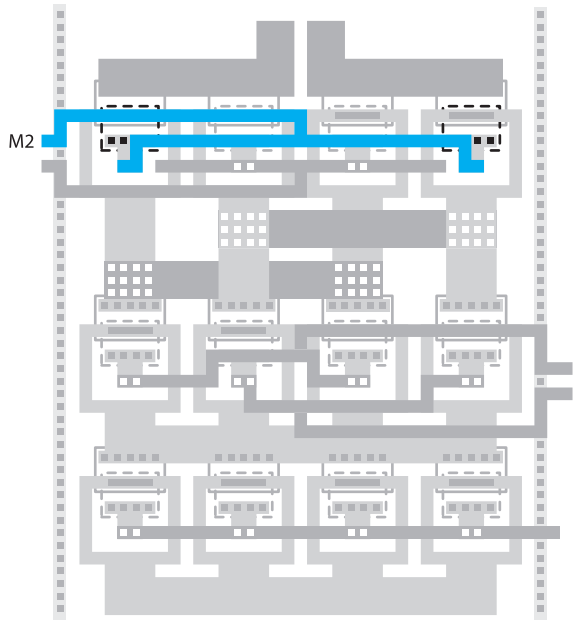


Figure CS2–48. *Bases are far apart due to interdigitation. We connect using Metal Two. Input to these bases is also highlighted.*

Similarly, the bases of QU2 and QU3 need to connect to each other. However, these are more easily connected since they are located near each other. (See Figure CS2–49. *Input wire also highlighted.*)

If you measure the distance between QU1 and QU4, and compare that with the distance between QU2 and QU3, you can see the base connections are of quite different lengths. If we were to just hook these bases to each other, we would see a very different parasitic. Likewise, the base for QU1 and QU4 runs over the emitters of almost everything, also yielding different parasitic conditions.

To try to resolve these matching problems, we stretch the Metal Two out from the QU2—QU3 connection to be a more similar length to the QU1—QU4 connection. This helps balance the parasitics to other nodes as well, i.e., the emitters. You can see the extensions in Figure CS2–49. The wire extends beyond the two bases, yet connect to nothing at the ends.

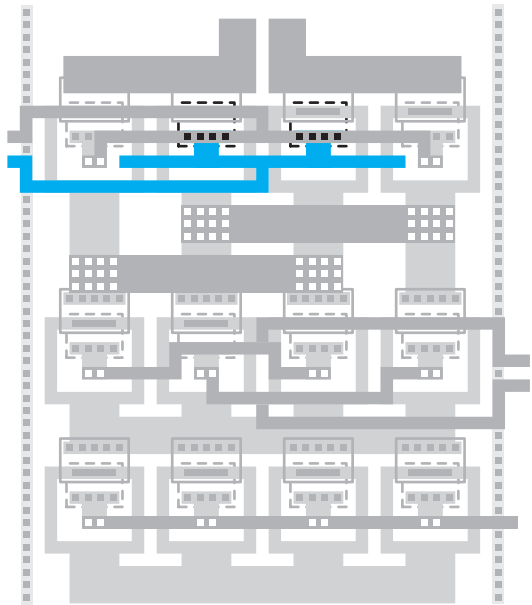


Figure CS2–49. *Bases of QU2 and QU3 easily connect. Extensions are added to try to even the parasitics with the other diff pair.*

Inputs

Similar to what we did with the Lower Pair inputs, we will also make the Upper Quad inputs of equal length. An easy way to do this, again, is to wrap one above and one below, so they can come into the wiring pair at the same central point. (See Figure CS2–50.)

Figure CS2–51 shows the completed input wiring scheme. You should be able to see by the appearance that the parasitics seem much more evenly matched and much lower than in the first layout of this Case Study. The lengths are fairly even. They run to the centers. They run over the tops of other layers in fairly similar conditions.

Loads

Let's examine the Load portion of our second layout. You can see the completed view in Figure CS2–52.

Interdigitation Plan

We have interdigitated the resistor Loads with each other as we have done elsewhere. The plan we drew before interdigitating is shown in Figure CS2–53.

