UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC180B **DIGITAL SYSTEMS II** **Spring 1999**

Lab 2: Xilinx Tutorial

Objective: The purpose of this lab is to introduce the Xilinx software which will be used throughout the quarter. In this tutorial, you will complete the design entry and simulation of a FIFO (First In First Out) buffer implemented in Xilinx library components. You will then test the design by downloading the bit file to a Xilinx XC4000 demo board.

I. Setting up the environment

You must first set up some environment variables. If you have not run the setup script for the Xilinx CAD tools before, you should type the command:

setup xilinx

In order for the changes to take effect, you should open a new window. You only need to run this script once in order to modify your login files.

II. Completing the FIFO design entry

You must have completed the Powerview tutorial, Lab Exercise 1, before attempting this tutorial. You will complete the FIFO design started in Lab 1.

Hex to Seven Segment Converter (Symbol and Schematic)

In order to display data values on the Xilinx XC4000 demo board, we will need a module which converts a 4-bit hex value into signals for driving a seven-segment display. We will demonstrate the use of Xilinx ROM components to implement a look-up table.

Create a symbol for your hex to seven segment converter as shown in Fig. 1. In our example, we named the symbol hex_7seg, although you can choose another name if you prefer. Since the outputs which drive the seven segment display are active low, we have changed the pin sense on the output pins. After selecting a pin, you can select *Change Pin Sense* from the menus with the mouse in order to toggle the polarity of a pin. Note that the pin polarity on a symbol only serves as documentation; it does not change the function of the underlying schematic.

Create a schematic with the same name as the symbol you just created. An easy way to do this is to select *File Level Push Schematic* from the ViewDraw pull-down menus. Add

a bsheet component and seven ROM16X1 components as shown in Fig. 2. Label the nets as shown in Fig. 2. Once you select a net, you can type **l** at the keyboard to bring up the label dialog box. Select each ROM and then choose *Add Attr...* from the menus. Type in the INIT value shown for each ROM component (i.e. INIT=2812). Also label each ROM component as ROM0, ROM1, ... ROM6 by selecting the ROM and then choosing *Add Label* from the menus.

You should understand how the INIT values for the ROMs have been determined. The following table shows the desired conversion from hex to seven-segment drivers. (NOTE: Output A drives the top horizontal bar of the seven-segment display; Outputs B, C, D, E, and F follow sequentially in clockwise order and output G drives the central horizontal bar of the display).

I[3:0]	A	B	C	D	E	F	G
0000	0	0	0	0	0	0	1
0001	1	0	0	1	1	1	1
0010	0	0	1	0	0	1	0
0011	0	0	0	0	1	1	0
0100	1	0	0	1	1	0	0
0101	0	1	0	0	1	0	0
0110	0	1	0	0	0	0	0
0111	0	0	0	1	1	1	1
1000	0	0	0	0	0	0	0
1001	0	0	0	1	1	0	0
1010	0	0	0	1	0	0	0
1011	1	1	0	0	0	0	0
1100	0	1	1	0	0	0	1
1101	1	0	0	0	0	1	0
1110	0	1	1	0	0	0	0
1111	0	1	1	1	0	0	0

The INIT value specifies the ROM contents from the highest addressed location (1111) to the lowest addressed location (0000). Thus, from the table above, we can verify that the INIT value for ROM0 (output A) should be 0010 1000 0001 0010 or 2812 hex. Similarly, the other INIT values can be read from the table and are shown on Fig. 2.

Once the hex_7seg symbol and schematic have been completed, add them to the top-level FIFO schematic as shown in Fig. 3. Select each hex_7seg component and attach a label to it, either CR3_ROM or CR4_ROM. These labels will be useful for specifying initial simulation values.

STARTUP Component

By using the STARTUP component, you can control the Global Set/Reset (GSR) signal from an external switch. When the GSR input is high, every flip-flop in the device will be set or reset, depending on the initialization state (S or R) of the flip-flop. The GSR input will be driven by an external Reset signal, which is normally high. Therefore, an inverter is used between the IBUF (input buffer) and the STARTUP component, as shown in Fig. 3.

OSC4 Component

The OSC4 component provides clock signals which can be used inside the Xilinx XC4000 family FPGAs. Although there are five available frequencies, only three can be used at one time: the 8M signal and any two of the other four outputs. The OSC4 output frequencies are approximate and are not intended for timing-critical applications.

IPAD, IBUF, OPAD, OBUF Components

In this design, each input signal will enter the FPGA through an IPAD. (IOPADs are also available for bidirectional signals.) The input signal can then be either buffered or latched. In this design, each input will be buffered by an IBUF component. Output signals are buffered through an OBUF component and then go off-chip via an OPAD component. Since we are using XC4000 demo boards which are pre-wired, we will need to specify the pad locations for the inputs and the outputs. The location attributes can be included on the schematic or in a constraint file. We will demonstrate the use of a constraint file for specifying pad locations. Use the same net names as shown in Fig. 3 so that the constraint file can be applied to your design without needing changes. **Note**: the Empty_N and Full_N labels have been deleted from their previous location and now label the nets connected to the OPADs.

