The Design and Simulation of an Inverter

Mentor Graphics Tutorial

This tutorial has been devised to run through all the steps involved in the design and simulation of a CMOS inverter using the Mentor Graphics CAD tools. Before invoking the Mentor Graphics tools it is essential to first set up the environment under which these tools will be used throughout the semester.

Setup

For the remainder of the course you will be working in your EECS 427 class directory which will store all the files that you create. Since Mentor files occupy a large amount of space it is essential that you work in your class directory otherwise your home directory quota will be quickly used up.

To avoid typing long pathnames, a link called *eecs427* is automatically created in your home directory when your class directory is set up. This *eecs427* link points to your class directory and hence, you can change to your class directory from your home directory simply by typing:

% cd eecs427

Note:

- The % symbol indicates the UNIX prompt of the c-shell and is usually preceded by the name of the machine you are working on.
- LMB, MMB, and RMB refer to the left, middle, and right mouse buttons. The LMB is always used for clicking on, selecting, and positioning components, shapes within the various Mentor programs, and for OK/ NO/Cancel operations in user interface items like dialog boxes, prompt bars, etc. The RMB is pressed to bring up the various pop up menus.

To get help at any time, click the menu

Help > Open Bookcase

in Mentor applications. The on-line documents use Acroread's search capability which often has problems when an index resides in afs. If you get bus error or segmentation fault pop-up windows when acroread opens, just close those windows. Also, you may have problems exiting acroread when this happens. If so, you can get rid of the acroread process by finding its process id with "ps -efl | grep acro" and killing it with "kill -9 <proc_id>".

First, select the proper version of the Mentor Graphics package. Do this with

% swselect &

If you see *Mentor Graphics* on the right side of the window, click on it and select *Unselect Package*. Then, regardless of wether the *Unselect Package* step was necessary, click on *Mentor Graphics* in the right half of the window and then choose *Select Package* and select version *D.1* in the pop-up window and select *Okay*. Then exit from *swselect*. To update the changes, type:

% source ~/.software

Next, open your *.cshrc* file, which gets read every time you log into a machine. Do this using the text editor of your choice (nedit, emacs, vi, etc.)

% nedit ~/.cshrc

Next, add the lines

```
setenv AMPLE_PATH ~/mgc/ample
setenv TSMC25 [enter your directory]
set path=($TSMC25/bin $path)
```

to the bottom of that file. The first line creates a link to the mgc file. The second line creates a link so you can use the $TSMC25 variable as a pointer to [directory]. The next step is

to update your session with the changes by typing

% source ~/.cshrc

The last step is to copy the setup file for the Mentor tools into your home directory (not your class space). Do this by typing:

% cp -r $TSMC25/mgc ~

If this command doesn't work (i.e. there is no *mgc* directory in your home space), then you must go back and make sure the edits to your *.cshrc* are correct. The files in the *mgc* directory sets up the checkpoint facility of *IC Station* . This periodically (every 15 minutes in our case) saves your design in a temporary format which can be restored if your design is corrupted in a system crash. To restore a checkpoint if the system crashes, use the *$salvage_cell()* command. Also for convenience the process file gets loaded, the grid set and a palette with the needed layers displayed. Now, we are ready to begin the design. Making sure you are in your class space, make a *cad1* directory to store your work. We are naming it cad1 because the inverter we will work on here is part of your first cad assignment.

% mkdir cad1
% cd cad1

Overview of Full-custom Design Flow

The following steps are involved in the design and simulation of a CMOS inverter.

- We need to capture the *schematic* i.e. the circuit representation of the inverter. This is done using the Mentor tool *Design Architect*. The last page of this document shows the schematic.
- The inverter may be needed in higher schematics and hence it needs to be represented by a *symbol*. This too is done using *Design Architect*.
- Verify correct logic functionality using Modelsim.
- Analog simulation is then performed using Eldo to further guarantee correct operation prior to beginning full-custom layout. This step isn't always necessary in the design flow.
- After verifying the functionality of the design, we need to physically *lay out* the inverter according to some CMOS process rules. In our case we will be using the TSMC.25 micron N-well CMOS process with MOSIS SCMOS SUBM design rules. Layout is done using the I*C Station* layout editor. The last page also shows the layout. (The key to the shadings is not part of the layout). Note that dimensions in IC Station for this process are in lambda, not microns.
- Check the layout to verify that it conforms to the process rules. This is done by *ICrules*, which is accessible from within IC Station.
- To prepare for layout verification, we need to generate a *design viewpoint* of the schematic. A design viewpoint can filter/modify schematic data for use in downstream tools. This is created with the *lvsvpt* command.
- Once the design rules have been checked, we need to confirm that our layout is a true representation of the correctly verified schematic. *ICtrace* is a layout versus schematic checker that does this and is also available from within IC Station.
- After the LVS (layout versus schematic) check returns a correct result, we need to extract layout parameters such as capacitances which result in circuit delays. This is done using *ICextract*. The resulting capacitance information along with external loading are the inputs required for analog simulation.
- Analog simulation is performed once again to determine circuit delays. These delays are then added to the schematic at the appropriate node as rise and fall delays.
- Finally, after adding accurate delays to the schematic, the circuit is re-simulated in Modelsim to check that the observed delays are correctly modeled in Modelsim.

Schematic and Symbol

Invoke Design Architect by typing the following. Make sure that you are in your cad1 directory.

% da_ic &

A window with the title *Design Architect* will open shortly. To expand this window to its maximum size, in the CDE windows manager click the LMB on the square button at the extreme top right corner of the DA window. In the *Session Palette* on the right part of the window, click the LMB on the *Schematic* button under the *Open* menu. A pop up dialog box will appear, prompting you for the name of the component. Use the mouse to highlight the component box (a red border should appear around the box) and enter the name of the component, say *inverter*. **Don't** delete the path name in the component name box, and don't change the sheet name. So the path should be *uniquename/cad1/inverter*. Then click the LMB on the OK button. A schematic window will appear and the *Session Palette* has changed to *schematic_edit*, you are now ready to edit a schematic. From the *schematic_edit* menu, click on LIBRARY. In the new menu that appears (*ic_library*), click on TSMC25 Lib. You will now see a list of components that you will be using to create schematics. These include:

nmos, pmos, gnd, portin, portout, vdd, delay, rnmos and rpmos

Click on pmos. A dialog box will appear at the bottom of the schematic window and if you move the mouse in the schematic window, a ghost image of the pmos transistor will appear. Position the transistor appropriately in the schematic window with the mouse and click the LMB. This causes the *instance* (in this case, the pmos transistor) to be fixed at one place. It will appear dashed as it is still selected. Press the F2 function key on the keyboard to *unselect* the transistor instance. Now, it should appear blue in color. This completes the *instantiation* of one transistor. The nmos transistor can be similarly instantiated by clicking on nmos in the TSMC25_lib. Add the vdd and gnd instances as well. The rnmos and rpmos parts are used in modeling weak feedback devices or pullups/pulldowns. You won't need them for this tutorial and their usage will be discussed in lecture. The directions of the little arrows on the transistor symbols are important. They show the direction of signal flow when the transistors are modeled in verilog. You can think of this as follows: for the p-channel, the signal flow is from vdd to the output node and for the n-channel, signal flow is from gnd to the output node. If a transistor is used as a pass transistor, signal flow is from input to output.

