Introduction to CMOS VLSI Design (E158)

Harris

Lab 1: Gate Design

The only way to become a good chip designer is to design chips. The hands-on portion of this class is divided into two segments. In the first six formal labs, you will use the Electric VLSI Design System to design an 8-bit subset of a MIPS processor. These 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 each week will save you time at the end. In the second portion of the class, you and a partner will design a chip of your own choosing, putting into practice what you have learned.

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'll draw schematics and simulate them. Then you'll draw the layout and verify 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.

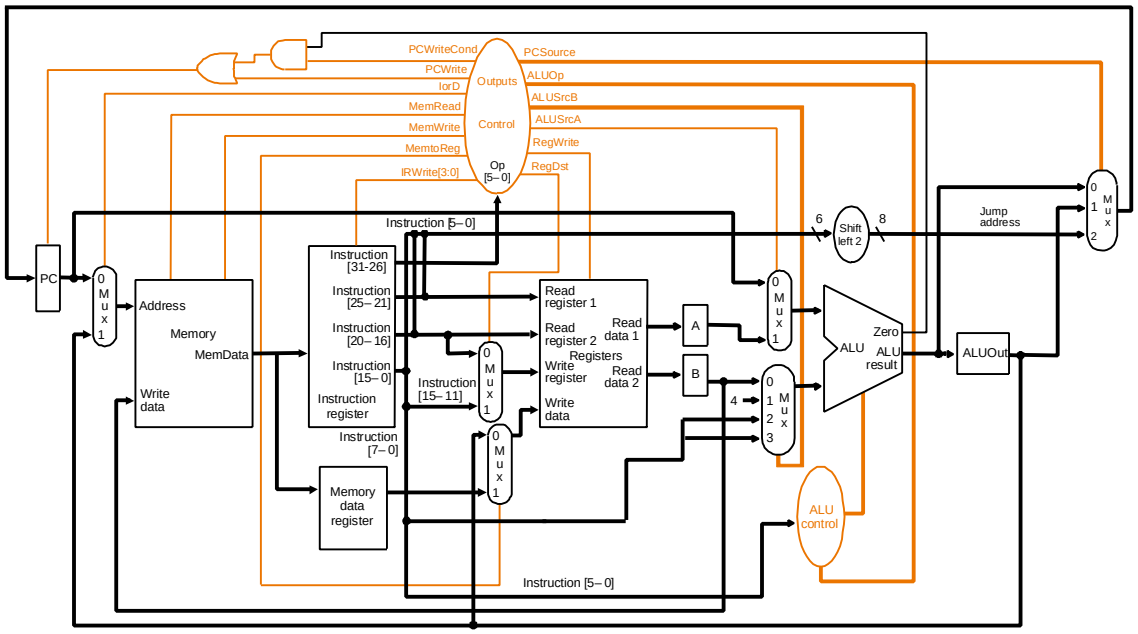

Figure 1: 8-bit MIPS Datapath

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 in this class for schematic entry, layout, simulation, and verification. Unfortunately, no CAD tools are perfect.

The 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. In Fall 2000, freshmen in a chip design seminar submitted over 90 requests for bug fixes and feature enhancements and over two thirds 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 on Kato at home\Eng\Classes\E158\Electric. 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. Read 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. Then 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.

Transistors are labeled ("annotated") with their length and width. Unfortunately, this label sits on top of the transistor and by default tends to get selected when you really want something else. To avoid this problem, choose Edit • Selection • Selection Options and make sure the "Easy selection of annotation text" box is unchecked.

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 layout 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 right-clicking and dragging. The transistors default to a width/length value of 2/2. We will ignore this for now, but will need appropriate sizes in layout.

