LAB #1: Schematic Entry and Logic Simulation

This lab is due at 5pm in the EE locker for CS/EE 3700 on Thursday, January 18, 2000
NO LATE HOMEWORK WILL BE ACCEPTED.

1 Objectives

In this laboratory, you will learn how to use a schematic capture system to draw the network of logic gates that constitute your design. The major advantage of computer-based capture of your design is that the computer can simulate the logical behavior of your design. This allows you to verify that the circuit operates as you expect it to, even before you build it in the laboratory. Thus, the laboratory also introduces you to a simple logic simulator that allows you to look at logic 1’s and 0’s attached to nodes within your schematic.

2 Prelaboratory Exercises

You should be familiar with the function of basic AND and OR gates from lecture. The following exercises should be completed on the attached Summary Sheet before you do the laboratory.

1. The half adder is a function that takes two binary inputs, X and Y, and produces two binary outputs, SUM and CARRY. It implements a simple binary addition. Fill in the truth table for the behavior of the half-adder on the lab summary sheet. Draw a schematic for the half adder as well. You may use only NOT (INVerters), AND, and OR gates.

2. The full adder is a similar to the half adder but it that takes three binary inputs, X, Y, and CARRY-IN, and produces three binary outputs, SUM and CARRY-OUT. Fill in the truth table for the behavior of the full adder on the summary sheet. Draw a schematic for the full adder in the space provided on the summary sheet. Once again, you may only use NOT (INVerters), AND, and OR gates.

3. You can build a full adder using half adders and logic gates. Draw a schematic for a full adder built in this way in the space provided on the summary sheet.

3 Overview of Powerview

This class will use the Powerview CAD tools from VIEWlogic. Actually, VIEWlogic has been bought and sold a few times in the last couple years, and is currently named Innoveda, but our tools date from when the company was still called VIEWlogic. This lab is designed to get people up to speed quickly on the schematic capture and simulation portions of the tool suite.

VIEWlogic Systems Inc. makes a set of computer aided design (CAD) tools for electronic circuit design that are widely used in industry. The VIEWlogic suite of tools includes a variety of programs that deal with different aspects of circuit design. These programs are all invoked from a top-level program called Powerview. Powerview is an umbrella program that provides a framework from which all the other VIEWlogic programs can operate. There are over 20 programs in the VIEWlogic suite, but you will only use a few this quarter. The main programs that you will use this quarter include:

Viewdraw – Schematic capture program. This is used to draw circuit schematics.

VSM – A conversion program to generate simulation files from schematics.
Fusion/Viewsim – Digital circuit simulation. Fusion and Viewsim are essentially the same thing. Viewsim is the name of the digital circuit simulator, and Fusion is the name of the simulator that also allows VHDL and Verilog simulation. In practice, you get the same code no matter which one you execute, but viewsim has fewer of the HDL widgets turned on by default.

Viewtrace – Waveform analyzer for digital simulation output.

The purpose of this lab is to make sure that you are able to run the VIEWlogic tools, and to introduce Powerview, Viewdraw, and Viewsim by having you create, simulate, and print a simple circuit schematic. This lab will in no way will teach you everything about Powerview or Viewdraw. It is only intended to get you started. If you would like to learn more, or are confused by something while using the programs, the documentation for all the VIEWlogic tools is on-line. This lab will show you how to access the on-line documentation. As always, the TAs and the instructor are also a good place to look for answers. Your fellow students are another good resource for learning how to use the tools. Note that the opers in the CADE lab do NOT necessarily know how to use VIEWlogic, so asking them may or may not do any good.

4 VIEWlogic Setup

The first task is to set up your environment so that the programs will run. First, and most importantly, the VIEWlogic tools will run only on Sun SPARC workstations running Solaris! The CADE lab has a bunch of SPARC's so this shouldn't be a problem, but you must be logged into a SPARC when you start up the system. Note that if you really like the NT lab, using an X-windows emulator it is possible to be sitting in front of an NT machine, running Powerview on a SPARC through one of your windows, and having that SPARC display on your NT box. You must complete the following steps before starting Powerview.

1. Make sure you are logged into a Sun SPARC and running X windows. You can use whichever window manager you are familiar with but you must be running one of them.

2. There is a special script that you will use to start Powerview. This script will make sure that certain system variables are set up correctly before invoking Powerview. This script lives in the CS3700 class bin directory (/home/cs/handin/cs3700/bin on the CADE machine, and /services/classes/cs3700/bin on the CS machines). It will be helpful to add this directory to your search path so that you don't have to type the whole thing every time. This is probably best done in your .customs.cshrc file. For example, you might add the following line to your .customs.cshrc file: set path = (/home/cs/handin/cs3700/bin $path) This will add the cs3700 class directory to the front of whatever search path you already had set up. Be sure to relink or resource this file afterwards.

3. In order not to clutter up your main directory, the Powerview startup script makes a new directory under your home directory from which to start Powerview. When you run Powerview a bunch of new files will be created by Powerview. It's nice to have these separate from your main directory. So, the Powerview script will make a directory for you called 3700-pv. In the extremely unlikely case that you already have a directory of this name, it would probably be best to rename it to something else so it doesn't get a bunch of strange files dumped into it.