Our next goal is to complete the connectivity of the schematic. Click the LMB on the *Back* button to return to *ic_library* menu and then click the LMB on *Edit*. This will take you back to the *schematic_edit* menu. Now click the LMB on *Wire* under the ADD area (or use the F3 function key on your keyboard) in the Menu Palette. A dialog box appears at the bottom of the window with the corresponding command name. To draw the nets click the LMB at the desired starting point in the schematic window and move the mouse to the destination and double click the LMB at the end point. You can draw as many different nets as you need since the DRAW NET command stays in effect until it is cancelled by clicking the LMB on the CANCEL button in the dialog box at the bottom of the screen (or by pressing the Esc key). Complete the connectivity as shown in the schematic at the back.

As a time saving hint, you might want to learn how to use the middle mouse button strokes. These are quick ways in any Mentor tool (da, ic, etc) to perform common commands. These are done by holding down the MMB and drawing a ? on the screen. This will pop up a menu that contains the shortcuts to many commands. To perform the command, close the window and draw the line(s) while holding down the MMB. You should see the strokes appear on the screen in red. If you use these often enough, you won't need to bring up the window with all the strokes because you'll have them memorized. This can save you a lot of time during the semester, especially when drawing layouts.

Having done that, we need to add input and output ports to the inverter schematic. Choose a portin from the TSMC25_lib and place it on the schematic as you did previously with the other symbols. Similarly, add the portout symbol at the appropriate place.

This completes the drawing of the schematic. Now we need to alter the *properties* associated with the nets and instances. For example, you can see that both the input and output ports have the word NET written near them. This is the *value* of the property *net* associated with these wires (nets). The transistors have length and width properties of 2 and 5 lambda, respectively. All these are default values for the corresponding properties and need to be altered according to our

requirements. Say, we want to name the input port as *in*. To do this, place the cursor over the word NET near the input port. Then press Shift+F7. A dialog box appears at the bottom of the window prompting you for a new value for the property. Type *in* and click the LMB on the OK button. The input port should now be named *in*. Similarly, to change the output port name to *out*, Place the cursor over the word NET near the output port, Press Shift+F7 and then enter *out* at the dialog box prompt before clicking on the OK button. Leave the transistor widths and lengths at the default values.

Please see Appendix B for legal cell names and cell and net naming conventions. The last step in the schematic capture is the Design Check. In the menu bar on top of the DA window you will see an entry *File*. Click the LMB on this and then click on *Check Schematic*. In the future such a menu selection will be denoted in the following manner.

File > Check Schematic

It is assumed that the LMB is always pressed during this time. If all the above design steps have been correctly implemented, you should not get any warnings or errors in the report generated by the check. If you do get errors, look for debugging hints in the DA handout. You may get a warning such as "schematic not registered with an interface," but this will go away once you generate a symbol and save the design.

Note: when you have entered more complex schematics later in the term, it pays to save your schematic before doing the check (and even when it only partially complete) in case there is a crash before the check is completed. You will get a warning that you are saving without checking. You should save again after the check is complete. How to save is explained at the end of this section.

The above completes the schematic capture. From the File menu at the top of the window you can save the schematic by doing the following.

File > Save sheet > Default

Next, we need to make a symbol for the inverter. Select the following.

Miscellaneous > Generate Symbol...

Choose *Yes* under the *Replace Existing* area in the lower left corner and then click on the *OK* button in the dialog box. A symbol for the design appears. This contains the pins *in* and *out* and their types i.e. in for input and out for output pins. Once again the symbol needs to be checked. Select:

File > Check Symbol

The first time you check a new design you get a warning saying that pins (*in* and *out* in this case) are not present on the interface. Ignore it, since it will go away after you have saved the symbol. Close the report window and save the symbol by selecting:

File > Save symbol > Default

The above completes the creation of the schematic and its corresponding symbol. We have now completed the first two steps outlined in the overview. Now, close the symbol window by double-clicking the left mouse button on the top left button of the symbol window, but keep the schematic window open for the next section of the tutorial.

In future designs you can change the symbol shapes to your liking. If you do this, try not to delete any of the pins or the pin properties while editing the symbol. If you do and you're unable to recreate them correctly, delete the symbol using Design Manager (described later in this document) and regenerate the symbol.

Logic Simulation

Modelsim will be used for all logic simulation. Modelsim can simulate verilog or VHDL models or mixed designs with both verilog and VHDL. We will use only verilog in EECS 427. The first step in simulating the inverter schematic is to write out a verilog netlist using the verilog netlister within Design Architect. Click the LMB on the background of the DA window and you should be back to the *Session Palette*. Select:

File > Export Verilog

Get into the directory where your inverter design files reside under the *Design Path* box and click on

the *Yes* button for Options and designate your output file to one you can easily recognize (i.e. inverter.v). Now click OK. A *Netlisting* window will pop up and prompt you to hit the *Return* key. If everything works correctly you should see a note at the bottom of the DA window that looks similar to this:

Note: Netlist has been written to: ~/cad1/inverter/inverter.v

In your shell go to this directory and look at the inverter.v file with a text editor or dump its contents to the shell with:

% more inverter.v

You should see a module definition for inverter with a nmos and a pmos instantiation. If the module definition for the inverter (or whichever cell you are creating) contains only ports called dummy_pin, then you didn't create the symbol in DA. Please go back and do it and re-run the *netlister*. Below the inverter module definition, you should see additional module definitions for nmos and pmos. The functionality of nmos is represented by the verilog primitive *nmos.* inverter.v is an example of *structural* verilog. Structural verilog is comprised mainly of module definitions, instantiations and a description of how the modules are connected. Later in EECS 427 you will write *behavioral* verilog to describe a circuit's operation. Behavioral verilog looks a bit like C-language and describes a circuit's operation at a more abstract level than structural verilog.

You can create input stimulus for your design in Modelsim using force commands or by creating a verilog "testbench" module that instantiates the design and forces the inputs. The latter is preferable in many respects and so we won't cover the Modelsim force command. Copy the arbitrarily named (meaning that you can name your testfile whatever you want, which may come in handy when you are testing different parts) *testbench.v* from the TSMC25 space by doing the following command:

% cp $TSMC25/misc/testbench.v ./testbench.v

This is a simple verilog module that will stimulate the inverter, but you can use this as a template for creating more complex input stimuli for other designs. Notice the line where the inverter is instantiated. The first field contains the instantiated cell or modules name. The second field contains the instance name. The remaining entries describe how the pins of inverter (in and out) are connected in testbench. The "in" pin is connected to the testbench signal "invin" and the "out" pin is connected to the signal "invout". The order of the pins is not important here.

Another useful construct to understand is the *always* block. As mentioned in the comments in testbench.v, an initial block is executed once per simulation. The first line after the begin in this example is executed at time=0, then the next line schedules an event at time=10 and the following line schedules another event at time=20, etc. Once the end of the initial block is reached, the block is never called again until the simulation is restarted from scratch. In this case, the line before the end stops the simulation. An *always* block is executed continuously throughout the simulation. Since it continuously executes, you must incorporate time control in the block otherwise you can create an infinite loop. We'll talk a lot more about the *always* block later in EECS 427 but for now we're really only interested in using it to generate input stimulus with periodicity -namely, clocks. An example of an always block in your testbench.v is shown below:

```
always
begin
#50 clk = ~clk;
end
```

This always block generates a 50% duty cycle clock with a period of 100 time units. Also note the need for an initial block to set clk to 0 or 1 at the start of the simulation, this should be added before the always block. The following always block creates an infinite loop which will cause the simulation to stall at time 0:

```
always
begin
clk = ~clk;
end
```

Finally, look at how logic values have been specified in testbench.v. Logic 1 was written as 1'b1 and logic 0 as 1'b0. We could have written just 1 or 0 but expressing them this way gives insight into how you might drive, say, a 16-bit data bus with an hex-encoded opcode of 1AF4. Here's how you would do that:

```
module testbench();

reg [15:0] databus;

initial
begin
databus = 16'h1af4;
```

Another ways of driving databus are:

databus = 16'b0001101011110100;

databus = 16'd6900; The first field of the number declares the bit-width, followed by a "'", then by an identifier: h -hex, d -decimal, o -octal and b -binary and finally by the number in the given encoding. The hex and binary representations will be the most useful. Also note how the reg signal databus was declared 16-bits wide with the "[15:0]". **Please learn bus notation and use it often**. It will make your lives easier when you move on to bigger projects during the semester.