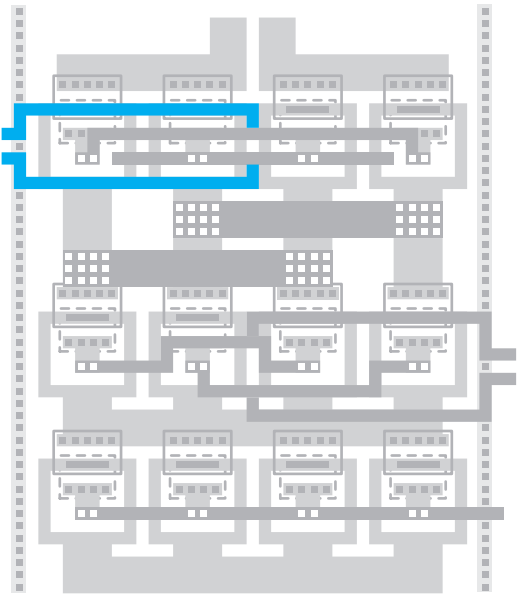


Figure CS2–50. *Equal length inputs help matching.*

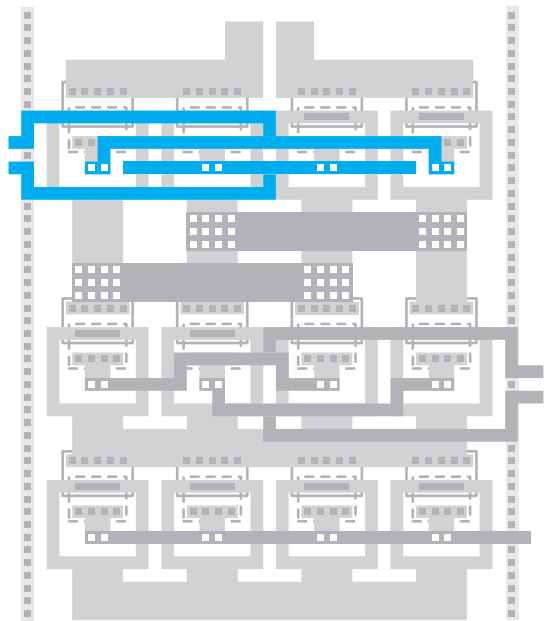


Figure CS2–51. *Completed figure of Upper Quad input traces.*

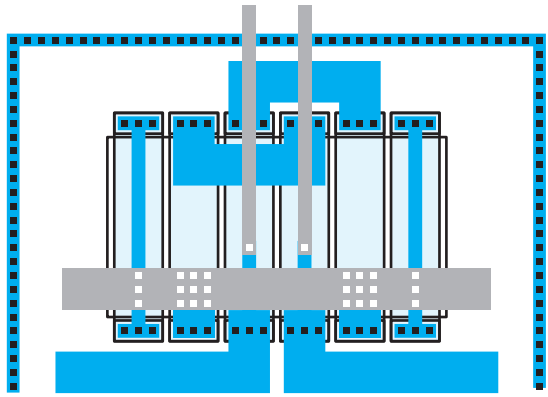


Figure CS2–52. *Load section of second layout.*

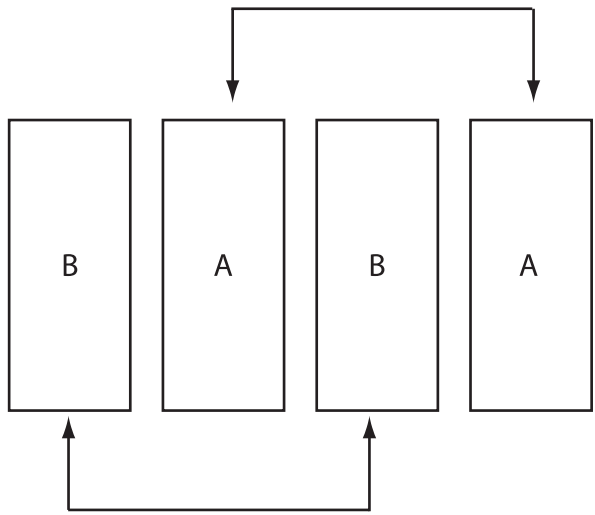


Figure CS2–53. *Planning to interdigitate the resistor Loads.*

Resistors

As with the current-source resistors, all the metalization has been widened to reduce any parasitic resistances and improve current handling capabilities. The collectors connect directly to the center of the resistors to ensure symmetry.

