Introduction to CMOS VLSI Design (E158)

Harris

Lab 1: Gate Design

The only way to become a good chip designer is to design chips. This is the first of five labs in which you will use the Electric VLSI Design System to design an 8-bit MIPS microprocessor. The labs will guide you through mastering schematic entry, layout, simulation, and verification of a complex system. Chip design is a very time-consuming activity, so certain portions of the processor will be provided for you to reduce the tedious work and to illustrate good design style. The design you do in each lab will form a component of the microprocessor, so careful work on each lab will save you time at the end.

This lab begins with a review of the MIPS processor microarchitecture that you will be implementing. It then guides you through the design of a 2-input NAND gate. You will draw and simulate schematics. Then you will draw the layout and verify that it satisfies design rules and matches the layout. Using your NAND gate and an inverter, you'll design a 2-input AND gate. Finally, you'll design your own 2-input NOR and OR gates.

As these are new labs, you will be asked in each lab how much time you spent on the lab and how you would modify the lab manual to make it more clear for students next year. These questions do not impact your grade but are much appreciated to improve the labs for the future.

1. MIPS Processor Overview

Your ultimate goal in this series of labs is to construct a MIPS microprocessor. In the interest of simplicity, your processor will only be responsible for the following instructions:

```
ADD, SUB, AND, OR, SLT, ADDI, BEQ, J, LB, SB
```

The MIPS architecture is a 32-bit architecture, meaning that registers and busses are 32-bits wide. This would involve much repetitive drawing, so we will construct an 8-bit subset. All datapaths and registers except the instruction register will be 8 bits wide. The instruction still must be 32-bits wide to contain a complete opcode, but it will be fetched from an 8-bit wide memory using four successive fetch cycles. Nevertheless, the memory address is only 8 bits so the design will support only $2^8 = 256$ bytes of memory. Finally, the MIPS architecture defines 32 registers. Again, to save drawing, we will use only 8. Register 0 is still hardwired to the value 0.

We will construct a multicycle implementation of the MIPS processor, as defined in section 5.4-5.5 of Patterson & Hennessy, Computer Organization and Design (2nd Ed.). If you are feeling rusty on the microarchitecture, you should review that section of the book.

There are a few changes between the design in Figure 5.33 of the book and the design you will be constructing. The book assumes a 32-bit path to memory, so instructions can be loaded in a single cycle. You will have only an 8-bit path to memory, so instructions will be loaded in four successive cycles with four separate IRWrite controls. Sign extension is not necessary because only the bottom 8-bits of constants are used. The book shows datapaths for the `lw` and `sw` instructions. You will be implementing `lb` and `sb` so the shift left 2 block is unnecessary. The modified datapath is shown in Figure 1 below. Finally, the control microsequencer of Figure 5.46 will be modified to handle the four-cycle instruction fetch and the `addi` instruction not in the textbook design.

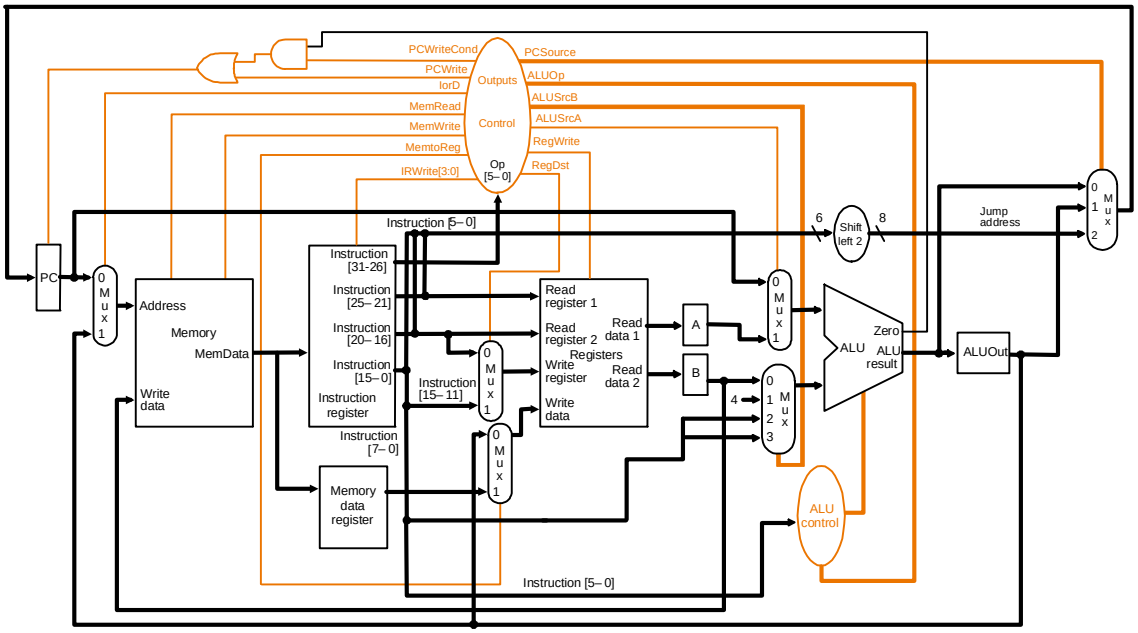

Figure 1: 8-bit MIPS Datapath

2. The Electric VLSI Design System

Integrated circuits have become sufficiently complex that Computer-Aided Design (CAD) tools are essential; nobody could design a 100-million transistor chip by hand on a schedule that would complete the chip while it was still interesting. We will use CAD tools for schematic entry, layout, simulation, and verification. Unfortunately, no CAD tools are perfect.