You now have a complete verilog model for your inverter along with a verilog module that instantiates and stimulates the inverter. At this point you're ready to bring up Modelsim to simulate your design. Enter:

% vsim &

and you should see two windows. The first is the main Modelsim window and the second is a "New Modelsim Features" window that highlights features of the new version of Modelsim in place. You can check the *Don't show this dialog again* box and close the second window if you wish. When first simulating a design, you'll follow these steps:

1) Create a project directory
2) Edit the project, adding verilog source files and creating a project build script
3) Compile the project (probably not necessary...)
4) Simulate

To create a project, select:

File > New > Project...

Decide where you want the project directory to reside with the project location box. Use *cad1/inverter_test* for this. Then enter a name (*inverter_test)* for the project in the "Project Name" field and click ok. In a shell, list the contents of the new directory. You should see a file named "<project>.mpf" and a sub-directory "work". Your verilog source files will eventually be compiled in the work directory and the .mpf file (Modelsim Project File) contains project-specific information. If you want to re-open this project, you will do so by browsing to the .mpf file and open it in vsim.

A new window *Add items to the Project* should have appeared after creating your project. Click the LMB on *Add Existing File* and provide the full UNIX path to the inverter.v file either by typing the path in or by using the "Browse" button.

Note: if you browse, don't try to browse the contents of /afs -it'll take a very long time and is difficult to interrupt. Instead, type a more specific path like "~/eecs427" in the "File name" field. The browser will recognize this as a directory and will simply browse it).

Once you've selected the correct file, click on "Open." Leave the *Reference from current location* selected.Then click on "OK" in the "Add File to Project" pop-up window. Do the same procedure for testbench.v. *The only time you should ever edit the project again is if you need to add another file to the project!* If you only want to make a change to file (like changing the forces in your testbench or re-running vnet on a new schematic), just re-compile the project (see next paragraph). You'll also want to be sure to add files to the project from the lowest level up (i.e. test-bench last). If you don't,

you can always fix this with the "Compile -> Compile Order" and choosing the test-bench to be compiled last.

At this point, we're ready to compile the project. Now select "Compile -> Compile All" from the main window. This will execute the project build script which will compile each source file that you've added to the project.

Next we need to load the design for simulation. Modelsim just needs to know which module will be considered the top-level design. In this case, it's testbench module. Select "Simulate." in the main window. In the "Simulate" pop-up window you should see a list of designs and your work directory. Click the "+" next to the work directory to expand it. You should now see the modules in your design: globals, inverter, and test-bench. Click on testbench then on "OK." You'll see additional messages in the main window. Note how Modelsim loads the necessary sub-modules found as it traverses the design hierarchy. You would get an error if you tried to load testbench when one of the sub-modules was missing from the project "work" library.

Now set up the simulation windows. To see all the available windows, select "View->All" in the main window. A lot of windows will open up but for our purposes we're interested only in the structure, signals and wave windows. Close the rest with "File->Close" in each window. In the future you can individually select which windows to open in the "View" menu of the main window. The structure window shows the hierarchical structure of the design. You should see testbench at the top of the structure and an instance of inverter called i1. Next, click (LMB) on "i1" and watch how the signals window changes. Click on "testbench" in the structure window and again watch the signals window. Signals names are displayed for each selected instance in the hierarchy. Next, click and drag (hold down the LMB) one of the signal names from signals window to leftmost gray area in the waves window and release the LMB. The wave window will display waveforms of the signals you specify once the simulation is run. Here's another way to add signals: in the signals window, select "Add->Wave->Signals In Design." All signals at all levels of hierarchy in the design are added in the wave window. Now try "View->Wave->Signals in Region," and only the signals at the current level of hierarchy are added in the wave window. For our purposes we're really just interested in the two signals "invin" and "invout" at the testbench level. To get rid of the others and the extra copies of "invin" and "invout" in the wave window, click (LMB) on the top-most signal in the wave window then, while holding the shift key, click (LMB) on the signal that is third from the bottom in the wave window. The selected signals should be highlighted white. Delete them by clicking on the scissors button in the wave window.

You should be left with "invin" and "invout" in the wave window. You're now ready to run the simulation and observe the response of your design. Try running the simulation each of the following ways (restart the simulator after observing the results by selecting "Run->Restart..." in the main window, then click "Restart" on the pop-up window):

1) In the main window, select "Run->Run - All"

2) In the main window, enter "run -all" on the command line

3) In the main window, enter "run 30" on the command line, look at the wave window, then enter "run -all"

Select "Zoom->Zoom Full" in the wave window to fill the window with the signal traces. The first two are equivalent in that the simulator runs until there are no more scheduled events. In this case, the $stop directive breaks the simulation at 60 time units (ns by default). The source window pops up to show which line in the verilog source caused the break. In the third case, the simulation is run for 30ns at which point you could observe signals, etc. Then the simulation is run to completion by entering "run -all".

Here are some miscellaneous features you may find useful in debugging your designs. Notice how commands are echoed back to the main window's transcript. This means you could type the commands rather than manipulate the windows.

In the wave window, try clicking (LMB) on a signal that has at least on transition on it, then click (again, LMB) in the wave window. You should see a time cursor appear and to the right of the signal in the signal pane you should see the signal values at that time in the simulation. Now click on the right (or left) arrow button in the wave window tool bar. The cursor should move to the next (or

previous) transition on the selected signal. This feature can be useful in large simulations.

Let's say you're in the middle of a complicated debug session and you've probed a 100 different signals in 20 different modules but you need to log out and go to class. You can save your environment in many ways but the most basic way is to save the wave window format. Do this by selecting "File->Save Format" in the wave window. Then when you return you can load your design, select "File->Load Format" in the wave window and have all signals probed and ordered in the manner you like.

For small designs, you can easily get away with displaying all the signals in the design. However, as your design grows you may run into two problems if you continue to display all signals in the design. One is that you'll have too much information on the screen and the other is that when you specify that a signal is to be displayed, its values are logged in a file on disk, such that a large design running for a long time may create very large files that start consuming too much disk space. You can control this by choosing which signals to log prior to running your simulation by using the "Add->Log->" menu in the signals window. By doing this you can log whatever signals you think you might want to display at some point. This signals aren't displayed until you do a "Add->Wave->" or drag them to the wave window. If a signal node has never been displayed or logged and you find that you need to view it, you'll need to restart the simulation since its values have not been stored. You can observe this by exiting Modelsim, opening your testbench design and viewing only the "invin" signal in testbench. Run the simulation to conclusion then try to display all the signals in the design.