The resistors are connected to VCC in Metal Two with a nice wide wire with lots of vias. (See Figures CS2–54 and CS2–55.)

Notice how we have connected the four resistors, in Figure CS2–56. This resembles our hand drawn interdigitation plan. We are interdigitating because we mentioned in our analysis that the Load could use some improved matching.

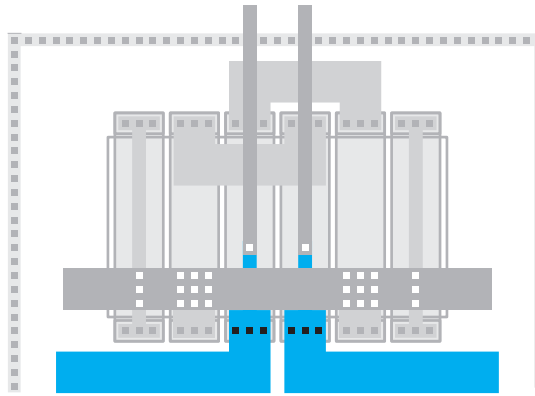


Figure CS2–54. *Collectors from below connect directly to center of resistors in wide metal.*

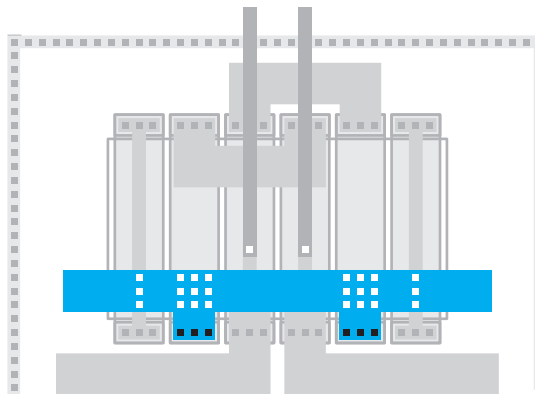


Figure CS2–55. *VCC connection.*

Similar to what we did with the resistors in the Current Source section, we have placed two dummy resistors here in the Load area as well, to try to improve the matching during manufacture. They are tied off to the power rail so they are of no consequence to the functioning of the circuit. (See Figure CS2–57.)

Outputs

The Load resistors are connected to the collectors of the Upper Quad directly in Metal One.

Again, to try to preserve the symmetry, we are connecting our start of the resistor chain in the middle of the cell. We bring the outputs symmetrically out through the middle of the cell in Metal Two. (See Figure CS2–58.)