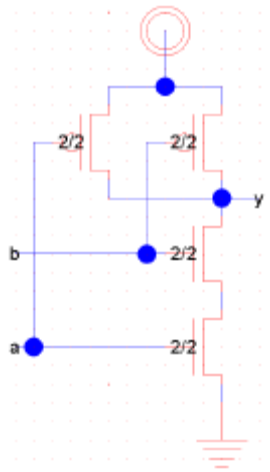

Figure 2: nand2{sch}

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 to save your library. Get into the habit of saving often because Electric crashes fairly often.

5. Switch-Level Simulation

Our next step is to simulate the schematic to ensure it is correct. Electric has a built-in *switch-level simulator*. Such a simulator treats transistors as switches that may be ON or OFF. It assigns unit delays between events. The simulator lacks the detailed timing information one would get from an analog simulation, but is fast and easy to use.

Start the simulation by selecting Tools • Simulation • 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 unnamed nets for VDD, GND, and the node between the series NMOS transistors. In this simulation, we can tell NET1 is power (VDD) and NET4 is ground because they are strongly driven high and low as indicated by the black lines. Thus, NET3 must be the node between the two NMOS transistors. The time scale at the bottom of the simulator is arbitrary and bears no relation to actual circuit performance.

The simulator has two vertical red cursors. The primary cursor with no x is used to create stimulus. Click and drag the cursor to about 40 ns. 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 have no reason to use it because the delays are arbitrary anyway.

If you position the simulation and schematic window so that you can see both simultaneously, you can watch the color coding of the wires in the schematic change as you drag the primary cursor back and forth. These colors correspond to the voltage levels on the wires, with blue indicating low and magenta high. Beware that this is not consistent with the color coding of the waveform window! Watching the voltage levels change on the schematic is helpful for debugging problems. Study your simulation and determine why NET3 behaves as it does.