5 Starting Powerview

You are now ready to run VIEWlogic. Remember that all the VIEWlogic programs are called from the top-level program called Powerview. However, enough extra stuff needs to be set up to make Powerview work that a script has been written that handles all this stuff behind your back. This is called pv-cade on the cade machines and pv-cs on the cs machines and is in the class bin directory.

This script does a few things for you. First, it will create a new directory for you in your home directory called 3700-pv. Once this directory exists, the script connects to that directory and fires
up Powerview. Every time you execute pv-cade or pv-cs you will be connected to that directory before invoking Powerview.

Once you execute pv-cade or pv-cs a new window will appear called the "Powerview Cockpit." This is the window from which all the VIEWlogic programs will be run. You should see a list of menu items across the top, a list of configuration information on the left side, and icons representing Viewdraw, Viewsim, VSM, Viewtrace, and Fusion in the right part of the window.

The interface to the various programs is through the keyboard and the mouse. The bindings of the mouse button are shown at the top of the window for whichever program you are using. In general, the left button selects and executes commands, the middle button finishes commands, and the right button brings up an extra menu. However, this isn't always true. You can see what each button should do by looking at the top of the window, and you'll get the hang of it pretty quickly. You can also press the space bar to get a command window and type a command if you like.

Most of the keys on the keyboard are so-called “hot keys.” This means that pressing any single key causes some action to happen. These actions are also available using the menus. In fact, if you mouse on a menu it will show you the keyboard shortcut at the right of the menu selection. The function keys on the SPARC are also programmed as “hot keys.” Their bindings depend on the program you are using and are shown in the menus.

### 6 Organizing into Projects

The first thing you need to do is set up a new project directory. Projects are simply ways of keeping different designs organized. There are a bunch of files that get generated for any design, and these files are best kept in a place of their own. In UNIX terms, each new project will be a subdirectory under your "/3700-pv directory. You should create a new project for this first lab (and for the following labs too). Call this first project "lab1." You need to have Powerview set up these projects for you so that it can do some bookkeeping and keep track of where all the files go.

So, make a new project by selecting the Create option under the Project menu. This will prompt you for a project name. It's a good habit to get into to always type the full name of a new project. This avoids making your new project a sub-project of an old project. So, type the full name starting from your home directory (i.e. "/3700-pv/lab1) and hit return. Powerview will make a new project directory, make some new directories in your new lab1 directory, and copy some init files there as well. You can also use the “browse” option in the create-project dialog box to help select which directory you create the new project in.

Once you have a project created, Powerview will assume that that's the project you want to work on. This is fine for now, but what about later when you have many projects? You can choose which project you want to work on by mousing on the “Current Project” selection in the Powerview window. Most of these selections are pretty obvious. The "Current Toolbox" selection just tells you that you are using the CS/EE 3700 set of tools, for example.

This is the end of your setup. Once you have a project set, each time you start Powerview (using pv-cade or pv-cs) your project will default to the one you were last working on. If this is not the project you want, use the current project box to change projects by selecting a new one.

### 7 On-Line Documentation

All of the documentation for Powerview is on-line. Look at the menu selection marked by the red square (the leftmost menu choice). This menu selection will always be there regardless of which tool you are using (the others will change for the different tools). Mouse on this menu and check it out. Notice the “Help” selection at the bottom. The arrow indicates that there is a submenu. Check out the submenu and you'll see a variety of choices for help. The main documentation is under the “Viewdoc” choice. If you mouse on “Viewdoc” Powerview will spawn a new window with the on-line documentation. Note that this is actually a separate program with its own, different, user interface. The documentation is hypertext based so you can mouse on the little boxes to go to that part of the documentation. The interface should be largely self-explanatory (I hope).
8 Schematic entry for the half adder

You are now ready to create the schematic for your half adder circuit. The drawing package used to construct schematics is called Viewdraw. Once you have selected which project you are using, you can construct the schematics that make up that project. Each project can consist of many schematics. Building new symbols that represent your schematics allows you to include your circuits into other schematics in a hierarchical way.

8.1 Drawing a Schematic with Viewdraw

You are now ready to open a schematic window and start building your circuit. The first step is to fire up the Viewdraw program. Double click on the Viewdraw icon. This will load Viewdraw, and pop up a dialog box asking you which schematic you would like to open. Of course, you haven’t built any yet so there aren’t any to choose from. Go to the “Enter Name” space and type the name of a new schematic file (i.e., HA). Later, when you have built a few schematics, they will be listed in the top of this dialog box and you can go to an old schematic by clicking on it.

Notice the list of things at the bottom of this dialog box which includes your new lab1 project, “(labkit)”, “(device)”, and “(builtin)”. These are locations of libraries of electrical components that Powerview will search for circuit components. The (builtin) library is the basic VIEWlogic library that all other cells are derived from. You should never have to use any cells directly from the (builtin) library.