The leading industry-standard tool is made by Cadence. It normally sells for six figures per seat, but is available at extremely generous discounts to universities. However, it runs only on Unix and is nearly a full-time job to setup and maintain. The Tanner tools run on

NT and are easier to support, but cost more than Cadence for universities! The freely available Magic and Sue tools are popular at some schools, but Magic again is limited to Unix and has a primitive, albeit powerful, user interface. All of these tools have their fair share of bugs.

The Electric VLSI Design System is a computer-aided design tool developed by Steve Rubin. It has powerful notions of connectivity, runs on all major operating systems, and is very well integrated. It is also GNU-licensed so you may freely download your own personal copy and even modify the source code. The drawback is that many features of Electric are still in development, so the tool crashes often and sometimes does unintuitive things. Steve Rubin and Harvey Mudd College have agreed to help each other develop Electric into an outstanding, freely available tool. Harvey Mudd has submitted over 300 requests for bug fixes and feature enhancements and the vast majority of the requests have been incorporated into the tool. One of the major goals in this class is to find more problems with Electric and improve the tool until it is a very stable and productive design environment. You will certainly run into frustrations along the way; this is typical of almost any cutting-edge field in which the tools have not caught up with design practices. You can help by submitting detailed reports of the problems you encounter so that Dr. Rubin can isolate the problem. You will receive extra credit for your reports if they are reproducible.

To submit a bug report, email Prof. Harris with the following information:

Your name:
Date:
Facet:
A facet demonstrating the problem: list the library name, facet name, and facet view and attach the library to the email.
Description:
A detailed description of the problem, including the exact steps necessary to reproduce the problem.

Thank you for your patience with the tool. Your work submitting bug reports will make the tool better for the entire design community!

3. Getting Started

The latest version of the Electric CAD tools is kept in the class directory at \Igor\Courses\Eng\E158\Electric. We receive frequent updates, so You may work from any Windows NT machine. Your personal PC or the Engineering Design Center computers are good choices. Electric is also available for the Mac and Unix. However, we will only be receiving frequent bug fixes for the NT version, so that is the recommended platform this semester. You may wish to map the Eng\Class directory as a network drive for convenient access.

Copy the e158.elib library to your account and rename it to lab1_xx where xx are your initials. This library contains some parts of the MIPS processor that are provided to you. You will add your new designs to the library as you work through the labs.

Double-click on Electric to start the program. Dismiss the splash screen. Choose Info • See Manual from the menu to bring up the Electric manual in a web browser. The manual is also available online at `www.staticfreesoft.com`. Skim through the following sections:

Chapter I: 1-2, 6-9
Chapter II: 1-6
Chapter III: 1-12

Don't worry if it doesn't all make sense yet. After you complete this lab, go back and skim over the sections that you initially found confusing. Refer back to the other chapters of the manual as you need help with specific features of Electric.

4. Schematic Entry

Your first task is to create a schematic for a 2-input NAND gate. Recall that each design is kept in a *facet*; for example, your schematic will be in the nand2{sch} facet, while your layout will eventually go in the nand2{lay} facet and your AND gate will go in the and2{sch} facet. Choose File • Open Library to open your lab1_xx library. Choose Facet • Edit Facet to bring up the Facet dialog. Click New Facet. Enter nand2 as the facet name and schematic as the view. A new editing window will appear with the title lab1_xx:nand2{sch} indicating the library, facet name, and view.

Electric defines various technologies for schematics and layout. To draw transistor-level schematics, you will need to select the Analog Schematic technology by choosing Technology • Change Current Technology and selecting the *schematic*, *analog* technology. This technology file contains basic circuit elements such as transistors, resistors, capacitors, and power and ground.

Your goal is to draw a gate like the one shown in Figure 2. Choose Windows • Toggle Grid to turn on a grid to help you align objects. Left-click on an nMOS transistor symbol in the components menu on the left side of the screen. Left-click on your schematic window to drop the transistor into your layout. Repeat until you have two nMOS transistors, two pMOS transistors, the circular power symbol, and the triangular ground symbol arranged on the page. You may move the objects around by left-clicking and dragging. The transistors default to a width/length value of 2/2. Double-click on the pMOS transistor and change its width to 12. Recall that nMOS transistors are roughly twice as strong as pMOS transistors. So a single nMOS transistor would only have to be 6 wide. However, because the nMOS transistors are in series, they should also be 12 wide.

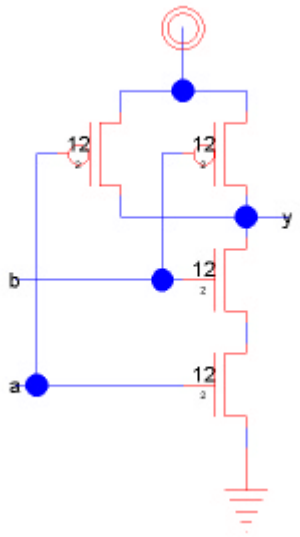

Figure 2: nand2{sch}

Now, make the connections. Left-click on a port such as the gate, source, or drain of a transistor. Right-click on another port to create a wire connecting the ports. Continue until all the blue wiring is completed.

Finally, you will need to provide *exports* defining inputs and outputs of the facet. Left click on the end of a wire where you need to create the export for input *a*. You should see a small square box highlighted at the end of the wire. If the entire wire is highlighted, you clicked on the middle of the wire instead of the end, so try again. Once you have selected the end of the wire, choose Export • Create Export. Give your export the name *a*. Give it the characteristic Input. Repeat with the other input *b*. Export *y* as an Output.

Use File • Save (*Ctrl-s*) to save your library. Get into the habit of saving often because Electric crashes fairly often. Also, learn the keyboard shortcuts for the commands you use frequently.