The usefulness of many of Modelsim's features will become apparent as the size of your design grows. The various windows and the features of those windows will be helpful later on when you're simulating large portions of the microcontroller project. To learn more about Modelsim, open up the on-line documentation by selecting "Help->SE/EE Documentation->SE/EE Documentation Index" in the main window.

To print modelsim waveforms, select "File -> Print Postcript" in the Waveform window. Select "File" for the printer and enter the filename you wish to output. Note that by default it puts the file in your modelsim project directory. You can change this using the "Browse" button.

Analog Simulation

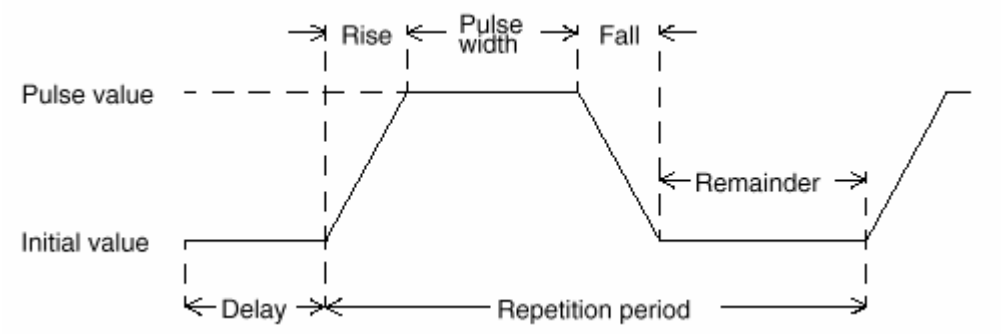

Figure 1. Waveform generated by Force Pulse command (from "Analog Simulators Reference Manual")

The analog simulator can be run within DA, and is called *Eldo.* If you closed your DA window, reopen the program by entering:

% da_ic

Open the schematic sheet for the inverter you drew earlier. You'll be running a transient analysis on the inverter and you'll need to provide input stimulus. To do this, click the LMB on *Simulation* in

the *schematic_edit* palette. Click OK in the pop-up menu that appears. You should now be in the Simulation mode and you will notice several options in the *schematic_sim_palette*. We first need to add forces for our simulation. Do this by clicking the LMB on the net you want to add a force to. In this case we would like to add a force to the input so click on the *in* port you created and you will notice that the wire connecting the port to the pmos and nmos transistors has been selected. Now click the LMB on *Forces* in the *Setup* area, and select *Add Forces*.

At the top of the *Add Forces* pop-up window you will notice entries for Name, Signal, and Reference Net. The Signal net is the name of the pin to insert the force on. The Reference Net is the name of the net or pin to appy the source to. The *Forcing:* section below allows you to toggle between forcing a voltage or current source, you can also select the *Source Type*. We want to force a voltage pulse trace in this case, so select *Voltage* in the *Forcing:* section and PULSE in the *Source Type* section. Fill in the trace data as follows:

Initial Voltage (V/A) = 0

Pulsed Voltage(V/A) = 2.5V

Delay(s) = 0N

Rise Time(s) = .5N

Fall Time(s) = .5N

Pulse Width(s) = 20N

Period(s) = 40N

Analog signals cannot change instantaneously, so you must specify the start and end of rising and falling signals. Usually, you can keep the rise and fall times of all input signals to be 0.5ns. Figure 1 shows the waveform generated by the force as it is now. Note that the N stands for nanoseconds in all the time units. You can also use M for milliseconds and others if you wish. Also note that you can also choose DC, or PWL for your source type. These options may come in hand later for different simulations. The DC source type forces a constant source value on the node throughout the simulation. The PWL, or piece-wise linear source type allows you to enter time/value pairs for the simulation.

Now that we have added forces, we need to choose which signals to probe. We're interested in seeing the effects on the input and output for this tutorial, so select both the *in* and *out* ports of your inverter by clicking them with the LMB. Now select:

Probes/Plots > Probe

in the *Setup* area of the *schematic_sim_palette*. In the pop-up window make sure you are set to Probe *Selected* nets, and click OK.

At times it may be necessary to change length of the transient analysis and the time step of it. To do this click the LMB on *Analyses* in the *Setup* area of the *schematic_sim_palette*. Now click on the *Setup* box next to *Transient* (Transient should be selected as default). In the *Setup Transient Analysis* pop-up window you will notice several options including *Stop Time* and *Max Time Step*. Obviously, a smaller time step will make for a more time consuming and more accurate simulation. We recommend the time step given as default. The length of the analysis is dependent on how long you need to simulate to see all the input combinations desired. You can click the LMB on *Cancel*, because we do not wish to change any of these values for this particular tutorial.

As you designs get larger in size, it may become tedious to select forces manually. In this case, you can edit the force file (sim.forces) generated by the simulation tool. Pay close attention to the format of the signals when doing this, to make sure you are generating signals in the way you desire.

Now we are ready to run the simulation. Click on *Run Eldo* in the *Execute* section of the schematic_sim_palette. Two windows should appear, one for netlisting, and another for running Eldo. They're just transcripts of activity and you can close them by hitting the Return key. Once the simulation is complete, you can view the waveforms by selecting *View Waves* in the *Results* area of the schematic_sim_palette.

The EZwave workspace should now open, and you will notice your design in the Waveform List on

the left side of the window. Click the "+" to expand the TRAN section and you can now select the waves for the signals you probed. With the LMB click and drag the V(IN) signal to the blue section of the EZwave window. You should now see a plot of the V(IN) signal. Now with the LMB also click and drag the V(OUT) signal to the wave window. (Place the signal outside of the graph space if you do not want the waves do not overlap). You can now Zoom in on the waves by selecting:

View > Zoom In X

Now we need to measure the delay time and we do this with Cursors on the chart. In order to add a cursor, RMB click inside the Chart window, then select *Add Cursor*. To move cursors around, LMB click to highlight and then drag the cursor where you want it. The cursor will display the time and Voltage levels of the signals where it crosses them. To measure delay, place two cursors at the %50 points of the signal and see what the time difference is. Cursors can be used for many other measurements like 90% to 10% rise and fall times also.

Layout

At this point you should be confident that the schematic you've entered correctly performs the function you desire. If it does, you are ready to perform mask layout of the design. If you don't adequately verify the functionality with Modelsim and/or Eldo prior to doing layout, you can waste a lot of time reworking the layout later when you discover a bug. We'll use the IC Station layout editor to enter the mask shapes. IC Station also includes a DRC checker, LVS checker and a layout parameter extractor (LPE). In this section we will concentrate specifically on the layout of the inverter. DRC, LVS and LPE will be covered in the following sections.

To invoke IC Station enter the following.

% ic &

A window titled *ICstation* will open up shortly thereafter. Use the same technique as explained for Design Architect to expand this window to its maximum size. On the right hand side of the window is a menu palette called *Session* (you may have to move the Layers menu to see it fully). Click the LMB on *Create* in the *Cell* area. A pop up box appears prompting you for the name of the cell to be created. Type in the cell name *inverter*. The IC Station Setup files you copied on page 1 take care of the default process and rules files, so you don't have to enter any more information in this window. Click on the OK button. Next, you should see a window in which the layout can be edited. The cell name is included in the title bar of this window. Towards the top right of the layout window, you should see X and Y coordinates of the cursor in the layout window being continuously tracked. The distance between two adjacent grid points will be more than 1.0 lambda at this point. This is because your layout window has been *zoomed out* to that extent. To get to the correct grid in order to start editing the layout, you need to zoom *in* to the window. From the *View* menu on the top menu bar, select:

View > Zoom > In (or use the following keyboard shortcuts: "+" -zoom in; "-" -zoom out; "/" -pan left; "" -pan right -be sure to use the numeric keypad versions of these keys for this. Also, Shift-F8 is view all. And don't forget the MMB strokes as shortcuts)*

Repeat this command till you see a second grid with smaller (1.0 lambda) separation between adjacent grid points. Zoom in until you feel you can conveniently lay out the features of the inverter. From the design rules and feature sizes, we can estimate the size of the inverter to be around 25 lambda wide by 45 lambda tall.