OK - once you’ve typed the name of a new schematic, or moused on an old one, a Viewdraw window will spring into being. Notice that the menu selection is a little different at the top of the Viewdraw window, but that the red square menu is still there.

8.2 Adding Components to a Schematic

The process of building a schematic is to place components (logic gates or larger circuits made of combinations of logic gates), and connect those components with nets. Nets are like abstract wires that make an electrical connection between the components in the schematic.

The set of components you will be using (in the (device) library) are the same components that you will have available in your lab kits. They are all logic gates in what is known as the 7400 series of logic gates. The 7400 series of gates is a standard way of numbering gates so that you can identify which chip has what type gates in it. The 74 just identifies the numbering scheme, the letter that follows after it identifies the technology that the chip is built from, and the following number identifies the gate type. The gates in the Powerview library that you will be using are all 74HCxxx parts which means High-speed CMOS. A partial listing of the numbers and the gate types is shown in Figure 1. You’ll see the rest once the lab kits are handed out.

**VERY IMPORTANT** Do **NOT** use components directly from the builtin library. It’s tempting to do so because there are all sorts of gates there and they have nice intuitive names like nand2

<table>
<thead>
<tr>
<th>Number</th>
<th>Gate Type</th>
<th># per package</th>
</tr>
</thead>
<tbody>
<tr>
<td>74HC00</td>
<td>2-input NAND</td>
<td>Four</td>
</tr>
<tr>
<td>74HC02</td>
<td>2-input NOR</td>
<td>Four</td>
</tr>
<tr>
<td>74HC04</td>
<td>Inverter</td>
<td>Six</td>
</tr>
<tr>
<td>74HC08</td>
<td>2-input AND</td>
<td>Four</td>
</tr>
<tr>
<td>74HC10</td>
<td>3-input NAND</td>
<td>Three</td>
</tr>
<tr>
<td>74HC11</td>
<td>3-input AND</td>
<td>Three</td>
</tr>
<tr>
<td>74HC20</td>
<td>4-input NAND</td>
<td>Two</td>
</tr>
<tr>
<td>74HC27</td>
<td>3-input NOR</td>
<td>Two</td>
</tr>
<tr>
<td>74HC30</td>
<td>8-input NAND</td>
<td>One</td>
</tr>
<tr>
<td>74HC32</td>
<td>2-input OR</td>
<td>Four</td>
</tr>
<tr>
<td>74HC86</td>
<td>2-input XOR</td>
<td>Four</td>
</tr>
</tbody>
</table>

Figure 1: Common 7400-Series Gates
instead of 74hc00, but they do not correspond with any real parts. They are there as building blocks for modeling other parts and are not supposed to be used directly. You will lose points on your lab assignments if you ever use parts directly from the built-in library! All the basic logic gates in your schematics should come from the class labkit library or the devices library that contains 7400-series parts. This warning is in effect for the rest of the semester! Never use built-in gates directly.

To add components to your schematic, go to the “Add” menu and select ”Comp” (alternatively, you can use the keyboard shortcut by hitting the “i” key for “insert.”). This will pop up a dialog box that lists the possible components that you can add. These components will either be primitive gates from one of the libraries, or brand new assemblies of gates that you made yourself. Of course, you haven’t made any yet, so there aren’t any listed. Later, once you have created components of your own, those components will be choices in this dialog box.

To add one of the standard components, you need to select them from the standard component library. You need to tell the dialog box which library you would like to use by clicking on one of the component libraries at the bottom of the dialog box. Try the (devices) library. What you should see is a bunch of components to choose from that are all identified by number. This is the library that contains the standard 7400-series gates. There are a lot of them.

You add components by selecting them from the top part of the dialog box, and then moving your mouse into the viewdraw window. Click the middle button to place the component. You can then go back and select a new component or this one again, and again use the middle button to place the component. When you’re finished placing components, select “cancel” to dismiss the dialog box. Clicking the left button will toggle whether you can see the component drifting around before you place it or not. Personally, I like to see it.

Note that there are multiple versions of some of the gates in the “devices” library. That is, there are gates called 74hc00.1 and 74hc00.2. You should look at both of these and see what the difference is! Remember that they are all the same gate because they all have the same 74hc00 number. Hint – remember DeMorgan’s Law! (or if you haven’t seen DeMorgan’s law yet, remember these gate designs when we talk about it in class)

8.3 Connecting Components with Nets

Once there are devices in your schematic, you need to connect them together. In Viewdraw, this is accomplished by connecting the components using nets. A net is like an abstract wire that represents an electrical connection between the components. Select Add→Net to begin adding nets, or press “n” to go into net-adding-mode.

You can now wire the components up by using the middle mouse button to draw nets between the components. Remember that the current mouse bindings show up at the top of the Viewdraw window. The connection points on the components are shown as little blue stubs, called pins, that you can connect red nets to. Position your mouse at the pin you would like to start at and press the middle button. You are now drawing a net as you move the mouse. To make the wire bend in a particular place, press the middle button again. When you get the net to the pin you would like it to stop at, press the middle button to finish the net.