5. Switch-Level Simulation

Our next step is to simulate the schematic to ensure it is correct. Electric has two built-in *switch-level simulators*: IRSIM and ALS. ALS is buggy, so we will use IRSIM. IRSIM treats transistors as switches that may be ON or OFF; it also understands resistance and capacitance based on transistor sizes and crudely estimates switching delays.

First select Tools • Simulation (Built-in) • Simulation Options. Make sure that IRSIM is selected for Simulation engine. Check that the scmos0.3.prm parameter file is selected; this contains resistance and capacitance data for the technology we are using. Start the simulation by selecting Tools • Simulation (Built-in) • Simulate. A waveform window will appear listing each of the *nets* in your design. If you created your exports properly,

you will see nets for *a*, *b*, and *y*, as shown in Figure 3. You will also see an unnamed nets for the node between the series nMOS transistors; in the figure it is named net3.

The simulator has two vertical white cursors. The primary cursor with no x is used to create stimulus. Click and drag the cursor near the left edge (near time 0). Click on the *a* input in the simulation window and press *h* or 1 to drive the input high. Drag the cursor to some later time. Click on the *b* input and press h to drive it high. Use the *l* or 0 key sometime later to drive *a* and *b* back low, as shown in the figure. Check that the *y* output matches the behavior you would expect for a NAND2 gate. If it does not, fix the bug in your schematic and resimulate.

The secondary cursor with an x is only used to measure delays relative to the first cursor. You will find that unloaded gate delays are under 100 ps. Gate delay depends on the capacitance being driven, so attaching a realistic load will slow the gate.

Use the Windows • Adjust Position • Tile Vertically command to arrange the windows so that you can see both the simulation window and your schematic at the same time. Watch the color-coding of the wires in the schematic change as you drag the primary cursor back and forth. On the schematic, magenta indicates high, blue indicates low, and black indicates x, an illegal value. When a signal is selected in the waveform window, the corresponding net will be highlighted on the schematic. Similarly, when a net is selected in the schematic then its waveform representation will be highlighted. Watching the voltage levels change on the schematic is helpful for debugging problems. Study your simulation and determine why the node between the two nMOS transistors behaves as it does. A thick purple bar means that the state is floating, undefined, or could not be made into a definite logic level.

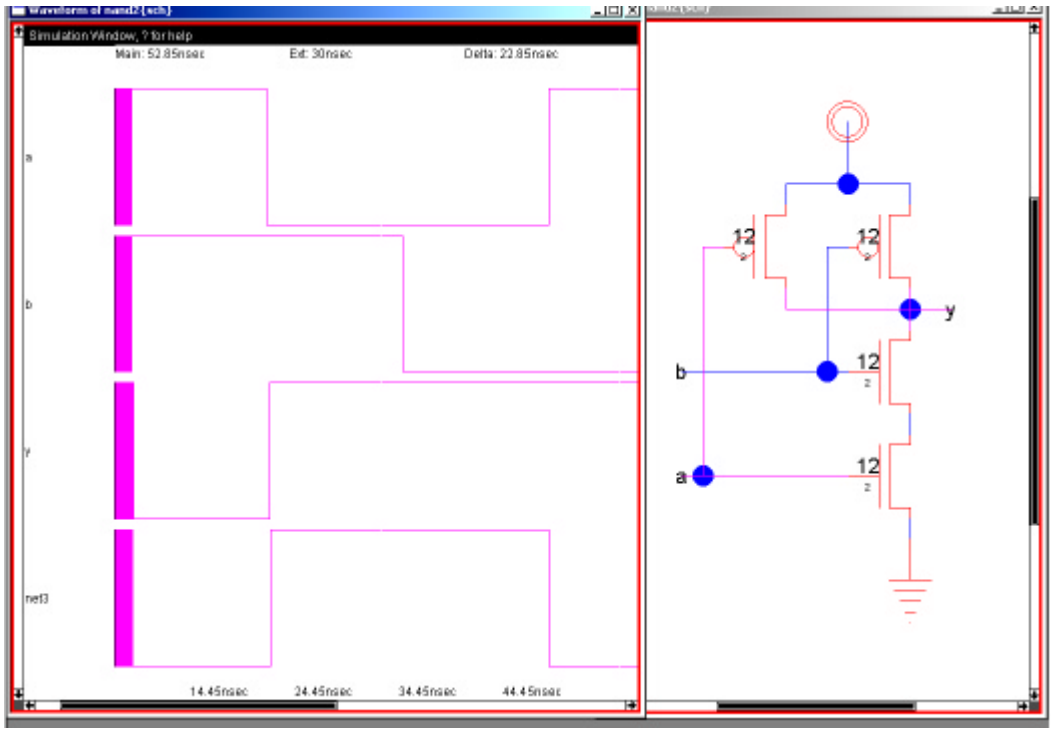

Figure 3: Simulation of nand2{sch}

Get in the habit of simulating each facet after you draw it so you catch errors while the design is fresh in your mind.

6. Layout

Now that you have a schematic simulating correctly, it is time to draw the layout. Choose Facet • Edit Facet to bring up the Facet dialog. Click New Facet. Enter nand2 as the facet name and layout as the view. We will be targeting the AMI 0.5 μm process but using MOSIS submicron scalable rules so we could easily adapt to the AMI 1.5 μm process or others. To setup the technology file, choose Technology • Change Current Technology and select *mocmos*, the MOSIS CMOS technology. Then choose Technology • Change Units and set Lambda for mocmos to 600 half-milimicrons, i.e. 0.3 μm.[1] Use the Arc • New Arc Options to set the default width for Metal-1 to 4 and for Metal-2 to 4 for convenience of layout. Finally, choose Technology • Technology Options and select 3 metal layers, submicron rules, and alternate active and poly contact rules. The submicron rules are documented on the MOSIS web page. Alternate Active and Poly contact rules are an older, cleaner set of design rules that don't involve half-lambda dimensions.

