UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC180B DIGITAL SYSTEMS II Spring 1999

Lab 1: Powerview Tutorial

Objective: The purpose of this lab is to introduce the Viewlogic Powerview software which will be used throughout the quarter, namely, ViewDraw, ViewSim and ViewTrace. In this tutorial, you will go through the design entry and simulation of a FIFO (First In First Out) buffer implemented in Xilinx library components.

Throughout this tutorial, the following conventions will be used. When the student is directed to type a command from the keyboard, the command will be displayed in a bold-face letters. For example, to start ViewDraw, the student should type **viewdraw &** at the UNIX prompt. When the student is directed to select items from the pull-down menus, the menu selections will be in italicized font, such as *File Level Push Schematic.*

I. Setting up the environment

You must first set up some environment variables by typing the following commands:

```
setup  powerview60

setup xilinx
```

For the changes to take effect, you need to open a new window or log-out and then log-in again. You do not need to run this setup script again. Now, create a directory for your Powerview project and make it your present working directory by typing the following commands:

```
mkdir  eec180b
cd  eec180b
mkdir lab1
cd lab1
```

Next you will copy a viewdraw.ini file into your working directory. (It must be in the directory from which you start Powerview programs). Note that the period at the end of the command indicates your current directory.

```
cp /afs/ece/classes/eec180b/lab1/viewdraw.ini  .
```

The viewdraw.ini file specifies which component libraries you will use in your design. In our case, we will be using Xilinx libraries so that we can target our design to a Xilinx FPGA device. Thus, the only libraries that should be in your viewdraw.ini file are Xilinx libraries.

Online Documentation

Virtually all of the Powerview documentation is available on-line through the ViewDoc utility. You can access this directly from the command line by typing:

```
online &
```

From the pull-down menus within the Powerview tools, you can also select with the mouse the Red box in the upper left corner, *Help* and *ViewDoc*. To view the table of contents of one of the manuals listed in the Main Menu, double click on the icon next to the manual name.

If the text is too small, you can change the default zoom by selecting *View Preferences* and setting the Default Zoom.

The Xilinx documentation is also available on-line through the command:

```
dtext &
```

The Viewlogic Interface/Tutorial Guide manual is extremely useful for our purposes. Also, the Libraries Guide documents all of the design elements in the Xilinx libraries. You will need to consult this manual in order to choose specific components to use in your design. You should also consult the Libraries Guide to learn how the components used in this tutorial function. For example, you will need to understand how an X74_157 or a CB4CLE component functions in order to understand the FIFO design.

II. Schematic and Symbol Entry in ViewDraw

In order to enter the FIFO design, you will use Viewlogic's ViewDraw schematic capture program. There are two ways in which you can run this program. You could use the Powerview cockpit which allows you to select individual tools using the mouse or you could directly start the ViewDraw tool. This tutorial will illustrate running the individual tools directly. Run the Powerview ViewDraw program by typing:

```
viewdraw ffblk &
```

This will bring up a blank schematic named ffblk.

Adding Components

Add two fd components and a bsheet component from the xc4000 library as follows:

1. Using the mouse, select *Add Comp* or type **i**.

This should bring up the Add Component dialog box that will list the various directories from which you can select components. Assuming your viewdraw.ini file is set up properly, you should see the directories "Symbols in memory", your primary directory, and the several Xilinx libraries.

2. Click the left mouse button on the "Enter name:" prompt.
Select a single D flip-flop by typing: **fd**

Note that you don't need to know which directory the fd component is in. However, you should have one of the top directories selected in the Add Component dialog box. ViewDraw will search in the selected directory and all directories below it in descending order.

3. Click the left mouse button on your schematic page and place the component using the middle mouse button.

Note that by clicking the left mouse button on your schematic page you can toggle the view of the symbol between an empty box and the actual symbol. When you have the component in the desired position, click the middle mouse button to fix the component's placement. If you need to re-position a component, you can select it with the left mouse button and then select *Move* from the *Edit* pull-down menu or use the short-cut command **m**.

4. Now add a second fd component. Since the Add Component dialog box is still open you can repeat steps 2 and 3. Place the second fd component near the first one. (Another way is to just type **i** again.)

5. Add a bsheet component by typing **bsheet** on the "Enter name:" line.
Place the bsheet so that it just fits inside the white border. You can use the box at the lower right to document your design. To add text, select *Add Graphics Text* from the menus or use the short-cut keyboard command: **t**.