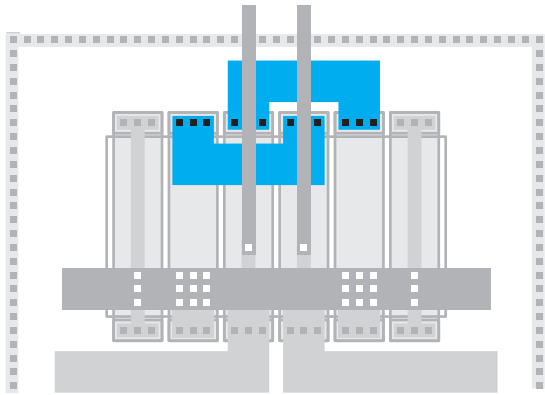


Figure CS2–56. *Following our interdigitation plan.*

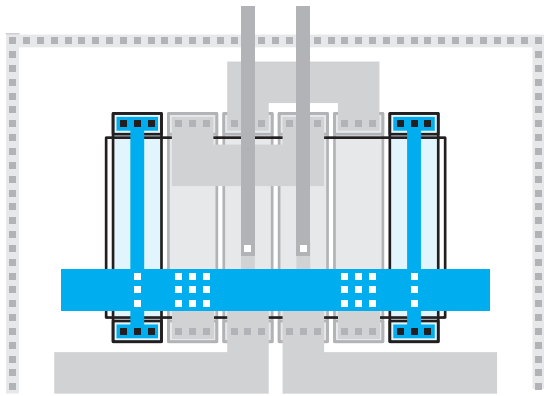


Figure CS2–57. *Dummy resistors placed to improve matching of inner resistors.*

That pretty well sums up a close look at how our second layout is put together. Yet, even though we may have spent many additional hours to implement improved methods into our layout, and even though it appears much improved over the first version, we might not settle for this level of complexity. We might decide we need to take our layout to the next Mario World, one level up.

Analysis of Second Layout

If you compare the complete figures showing the first and second layouts, Figures CS2–5 and CS2–28, you can surely see a large improvement regarding our matching and parasitic concerns. This might be as much effort and chip size necessary for your particular requirements. However, if our circuit calls for additional measures, we can improve this layout even further.

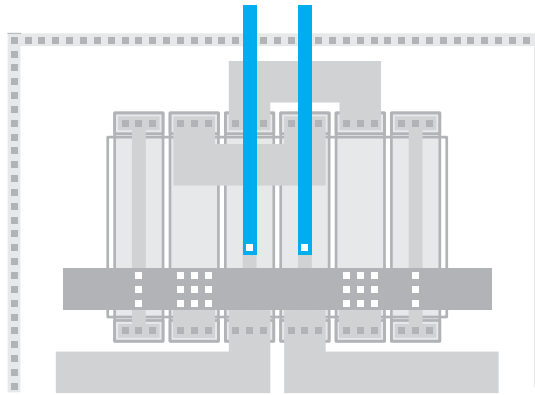


Figure CS2–58. *Outputs run north centrally and symmetrically.*

At the moment all we have for the Lower Pair is interdigitated transistors. Interdigitation gives you good match, but cross-quading can give you even better matching. Depending on the requirements of your circuit, you might be asked to do some cross-quading on top of what you have already done.

If we wanted to cross-quad the diff pairs of the Upper Quad we would have to break the devices in two. This is a Bipolar process, and as we stated earlier, we only have certain transistor types we can work with. Let's assume for this exercise we do not have a half-size device, so the only pair of transistors we can cross-quad is the Lower Pair since they are M=2 anyway.

As another matching improvement, if you are really unlucky, you may even be asked to place dummy Bipolar transistors around your active devices, just as we placed dummy resistors in the Current Source and the Loads. You can do it, but we have not bothered for this example. Some people even go to the extreme of dummying the wiring. There seems to be no end to what you can do with enough time and space.

So, in our final layout improvement, we will cross-quad the Lower Pair. In fact, since the rest of the third layout is not altered, we will only examine the cross-quading of the Lower Pair, with a close look at some specific techniques used.

Let's have a look.

Third Layout

Refer to the layout in Figure CS2–59 for this section of the Case Study.