Complete the top level schematic as shown in Fig. 3.

III. Functional Simulation

Since our schematic contains ROM components which depend on the assigned INIT attribute, we need to go through some additional steps to prepare for functional simulation. The on-line Viewlogic Interface/Tutorial Guide describes the procedure for preparing for functional and timing simulations and is a useful reference. We will summarize the procedures given in that on-line document.

edifneto –l xilinx fifo - generate an EDIF file

ngdbuild –p xc4005epc84-3 fifo.edn - create a flattened netlist

ngd2edif –v viewlog fifo.ngd func_sim.edn - create a new EDIF file

edifneti func_sim.edn - produce WIR files from EDIF netlist

vsm func_sim - create a VSM file for use by ViewSim

viewsim –vsm func_sim & - functionally simulate design

A func_sim.xmm file should have been created by one of the programs listed above. You will execute this file in ViewSim in order to load the data into the ROM components. You can then simulate the design using a command file. We have written a small fragment of a simulation file called sfifo.cmd which is available in the /afs/ece/classes/eec180b/lab2 directory and should be copied into your directory. Thus, the commands to enter in ViewSim are as follows:

func_sim.xmm

sfifo.cmd

Verify that your FIFO is working as expected. An up-counter is used for entering the data. The INC signal is debounced to generate the clock signal to increment the counter. Similarly, READ and WRITE are debounced signals which drive the RREQ and WREQ signals, respectively. Note that in the simulation command file the DEBOUNCE clock signal has a much longer period than the CLK signal to simulate the frequencies which will be generated by the OSC4 component. (The STARTUP and OSC4 components cannot be simulated directly.)

IV. Timing Simulation

You can get an estimate of the timing characteristics of your design through a timing simulation, which uses worst-case routing and block delay values in the simulation netlist. Although a timing simulation provides a more accurate picture of the design's performance, it is more time-consuming to process the files than for a functional simulation. Therefore, you should perform a functional simulation to verify your design before attempting a timing simulation. Before you can do a timing simulation, you must route your design using the Xilinx Design Manager.

Preparing for Timing Simulation and Generating a Routed Design

Start the Xilinx Design Manager using the following command:

dsgnmgr &

Create a new project by selecting *New Project…* from the *File* pull-down menu. Click on the *Browse* button and select the fifo.edn file. Then click on the *OK* button to create a project directory.

Next select *Design Implement…* from the pull-down menus. Select part XC4005E-3-PC84 as the Xilinx part by clicking on the *Select…* button and filling in the following information:

Family:	XC4000E
Device:	XC4005E
Package:	PC84
Speed Grade:	-3

In the *Design Implement…* dialog box, click on the *Options…* box. There are three different option settings that must be set, as described below.

1. We need to specify the PAD locations of the input and output signals so that our design will work on the Xilinx demo board. The constraint file, fifo.ucf, is available in the class afs directory and should be copied into your project directory. Then click the *Browse…* button in order to fill in the *User Constraints:* box with the fifo.ucf file.

2. In order to produce an annotated NGD file, you must select the Optional Target *Produce Timing Simulation Data* in the dialog box.

3. To create a new EDIF file, select *Viewsim EDIF* from the Simulation Program Option Templates. You can click on *Edit Template…* to verify the Simulation Data Option is set to EDIF and the CAE vendor under the EDIF tab is set to ViewLogic.

Once the options have been entered, click the *OK* button and then click on *Run*.

When the Design Manager has finished processing the design, it will have produced a bit file which can be downloaded to the Xilinx board as well as the timing simulation data. To finish preparing for timing simulation, enter the following commands:

edifneti time_sim.edn - produce WIR files from EDIF netlist

vsm time_sim - create a VSM file for use by ViewSim

To simulate your design, enter the same commands given below.

viewsim –vsm time_sim &

Within the Viewsim window, type:

time_sim.xmm

sfifo.cmd

To measure an elapsed time, move the cursor to the reference transition, hold the Ctrl key and click the middle mouse button. Then move the cursor to the transition to which you want to measure the elapsed time and again click the middle mouse button. The Info box should display the time difference between the transition and the reference transition.

V. Testing in the Xilinx Demo Board

Transferring the Bit File to a PC

To test your design in a Xilinx demo board, you will need to be able to access your bit file, fifo.bit, from one of the PCs in 2112. The easiest way to access the bit file is to log-

on to the PC, which should mount your workstation home directory as drive H:. You can use the bit file directly from the workstation drive without transferring the file to the local PC drive.