If you would like to make a connection to the outside world (i.e. to a circuit not shown on this particular schematic, or as an input or output to the world), make a net starting at a pin, pull it away from the component, click the middle button to end the net, and the press the right button to make the net dangle. This will leave a square at the end of the net to show that it ends there.

As a matter of style, if a signal comes from off of the schematic page or goes off of this page, you should add an inpin or outpin to that net. More on that later.

8.4 Moving Things Around

Once components and nets are placed in a schematic, you may want to pick them up and move them around to make the schematic look better. You can do this by selecting an object (net, component, label, text, etc.) and choosing Edit→Move from the menu, or using the “m” key to move. Drag the objects around the screen and use the middle button to pin them into place.
8.5 Labeling Nets in a Schematic

The nets will make connections between the components. It is often very helpful to name these nets so that you can refer to them later and know which net you are talking about (i.e. during simulation). This will be essential when building a symbol for the schematic as mentioned later. To label a net, select a net using the right button, and choose Add→Label from the menu (or press "I"). You will be prompted for the label with a dialog box. Type the label and hit return. The label now shows up as a little white box that you can move around. If you would like to see inside the box, press the right button. When the label is where you want it, press the middle button. This label is now associated with that net.

These labels are also important (and also require some care in their use) because the Viewdraw program, and other programs such as Viewsim, will assume that any two nets in the same schematic with the same name are connected, even if there is no net directly connecting them! You can use this trick later in the quarter when your schematics get more complicated to reduce the tangle of nets that can clutter up a schematic. However, you need to keep this in mind so that you don’t get confused! If you happen to have two nets named the same thing in the same schematic that really are different, you will get very confused when you try to simulate things! The ability to connect things by name is one of the more powerful and useful features of Powerview. It’s also the number one cause of simulation frustration later on, so be careful and pay attention!

8.6 Saving Your Work

Once you have created a schematic, you need to save the file. Use File→Write to write the file. The system will check for errors, and report the status. You can also use “w” as a shortcut. IMPORTANT: because Powerview is a large and complicated system, and our CADE network is a large and complicated system, sometimes unanticipated things happen that cause things to quit unexpectedly. Get in the habit of saving your work frequently!

8.7 Adding a Frame to Your Schematic

A good practice for drawing schematics, and one required by this class, is to put the schematic inside a frame. This frame identifies the edges of the paper so you don’t draw outside the edges, and leaves space for identifying the schematic, and for listing other information about the schematic. For this class, there is a standard frame that you must use for all schematics that you draw. This frame is in the labkit parts library (use Add→Comp, or "i", and select “(labkit)” in the dialog box to start selecting components from the cs3700 library) and is called “bsheet.” To add the frame to your schematic, select the bsheet component and place it in your file. Center the frame in the large white box in the Viewdraw window and place the component using the middle button. In fact, it is usually a good idea to make the bsheet frame the very first component you add to any new schematic.

You can add text (using Add→Text or “t”) to the information section of the bsheet to document the schematic’s function and to identify the author of the schematic. When you add text, it is sometimes not the size you want. Use Change→Text→Size to change the size of the text. Make the text large enough to read when the schematic is printed! See the schematic style section for further details.

8.8 Building a Symbol for a Schematic

In order to include a schematic that you’ve designed into another schematic (in other words, use hierarchy), you will need to make a symbol for that schematic. A symbol is simply a picture of the schematic that can be placed, like a gate, in another schematic. When you place components in a schematic, you are placing symbols that represent those components.

To build a symbol for a schematic, you simply design a picture to represent that schematic, and give that symbol file the same name as the schematic file. You also need to make the blue pins on the symbol that will be used for connecting nets when you place that new component in a schematic.
For example, consider your half adder circuit. If I want to use this compound gate as a component in another schematic, I need to create a symbol for that schematic. If the schematic file is named HA, then I need to create a symbol called HA. This can also be done in Viewdraw by telling Viewdraw to open a symbol window rather than a schematic window. Notice that when you get the viewdraw dialog box you have a choice to select a schematic or symbol (schematic is the default). A symbol window is a Viewdraw window specifically designed for drawing symbols. Make sure to specify exactly the same name as the schematic you are making the symbol for! You can also Push → Symbol (try mousing the right button to get the push menu) to push into a symbol with the same name as the schematic automatically.

Note when you try this that the white frame is quite a bit smaller than the frame for a schematic. This frame denotes the extent of the symbol when you place it in a schematic.

Start by drawing a box (Add → Graphics → Box) to denote the new gate. Draw this box inside the white frame leaving one grid space between the box and the frame. Add some text so that you can identify the new gate (i.e. AND-OR). The text will probably not be the right size. So, select the text and change its size using (Change → Text → Size) to 15. You can draw whatever picture you like for the new symbol. The gates in the AND-OR circuit are just symbols that were drawn by someone else using lines and arcs. Symbols are just graphics that represent some other piece of circuit.