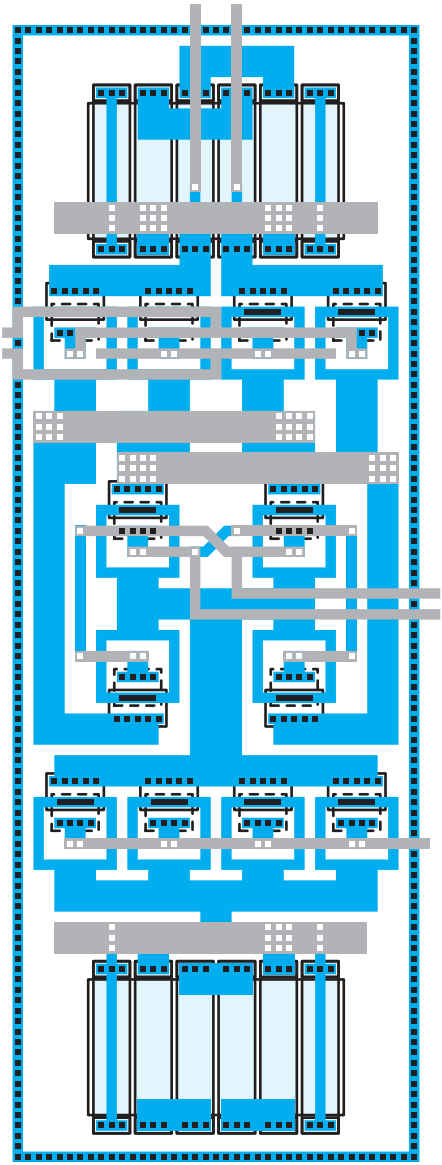


Figure CS2–59. *Full view of third layout.*

Lower Pair

Figure CS2–60 shows a complete view of the finished Lower Pair. Let's begin to dissect this layout to see what techniques were used in the cross-quad.

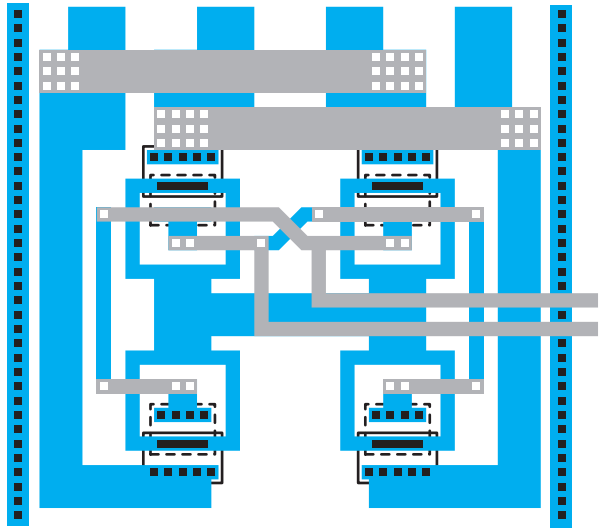


Figure CS2–60. *Finished view of cross-quad of Lower Pair.*

Cross-Quading Plan

You can see our plan for the cross-quad in Figure CS2–61.

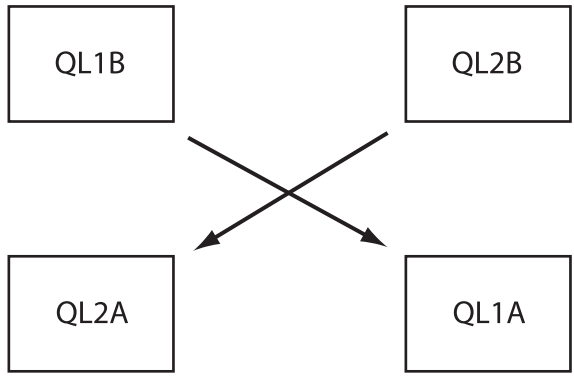


Figure CS2–61. *Planning the cross-quading of the Lower Pair.*

Emitters

We have pulled our current out through the center of the devices from a central common point. So, now, each transistor's emitter sees the same impedance in relation to this common central point.

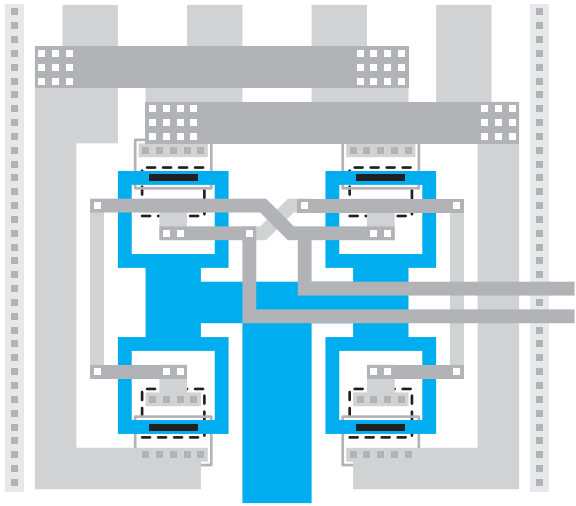


Figure CS2–62. *Evening the impedance by routing through a common central point.*

Collectors

The wiring for our collectors is a bit more complicated now. We have to wire collectors from corner to corner, so we have to wrap our wiring around the outside of the devices. I am following the horizontal/vertical metal layer scheme. We could have continued to wrap the collectors around still using Metal Two, but we need to run the inputs in Metal Two. Likewise, on the base connections, we could have done some of that wiring in Metal Two as well, but we needed to get the base connections in. If we had more levels of metal we could have done whatever we wanted. With only two layers the horizontal/vertical scheme works well.

