1 CAD Tool Installation

This lab assumes use of the Electric CAD system (v6.06), which is a free package distributed under the GNU public license. The package is available for a number of operating systems and is freely distributed in source code format from http://www.staticfreesof.com.

If you choose to use your own toolset, make sure that it is capable of performing all the following tasks. Otherwise, you may find yourself unable to complete the project requirements. The directions are divided into a CAD tool independent background and Electric-specific parts.

2 CAD Tool Independent Background

This lab deals with schematics for integrated circuit designs. A schematic is defined as a document that has a list of elements and connectivity information. Most designers are familiar with the graphical view of this data and can recognize transistor-level schematics that use common symbols for NMOS, PMOS, VDD, GND, and capacitors. Schematics abstract away the actual geometry of the layout, in favor of a more simplistic connectivity model. A schematic can be used to generate a netlist, which is typically a text file containing this connectivity information. Note that many CAD tools may use the term schematic and netlist interchangeably.

Schematics and netlists can be flat or hierarchical. A flat schematic consists of only circuit primitives like transistors, power rails, capacitors, etc. If a schematic contains another schematic, it is hierarchical. One can think of a hierarchical design as a tree with the root being the top level schematic and the leaves being flat schematics. Hierarchical schematics are easier to understand and often improve the speed of CAD tools. Signals (wires) can traverse levels of the hierarchy by placing ports on the schematic page. Most graphical CAD tools have an automatic method to create a symbol view with pins for the schematic ports. This symbol view can then be instantiated (possibly multiple times) in other schematics.

A concrete example of a simple hierarchical design is a buffer composed of two inverters. To make this schematic, the inverter schematic will be created first. It will then be instantiated twice to make the buffer.

Once the schematic is created, it can be simulated for functional correctness. Functional correctness means that the logic is correct, assuming ideal timing (delay) conditions. At this stage of the design, the models for the circuit primitives (particularly the transistors) are accurate, but the wiring between them is assumed ideal. This means that when the actual layout is made, the parasitic resistance and capacitance of the interconnect (often referred to as simply parasitics) have the effect of slowing the circuit down. This speed reduction is always significant, but can be critically long for a design with long wires, busses, and/or high fanout. Lab 2 will address this issue in greater detail after the layout is created.

For speedy simulation, a simple switch model can be used to replace the transistors in a digital circuit to test the function of the logic. For estimated timing delays, a more accurate simulator like HSPICE can be used. After parasitics are added from the layout, HSPICE will be required to get reasonably accurate delay estimates.
3 Getting Started with Electric

Every CAD tool vendor has an organizational structure for its databases of design data. In Electric, a particular circuit is called a cell. Think of a cell as all the design information (including connectivity information) for a particular circuit. Each cell has many possible facets, which are simply different ways of looking at the cell. For example, the schematic facet and the layout facet for a cell are simply two different ways of looking at the same circuit. There are many types of facets, but this lab will focus on the schematic type. Lastly, a library is simply a collection of related cells. For this lab, you will create:

- **Library:** labs
- **Cell:** inverter
- **Facet:** schematic
- **Cell:** buffer
- **Facet:** schematic

When Electric is started, there are three panes: a large empty facet window in the top right, a message window at the bottom, and a component window. Multiple facet windows can be open simultaneously and are the main workspace for the designs. The message window is where information, warnings and error messages appear. The component window shows a selection of symbols for the current technology.

Electric computes distances in terms of ‘lambda’, which is defined as half the minimum feature size in half-millimicrons (for a 0.25µm process, we set lambda to 250 which maps to 0.125µm). Once lambda is set correctly, you should enter distances in terms of lambda (e.g. setting a transistor width to W=2 means 2*lambda or 0.25µ). To set the lambda values for the current library:

- Select menu: Technology|Change Units…
- In the listbox, choose: mocmossub
- Set ‘Lambda size’: 250
- Press: Ok

To set the correct technology file for your schematics:

- Select menu: Technology|Technology Options…
- Set ‘Schematics|Use lambda values from this technology’: mocmossub
- Press: Ok

Lastly, we need to load the familiar schematic symbols into the component window:

- Select menu: Technology|Change Current Technology…
- Choose: schematic, analog
- Press: Ok

The component window changes to include symbols for transistors, power rails, etc.