6. Close the Add Component dialog box by selecting *Cancel* with the left mouse button.

<u>Useful Function Keys</u>

The following Function Keys are useful for displaying a particular region of your schematic.

F5	- refresh display
F6	- pan (center display around cursor)
F7	- zoom in
F8	- zoom out
F9	- zoom in on area
F10	- show full schematic

<u>Adding Nets</u>

Now you will add nets as shown in Figure 1.

1. Position the cursor on a pin where you want to start a net.
Type **n** at the keyboard to start the net.

2. Move the cursor to draw the net. You can anchor a right-angle in the net by clicking the middle mouse button where you desire the corner.

3. End the net by clicking the middle mouse button on the destination pin. To create a dangling net, click the right mouse button immediately after clicking the middle button. This will create a small square at the end of the net.

Adding Labels

The nets should be labeled as shown in Figure 1.

1. Select a net by using the left mouse button.

2. From the menus, select *Add Label*. At the Label String prompt, type in the label name.
The options should be Visible, Local and Noninverted.

3. Place the label on the schematic using the middle mouse button to anchor it.

4. After the first label has been added, you can bring up the Add Label dialog box using the middle mouse button. First, select a net and then click the middle mouse button.

Saving your File

To save your design, select *File Write* from the menus. This will save your schematic in the sch subdirectory and it will also generate a wirelist file that will be stored in the wir subdirectory. You should see a message regarding errors and warnings. If you didn’t get the message “0 Errors and 0 Warnings”, then go back and fix the problems.

Creating a Symbol for your Circuit

The circuit you just entered will be a small part of a larger, hierarchical design. You will create a symbol for this circuit so that you can use it easily. Symbols, as well as schematics, are created using ViewDraw.

1. Select *File Open* from the ViewDraw menu and select *Symbols* from the File Open dialog box.

2. Click the left mouse button on the “Enter name:” prompt and type **ffblk**. Note that the symbol must have the same name as the schematic if the symbol is to represent that schematic.

Once you open your symbol window, you will see an empty white box on your display with a default size of 1” x 1” with a grid size of 10 dots per inch. (See the text-bar at the top of the window for the dimensions).

3. To change the size of the symbol, select *Change Block Size Z-WxH* and enter the new dimensions at the prompt. For example,

Block width **120**	| 1.2 inches wide
Block height **100**	| 1.0 inches high

4. Select *Add Graphics Box*. Use the mouse to draw a box inside of the white box, where the sides of the inner box are 1 or 2 grid spaces inside the white box on all sides. Move the cursor to the upper left corner of your desired box, click the middle button to start the box, move the cursor to the lower right corner and click the middle button again to end the box.

5. To add pins to your symbol, select *Add Pin*. Move the cursor to an edge of your inner box and double click the middle mouse button. Add four pins as shown in Fig. 2.

6. To label the pins, first select a pin by clicking on it with the left button and then select *Add Label*. Label the pins in the same manner as you labeled nets. You must use the same names for the pins as for the nets in the schematic that they represent. (See Fig. 2).

7. Select *Add Graphics Text* from the menus and label your symbol. An appropriate label is the name of the symbol (and schematic) file so that the component on a schematic identifies the symbol name. In Fig. 2, the symbol is labeled FFBLK.

8. Finally, save your symbol with *File Write*. The block type should be Composite, meaning that the block represents an underlying schematic. After selecting your symbol with the mouse, you can select *File Level Push Schematic* to view the circuit that corresponds to your symbol block. You should see the circuit you created earlier. Select *File Level Pop* to return to the symbol and then close the symbol window by selecting the Red box and then *Dismiss Window*.

<u>FIFO Control Schematic Entry</u>

Now that you have the basics down, you will finish entering the FIFO design.

1. Open a new schematic file named **control**.

2. Add the following components as shown in Fig. 3.

bsheet	
and2	(2)
and3	(2)

ffblk (2)
cb4cle (2)
cb4cled
x74_157
or2
or4
inv

3. Add nets, buses and labels as shown in Fig. 3.

Shortcuts and Hints

To add a number of dangling nets to a component, draw one net and then use the copy command (**c**) to add the remaining nets. i.e. select the net, type **c**, move the net to the desired pin using the mouse, and attach the net to the pin by clicking the middle mouse button.