An example inverter layout is included at the end of this tutorial. A color version is available at: http:// www.eecs.umich.edu/courses/eecs427/inverter_layout.html.

• Drawing a layout

You are now ready to start adding *shapes* to your inverter layout. From the IC Palettes menu click on *Easy Edit*. This causes the IC Palette to be replaced by an edit menu. Click on *Shape* in this menu. A pop up dialog box appears on the bottom of the layout window. It indicates the current command that has been invoked. On the layer palette in the upper right hand corner of the IC Station window, click on the METAL1 layer with the LMB. This selects METAL1 as the current layer for the next shape to be drawn. In the layout edit window you now see that the cursor has change shape to a *cross*. This means that you can now execute a command. Keeping the LMB pressed,

drag the mouse till you can draw a vdd line as shown in the layout attached to this tutorial. This polygon shape is 5λby 25λ. Release the LMB after you have satisfactorily drawn the shape. You should see a blue highlighted rectangle. To *unselect* this shape press the F2 key. To reselect this shape place the cursor on the shape and press the F1 key. In general, F1 selects the nearest unselected shape whereas F2 unselects all selected objects. To clearly identify a selected shape it is always highlighted. For other function key mappings, look just below the edit window in IC Station. This is the Softkeys and can be interpreted as follows, using the F2 key for example. Pressing F2 by itself while the edit window is active executes "Unselect All." If you hold down the Shift key while pressing F2, you can drag a box with the mouse to Unselect Area contained by the box. Finally, if you hold down Control while pressing F2, you can Move any selected objects in the edit window.

You can also select shapes by clicking on their edges, and by clicking and dragging the mouse to define a rectangle. All shapes intersected by the rectangle will be selected. A useful example of using the *Strokes* technique is the method of selecting just one edge of a rectangle, since there is no easy way to do this otherwise. You need this if you want to change the width or height of a rectangle, without deleting it and redrawing. To do this place the cursor close to the edge of an unselected object you wish to select, press the middle-mouse button (MMB) and, keeping the MMB pressed, drag the cursor a short distance *horizontally* (even if selecting a vertical edge) to the right. For more information on Strokes, select Help->On Strokes->Quick Help... or draw a question mark on the screen with the MMB depressed and the help menu will come up also.

You will notice now that the pop up dialog box for the *add shape* command is still present at the bottom of the layout edit window. This is because the *Repeat On* function is in effect. This causes a command to remain until it is cancelled by clicking on the CANCEL button, or by pressing the Esc key. The repeat function is indicated at the bottom of the Easy Edit menu and you can toggle the setup to *Repeat Off* by simply clicking on the Repeat On bar. It is a matter of personal choice as to which setup to use. The documentation will assume *Repeat On* since that is the default mode. The repeat mode is indicated by an asterisk at the end of each command in the Easy Edit menu.

It is best to get acquainted with the editing features of IC Station at this juncture. The most commonly required edit commands are *Copy*, *Move*, *Delete* and *Notch*. All of these are present in the Easy Edit menu. All of them operate on *selected* shapes i.e. you must first select the shape to be edited by placing the cursor over it and pressing the F1 key. Choose such an area to place the cursor that is not overlapped by other layers else you might not be able to select the layer of your choice. In case you select the wrong layer; press F2 and then reselect the correct layer. Let’s start with the simplest -Delete. If you erroneously enter an incorrect shape, select it and then click the LMB on *Delete* in the Easy Edit menu. Click on OK in the dialog box that appears at the bottom of the window. This causes the selected shape to be deleted. Copy -select the shape to be copied. Then click on *Copy* in the Easy Edit menu. Click on OK in the dialog box and you should then see a ghost image of the selected shape which is dragged about with the mouse. Place it at the correct location and click the LMB to fix the position of the copy. You will still see the ghost image and so if you wish you can copy it to yet another place. When you are done copying press the F2 key and all copied shapes will appear in their new locations. Move -is similar except that the original shape does not stay in place. Place F2 at the end of the move to fix the location of the shape. Notch -is used to edit shapes that are in the right place but are of the wrong shape or size. Select a shape using F1. Assume that the shape is smaller along one dimension than it needs to be. Click on *Notch* in the Easy Edit menu. Click OK in the dialog box. Now, position the cursor on the boundary of the selected shape and keeping the LMB pressed, drag the cursor to form a box that represents say the additional size that the shape needed to be. Release the mouse button and you should see that the shape has now increased to the desired size. You can similarly reduce sizes, add different shapes etc. using the Notch command. The one important thing to keep in mind is that before *notching* i.e. dragging the mouse with the left button pressed -you must position the mouse on the edge of the selected shape. If you press the RMB in the layout edit window you can see some more commands of which *Undo* and *Redo* are of much help. If you make a mistake while editing, you can immediately erase it using Undo. Similarly Redo may be used if you change your mind and it was not a mistake after all!

Note that you can enter non-rectangular polygon shapes by clicking and releasing (rather than click and drag) on each vertex of the polygon. If you double-click on a vertex, the polygon will be

completed. Typically you should double-click when you return to the initial vertex that you entered.

Path editing is another useful technique for creating layout data, especially when you're routing blocks together using standard width wires. Paths are fixed-width routes on a given layer. Click on Path in the Easy Edit palette, then on Options in the pop-up Add Path window. Select the METAL1 layer and set the width to 3, then click Ok. Now single-click on 3 non-rectilinear points in the edit window, double-clicking on the final point. You should see an L-shaped METAL1 path, 3-lambda wide path. The points you clicked on define the center line. Paths are easy to enter but can't be modified as much as shapes. After a path is created you can only edit its center line, width, layer and endstyle.

Note on Design Style:

Let's get back to the inverter layout. Usually cells (like the inverter) are designed so that they can be assembled side by side such that the power and ground lines abut each other. This is called *pitch matching*. This reduces need for some amount of routing and also enables a more orderly and regular layout. The pitch is therefore a standard value which you have to decide on at the onset. Let us assume that this height is 40λ. This is the height measured from the top of the power rail to the bottom of the ground rail. You have already laid out the vdd line so using the coordinate tracker you can now place your gnd line. Similarly place all other shapes as illustrated in the layout and remember to follow the design rules. Use the editing features described above if you make mistakes or need to copy or move objects. Note that in the example layout, the height of the cell is greater than the pitch. This must be take into account when there are cells above or below this one. However, this space can be saved if the cells above and below are the same cell inverted, so that they can share the vdd and gnd lines.

Now that you have completed laying out all the features that make the inverter, you need to add *properties* to some of the features. Properties are identifiers which are used by the layout verification tools to compare with the schematic. In other words, you have to now specify which are the input, output and power lines in your layout which correspond to similar lines in your schematic. Let's start with the vdd line. Select the shape representing the vdd line. From the Easy Edit menu click on *Prop Text*. In the dialog box type the *Name* of the property as *net*. Type the *Value* of the property as vdd. Click the options button. Another pop up box appears. Change the *height* to 1 if it differs from 1. Click on OK. Click on OK in the dialog box as well. Now, position the cursor to the place where you would like the property to be displayed and click the LMB. You should see vdd entered at that location. Similarly add *net* property called gnd to the ground line. Also add *net* properties called *in* and *out* to the input and output lines respectively. It is important to retain the same names as were used in the schematic otherwise LVS will report discrepancies.