The wire for this connection has to be a reasonably thick piece of metal because we do have substantial current flowing in the collectors. (See Figure CS2–63.)

For the other collector, we run metal in the same shape (for matching reasons) out and around the outside. Both sides are matched fairly evenly.

Bases

Finally, we get to cross-quad our input traces. Unfortunately, we have that big, fat emitter coming through the middle. Otherwise that would have been a natural place to run our cross-over connection.

We will have to do some messing around in order to get the connections wired nicely, because of the emitter wire. In Figure CS2–65, you can see what we

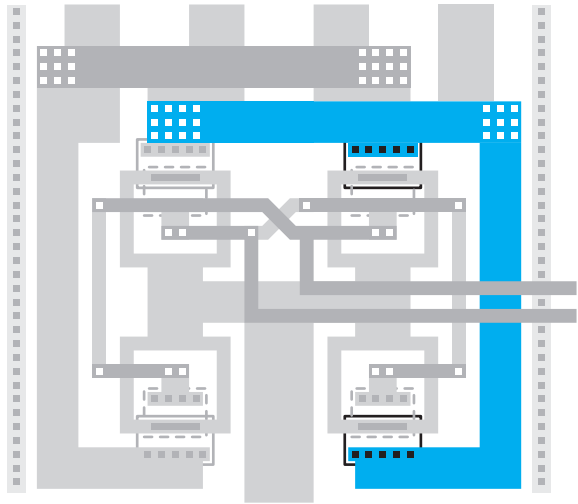


Figure CS2–63. *First collector connection.*

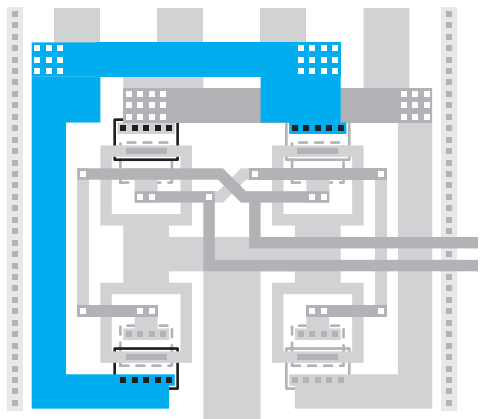


Figure CS2–64. *Second collector connection.*

have done for QL1. We string the base connection across the center then pull it south. Notice we had to extend the wiring so that it travels outside the devices.

Likewise, for QL2, we must run all over the place in Metal Two in order to connect bases. We try to keep the connection as similar to the QL1 connection as possible. (See Figure CS2–66.)