To add a series of labels, i.e. RP0 - RP2, select the net to be labeled RP0 and enter **RP[0:2]** as the net name. This net will be labeled RP0. Then select the next net and click the middle mouse button. The net will automatically be labeled in consecutive order.

Buses must be labeled using square brackets to show the index numbers. Nets coming off a bus must have the bus name and an index number.

In order to label a net active low, such as Empty or Full, you can devise a naming convention. For example, you could add _n to the end of active low signal names. However, you should avoid using characters such as '\', '/', or '*' since these symbols have special meanings to the Xilinx or Viewlogic tools. Also avoid clicking on the "Inverted" box in the Add Label dialog box since this adds a symbol which has special meaning to Xilinx tools.

Use PWR and GND components to tie pins high or low as shown in Fig. 3.

<u>Be sure to save your schematic periodically in case of some kind of system failure.</u>

FIFO Control Symbol Entry

Enter a symbol for the control schematic which you have just entered. One way to do this is to select *File Level Push Symbol* from the control schematic. Since the control symbol does not exist, a blank page will be displayed where you can create the symbol.

The symbol should be defined as in Figure 4.

The block size can be set to Width 150 (1.5”) x Height 180 (1.8”).

When adding labels for buses such as A[3:0], do not expand the label.

Remember that pin names must be the same as the net name in the schematic.

Top Level Schematic Entry

Finish the schematic entry by creating a schematic called fifo. Complete the schematic as shown in Fig. 5.

Note that the RAM component must be specified in lower case, i.e. ram16x4.

Once you have saved your schematic, exit ViewDraw.

Printing a Schematic

To print a schematic or symbol, select the Red Box and *Plot Setup* from the menus. Select the following options:

Device - **POSTSCRIPT**
Papersize - **A**
Output - **FILE**
Extents - **FULL SHEET**
Output > **filename.001p**

You can accept the default output file name or specify a different name if you prefer. Click on *OK* once the options are set. This will generate a postscript file in your directory which you can print using the standard UNIX print command lpr or lp.

III. Simulation in ViewSim

Before simulating your design, you must process your design to create a ViewSim netlist. In this tutorial, we will introduce a simple way to process your design in order to perform a functional simulation. In the next lab, we will demonstrate how to process a design in order to do a timing simulation.

First, run the check program to update your WIR files and verify that your design does not have errors. At the UNIX prompt, type

```
check -p fifo
```

Next, to generate the ViewSim netlist, type

```
vsm fifo
```

Now run the ViewSim program by typing

```
viewsim –vsm fifo &
```

You will also want to view the back-annotated values on your schematic. This can be done by opening a ViewDraw window for the fifo schematic.

```
viewdraw fifo &
```

You should see an X on each net of your fifo schematic, indicating that its value is undefined.

Asserting Global Set Reset (XC4000 family only)

The Global Set Reset (gsr) signal is an active high signal which initializes every flip-flop in the Xilinx XC4000e family FPGA. The gsr signal should be set high for 100 ns at the beginning of each simulation. In the next tutorial, we will illustrate using the Startup component to drive the gsr net in hardware.

An Example Simulation Command File

You can type simulation commands directly at the ViewSim prompt and observe the results on your schematic or on a waveform display. This is useful for debugging a problem. However, this method is tedious for long simulations and many users prefer to use a command file instead. By looking at an example command file, you can learn the basic ViewSim commands that can be used interactively or in a command file.

Copy the fifo simulation command file into your working directory.

```
cp /afs/ece/classes/eec180b/lab1/fifo.cmd  .
```

This command file is also included in this handout.

The basic commands for simulation are as follows (not all of these are used in fifo.cmd):

```
restart                  | Roll back the simulation time to 0
stepsize                 | Show the default step size
stepsize 100ns           | Set the step size to arbitrary value of 100ns
v D d[3:0]               | Define a vector D consisting of the bus d[3:0]
w  rd wr empty ...       | Watch signals of interest in ViewSim
                         | (display their value after each simulation event)
wave fifo.wfm rst clk    | Select nets and vectors to display in ViewTrace
clock clk 0 1            | Define clk as a clock with a 50% duty cycle
sim                      | Simulate until all signals are stable
sim 1000                 | Simulate for 100.0 ns
c                        | Perform one complete clock cycle of simulation
h gsr                    | Set the net gsr high
c                        | Simulate for 1 clock cycle
```

```
l gsr                          | Set the net gsr low
c 3                            | Simulate for 3 clock cycles
a D e\h                        | assign (force) the d[3:0] nets to 1110
```

Run the simulation command file by typing **fifo.cmd** in the ViewSim window. Verify that the fifo works as expected. Your ViewTrace waveforms should look similar to Fig. 6.