The net properties are for the internal representation of the signal names. To define these at a higher level (say if we were to instantiate this inverter into a higher level layout) we need to also define *ports* on the above signal lines. Select a line to be made a port. Say, this time we choose *out*. Select the shape representing the output port. From the *Objects* menu on the top menu bar, select the following.

Objects > Make > Port

A dialog box appears at the bottom of the window. There are two boxes with scroll bars on them. One denotes the type of the port -default *signal* and the other denotes the direction in which the signal is transmitted -default *in*. Now, *out* is a signal so the type remains the same. However, it's direction is outward since it is an output port and hence you should click on the lower arrow on the second box till you see *out*. Enter the name of the port. It should have the same value as the *net* property i.e. *out* for the output port. Click on OK to complete the command. For the input port, the type of the port is *signal* and the direction is *in* and the name is *in*. For the vdd and gnd line, the type of port is *power*. You can select this by clicking on the lower arrow on the first box in the dialog box. The direction is *in*. The names remain the same as before.

If you realize that you made a mistake and did not want something to be a port, you can remove it by selecting the port and choosing C*onnectivity > Port > Remove from Port*.

For this assignment, nothing in this couple of indented paragraphs needs to be done, but refer back to it for later CADs. On most of the later CAD assignments, you will need to include cells

that you have previously designed. IC Station allows you to do this with the *Cell* command on the right side of the window. When you click on this command, a small window will pop up in the lower left side of the IC window. In this window you will need to enter the path and name of the cell that you wish to include in the current layout. Press Return. At this time if the cell you selected is a valid cell, an outline of the cell will appear in the layout. Move the cell to where you want it and click the mouse to drop the cell. Press ESC to close the *Cell* window, then press F2 to deselect the new cell. At this time you will see only the shapes that are ports in the new cell. To see more of the new cell, select that cell with the mouse. From the pulldown menu, select:

Context > Hierarchy > Peek

A new window will appear in the lower left corner. Enter the number of levels you wish to peek (since cells can consist of multiple levels of cell importation such as this). In this case, enter 1 and click OK. You will now see more of the included cell. An alternative method of peeking a cell is to select it and then type *peek <# of levels>* anywhere in the layout window. The last step of a cell design is the placement of the cell origin. You have probably noticed the white cross on the layout. This is the origin of the cell. All coordinates of the layout are based of this point. In order to move the cell origin, use the menu item *Context > Set Cell Origin* or type *set cell origin* anywhere in the layout window. You will then need to move the mouse to the location that you want the cell origin to be in the layout. We recommend moving it to the lower left corner of the layout. It is important to note that once you include the cell in a higher level design, you can not move the origin. The placement of the cell is based off the cell origins of the individual cells.

This completes the layout of the inverter. You are now ready to verify the layout for Design rule errors. Before you do this, save the cell with File->Cell->Save Cell.

Some Layout Extras:

1) Let's say you want to place one cell multiple times in an organized manner (e.g. a Register File). IC contains command that will place cells in a 2-dimensional grid of any size. To do this, unselect everything on the layout. Then select the cell that you want to make an array of (e.g. the 1-bit register or the 16-bit register).Then select the following:

Objects > Make > Array

A new window will show up at the bottom of the screen. Enter the # of row and columns that you want the array to be. Click on OK and the new structure will be shown in the layout.

2) If you have two or more nets which are connected together in a higher level of the hierarchy, but which are inconvenient to connect at the cell level, you can define a virtual connection between them (called a *must connect*). **This is particularly common for vdd and gnd lines.** You can make ports for all of these and give them the same name (e.g., vdd). Then select all of them with the same name and use the menu item:

Connectivity > Port > Define Must Connect

Then leave the default to *Unconditional* and click *OK*. This will allow LVS to function correctly, although you may see error comments about the multiply defined ports at the head of the LVS Report File. These should disappear when you do global routing of the power and ground lines because, of course, all electrically connected ports in the layout must be physically connected in the complete chip.

Design Rule Checking

At the top of the Easy Edit menu, you should see the word *Back*. Click on it to return to the previous IC palette menu. In the IC palette menu click on *ICrules*. The IC palette menu will change to the ICrules menu. Click on *Check* in the ICrules menu. A dialog box appears at the bottom of the layout edit window. If you need to save the DRC report, click *Options*.In the *Summary Report File*, enter the path and name of the file to save the report file in (e.g. *drc.rep)* and click on OK. This runs the design rule check. At the end of the check the transcript bar the bottom indicates a brief result of the check. If you see *total results 0*, it implies that the check was completed successfully. The DRC also generates a report file with all the DRC checks and their results. This is under the *Report > All* heading of the ICrules menu. Also a graphical interface is available to see the design rule errors.

Under the *Show* menu in the ICrules palette you can choose to see either the *Current* DRC error or *All* DRC errors. The errors are highlighted in the layout window by a separate error color. You can then easily check what the mistake is by referring to the process rules handout. In the *Show* menu you can choose *Unshow* to remove the error displays. Also available is a means to *scan* the errors sequentially. Under *Set Scan To* of the ICrules palette you can choose *First* to display the first DRC error. The corresponding design rule violated then is listed in the status bar at the bottom of the window. You can then choose *Next* to review the next error and so on. After you have made corrections, if any, run the *Check* once again and verify that you get no DRC errors. When you scan sequentially through the errors, the message area at the bottom of the IC Station window will provide a brief explanation of the rule that was violated along with a number that matches the corresponding SCMOS rule. You can modify the layout and immediately recheck without saving the cell. When you've fixed all of the DRC errors, click on *Back* at the top of the ICrules menu to return to the ICpalette menu.

Design Viewpoint

A *design viewpoint* is needed to translate the schematic design into a form recognizable by the layout versus schematic program. To create this go to a shell and make sure you're in the directory containing you inverter design files (e.g. when you *ls* you should see inverter.mgc_component.attr) and enter:

% lvsvpt inverter &

A transcript will be dumped in your shell as the command executes. The last few notes in the transcript should say something like "Created a new viewpoint:..," and "Viewpoint "lvs" is saved," if the command executed properly.

Layout vs. Schematic

Now you need to verify if the layout corresponds exactly to the schematic that had been designed and verified. LVS extracts a netlist from the layout and compares it to a netlist from your schematic. If all devices are interconnected via the same nets, if there are no extra devices on any given net and if all transistor widths and lengths are the same between layout and schematic, the check will be clean. ICtrace is the LVS checker that is invoked by clicking on *ICtrace(M)* in the ICpalette menu. You should now see the ICtrace menu on the right hand side of the window.