Your goal is to draw a layout like the one shown in Figure 4. It is important to choose a consistent layout style so that various cells can "snap together." In this project's style, power and ground run horizontally in metal2 at the top and bottom of the cell, respectively. The spacing between power and ground is 80λ center to center. No other metal2 is used in the cell, allowing the designer to connect cells with metal 2 over the top later on. nMOS transistors occupy the bottom half of the cell and pMOS transistors occupy the top half. Each cell has at least one well and substrate contact. Inputs and outputs are given metal1 ports within the cell.

You may find it convenient to have another sample of layout visible on the screen while you draw your gate. Use the Edit • New Facet Instance command and select inv{lay}, then click to drop this inverter in the layout window. Highlight the inverter and use the Facet • Expand Facet Instances • All the way command to view the contents of the inverter. Study the inverter until you understand what each piece represents.

[1] Lambda is normally defined as half of the minimum drawn gate length. Therefore, the minimum drawn gate length (polysilicon width) is 0.6 μm even though the vendor describes the process is 0.5 μm.

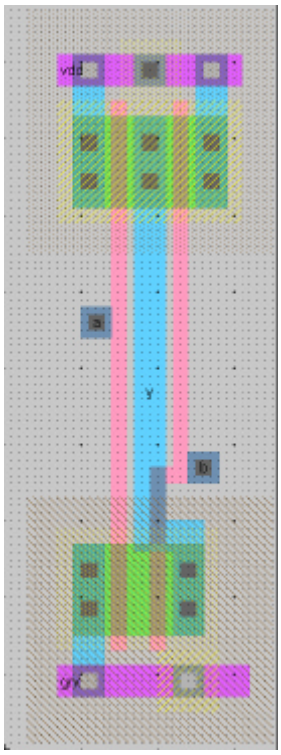

Figure 4: nand2{lay}

Start by drawing your nMOS transistors. Recall that an nMOS transistor is formed when polysilicon crosses N-diffusion. N-diffusion is represented in Electric as green diffusion surrounded by a dotted yellow N-select layer all within a hashed black P-well background. This set of layers is conveniently provided as a 3-terminal transistor node in Electric. Move the mouse to the components menu on the left side of the screen. As you move the mouse over various objects, the node name will appear on the status line next to the word NODE near the bottom left corner of the screen. Left click on the N-Transistor, shown in Figure 5, and click again in the layout window to drop the transistor in place. Use the Edit • Rotate • 90 Degrees Counterclockwise command to rotate the transistor so that the red polysilicon gate is oriented vertically. There are two nMOS transistors in series in a 2-input NAND gate, so we would like to make each wider to compensate. Double-click on the transistor. In the node information dialog, adjust the width to 12.

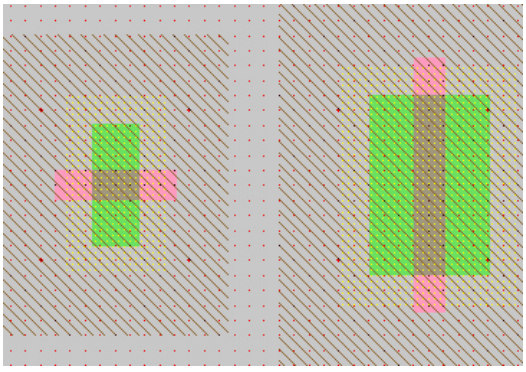

Figure 5: nMOS transistor before and after rotation and sizing

We need two transistors in series, so copy and paste the transistor you have drawn or use the Edit • Duplicate command. Drag the two transistors along side each other so they are not quite touching. Left click the diffusion (source/drain) of one of the transistors and right click on the diffusion of the other transistor to connect the two. Then drag the two transistors until the polysilicon gates are 3λ apart, looking like they do in Figure 4. You will probably find it helpful to turn on the grid using the Windows • Toggle Grid command. The grid defaults to small dots every lambda and large dots every 10λ. You can change this with the Windows • Grid Options command. Also by default, objects snap to a 1-λ grid. This can be changed by using Windows • Alignment Options. You can also move objects around with the arrow keys on the keyboard. Pressing the *h* or *f* keys cause the arrows to move objects by a half or full lambda, respectively. You will avoid messy problems by keeping your layout on a lambda grid as much as possible. Inevitably, though, you will create structures that are an odd number of lambda in width and thus will have either centers or edges on a half-lambda boundary.

Electric has an interactive *design rule checker* (DRC). If you place elements too closely together, it will report errors in the Electric Messages window. Try dragging one of the transistors until its gate is only 2λ from the other. Observe the DRC error. Then drag the transistors back to proper spacing. When you are in doubt about spacing, use the Tools • DRC • Hierarchical Check command to ask Electric to recheck the entire facet and any subfacets it might contain.