4 Create an Inverter Schematic

When Electric starts, the current library is called ‘noname’ and is empty. Cells are automatically created by creating a facet for that cell:

- Select menu: Facets: Edit Facet…
- Press: New Facet
- Enter the name: inverter
- Choose view: schematic
- Press: Ok
Note that the name of the facet window changes to indicate the library, cell, and view for the facet being displayed.

Figure 1 shows the components for a simple static CMOS inverter. Drop these components into the facet window by first clicking on a component in the component window and then clicking in the facet window. When you drop in the ports, make sure to specify that ‘In’ is an input port and ‘Out’ is an output port. Arrange your components as shown in Figure 1.

Now connect the wires by first selecting the power port (ring symbol) using the left mouse button. Connection points are denoted by small crosshairs (+). Click and drag the right mouse button from the connection point on the power port to the source of the PMOS transistor. In the same way, connect the remaining wires to make a static CMOS inverter.

Now we need to configure some of the components. As explained above, distances are entered in ‘lambda’ units. The minimum size transistor allowed by the layout technology library is W=3 (0.625 µm) and L=2 (0.25 µm), so set the W/L for the NMOS to 3/2. Open the properties for the NMOS by left double-clicking on the NMOS. For approximately symmetrical switching, we should make the PMOS width twice that of the NMOS. Change its W/L ratio to 6/2 and hit Ok.

Signals that are required to use this gate in another schematic must be labeled as exports. The ‘In’ and ‘Out’ off-page ports are automatically considered exports. Unfortunately, the power and ground symbols are not automatically exported and must be manually labeled. The easiest way to do this is select the power symbol, choose Export|Create Export…, enter the name ‘vdd’, choose the ‘power’ type, and hit Ok. Do the same for the ground symbol, naming it ‘gnd’ and selecting the ‘ground’ type. The exports are what make the power and ground nets special. The symbols are purely ornamental. Once there is an export labeled ‘vdd’, you can just double-click on any wire you want to connect to power and just name its network ‘vdd’. In schematics, any arcs (wires) that have the same network name are considered connected. The use of exports will be a little clearer later.

At this point, you should have a full schematic for an inverter. Now would be a good time to save the library:

Select menu: File|Save Library As…
Select the folder you want
Enter filename: labs
5 Using the Internal Simulator

Electric has a simple built-in simulator that can be used to test functional correctness. Activate a simulator window by selecting Tools|Simulation (Built-In)|Simulate. A window with waveforms should appear. The two vertical lines that cross the waveforms are the main time and the extra time (the line has an X on the top). Select the input signal net (probably called INNV), and drag the main time line to 100ns. Press the number ‘1’ to indicate you want logic high there. If everything is correct, you should see the output switch to logic low about 10ns later. You can use the extra time line to measure the delta between the input change and the output change. Note that this time has no physical meaning, since actual delays are not modeled by this functional simulator.

If you get unexpected output, check your circuit carefully and resimulate until it works correctly.

6 Hierarchical Designs

In this step, you will create a buffer using two instances of the inverter you just created. First, an icon for the inverter schematic must be created. This icon can then be dropped into another schematic to use the inverter. Electric has the capability to generate related facets automatically. This feature can be used to create a symbol for the inverter. Make sure that the inverter schematic is the active facet, then select: View|Make Icon View. A simple block is dropped into the schematic that show the ports of the block. The block should contain all the exports you created, whether implicitly through off-page symbols or explicitly through the Create Export menu item. The block it creates should contain ‘In’, ‘Out’, ‘vdd’, and ‘gnd’. After verifying the ports are correct, simply select and delete the new block, since it’s actually contained in another facet view.

Open up the icon facet that was just created using View|Edit Icon View. The facet type in the title bar has changed to {ic} to indicate this is the icon view. Change the component window to a selection of geometric shapes by choosing Technology|Change Current Technology… and selecting artwork. Create a typical inverter symbol by first selecting and deleting the bounding box. Then place a triangle and a circle. The edit menu has rotate (Ctrl-J) and sizing (Ctrl-B) commands that operate on the selected shape. The editor has a snap-to-grid feature that sometimes helps alignment, but also sometimes makes it more complicated. The easiest way to override the grid is to double-click on the element and manually change the X and Y positions to the desired values.