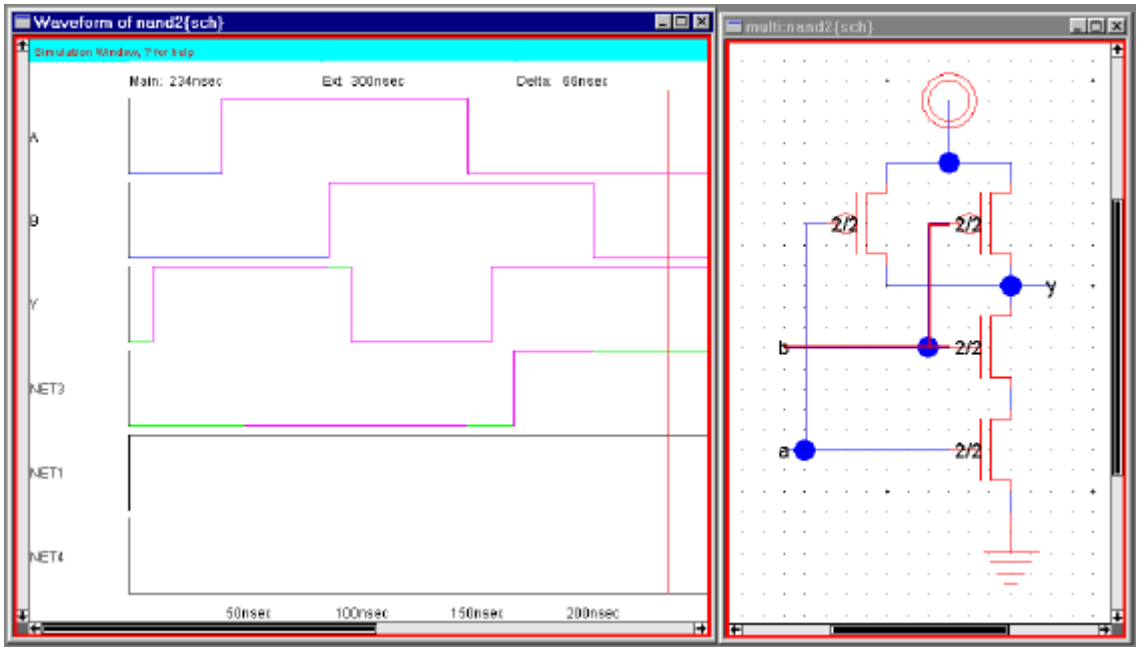

Figure 3: Simulation of nand2{sch}

When you request a simulation, the schematic is translated into a VHDL netlist. VHDL stands for VHSIC Hardware Description Language, and VHSIC in turn was a Department of Defense project on Very High Speed Integrated Circuits. VHDL is similar to Verilog, though somewhat more verbose. The VHDL in turn is translated into an internal text netlist format called net-als. Use the Facet • Edit Facet command to view the nand2{vhdl} facet. You should see something like this:

```
-- VHDL automatically generated from facet nand2{sch}
entity nand2 is port(b, a: in BIT; y: out BIT);
  end nand2;
architecture nand2_BODY of nand2 is
  component PMOStran port(g: in BIT; s, d: inout BIT);
    end component;
  component ground port(gnd: out BIT);
    end component;
  component power port(pwr: out BIT);
    end component;
  component nMOStran port(g: in BIT; s, d: inout BIT);
    end component;
  signal net3, net4, net1: BIT;
begin
  node7_1_1: PMOStran port map(b, y, net1);
  node3: ground port map(net4);
  node5: power port map(net1);
  node7: PMOStran port map(a, y, net1);
  node8: nMOStran port map(b, net3, y);
  node9: nMOStran port map(a, net4, net3);
end nand2_BODY;
```

A VHDL file is divided into entity and architecture descriptions of each cell. The opening *entity* line defines the inputs and outputs. The *architecture* describes the components used by the NAND gate (NMOS and PMOS transistors and power and ground) and how these components are connected. Study the file and see how it relates to the schematic you have drawn. Do the same for the net-als file. If you have simulation problems, it is sometimes helpful to examine the vhdl or net-als files to ensure that the input to the simulator matches what you'd expect.

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 1.5 μm process but using MOSIS submicron scalable rules so we could easily adapt to the AMI 0.5 μm process. To setup the technology file, choose Technology • Change Current Technology and select *mocmossub*, the MOSIS submicron CMOS technology. Then choose Technology • Change Units and set Lambda for mocmossub to 1600 half-milimicrons, i.e. 0.8 μm. Use the Info • 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 2 metal layers, submicron rules, and disallow stacked vias so that the design rule checker will forbid vias placed on top of contacts. Note that the AMI 1.5 um process has two polysilicon layers, but the AMI 0.5 um process has only one. The mocmossub technology shows two polysilicon in the palette. In the interests of being able to target either process, do not use the orange Polysilicon-2 layer.

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 60λ 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.

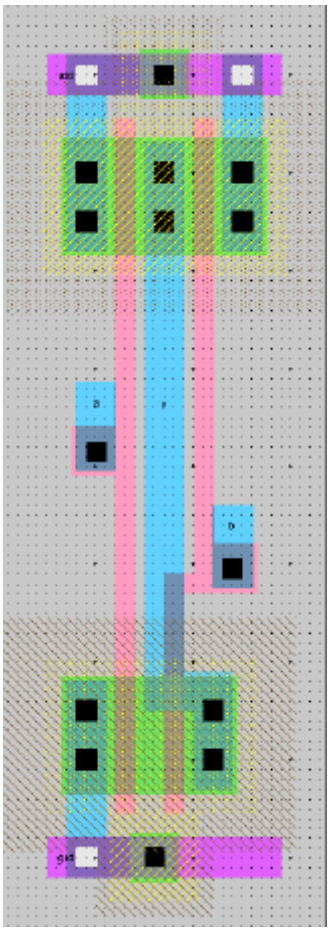

Figure 4: nand2{lay}

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. Study the inverter until you understand what each piece represents.

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.