Now you need to put pins between the box and the frame. These pins are where nets will attach when this is included in a schematic. Select Add → Pin (or use “p”) and draw the pins between the box and the frame. Pins can only be drawn up to the edge of the frame, so it’s best to leave a little space between the graphics that make up the symbol and the frame. So that these pins can be related to the nets in the schematic, they need to be labeled. Use Add → Label the same way that you labeled nets. In particular, any wire that you want to connect to must be labeled and have a pin in the symbol with the same label. For this circuit, the inputs are X and Y, and the outputs are SUM and CARRY so I’ll make four pins and label them with these names. Save the symbol file using File → Write.

Once this symbol file is created and saved, it will show up in the dialog box when you try to add components. For example, I can now use this HA circuit in a new schematic.

8.9 Hierarchy Shortcut

In your class version of Powerview, the mouse clicks have been modified from the “vanilla” Powerview to make traversing a circuit hierarchy easier. If you double-click the left button on a symbol, Powerview will push into that schematic. For example, if you double-clicked on the and-or symbol from Figure 3, Powerview would push into the schematic that is represented by the and-or symbol. To pop back out to the schematic you started in, double-click anywhere else in the schematic (i.e. anywhere not on a symbol).

8.10 Printing

To print any file in your design, you must create a plot file. The plotting commands are in the red square menu. Select the plot-setup item in the red-square menu and make sure that the type is set to ’postscript.’ This is very important. If you try to print files that were generated in other formats, you will most likely end up printing lots and lots of pages full of numbers. You can check a plot file to see if it’s postscript by looking at the first few lines of the file. The very first line in a proper postscript file will be:

%!PS-Adobe-2.0 EPSF-1.2

If the very first line in the file does not look like this - it’s not postscript and won’t print (at least not the way you want to print) on the cande lab postscript printers!

OK - once this is set up, you can use plot-go in the red square menu to produce a plot file. This will be called schematic-name.001p. You can queue this file up for plotting like any other postscript file using lpr from the UNIX prompt.
8.11 Schematic Style

Although good schematic design is a matter of individual taste, there are some basic conventions that will make your schematics much easier to read. The basic rule is that neatness counts! If your schematic is well organized, uncluttered, and well documented it will be much easier for me to figure out what you did. A few minutes spent lining up your components, taking the jogs out of the wires, making the labels large enough to read, and organizing the flow of the signals and components on the page will greatly improve your schematics. Neatness and organization of your schematic will be part of your grade! Well organized and well documented schematics will be required for full credit.

There are a number of different sheet sizes you could use for your schematics. The preferred sheet to use for this class, and the default sheet size in VIEWlogic, is a B-sheet (other letters are used for different sizes of sheets, the higher the letter the bigger the sheet). Don’t try to jam too much onto a single schematic sheet! That’s like trying to put your whole program in one huge “main” procedure! If you can’t squeeze the schematic onto a B-sheet, think about making it more modular and hierarchical before going to a larger sheet size. Use the B-sheet that is given to you by ViewDraw and you’ll be fine.

Each schematic should have a border. The component that you insert to get the border is called a bsheet. There is a class bsheet in the (labkit) cell library. You should also fill in the name of the schematic and a short description in the documentation box in the lower right of the bsheet. You may want to make a bsheet of your own that has your name already entered in the name portion of the documentation box. To do this, make a copy of the bsheet in your own project (use write-to to write the bsheet to your own project), then edit the bsheet in a symbol window and add your name and whatever else you would like to to personalize your bsheet.

In general, schematics should flow either left-to-right or top-to-bottom. That is, the inputs are generally placed along the left or the top of the schematics and the output on the right or at the bottom. Of course, depending on the particulars of the schematic, there may be exceptions to this rule.

You will have to put labels on all nets in the schematic that go to the interface of the schematic (pins of the symbols for example). These labels should be in size 20 type (the current default font size) or larger. Please consider putting descriptive labels on interior nets in the schematics too for ease of understanding (also, you must label all buses even if they do not go to pins). These extra labels should be at least size 10 type. All pin labels in new symbols that you create should be in at least size 10 type. Other text in your new symbols should be in larger type (at least size 15).

Please put comments in your schematics that explain what the schematic does, or how it does it, or what that tricky bit in the middle is doing etc. This is the same as a commenting your code in a programming class. You may remember what that tricky bit of code does, but it will be hard for someone else to figure it out without some hints. This text should be at least size 15.

All schematics should use a size 10 grid (the default). When you are making your own symbols it is sometimes useful to use a smaller grid to move text around and make things look nice, but make sure that all the pins in the symbol are on a size 10 grid.

Any nets that come from off the page (i.e. from the symbol’s interface) should have “inpin” components on them. These are in the (labkit) library and are just graphics that make it clear that the signal is coming from the interface, and not from another circuit in this schematic. Likewise, any signal that is going outside this schematic page (i.e. out to an interface pin on the symbol) should have an “outpin” component attached to it.