Click on *LVS* in the ICtrace menu. You will be prompted for a *Source Name*. This is the name of the design viewpoint created for the LVS. So move the mouse to highlight the Source Name box and type in (or browse):

inverter/lvs

If you want to save the LVS report to a file, you need to enter the file name in the *Report Name* entry. *lvs.rep* is the default entry. Click on *Setup Trace Props...* and turn off tracing for as, ad, ps and pd properties for both mn and mp type transistors. Do this by clicking (LMB) on the yes/no toggle button on the far right column next to each of the eight properties. Click Ok when finished.
Click on OK to initiate the LVS. Hopefully, if things have gone well you should not get any errors in the execution of the LVS. The results of the LVS are written to a report file which can be viewed from the ICtrace menu:

Report > LVS

If there are no *discrepancies* between the schematic and the layout, you will see CORRECT and a smiley face in the report file. You then need not proceed further. Close the report window by keeping the LMB pressed on the top left corner and choose *Close*. If you do get discrepancies, there are several debugging features available in ICtrace. Foremost, you can read the report file to see which layout and schematic nets and instances did not correspond. Then from the ICtrace menu select:

Logic > Open

In the pop up box that appears, enter the LVS viewpoint name at the *Logic Name* prompt i.e. *inverter/lvs* Click on OK. You should see the schematic appear briefly and then it gets buried under

the inverter layout. You need to resize the layout window to see the schematic. To do this, Keep the LMB pressed at the lower left corner of the layout window till you see a bracket shaped cursor. Now, keeping the mouse button pressed, drag the cursor upwards. You should see a ghost image of the lower border of the layout window moving along with the cursor. Place it about halfway up the screen and release the mouse button. Your layout window should have reduced in size. The above procedure can be used for all windows and at all places on the borders of windows. The inverter schematic should now be visible at the lower right corner. You can use the F1 and F2 keys to select nets and instances in the schematic and corresponding nets and instances will be highlighted in the layout. You can select an area by keeping the LMB pressed and dragging the mouse to form a rectangle over the desired area. This correspondence feature will be of vital use later in the semester while debugging LVS discrepancies.

Under *Scan* in the ICtrace menu you can select:

Discrepancies > *First* to highlight the first discrepancy,
Discrepancies > *Next* to highlight the next discrepancy, and
Discrepancies > *Previous* to highlight the previous discrepancy.

Similarly, you can Show discrepancies by:

Show > *Current discrepancy*
Show > *All Discrepancies*
Show > *Discrepancy by Number Show*n discrepancies and features can be *Unshow*n by:
Unshow > *All* to unshow all that is shown,
Unshow > *Current Discrepancy* to unshow current discrepancy only, and
Unshow > *All Discrepancies* to unshow all discrepancies.

This completes the LVS. Click on *Back* at the top of the ICtrace menu to get back to the ICpalette. Debugging LVS errors on the inverter is simple since there are so few nodes and devices to deal with. Things can get a lot more complicated even when you're dealing with only 10-20 devices and 8 or 10 nodes. A simple mistake like shorting vdd to gnd can lead to very few devices matching up and the information in the report file may not immediately lead you to the problem area. It helps to understand a bit about how LVS goes about comparing the layout and the schematic when debugging more complex circuits. First, the ports and any named internal nets are taken as initial correspondence points. The program will work its way in from these initial correspondence points, attempting to match nodes by the number and types of devices and pins that connected to them. Similarly, devices are matched if their pins share similar node connections. When LVS is unable to make a complete match, it will report nodes and devices that don't match up. The report will show the source (schematic) device or node in one column and the layout device or node it attempted to match it to in another column. A good place to start debugging is to look at the area of the report file called Number of Objects After Transformation (or just Number of Objects if no parallel connected devices were collapsed). See if you're at least getting the same number of n-channels and p-channels between layout and schematic. Check the node counts too. Shorts in the layout will show up as fewer nodes in the layout while opens in the layout will show up as extra layout nodes.

Again, sometimes simple mistakes (vdd and gnd shorts) can lead to many mismatches. You may have hundreds of discrepancies that will eventually disappear after making 3 or 4 layout changes. Take advantage of the initial correspondence points for tough debug problems. You can put net names on unnamed nets in the schematic and layout that you know should match. LVS may be able to use this information to match up a lot more devices and nodes and give you better information about where the real problem is.

Layout Parameter Extraction

It's now time to extract parasitic capacitances from your layout. This step cannot be done until after LVS is clean since the capacitances and transistor parameters must be back annotated to the viewpoint. These parasitic capacitances can then be accounted for in Eldo, leading to more realistic circuit delays. Note that in the original Eldo model you simulated, there are no interconnect capacitances and only estimated source/drain areas and perimeters.

Click on *ICextract(M)* in the ICpalette. You will now get the *ICextract* menu on the right hand side of the screen. First you'll extract parasitic capacitances for the metal and poly layers and model them with a single capacitor to ground for each node. To do this, click on *Lumped* in this palette. In the pop-up window that appears, click on Yes for *Specify Schematic Source* and provide the path to the inverter/eldonet viewpoint. Next, click on Yes for *BackAnnotate* and enter *sim_ba* for the *BA Name*. Leave the *ASCII BA Name* blank. Change the *Lumped Capacitance* name from *icap_net* to *cap_net*. If you wish to save a report file, type that name in the *Export File Name* box on the left side. Leave all other settings at default values and click on ok to run the extraction. An LVS will be run again and should again be clean unless you've made a change to the layout or to the trace properties in Setup Trace Props.

Next, you need to extract the real areas and perimeters of the source/drain regions. To do this, click on Netlist in the ICextract(M) palette. Once again, click on Yes to Specify Schematic Source and provide the path to the inverter/eldonet viewpoint. This time you want to trace as, ad, ps and pd, so enable their extraction by clicking the yes/no toggle button to yes in the Setup Trace Props... window. Click on Yes for Back-Annotate Device Properties and once again enter sim_ba for the BA Name. To save a report file, enter its name in the *Name* box in the upper left hand corner. Because you changed the lvs properties, **do not** change the name in the *LVS Report* box. You will write out an incorrect lvs report to the name in the *LVS Report* box and you don't want to overwrite your correct one. Click Ok to start the extraction.

Note: You only want to trace as, ad, ps and pd properties in the step above. If you inadvertently leave the toggles for these properties at yes during an ordinary LVS check, you'll get errors since the extracted values almost never match the estimated values on the transistor symbols.

This completes the layout parameter extraction. You'll get to see the results of the extraction when you open up the viewpoint to add load capacitance to your model. If you haven't saved the layout yet, do so then select File->Exit Session to exit ICStation.

Load Simulation

The capacitances that have been extracted do not account for any load connected to the circuit. Add 50 fF of capacitance to the output node to represent a typical external load on the circuit. To add this capacitance to the analog viewpoint, you need to use *dve* to edit the viewpoint. Invoke dve:

% dve_ic &

Then, select:

File > Open > Design Viewpoint Enter the *Component Name* (e.g. inverter) and the analog viewpoint (e.g. eldonet) in *Viewpoint Name*. Then, click *OK*. It will bring up two design viewpoint windows. From the pulldown menu select

File > Open > Back Annotation Use the Navigator to browse to the inverter directory and select the *sim_ba* file if it's not already selected. Click *OK*. After you click on *OK*, a new window will appear, which shows the extracted capacitances and

transistor parameters. We need to modify the output capacitance. Click on your output signal (it may be
called '*out*,' for example). It should appear highlighted.
From the pulldown menu select:

Edit > Change > Property A new window appears. Click on *cap_net* value. Click *OK*. In the dialog box that appears you should add the capacitance value of 0.050 to the current value since the value in the box is in pF. Click on *OK*, and then save the design viewpoint.

File > Save Design Viewpoint > with same name

Exit dve. You can now re-run Eldo as you did before. *Note: You should not have to re-add forces or choose probes again, but simply "Run Eldo" to obtain a simulation with the backannotated parasitics.* Now, re-measure the rise/fall delays to the output in EZwave. You'll use these values to model real circuit delay in Modelsim.