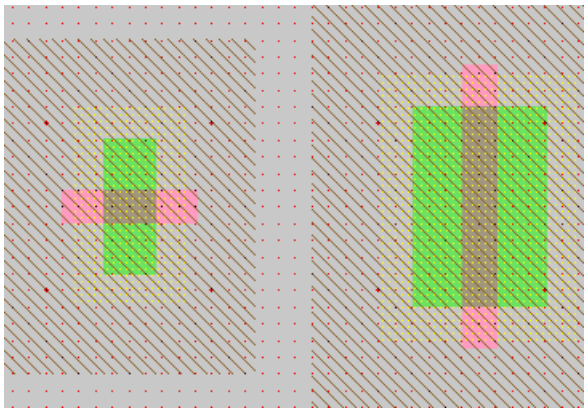

Figure 5: NMOS transistor before and after rotation and sizing

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. You may occasionally need to snap to a 0.5-λ grid instead; if so, use the Windows • Alignment Options to change the alignment of the cursor to 0.5 rather than 1.

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. 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

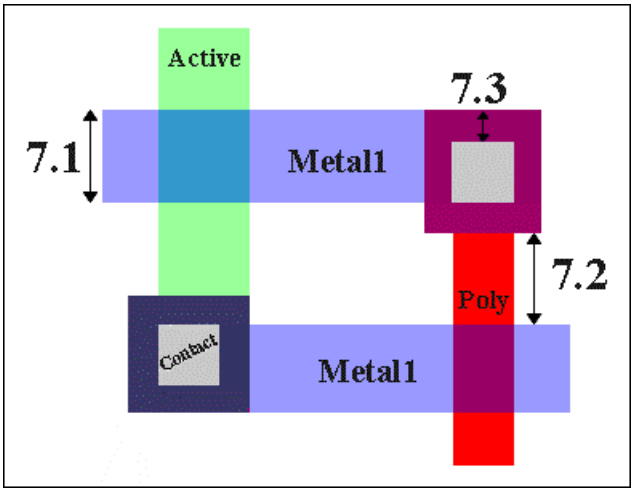

Figure 6: SCMOS Metal1 Rules

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, create contacts from the P-diffusion to metal1. At this point, your layout should look something like Figure 7:

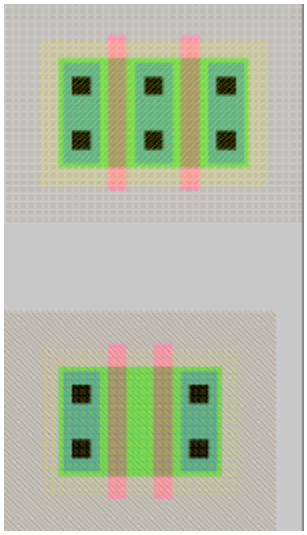

Figure 7: Contacted transistors for nand2 layout

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. Use the grid to ensure they are 60λ apart from the center of each line. This is the same spacing as the power/ground lines of the inverter. Place metal-1-metal-1-contacts, also known as vias, on the power lines. In Electric, you must explicitly draw a line from the via to the metal2 to form a connection. Otherwise you will be confused when you appear to have the via sitting on the metal2 line but discover in simulation there is no connection. Therefore, it is often easiest to place the via some distance away from the metal2, draw a wire to connect, then drag the via to its desired location over the metal2. Add more metal lines to connect power and ground to the transistors.

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. Place the P-well contact under the ground rail and connect it to the ground via with metal1. Place the N-well contact under the power rail and connect it to VDD with metal1.

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.

If you would like to make your layout somewhat cleaner, double-click on the metal lines and change their width to 4. This is a particularly good idea for the power and ground lines because it provides more metal area to carry supply current.

Finally, define exports for 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. Also export *vdd* and *gnd* from the metal 2 lines; these should be of type power and ground, respectively.