An Aside on Design Rules

The definitive design rules are available on the MOSIS web page. MOSIS is a service that collects small orders for designs and combines them into an order large enough to interest a semiconductor manufacturing facility. For best density, one could optimize for a particular fabrication process and design in units of microns rather than lambda. However, the layout would require redrawing when ported to a different process because the design rules and feature size will have changed. MOSIS has developed a set of *Scalable CMOS* design rules that are sufficiently conservative to work for virtually all processes. The original rules, *SCMOS*, apply to older processes. We will be using the *SUBM* submicron rules that are even more conservative and suffice for more advanced processes. They are adequate for both the AMI 0.5 and 1.5 micron processes. Finally, the *DEEP* deep submicron rules are slightly more conservative and are necessary for best performance in the 0.25 μm and below processes. For historical reasons, all three sets of rules are selected in Electric by choosing the scmossub process, then using the Technology Options dialog. Tables 3 and 4 on the MOSIS web page list the differences between these design rules.

http://www.mosis.org/Technical/Designrules/scmos/scmos-main.html

We will be fabricating using the SCNE technology code defined in Table 5 of the web page. SCNE means Scalable CMOS with N-wells and an Electrode layer, i.e. polysilcon 2. Click on each of the layers in the table to see the design rules. For example, the Metal1 rules are shown in Figure 6 below. We will follow the SUBM rule set, requiring metal width and spacing of 3λ.

SCMOS Layout Rules - Metal1

Rule	Description	Lambda		
		SCMOS	SUBM	DEEP
7.1	Minimum width	3	3	3
7.2	Minimum spacing	2	3	3
7.3	Minimum overlap of any contact	1	1	1
7.4	Minimum spacing when either metal line is wider than 10 lambda	4	6	6

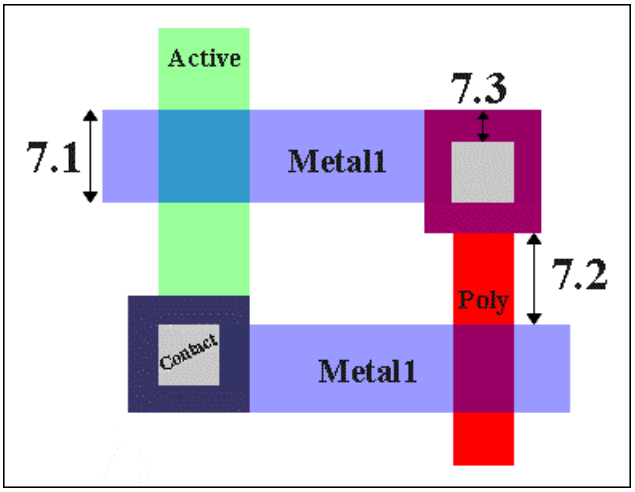

Figure 6: SCMOS Metal1 Rules

Next we will create the contacts from the N-diffusion to metal1. Diffusion is also refered to as *active area*. Drop a square of Metal-1-N-Active-Contact in the layout window and double-click to change the properties to a Y size of 12. You will need a second contact for the other end of the series stack of nMOS transistors, so duplicate the contact you have drawn. Move the contacts near each end of the transistor stack and draw diffusion lines to connect the transistors. Then move the contacts even closer; you only need a gap of 1λ between the metal and polysilicon. Use the design rule checker to ensure you are as close as possible but no closer. Using similar steps, draw two pMOS transistors in parallel and create contacts from the P-diffusion to metal1. At this point, your layout should look something like Figure 7:

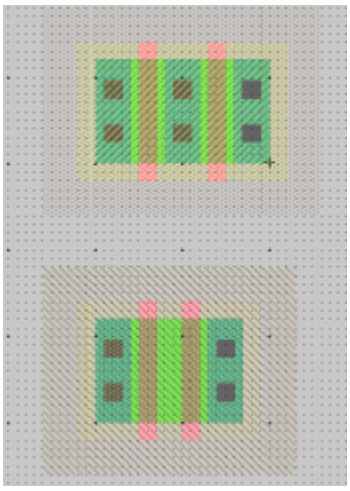

Figure 7: Contacted transistors for nand2 layout

Draw wires to connect the polysilicon gates, forming inputs *a* and *b*, and the metal1 output node *y*. Then add metal2 power and ground lines. You can add a line by selecting a pin such as metal-2-pin from the palette then right-clicking nearby to draw the line. Use the grid to ensure they are 80λ apart from the center of each line. This is the same spacing as the power/ground lines of the inverter. A metal1-metal2-contact, also known as a via, is required to connect the metal1 to the metal2 lines. Left-click on an active contact. Right-click on the ground line. Electric will automatically create the necessary via for you while making the connection. Complete the other connections to power and ground. Let power and ground extend 2 lambda beyond the contents of the cell (excluding wells) on either side so that cells may "snap together" with their contents separated by 4λ so design rules are satisfied.

Recall that well contacts are required to keep the diodes between the wells and source/drain diffusion reverse biased. We will place an N-well contact and a P-well contact in each cell. It is often easiest to drop the Metal1-N-Well-Con near the desired destination, then right click on the power line to create the via. Then drag the contact until it overlaps the via to form a stack of N+ diffusion, the diffusion to metal1 contact, metal1, the metal1-metal2 via, and metal2. Repeat with the P-well.

In our datapath design style, we will be connecting gates with horizontal metal2 lines. Metal2 cannot connect directly to the polysilicon gates. Therefore, we will add contacts from the polysilicon gate inputs to metal1 to facilitate connections later in our design. Place a metal-1-polysilicon-1-contact near the left polysilicon gate. Connect it to the polysilicon gate and drag it near the gate. You will find a 3λ separation requirement from the metal1 in the contact to the metal forming the output *y*. Add a short strip of metal1 near the contact to give yourself a landing pad for a via later in the design. You may find Electric wants to draw your strip from the contact in polysilicon rather than metal1. To tell Electric explicitly which layer you want, move the mouse over the palette until it is over the blue Metal-1 arc square and click. Then draw your wire.