Re-simulating the Schematic

You need to add the delays obtained in Analog simulation to the schematic and simulate it once again in Modelsim to verify that it still behaves as you expect. To do this, you need to reopen the schematic in Design Architect. On the output of the inverter, you will need to add a delay box from the TSMC25_lib just before the output port. You will then need to change the rise and fall times of the delay box to the values that you found in EZwave. These numbers go in the top and bottom property texts, respectively, near the output of the delay box. Delays must be in ns, e.g. 0.5 but *not* 0.5n. After adding these, save the schematic and exit Design Architect. Note that our default Modelsim setup has a time resolution of 10ps, so any delays of finer resolution will be rounded to the nearest 10ps in simulation. Re-netlist your schematic with *Export* > *Verilog*, place the new inverter.v in your Modelsim project directory and recompile your project. When you re-run Modelsim you should now see the delays you've added to your model.

Design Manager

The time will come when you are going to need to move a cell or a schematic from one location to another. The program that allows this is called Design Manager. Start with the command:

% dmgr_ic &

This will open the Design Manager window. In order to copy or move a complete layout and schematic, you need to first copy or move the layout. This is because the schematic has a directory associated with it and if it is moved first, the layout is often placed in the directory and not in the intended location.

To copy a layout, select the layout on the right side that you wish to copy. It looks like a small IC layout. On the left side of the screen, select *ic*, this also has a small IC layout on it. Reselect the layout you wish to copy. Then on the far right side click on the *Copy* command. A small window will appear in the lower left corner. Enter the location you want the layout to be copied to.

To copy a schematic, repeat the above steps but select the schematic on the right side. It looks like a small DA schematic. On the left side, instead of the *ic* icon, select the *design_arch* icon.
In order to move a layout and a schematic, select the move icon on the far right side instead of the copy icon. An important thing to note is that when you change the name of a schematic, it does not change the name of the symbol associated with the symbol. You will need to recreate the symbol and give it the new name of the schematic manually. You will also need to delete the old symbol. To do this, double click on the schematic in Design Manager. Select the symbol (a rectangle with six inputs and outputs on it) and click on *Delete* on the far right side of the window.

Design Manager provides several other features that you will discover throughout the term. For example you can find the exact cells that you used in all levels of your design with one of the Report options.

Printing

You should also learn to print schematics, layouts and traces of simulations, even though we do not require hardcopy submissions. To print in **da_ic** select

File > Print Sheet with ICprint...

Type in the printer name and click OK.

To print from **EZwave**, activate the window you want to print by clicking in it, then choose

File > Print

Enter the printer name and select OK.

To print in **ic** select

File > Print > Hot Plot

Once again enter the printer name and click OK.

To print Modelsim waves select

File->Print Postscript...

and either give the print command (lp -d<printer_name>) or a filename if you just want to create a .ps file. Note that it defaults to dumping the output file to the modelsim project directory.

Conclusion

This is the basic flow you'll use for full-custom design in your EECS 427 project. Keep this tutorial on hand while you're doing your CAD assignments for your projects since it contains a lot of useful information. Also feel free to explore better ways of doing things in the tools. There are many shortcuts and alternative ways of doing things in the tools and we're unable to cover them all in this document.

Appendix A: Use of Bus Rippers

You will find that using bus rippers in da will make your schematics much cleaner and the handling of bus variables in simulation much easier. Bus rippers allow you to group single lines of the same 'type' into a bus. For example, rather than thinking of the result of an addition in an 8-bit adder as: s0=0, s1=1, s2=1, s3=1, s4=0, etc., you can think of the sum as a 16-bit entity: s[15:0]. You're probably all familiar with that. In 427, we will require that you always use [] and not () when naming buses and always do it [MSB:LSB]. This will avoid confusion later down the road. Use the bus rippers in the Generic_lib in da. There are several types (16x1, 8x1, 4x1, 1x1 etc.) which are available. A bus is usually drawn with a thick line (Explore the ADD NET option). Also its name is of the form NET[15:0] for a 16-bit bus. Each line of the bus is referenced by its bit number. For example, the 5th bit of a bus called NET would be NET[5]. So the corresponding signals in the layout would be named NET[0], NET[1],..., NET[15]. The bus rippers have a property called RULE attached to them. This indicates the numerical position of the net in the bus. You need to change this property to the appropriate number for each ripper. You might find the 1x1 ripper to be most useful. The thick end of this ripper is connected to the bus and the thin end to the individual line.

While changing the "*R*" text on the bus rippers to whatever value you want, you can use some helpful macros like

Sequence Text

from the Property/Text menu. After clicking on Sequence Text, a pop-menu within the editing area in DA will appear. This lets you change the "R" values quickly rather than having to change them one by one. This macro also lets you add a prefix and suffix to the numerical index. For example, if you want to name a bunch of nets as

input0pin
input3pin
input6pin
input9pin you can specify input as the prefix, pin as the suffix, 0 for the beginning index, and 3 for the step, then click

away on the nets to change their names from whatever their present value is.

Be careful when you use this macro. It takes a couple of AMPLE commands to execute this macro and

hence it will take a couple of seconds. If you click too fast, it will change the previously selected net again.

This macro is also somewhat smart as it will not let you change different properties by mistake.

Similarly,

Change Values

from the Property/Text menu will let you change multiple prop values at one time. In the dialog box for this macro, just enter all the values one by one by using the TAB key to move from one field to another and when you are done, OK the dialog box. Then you can click on the property values to be changed, one by one. This command, unlike Sequence Text, will let you change different properties at one time, for example the name of a net, the value of a resistor, etc.

Appendix B: Cell Naming Conventions

When naming cells (e.g. inverter), nets (e.g. in, out) and instances, use the following naming conventions:

1) Names can contain only lower case alphabet, digits or the underscore: "_".

2) Names should begin only with lower case alphabet.

3) Names should not end with "_".

4) Don't use names of verilog keywords. The common ones are given in the list below. By following these naming conventions you can avoid problems in other tools that use the design data you enter in Design Architect. In addition, the following list of names cannot be used as cells you create. They are either cells that exist in our standard cell library, in our memory components that will be used later in the semester, or are verilog keywords. Also, do not name two different cells with the same name even in different cad assignments. You will later be combining all your cad assignments into one project so this will cause problems also. If you follow these rules, you'll save a lot of problems from occurring later in the semester.

Illegal Cell Names

and02	and03	and04	ao21	ao22	ao221
ao32	aoi21	aoi22	aoi221	aoi222	aoi32
aoi321	aoi322	aoi33	aoi332	aoi333	aoi43
aoi44	buf02	buf04	buf08	buf12	buf16
dff	dffr	dffs	dffsr	fadd1	hadd1
inv01	inv02	inv04	inv08	inv12	inv16
latch	latchr	latchs	latchsr	mux21	nand02
nand03	nand04	nor02	nor03	nor04	oai21
oai22	oai221	oai222	oai32	oai321	oai322
oai33	oai332	oai333	oai43	oai44	or02

or03	or04	sff	sffr	sffs	sffsr
tri01	trib04	trib08	xnor2	xor2	rom256
rom512	rom1k	rom2k	ram256	ram512	ram1k
ram2k	and	or	xor	buf	not
nand	nor	xnor	reg	tri	tri0
tri1	dff_p	dff_r	dff_s	dff_sr_1	latch_p
latch_r	latch_s	latch_sr_1	mux2	mux4	dff_sr_0

Appendix C: Inverter Schematic and Layout