Note that you will see a MEMORY HAZARD warning as your simulation starts. This is because the data in the RAM is unknown until you write into it.

IV. FIFO Implementation Details

The FIFO is implemented as a circular buffer using RAM and read and write pointers. The FIFO buffer is 8 words deep and 4 bits wide. RAM addresses 0 through 7 are used for the buffer. An up-down counter is used to keep track of how many valid data words are in the counter.

To write into the FIFO, data is placed on the data bus D[3:0] and the WREQ signal is pulsed. To read a value from the FIFO, the RREQ signal is pulsed. Note that the WREQ and RREQ signals should never be high at the same time. Also, we have assumed that the WREQ and RREQ pulses will last for several clock cycles. This is a valid assumption because these signals will be driven from switches in the final implementation.

V. Lab Report and Questions

1. Generate a printout of your schematic diagrams and your waveform output.

2. Add a global reset somewhere in the middle of the simulation command file. Verify that the FIFO is reset as you expect.

3. Modify the design to make the FIFO 15 words deep. Turn in a schematic of the modified control unit entered in ViewDraw, a simulation command file, and the ViewTrace waveform output.

fifo.cmd

```
restart
v A a[3:0]
v C c[3:0]
v OUT out[3:0]
v D d[3:0]
w gsr rd wr empty_n full_n A C D OUT
wave fifo.wfm clk gsr rreq wreq rd wr empty_n full_n A C D OUT
l  rreq wreq            | initialize inputs
a D 0
clock clk 0 1           | set up the clk as clock net
h gsr
echo                    -- assert Global Set Reset signal
c
l gsr
echo                    -- de-assert Global Set Reset signal
c 4
echo                    -- read request (empty buffer)
h rreq
c 4
l rreq                  | set read request low
c 4
a D e\h
c 4
echo                    -- write E to FIFO (C -> 1)
h wreq
c 4
l wreq
c 4
echo                    -- read E from FIFO (C -> 0)
h rreq
c 4
l rreq
c 4
a D 5\h
c
echo                    -- write 5 to FIFO (C -> 1)
h wreq
c 4
l wreq
c 4
a D 1
c
echo                    -- write 1 to FIFO (C -> 2)
h wreq
c 4
l wreq
```

```
c 4
a D a\h
c
echo                    -- write A to FIFO (C -> 3)
h wreq
c 4
l wreq
c 4
a D 0
c
echo                    -- write 0 to FIFO (C -> 4)
h wreq
c 4
l wreq
c 4
a D f\h
c
echo                    -- write F to FIFO (C -> 5)
h wreq
c 4
l wreq
c 4
a D 3\h
c
echo                    -- write 3 to FIFO (C -> 6)
h wreq
c 4
l wreq
c 4
a D c\h
c
echo                    -- write C to FIFO (C -> 7)
h wreq
c 4
l wreq
c 4
a D 7\h
c
echo                    -- write 7 to FIFO (C -> 8)
h wreq
c 4
l wreq
c 4
a D 8\h
c
echo                    -- write attempt (full)
h wreq
c 4
```

```
l wreq
c 4
a D 9\h
c
echo                    -- read (C -> 7)
h rreq
c 4
l rreq
c 4
echo                    -- read (C -> 6)
h rreq
c 4
l rreq
c 4
echo                    -- read (C -> 5)
h rreq
c 4
l rreq
c 4
echo                    -- write 9 to FIFO (C -> 6)
h wreq
c 4
l wreq
c 4
echo                    -- read (C -> 5)
h rreq
c 4
l rreq
c 4
echo                    -- read (C -> 4)
h rreq
c 4
l rreq
c 4
echo                    -- read (C -> 3)
h rreq
c 4
l rreq
c 4
echo                    -- read (C -> 2)
h rreq
c 4
l rreq
c 4
echo                    -- read (C -> 1)
h rreq
c 4
l rreq
```

```
c 4
echo                    -- read (C -> 0)
h rreq
c 4
l rreq
c 4
echo                    -- read attempt (empty)
h rreq
c 4
l rreq
c 4
```