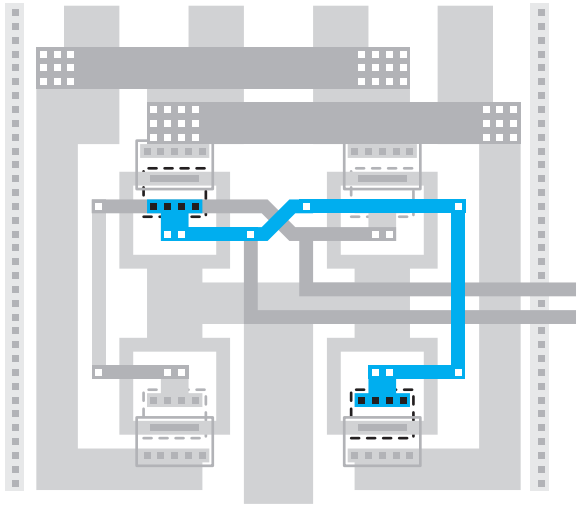


Figure CS2–65. *Connecting QL1 bases requires gymnastics in Metal Two.*

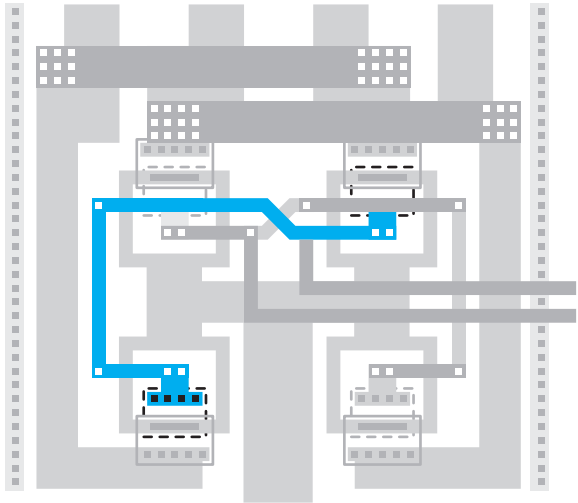


Figure CS2–66. *Connecting bases for QL2 in Metal Two is likewise difficult. By the way, if you are only looking at the pictures and reading captions, you are missing information. Just thought we'd let you know.*

Inputs

If you have noticed, we have a weird cross-over in the middle. Then we pull the bases to the outside. (See Figure CS2–67.)

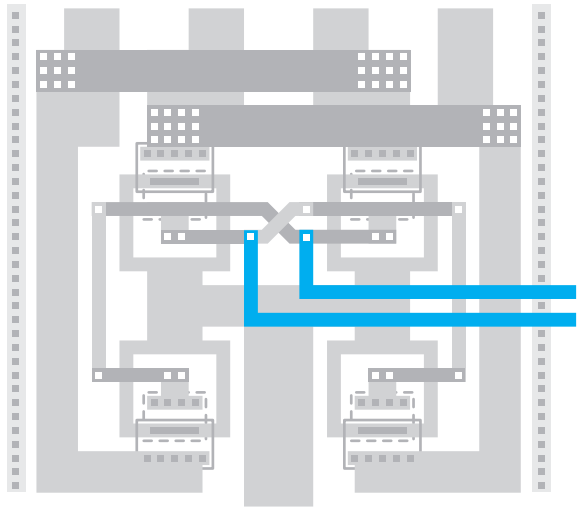


Figure CS2–67. *Inputs run to the central cross-over point. And, if you are just looking at the pictures and not even reading the captions, then someone sitting next to you will have to tell you that you are missing information, because I can't.*

As we did with our input traces before, we bring the two traces into a central common point to balance the parasitics on both of them as much as possible.

Figure CS2–68 shows the two complete input paths. Despite a first impression of chaos, if you examine the traces, they should appear well thought-out for matching, symmetry, and parasitics.

Remember how we used poly as a third conducting layer in our cross-over from the op-amp Case Study? That was a creative solution. However, we don't need three conducting layers this time, so we can get away with using just two metal layers for all our cross-over wiring.

The connections to the Upper Quad can just go straight in, as you can see in Figure CS2–69.

The other sections of our layout, as we mentioned, are identical to the second layout we discussed before. All we have changed in this third version is the cross-quading of the Lower Pair.

Connecting the emitters together with big, fat wire provided a nice solution to our potential current density issue. Since that grand scheme took so much space in the center of our cross-quad, we were forced to do the actual crossing

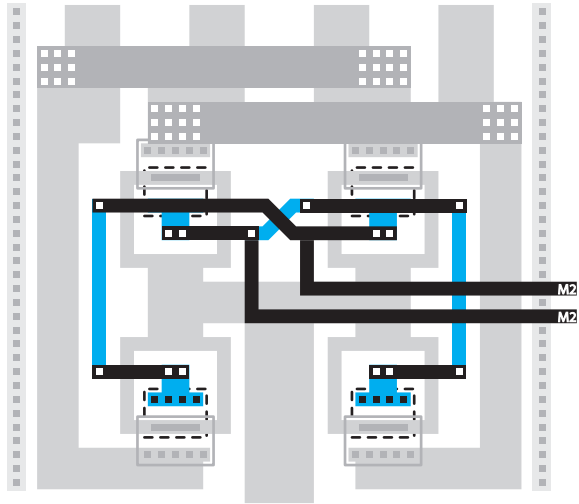


Figure CS2–68. *Completed figure of all input traces. Metal Two is in black. Metal One is in blue.*