Electric is agnostic about the polarity of well and substrate; it generates both n- and p-well layers. In our process that has a p-substrate already, the p-well, indicated by black slanting lines, will be ignored. The n-well, indicated by small black dots, will define the well on the chip. Electric only generates enough well to surround the n and p diffusion regions of the chip. It is a good idea to create rectangles of well to entirely cover each cell so that when you abut multiple cells you don't end up with awkward gaps between wells that cause design rule errors. To do this, choose Edit • New Pure Layer Node. Create an N-Well-Node and a P-Well-Node. Double-click on each to change the N-Well-Node size so that it entirely covers the existing n-well. Similarly, change the P-Well-Node. You will find the pure layer nodes are annoying because you will tend to select them when you really wanted to select a transistor or wire. To avoid this problem, shift-click to select both pure layer well nodes and use the Edit • Selection • Make Selected Hard to make the layers hard to select. You will then need to hold the Alt key while clicking to select a well. You can use the Make Selected Easy command if you want to restore a well to be easily selected.

If you have not already deleted the inverter that was placed for reference, select it and press delete.

Finally, provide exports for the cell. When you use the cell in another design, the exports define the locations that you can connect to the cell. Click near the end of the short metal1 input lines that you just drew on the left gate. You will see a small white box highlighted, corresponding to the pin at the end of the cell. If you accidentally selected the entire line instead, click elsewhere in space to deselect the line, then try again to find the pin. You may also try holding the *ctrl* key while clicking to cycle through selections. Add an input export called *a*. Repeat for input *b*. Export output *y* from the metal line connecting the nMOS and pMOS transistors. You may have to place an extra pin and connect it to the output line to give yourself a pin to export as *y*. Also export *vdd* and *gnd* from the metal 2 lines; these should be of type power and ground, respectively. Electric recognizes vdd and gnd as special names, so be sure to use these names.

Your design should now resemble Figure 4 and should pass DRC. When you are done drawing the nand2 layout, click on the inverter and press delete to remove it from the facet. Save the library.

Electric has a fun feature to show a 3D rendition of your layout. Use the Windows • 3D Display • View in 3 Dimensions command. Click and drag with the mouse to rotate the layout. You will see the layer stack from diffusion on the bottom through polysilicon, metal1 and metal2. Vias are shown in white and contacts from metal1 to poly or diffusion in black. Does the 3-D visualization match your mental picture of the layout? When you are done, use Windows • 2D Display • View in 2 Dimensions to restore normal viewing.

7. Layout Verification

Layout verification involves more checks than schematic verification. The checks include *Design Rule Checks* (DRC), *Electrical Rule Checks* (ERC), *Network Consistency Checking* (NCC), and simulation.

You will likely find yourself invoking these menu options often and may wish to give yourself keyboard shortcuts. For example, you might use the Info • User Interface • Quick Key Options dialog to map the Tools • DRC • Hierarchical Check command to the F1 function key, Tools • Network • NCC Options to F2, and Tools • Network • Do NCC(LVS) to F3.

First, be sure the design satisfies the layout design rules by running a hierarchical DRC. This checks the layout and any facets it might incorporate; in this case the nand2 is a *leaf* cell and has no subfacets. Correct any DRC violations that might remain.

Next, run Tools • Electrical Rules • Analyze Wells. This check ensures that every N-well has a contact to VDD and every P-well has a contact to GND reasonably close by. ERC will report the number of transistors found. Check that this matches your expectation; it should be 2 nMOS and 2 pMOS for the nand2. It also reports the farthest distance of any part of the layout from a well or substrate contact; if this is greater than 100, consider adding more contacts to avoid latchup risks. If any errors are reported, fix them.

Next, simulate the layout. Apply a complete set of stimulus to the two inputs to convince yourself that the gate is working correctly.

Electric includes a powerful tool called a network consistency checker that checks that the schematic and layout are equivalent. This is especially valuable when your design is too complex to verify completely through simulation. The checker relies upon graph theory algorithms. If your layout and schematic match, you have much greater confidence that a bug hasn't crept into your design. Unfortunately, if the two don't match, the tool becomes very confused and provides few hints. Also, NCC uses a rather complex algorithm and has been the source of quite a few bugs in Electric, so don't trust it blindly.

To use NCC, choose Tools • Network • Network Options. In the dialog box, set for all facets to Recurse through hierarchy. Check the Check export names, Check component sizes, and Ignore Power and Ground boxes. Click OK to close the dialog. Use Facets • Edit Facet to be sure both the nand2{sch} and nand2{lay} facets are open. Close all other facets; NCC assumes the two windows open are the schematic and layout views of the cell being verified. Then choose Tools • Network • Do NCC(LVS). If your design is correct, you should get a message of:

```
Comparing facet nand2{lay} (4 components, 6 nets) with facet nand2{sch} (4 components, 6 nets)
- Flattening hierarchy; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names; Ignoring component sizes
Facets nand2{lay} and nand2{sch} are equivalent (0.1 seconds)
```

This message means that the layout and schematic each have four components, i.e. the two nMOS and two pMOS transistors. They also each have four nets: A, B, Y, and the wire between the series nMOS transistors. Note that power and ground are ignored and therefore do not appear in this count. NCC finds that the facets are equivalent.