Another alternative is to transfer the bit file from the workstation to the PC using ftp. From the DOS prompt, create a directory for yourself and run ftp from within your own directory.

```
cd usr                  ! Switch to the user directory
mkdir my_name           ! Make your own directory
cd my_name              ! Change directory into your own directory
ftp workstation_name    ! Log in using user name and password
cd eec180b/lab1         ! Change directory to your project directory
bin                     ! Use binary transfer mode!
get fifo.bit            ! Transfer the file
bye                     ! Exit your FTP session
```

Downloading to the Demo Board

Connect the XChecker cable to a com port on the PC. There will be a 9-pin serial cable connected to com1 on the PC; The XChecker cable should be connected to this cable. The demo board is powered by +5V applied to the J2 connector. Check your power supply voltage and connections to ensure that you do not damage the demo board. Then turn on the power to the demo board. From the DOS prompt, type

xchecker fifo

This will use the default settings to download your fifo.bit file to the demo board.

The DIP switches labeled SW4 control the configuration of the demo board. All of the switches should be in the OPEN position except the RST switch which should be CLOSED. If M0, M1 and M2 are not in the correct position, you will not be able to download a design using the XChecker cable. Closing the RST switch will connect the RESET push-button switch to pin 56 of the Xilinx part, which has been assumed in this design. Thus, this switch must be closed for the RESET switch to affect your circuit.

Testing your Design

The WRITE and READ signals are controlled by the two left-most switches on the SW5 dipswitches. (WRITE is the end switch and READ is next to it). Your WRITE and READ switches should start in the OPEN position. The INC signal is generated from the momentary SPARE button and the RESET signal is generated from the momentary RESET button. The left-most LED is the EMPTY flag and the one next to it is the FULL flag. The four LEDs on the right-hand side will display the number of values in the FIFO. (NOTE: we have wire-wrapped the SPARE button jumper to pin 18 on the FPGA and ground. This is the only board customization that is necessary for this lab as written.)

First, press the reset button. Both seven-segment displays should show zero and the EMPTY flag should be on. By pressing the INC button, you should be able to increment the counter and see its value on the left seven-segment display. Toggle the WRITE switch to write a value into the FIFO. The count should increment to one. Then by toggling the READ switch, you should see the value displayed on the right seven-segment display. Completely test your FIFO.

When you are done testing your design, delete your bit file from the PC if you transferred the file to the local disk.

VI. Lab Report and Questions

1. Discuss the advantages and disadvantages of implementing the hex to seven segment converter in ROM rather than in combinational logic. Generate the output equations for the A and B segments based on the table presented earlier.

2. Perform a timing simulation and measure the delay between the following signals:
(i.e. The ↑ indicates a signal going from low to high and ↓ indicates a transition from high to low).

Signal Transitions	Action which caused transitions
RD ↑ EMPTY_N ↓	Emptying the FIFO by reading the last datum
WR ↑ EMPTY_N ↑	Writing a value into an empty FIFO
WR ↑ FULL_N ↓	Filling the FIFO by writing the final datum
RD ↑ FULL_N ↑	Reading from a full FIFO

You will need to modify the simulation command file in order to produce the desired EMPTY_N and FULL_N transitions. Turn in your simulation command file as well as the timing measurement results.

3. Demonstrate your working FIFO to the TA. Turn in a signed verification with your schematics.

sfifo.cmd:

```
v A a[3:0]
v C c[3:0]
v OUT out[3:0]
v D d[3:0]
v CR3 cr3_a cr3_b cr3_c cr3_d cr3_e cr3_f cr3_g
v CR4 cr4_a cr4_b cr4_c cr4_d cr4_e cr4_f cr4_g
v LED led[3:0]
w gsr rd wr empty_n full_n A C D OUT CR3 CR4 LED
wave sfifo.wfm clk debounce gsr inc read write rreq wreq rd wr empty_n full_n A C D
OUT CR3 CR4 LED
break clk 1 do (display OUT >test.out)
clock debounce 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
clock clk      0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
stepsize 50ns
l read write            | initialize inputs
h inc
h gsr                   | activate GSR
l gts                   | needed for timing simulation!
c 2
l gsr
l inc                   | increment input data (displayed on CR3)
c
h inc
c
l inc                   | increment input data to 2
c
h inc
h write                 | write 2
c
l write
c
h read                  | read 2
c
l read
c 2
```

fifo.ucf:

```
# This is a schematic constraints file for use with the
# FIFO tutorial design. The FIFO will be implemented on
# the XC4000 demo board in an 4005epc8C4-3 part.

net CR3_A     LOC=p39;
net CR3_B     LOC=p38;
net CR3_C     LOC=p36;
net CR3_D     LOC=p35;
net CR3_E     LOC=p29;
net CR3_F     LOC=p40;
net CR3_G     LOC=p44;

net CR4_A     LOC=p49;
net CR4_B     LOC=p48;
net CR4_C     LOC=p47;
net CR4_D     LOC=p46;
net CR4_E     LOC=p45;
net CR4_F     LOC=p50;
net CR4_G     LOC=p51;

net LED0      LOC=p60;
net LED1      LOC=p59;
net LED2      LOC=p58;
net LED3      LOC=p57;
net FULL_N    LOC=p62;
net EMPTY_N         LOC=P61;

net INC       LOC=p18;
net READ      LOC=p20;
net WRITE     LOC=p19;
net RESET     LOC=p56;
```