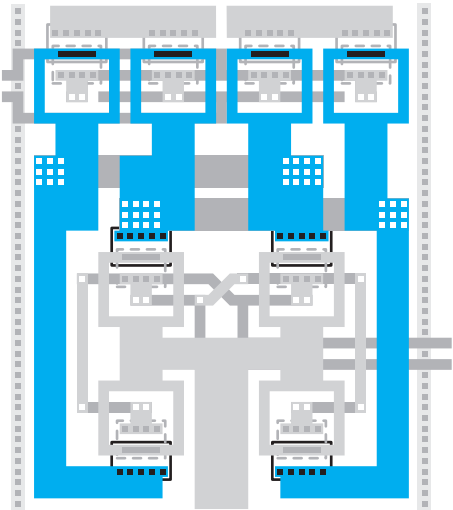


Figure CS2–69. *Connections to the Upper Quad.*

over a little higher than center. But, we were able to keep our wiring fairly equal, getting to this point. The bases are wired fairly symmetrically. This should be a nice mixer.

And that's it. We have finished our layout of the Bipolar Gilbert cell mixer, completed to three different levels of complexity. Although it may resemble a

plate of spaghetti, we have done a lot to improve the matching, and to lower the parasitics on this circuit. Each of the three layouts is a job well done if it fits your given assignment.

Final Analysis

There are plenty of other adjustments we could employ to improve matching even further. For example, our input wires still experience different parasitics. That could be addressed.

But after a point, you start to get into minutia. It's the Law of Diminishing Returns. After awhile, changes do not result in enough gain to justify taking the time to work on them. You have to finally just call it done at some point, even though you could continue to improve a design forever.

Some circuit designers insist everything be totally, 100% matched. Sometimes that is absolutely necessary. The same techniques you have just seen are the ones you will continue to employ to achieve that perfection, if that is what you need to do.

The performance of this third version of our Gilbert mixer layout is probably good enough for most purposes. At any rate, our objective in this book is to demonstrate techniques, ways of doing things, thought processes, let's-have-a-look-at-how-someone-else-has-done-some-layout sort of coverage, not necessarily extend to every potential nuance of one particular example.

So, this is as far as we need to go in our second Case Study in order to give you all the proper tools. You can use these techniques in your everyday layout, even in digital layout assignments. The more you practice these techniques routinely, the better your circuits will perform, and the easier your job will be.

Comparison of Case Study 1 and Case Study 2

The first Case Study demonstrates the techniques and communication avenues you experience in working with a team to create your CMOS layout. The second Case Study demonstrates the kinds of worries you battle at really high frequency, particularly radio frequency circuit applications. Each Case Study illustrates the challenges facing a typical mask designer, and the different working styles of CMOS and Bipolar.

In CMOS, the challenges are speed and quantity of design. A person walking past your desk might see you madly duplicating finished chunks of cells all

over the screen. Some of the best CMOS design is done with the *copy, reflect*, and *paste* functions.

Generally, the challenges with Bipolar layout result from the very high frequencies involved. You will devote your time to thinking about matching, symmetry, and parasitics. One wire might take all day to plan. And, the higher your frequency, the more planning your work requires. A person walking past your desk might see you leaning back in your chair, staring blankly at your computer screen. Some of the best Bipolar design is done this way.

As you can see, there are very different challenges in Bipolar layout than there are in CMOS. The two processes are beginning to mesh in the field. What is good practice for one can be good practice for the other.

We hope you will reread both Case Studies to discover more depth and nuances that you did not quite catch before, so that you can incorporate all these techniques into your daily routine. With practice, you can be the critical backbone of your design team, if you aren't already.

Have fun. Keep in touch.

END OF CASE STUDY 2