8.12 Schematic entry part of your lab

At this point, you should take what you have learned and complete the schematic for your half adder circuit. Repeat the previous steps to grab all the gates that you need to build your half adder, and wire your gates together. Next, create a symbol to stand for the half adder in other higher level schematics, such as the full adder. At this point, you should create a new schematic for the full adder using two instances of the half adder and an OR gate.
9 Simulating your half adder using Viewsim/Fusion

The simulator that is integrated with the VIEWlogic tools through Powerview is a program called ViewSim (in the new version of the Powerview tools, it is also called the FusionHDL simulator, so don’t be confused if you see that on your screen: ViewSim and Fusion are really the same thing). ViewSim is a gate-level simulator for digital circuits. What this means to you is that you can apply 1’s and 0’s to the inputs of your circuit, and the simulator will report what the resulting values are at the outputs of your circuit. This can be reported as a list of values in a table, as values attached to your schematic in ViewDraw, or in the form of a waveform showing the input signals and output signals over time. This simulator is integrated into the powerview umbrella and interacts directly with the ViewDraw schematic capture system. You will now learn how to prepare your circuit schematics to be used with the simulator, how to run the simulator, and how to interpret the results.

9.1 Preparing Your Schematic for Simulation

In order to simulate your circuit, the first thing you must do is put labels on any nodes that you want to set, and any nodes whose values you want to check. Use Add→Label (or “L”) to add labels to nets as described in the previously. Attach labels to the input nets in the circuit, to the output nets in the circuit, and to any internal nets whose values you are interested in checking. Note that these labels must be unique! Remember that any two nets in the same schematic that are named the same thing will be considered by ViewDraw to be connected even if there is no net drawn between them.

Please note that labels are quite different from text in ViewDraw. A label is an identifier that is attached to a net. When you refer to that net (which you need to do in ViewSim), you can refer to it by name only if it has had a label attached to it. They are like named variables in a program. Text, on the other hand, has no meaning as far as the circuit is concerned. It is simply a comment, much like a comment in a program. It is useful to the human reading the program, but makes no difference to the system running the program. For using the simulator, it is important to have labels on interesting nets so that you can refer to them by name during simulation. You do not have to put labels on every net in the system! (In fact, these other nets are also labeled, but are named very obscure things by ViewDraw). When you select any net or component in ViewDraw its name (label) shows up in the lower left corner of the ViewDraw window. If you’ve named that object something with a label, that’s the name that shows up. If not, you see the ViewDraw-generated label.) You only need to label those nets that you are interested in during simulation.

You can, and should, also use text in your schematics as comments. It’s very helpful to explain things about the schematic by putting in a little text. This is exactly like comments in programs. You don’t need a comment on every line, but well-chosen comments can make things much easier to understand. The same thing applies to schematics. On the screen, text and labels will be drawn in different colors so they will be easy to differentiate. Labels are in white, and text is in green.

Generating the .vsm File for ViewSim Once you have the schematic labeled, you need to generate a version of that schematic that the simulator can read. This operation is known as creating a netlist of the circuit. It extracts the entire circuit (remember that a more complicated circuit might be made of many schematics nested hierarchically) into a single file consisting of gates and their connections. In powerview, this conversion is done using a program called VSM. So, double click on the VSM icon in the powerview window. This will result in a dialog box asking about the file that you want to convert. If you already have a schematic open using ViewDraw, that file will be the default choice for which one to convert. If you don’t have ViewDraw active, you will have to type the name of the schematic that you’d like to convert. You won’t need to mess with the rest of the slots in the dialog box. You want the long format list (the default) and you won’t be specifying levels, or using a delay table file. The default output file is <schematic-name>.vsm. Do not name it something else, or powerview won’t be able to annotate the schematic with simulation information. The vsm extension identifies this as a ViewSim file. This file will be created in your project directory. Press “ok” to start the conversion process.
9.2 Running ViewSim

Once you have the .vsm file generated, you can execute ViewSim by opening a ViewSim window. Double click on ViewSim to fire it up. If you have a schematic window open, the default design name will be the name of that schematic file. If not, you will have to type the name. The most useful way to use ViewSim is to also have the schematic file open using ViewDraw. That way, the values produced during simulation will be annotated onto the schematic and you can easily see what’s going on with the circuit.

There are some options to ViewSim that are controlled by the on-off buttons in the dialog box. It doesn’t really hurt anything to leave them on, but really the only box you want on is the first one labeled “Graphical Interface.” So, click the “off” side of the others (you should only have to do this the very first time you use ViewSim). You also don’t need to specify a command file at this time. A command file is just a list of ViewSim commands that you can have executed when things start up. Later on you may find it useful to set up a small initializing command file for your design, but it’s not necessary. So, click on “ok” to get the ViewSim window.

At this point, you will get a ViewSim window that reports some statistics about your design (how the delays are scaled, how many components there are, etc.). The bit about delay scaling can be ignored. This simply tells you that the delays assigned to each of the gates are scaled by one. If you wanted to see how well your circuit worked when it got hotter (and therefore slower) you could scale all the delays by a number greater than one, or whatever. You won’t need to adjust this at all. Notice that if you look at the schematic in the ViewDraw window, all the nets in the schematic have been annotated with an X. This shows that the circuit has not been initialized yet so all the nets are at unknown values.