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 • Network Options to F2, and Tools • Network • Network Consistency Check 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 Substrate and Wells. This check ensures that every nwell has a contact to VDD and every psubstrate/pwell 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 Flatten Hierarchy. Check the Use Port Names, Check Port Order, 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 Consistency Check. If your design is correct, you should get a message of:

```
*** Comparing facet nand2{lay} (4 components, 4 nets) with facet nand2{sch} (4 components, 4 nets)
Facets are equivalent
```

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, NCC will likely mark too much mismatching to be useful. You are best off carefully examining your design by hand to look for errors. Remember that the order of inputs is important; if A and B are interchanged, you will get an error.

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.

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 se the inverter schematic and icon provided. You will need to create an icon for your 2-input NAND gate.

Open your nand2{sch} and choose View • Make Icon. Electric will create a generic icon based on the exports looking something like Figure 8.

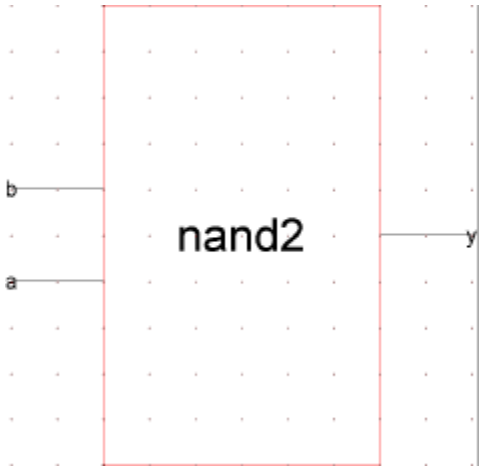

Figure 8: nand2{ic} from Make Icon

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.

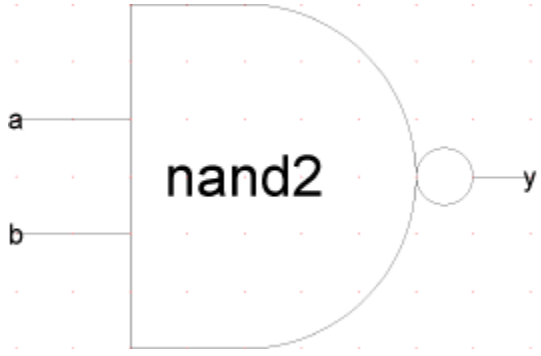

Figure 9: nand2{ic} final version

Start by changing the technology to *artwork*. A palette will appear with various shapes. Delete the generic box but leave the input and output lines. 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.

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.

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 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. 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. Try the Tools • Simulation • Down Hierarchy command to descend into each of the gates and observe their internal waveforms. The gates are given generic names such as NODE4 and NODE5 when created, so it can be

hard to determine which is the nand and which is the inverter. 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 mocmossub. Instantiate the nand2{lay} and inv{lay} layouts. 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. 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.

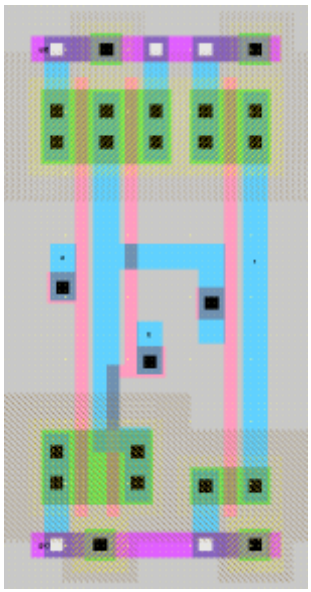

Figure 12: and2{lay}

Finally, do a network compare. NCC has three modes in the Network Options: Check Current Facet Only, Flatten Hierarchy, and Recursively Check Subfacets. Check Current Facet Only assumes all subfacets are perfect. Don't trust this mode. Flatten Hierarchy smashes your entire design into one giant netlist for its internal comparison. Recursively Check Subfacets checks each subfacet. NCC is a relatively new feature that has received many bug fixes, so it is wise to try both Flatten Hierarchy and Recursively Check Subfacets to catch any problems that one mode misses. In each case, be sure to check Use Port Names, Check Port Order, 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.

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.