The connection points to this icon are the exports. The export locations can be shown by selecting Export|Show Exports. You can move the exports around by selecting and then dragging their text labels. Make sure that you have the exact same exports as your schematic. Double-click the large black text ‘In’ to see the export properties. You can change the location of the text relative to the export point, select when the label should be drawn, and many other options. You can control whether the export names are printed using Export|Port and Export Display Options…. All this is aesthetic because, when instantiated, the connections will be made to the export location no matter where it is. It is, however, easier to read your top-level schematics if the icons are intuitive. When you are satisfied with your icon, make sure to save the library. As an example, my icon is shown in Figure 2.

![Figure 2: Icon facet for inverter cell](image-url)
First, create a new schematic facet called ‘buffer’ using the same technique as for the inverter schematic above. Note that if you check the ‘Make new window for this facet’ option, multiple facets can be open simultaneously. You may need to change the technology to get the analog schematic symbols to appear in the component window. Place two instances of the inverter by selecting the ‘Inst’ component and choosing the ‘inverter {ic}’ facet. Then place an input and an output port, making sure to set the type correctly. Pick one of the power pins, extend the wires, and export the wire as ‘vdd’. For the other power pin, extend the wire, double-click on it, and change its network to ‘vdd’. When you select the inverter symbol near the power pin, it should highlight the entire ‘vdd’ network. This is an easy way to verify that all your ‘vdd’s are connected correctly. Do the same for the ground network. Lastly, connect the wires to form a buffer out of the two inverters.

You should now be able to simulate the buffer in the same way as the inverter. Once the functional simulator shows the correct logic for a buffer, the next step is to simulate with real transistors instead of the simple switch model.

7 Exporting to HSPICE

Electric has the capability to export the current design to a SPICE deck. As an historical aside, the term ‘deck’ is used because the original SPICE used punch cards and a design consisted of a deck of these cards. Models and subdesigns in the SPICE file are called ‘cards’ for the same reason. The model card used for this class is called g25.mod. It contains complete parameters for PMOS and NMOS devices over several process corners (fast, slow, and typical).

A SPICE simulation has more than just your circuit. Essentially, you must build a testbench around your design to provide the correct inputs, power supply, and measure the outputs. Although Electric has the capability to graphically add SPICE elements, it is far easier to make the testbench by hand in a text editor. The testbench for the buffer is shown in Figure 3. The purpose of each line is described in detail below:

1. Name of the SPICE deck (always include a name in your decks)
2. Example of parameterization. The term ‘vddval’ can be used anywhere else in the deck and it will be replaced by the value ‘2.5’
3. Library specification. This uses the Typical-nmos/Typical-pmos (TT) process corner.
4. Comment
5. Power rail connected between node ‘vdd’ and ground. Be sure to use the same node name as the exported power port.
6. Input stimulus is a Piece-Wise-Linear (PWL) wave that has a voltage of 0V at 0ns, 0V at 3ns, and ‘vddval’ V at 3.2ns.
7. Comment
8. Simulation options to use binary output files and only output signals that are explicitly probed (this speeds execution and reduces disk space requirements)
9. Transient analysis command with a print step of 0.2ns and a simulation time of 10ns. The print step is used change the amount of data saved for the waveform viewer.
10. Comment
11. Probe statements tell what values to output in the data files. In this case, it will output the voltage at nodes in and out and the current through the vdd voltage source.
12. Include statement. This includes the netlist produced by Electric.
13. End statement is required at the end of the file.

Type this testbench or copy it from ‘~msheets/ee141/lab1_testbench.sp’. Inside Electric, choose the Tools|Simulation (SPICE)|SPICE Options… menu item. Pick HSPICE, level 3. Hit Ok to set this change. Now choose Tools|Simulation (SPICE)|Write SPICE deck and write to file ‘buffer.spi’. Be sure to examine the generated SPICE deck in a text editor to make sure you understand the contents. The hierarchy of the design is clear in the SPICE deck because the buffer is formed using a subcircuit card (type X) to make two instances of the inverter.

Now FTP this file to ‘cory.eecs.berkeley.edu’ and log in to your account using SSH. Make sure the testbench and netlist files are in the same directory and run HSPICE on the testbench:

```
hspice lab1_testbench.sp > lab1_testbench.lis
```

You can then use ‘awaves’ to view the results. Verify that your circuit works by graphing the waveforms for ‘in’ and ‘out’. If there is a problem, correct it in Electric, rewrite the SPICE deck, FTP the new netlist to cory, and rerun HSPICE.