Also notice that the title bar of the ViewSim window says that you are actually running FusionHDL. This is just the new name of ViewSim and has to do with enhanced support for using Hardware Description Languages (HDLs). It has no effect on the way you are using ViewSim.

Simulation Model The simulation that ViewSim and many other simulators use is known as ternary simulation. This means that the nodes (wires) in the circuit are allowed by the simulator to be in one of three states: 1, 0, or X. A 1 or 0 state is used when a circuit node is electrically driven to that state. The X state is used to show that the node is either uninitialized and not driven to either 1 or 0, and thus the simulator does not know which state it is in, or it is being driven to both 1 and 0 at the same time and the simulator can’t figure out which driver will win (in a real circuit, things usually just melt at this point). Your circuit, before anything is initialized, will have all its nodes in an X state. In Viewsim there is actually another state shown as ? which means that Viewsim doesn’t have any record that this node exists. You will hopefully never see this state. If you do, it might mean that your schematic is newer than the vsm file so there are nodes in the schematic that are unknown to Viewsim. There is some anecdotal evidence that you can also get this state if you try to write a vsm file at a time when your disk quota won’t let you write the entire file. In this case it seems that sometimes you can get truncated vsm files that leave out nodes. This is unlikely though so don’t worry about it unless you actually start seeing ? nodes.

The process of simulating the circuit is to drive the inputs of the circuit to a 1 or 0 and then examine the outputs of the circuit (and possibly internal nodes too) to see if they are at the correct values. At its simplest, testing is simply a verification of the truth table. That is, you apply all the possible input combinations and verify that the outputs are as predicted by the truth table. In practice, as the number of inputs grows, it becomes impractical to test the circuit exhaustively. There is a whole branch of electrical engineering devoted to generating tests for circuits that will test the circuit without trying every possible input combination. There are also schemes for adding things to the circuit to make it easier to test. For this class, you will have to think carefully about the right set of tests to apply to your circuits to test them adequately.

9.3 ViewSim Commands

The commands you give the simulator are simply commands to force the inputs of your circuit to 1 or 0. You do this by referring to the inputs using the labels you put in your schematic. You can use the menus in the ViewSim window, or type commands into the text part of the ViewSim window, or put commands in a command file and execute that. The command set for ViewSim is
described in great detail in the on-line documentation (Use Help→Viewdoc to start up the on-line documentation). The most important ones are:

**h node node ...**  – Force the nodes in the node list to 1

**l node node ...**  – Force the nodes in the node list to 0

**assign node value**  – Assign a value to a node, 1, 0, or X. This command also lets you assign a value to a bus by first defining a vector, and then assigning a value to that vector (see below).

**sim, or sim time**  – Apply the inputs that you specified using the h and l commands. “Sim” will run the simulation for a fixed amount of time (Default is 100ns. Set it to something else using the “stepsize” command). Note that the units in the “stepsize” command are actually 10ths of nanoseconds so if you want to set the sim time to 500ns you need to say “stepsize 5000.” “Sim time” will run for ”time” nanoseconds. An important note is that the h and l commands tell which nets to set to 1 and 0, but those settings don’t take effect until you let simulation time pass with a “sim” command.

**watch node node ...**  – The value of the named nodes will be reported at the end of each simulation step.

**check node value**  – checks to see if a node is at a particular value and reports an error if it is not. This is very helpful when building simulation command files. You can run the command file and see if errors are generated using the “check” command. Good command files have lots of check statements!

**execute cmdfile**  – Applies the ViewSim commands in the file “cmdfile” to the circuit. If you name your command file file.cmd then you can execute it simply by typing file at the viewsim prompt.

**logfile filename**  – Set up a log file to record your simulation activity.

**echo message**  – Prints the message to the screen (and logfile). Useful for annotating your simulations with comments that will appear as you run the simulation.

You can use vectors to collect signals together where appropriate. You could, for example, collect three input signals to a random cell into a vector and then assign them all at once. The syntax is:

```v vecname node node node ...
```

Once the vector is defined, you can refer to all the bits by using the vector name. For example, if you defined a vector for a 4-bit adder to an input called ain, the syntax would be:

```v ain A3 A2 A1 A0
```

Note that the A input to the adder is probably a bus labeled A[3:0] so you could also have used that shortcut in the “v” command:

```v ain A[3:0]
```

In any case, now you can refer to the four input bits as the single vector called ain. You can set this vector using the “assign” command. For example,

```a ain 0011```
will set the value of the ain vector to 3 (0011 in binary). That is, A3 = 0, A2 = 0, A1 = 1, and A0 = 1.

There are many more commands in ViewSim, and you will use some of them in future labs, but these are the most important.

So, assume that we have opened a ViewDraw window with the HA circuit, and a ViewSim window using HA.vsm as the simulation file. You could say, for example:

```plaintext
watch X Y SUM CARRY
h X Y
sim
```

The result from ViewSim would be:

```plaintext
time = 100.0ns  X=1  Y=1  SUM=0  CARRY=1
Simulation stopped at 100.0ns.
```

At this point you can see the values updated in the ViewDraw schematic window as 1’s and 0’s attached to nodes in the schematic.

You can also put your commands in a command file so that you can apply the entire sequence of commands at once, and apply them again if you change the circuit. For example, a command file to simulate your half adder might look like the following:

```plaintext
restart | reset simulation time to 0
logfile ha.log | open a log file for results

echo Set up a watch list for the inputs and outputs
watch X Y SUM CARRY

echo Drive the inputs and check the outputs
h X Y | Set a and b high
sim | simulate to propagate values
check SUM 0 | check to see if result is 0
check CARRY 1 | check to see if result is 1

1 X | raise c to 1
sim | simulate to propagate results
check SUM 1 | check and see if result is 1
check CARRY 0 | check and see if result is 0

1 Y | lower b to 0
sim | simulate to propagate results
check SUM 0 | check and see if results is 0
check CARRY 0 | check and see if results is 0

h X | lower b to 0
sim | simulate to propagate results
check SUM 1 | check and see if results is 1
check CARRY 1 | check and see if results is 0

echo simple check done
log | close the logfile
```

Notice that the command file sets up a logfile to record the results of the simulation, then sets up a list of which inputs and outputs you wish to see at the end of each simulation cycle (the “—” is the comment character for ViewSim). It also checks to see that the output is at the correct state at the end of each cycle. In this case, the HA.log file would contain the following:
SIM>execute HA.cmd
Simulation time rolled back to 0.0ns.
Closing current logfile (fsimtemp.log) before opening ha.log.
Set up a watch list for the inputs and outputs
Drive the inputs and check the outputs
time =  100.0ns  X=1 Y=1 SUM=0 CARRY=1
Simulation stopped at 100.0ns.
time =  200.0ns  X=0 Y=1 SUM=1 CARRY=0
Simulation stopped at 200.0ns.
time =  300.0ns  X=0 Y=0 SUM=0 CARRY=0
Simulation stopped at 300.0ns.
time =  400.0ns  X=1 Y=0 SUM=1 CARRY=0
Simulation stopped at 400.0ns.
Value '0' on CARRY does not
match '1' at time 400.0ns.
simple check done
ha.log closed.
SIM>

Notice that the last check command reports an error. It says that the value of 0 that the
simulation saw at the CARRY output is not what the check command said to expect. In this case,
however, it is the check command that is incorrect and not the circuit. If you are sure that your
check command is looking for the correct circuit outputs this would alert you to a problem with
the circuit that you need to fix. The ability to write command files with statements that will alert
you when the outputs do not correspond to what you expect is one of the most powerful features
of ViewSim. Command files that you write should always make heavy use of “check” statements!

When you close the ViewSim window, the logfile is written, and the values in the schematic
will go away. (Actually, this is not entirely true. The values will stay there until you make them
go away by redrawing the screen. Strangely, refreshing the screen (using view→refresh or F5) does
not work. One easy way to get rid of the annotations is to push into another schematic, then pop
back)

9.4 Simulation part of your lab

At this point, you should simulate both your half adder and your full adder which you built out of
half adders. You should turn in a copy of your command files and the resulting log files.

10 Quitting

To quit Powerview you will need to go to the menu again and select "Quit Powerview" from the
red square menu.

NOTE: You MUST quit Powerview using this menu selection! DO NOT quit using the window
system. That is, in the title bar of the X window there is an option to quit the application running
in that window. DO NOT quit using this X-window menu! Doing so may lock up a license token
and prevent others from using Powerview!

11 Laboratory Summary

In this laboratory, we have seen how to enter schematics, create symbols, and use a logic simulator
to verify the behavior of a logic circuit. This is just the beginning of your exposure to CAD tools.
You will be using PowerView throughout the semester to design and simulate circuits.
12 Summary Sheet

Name:
Discussion Section:
TA:

1. The Half Adder (15 pts).

   (a) Fill in the Half Adder Truth Table:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>SUM</th>
<th>CARRY</th>
</tr>
</thead>
</table>

   (b) Draw the schematic of the half adder circuit, using only AND, OR, and NOT gates.
2. The **Full Adder (25 pts)**.

(a) Fill in the Full Adder Truth Table:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Carry-In</th>
<th>SUM</th>
<th>Carry-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Draw a schematic that implements this function directly, using only AND, OR, and NOT gates.
(c) Draw another schematic, this time expressing the full adder in terms of wired together half-adders and any other logic gates you may need:
3. **Using Powerview to Capture and Simulate the Half Adder (40 pts).**

(a) Print out your schematic and attach it to this laboratory write up.
(b) Attach copy of your command and log files from simulation.
4. The **Full Adder** (20 pts).

   (a) Attach a print out of the full adder schematic to the summary sheet.
   (b) Attach copy of your command and log files from simulation.