If your design is not equivalent in any nontrivial way, NCC will likely mark too much mismatching to be useful. Identifying the error is part of the art of using NCC. If possible, simulate both the layout and schematic carefully to eliminate obvious errors. Carefully examine your design by hand to look for errors. Here are some common errors:

- The layout and schematic being compared should be the only two windows open.
- Port names must agree and be in the same order. For example, the order of series transistors in the nMOS stack on the NAND gate matters. One input is closer to GND than the other. If the inputs a and b are in the opposite order in the layout than in the schematic, you will see the message:

  ******* Export differences have been found! (0.01 seconds)

  If you press the > key repeatedly to show each error until no more errors exist, you will see the inputs highlighted and see the following messages printed:

  NCC: Export names 'nand2{sch}:b' and 'nand2{lay}:a' do not match
  NCC: Export names 'nand2{sch}:a' and 'nand2{lay}:b' do not match
- Transistor sizes must agree.
- Forgetting to export power and ground in the layout will cause many errors.
- Be sure the *Ignore Power and Ground* network option is checked. Otherwise, you will get errors like:

  Note: facet flop{lay} has 1 power and 1 ground net(s),
  while facet flop{sch} has 0 power and 0 ground net(s)

Another powerful tool for tracking problems is NCC Preanalysis. This command lists all the components and networks in the design. You should expect equal numbers of connections in the layout and schematic, so by looking for discrepancies, you can find clues about your design error. Try running Preanalysis on a good design so that you know what the report looks like when the design is correct; that way you will have an easier time interpreting the results when the design is incorrect.

If desperate, you can turn on the Verbose NCC (text) or (graphics) options in the Network options dialog to get hints about how NCC seeks to match components between the layout and schematic.

8. Hierarchical Design

Now that you have a 2-input NAND gate, you can use it and an inverter to construct a 2-input AND gate. Such hierarchical design is very important in the design of complex systems. You have found that the layout of an individual cell can be quite time consuming. It is very helpful to reuse cells wherever possible to avoid unnecessary drawing. Moreover, hierarchical design makes fixing errors much easier. For example, if you had a chip with a thousand nand gates and made an error in the nand design, you would prefer to only have to fix one nand cell so that all thousand instances of the nand inherit the correction than to need to fix each nand individually.

Each schematic has a corresponding symbol, called an *icon*, used to represent the cell in a higher-level schematic. For example, open the inv{sch} and inv{ic} to see the inverter schematic and icon provided. You will need to create an icon for your 2-input NAND gate. When creating your icon, it is a good idea to keep everything aligned to the 1-λ grid, this will make connecting icons simpler and cleaner when you need it for another facet.

Open your nand2{sch} and choose View • Make Icon. Electric will create a generic icon based on the exports looking something like Figure 8. It will drop the icon in the schematic for handy reference; drag the icon away from the transistors so it leaves the schematic readable.

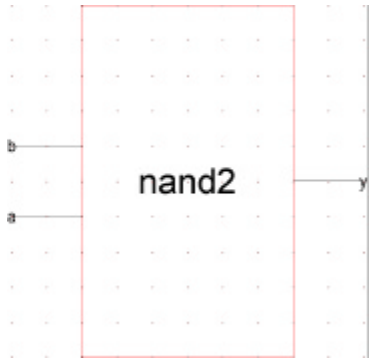

Figure 8: nand2{ic} from Make Icon

A schematic is easier to read when familiar icons are used instead of generic boxes. Modify the icon to look like Figure 9. Pay attention to the dimensions of the icon; the overall design will look more readable if icons are of consistent sizes.

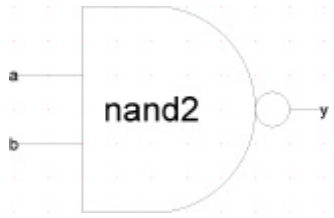

Figure 9: nand2{ic} final version

Click on the icon and choose Facets • Down Hierarchy to drop in to the icon. The technology will automatically change to *artwork*. A palette will appear with various shapes. Delete the generic red box but leave the input and output lines. Turn on the grid.

Place a *facet center* in the middle[2] of the icon using the Edit • New Special Object • Facet Center command. Drag the facet center where you want it. If you deselect it, then try to select it again, you'll find you can't pick it up. The facet center is by default *hard to select*. To select hard to select nodes on a Windows machine, hold the Alt key down while clicking. On other machines, see the Special Select key combination in section 1.8 of the Electric manual. Using facet centers on all your icons and layout facets will avoid nasty off-grid errors later in your work.

The body of the NAND is formed from an open C-shaped polygon, a semicircle, and a small circle. To form the semicircle, place an unfilled circle. Double-click to change its size to 6x6 and to span only 180 degrees of the circle. Use the rotate commands under the Edit menu to rotate the semicircle into place. Place another circle and adjust its size to 1x1. You will need to change the alignment options under the Windows menu to 0.5 to move the circle into place, then set alignment back to 1. Alternatively, you can press h and use the arrow keys to move objects by ½ grid increments, then press f to return to full grid movement.

The opened-polygon shown in Figure 10 can be used to form the C-shaped body. Drop an opened-polygon object. Select it and choose Edit • Special Function • Outline Edit to enter outline edit mode. In this mode, you can use the left button to select and move points and the right button to create points. You should be able to form the shape with four clicks of the right button to define the four vertices. Outline edit mode is not entirely intuitive at first, but you will master it with practice. Choose Edit • Special Function • Exit Outline Edit when you are done. If your shape is incorrect, delete it, drop another opened-polygon, and try again.

Figure 10: Opened-Polygon

Electric is finicky about moving the lines with inputs or outputs. If you left-click and drag to select the line along with the input, everything moves as expected. If you try to move only the export name, it won't move as you might expect. Therefore, make a habit of moving both the line and export simultaneously when editing icons. Note that the line is just an open-polygon and can be shortened if desired by entering Outline Edit mode.

Use the Text item in the artwork palette to place a label "nand2" in the center of the icon.

Now that you have an icon with three exports, create a new schematic called and2. Change the technology back to *schematic, analog* because you are drawing a schematic

[2] It is not strictly necessary for the facet center to be in the true center of the facet. Electric just uses the center to provide a reference point within the facet. This helps Electric keep cells on a clean grid.

again now. Use Edit • New Facet Instance to create ("*instantiate*") the nand2{ic} and an inv{ic}. Wire the two together and create exports on inputs *a* and *b* and output *y*. Double-click on the wire between the two gates and give it a name like *yb* so you know what you are looking at in simulation. It is good practice to label every net in a design if practical. When you are done, your and2 schematic should look like Figure 11. If the line between the gates is black rather than blue, you neglected to return to the *schematic, analog* technology and were still drawing using the *artwork* palette.

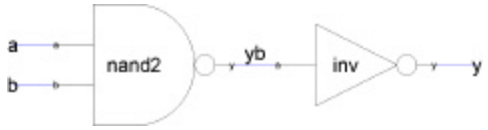

Figure 11: and2{sch}

Simulate your and2 gate to ensure it works. You can use the Tools • Simulation • Down Hierarchy command to descend into the nand gate and see its internal node. In the schematic editor, you can double-click on each gate and assign it a name to help differentiate cells when you simulate.

Next, create a new layout called and2. Change the technology to mocmos. Instantiate the nand2{lay} and inv{lay} layouts. ALWAYS use the New Facet Instance to create layout from preexisting facets. NEVER build a cell by cutting and pasting entire existing cells. If you do, then make a correction to the original cell, your correction will not propagate to the new layout. However, this does mean that you will need to make all the connections to existing ports in your instantiated cells.

Initially the layouts appear as black boxes with ports. Select both and use the Facet • Expand Facet Instances • All the Way command to view the contents of each layout. Wire together power and ground. Move the cells together as closely as possible without violating design rules. Connect the output of the nand2 to the input of the inv using metal1. Remember that connections may only occur between the ports of the two cells. Export the two inputs, the output, and power and ground. The final and gate should resemble Figure 12. Run DRC, ERC, and simulation to verify your design.

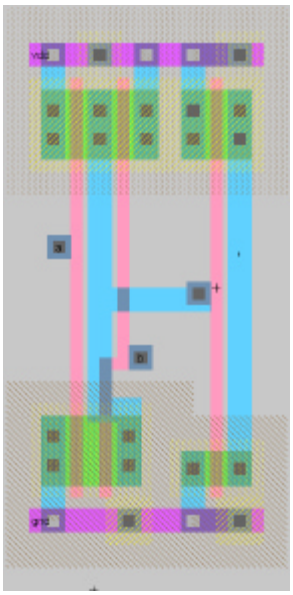

Figure 12: and2{lay}

Finally, do a network compare. NCC has three modes in the Network Options: Check Current Facet Only, Expand hierarchy, and Recurse through hierarchy. Check Current Facet Only is the default if none of the boxes are checked; it assumes all subfacets are perfect. This is usually not what you want. Expand hierarchy smashes your entire design into one giant netlist for its internal comparison. Recurse through hierarchy checks each subfacet on each hierarchical level. NCC is a relatively new feature that has received many bug fixes, so it is wise to try both Expand hexarchy and Recurse through hierarchy to catch any problems that one mode misses. Do not check both Recurse through hierarchy and Expand hierarchy, as it will give meaningless results. In each case, be sure to check Check export names, Check component sizes, and Ignore Power and Ground. Do not set any individual facet overrides. After you check a facet, Electric marks it as clean and thus not in need of rechecking. Press the Clear Valid NCC Dates before each NCC run to ensure NCC will recheck your whole design each time.

If NCC reports a problem, check that your inputs are in the correct order. If *a* and *b* are reversed between the layout and schematic, you will get an error that nodes are wired differently. Tracking NCC mismatches is very difficult, so you are best off ensuring your design is correct by means of careful drawing and simulation rather than drawing sloppily and hoping to catch problems with the checker. Keep a list of any design errors you made that caused NCC problems and note the symptoms reported by NCC. In the future, you can use this experience to help you find problems in complicated designs more quickly.

Now that your and2 gate is complete, use the Info • Check and Repair Libraries command to look for any inconsistencies in your library. You shouldn't expect any, but Electric has been known to do strange things from time to time, especially in the hands of novice users. Therefore, you may wish to check your library every few hours.

9. NOR / OR Gate Design

Your MIPS processor supports the OR instruction as well as the AND instruction. Therefore, you will need to create a 2-input NOR and a 2-input OR gate. Call these nor2 and or2. For each gate, create schematic and layout facets following the same steps you used with the nand2 / and2. Simulate each to ensure correct operation. Run DRC, ERC, and NCC to check each layout. Be sure to save your work frequently. It is wise to keep a backup of your work from time to time in case your library becomes corrupted.

10. What to Turn In

Please provide a hard copy of each of the following items:

1. Please indicate how many hours you spent on this lab. This will not affect your grade, but will be helpful for calibrating the workload for the future.
2. What was unclear in this lab writeup? How would you change it to run more smoothly next time?
3. A printout of your nor2 schematic.
4. A printout of your nor2 layout.
5. A printout of your or2 schematic.
6. A printout of your or2 layout.
7. Simulation waveforms demonstrating correct operation of the or2 layout.
8. What is the verification status of your or2 layout? Does it pass DRC? ERC? NCC?

Extra Credit

As you are probably aware by now, Electric has plenty of bugs and idiosyncrasies. A major goal of this class is to improve the stability and ease-of-use of Electric. Please email your bug reports directly to Prof. Harris in the format described in this lab manual.

Also, if you had any NCC errors that you had to track down, please report what they were. Prof. Harris is compiling a list of common design errors.