Lab #4: VHDL FSMs and Gizmo Interfacing

ECE352F: Computer Organization

Fall 1999

The goal of this lab is to develop some basics that can be used for the remaining labs. The first part will be to develop a circuit that works solely on the Xilinx board. The next part will interface the circuit to the Ultragizmo board PIT. In particular, you will:

- exercise your skills in writing state machines in VHDL and control a simple data path.
- interface a simple circuit to the Ultragizmo PIT.

1 Functional Description

The circuit, to be called the *Widget 1099* will do a predefined set of computations on four 4-bit numbers that are input through a 4-bit data port. The result of the computation is available at a 4-bit output port.

You will build a circuit with the inputs and outputs as shown in Fig. 1. The circuit should be synchronous, meaning that it will look at signals based on the timing provided by the clock. The function of each pin is described below:

Clock Clock input.

$\overline{\textbf{Reset}}$ Input that asynchronously resets the circuit when set to 0.

D3–D0 (D) Data input lines, where **D3** is the MSB

Z3–Z0 (Z) Data output lines, where **Z3** is the MSB

$\overline{\textbf{Load}}$ Input set to 0 indicating that there is valid data at **D**. Data value is synchronously loaded after a falling edge on this signal. The next data value is not loaded until $\overline{\textbf{Load}}$ first goes high.

NV1–NV0 (NV) Output that shows which of the next values to put on the data (**D**) lines.

00 x_0

01 x_1

10 x_2

11 x_3

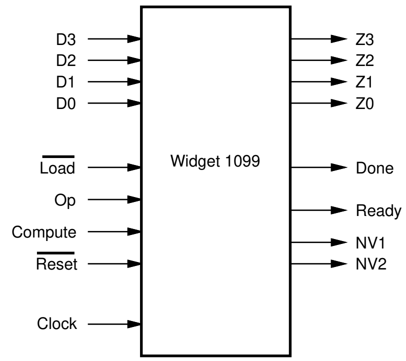

Figure 1: Input and Output Signals

Ready Output indicating that the circuit is ready to accept the next piece of input data indicated on **NV**. This signal should be low during compute cycles.

Op Input specifying the computation to perform.

Op = 0 Compute $(x_0 + x_1 + x_2 - 2x_3)/4$

Op = 1 Compute $(x_0 + 3x_1 + x_2 - x_3)/2$

Compute Input that starts the computation when set to 1.

Done Output set to 1 when a valid result is available on **Z**

The typical way to use this circuit is:

1. Reset circuit
2. Watch for **Ready**
3. Put data at **D**
4. Set $\overline{\textbf{Load}}$ low for at least one cycle and then high
5. Check that **NV** increments. This is not really necessary but it just tells you that something happened.
6. Repeat Steps 2–5 till all data input
7. Select which operation is desired on **Op**
8. Set **Compute** high and watch for **Done**.
9. When **Compute** is set low, then the cycle can begin at Step 2 again.

Signal	PIT	FPGA	7-Seg
D0	Port A 0	XCBUS07	
D1	Port A 1	XCBUS08	
D2	Port A 2	XCBUS09	
D3	Port A 3	XCBUS06	
Z0	Port A 4	XCBUS77	
Z1	Port A 5	XCBUS70	
Z2	Port A 6	XCBUS66	
Z3	Port A 7	XCBUS67	
$\overline{\text{Reset}}$	Port B 0	XCBUS68	
$\overline{\text{Load}}$	Port B 1	XCBUS19	Top
Op	Port B 2	XCBUS23	UR
Compute	Port B 3	XCBUS18	UL
Done	Port B 4	XCBUS20	Mid
NV0	Port B 5	XCBUS26	LR
NV1	Port B 6	XCBUS24	LL
Ready	Port B 7	XCBUS25	Bot

Table 1: Pin Connections

As a means of providing some standardization and inter-operability between designs, you should use the connections as shown in Table 1.

Preparation

1. Write and simulate a VHDL program that will implement the desired functionality. Provide some evidence, such as a timing diagram, that your simulation is working.

   Things to think about: You can think about the numbers as signed or unsigned integers. It is only a matter of how you interpret them. In either case, it is possible that the intermediate computations will be larger than 4 bits while the result is correct. You should take this into account as much as possible.

   Any circuit can be controlled by one large FSM, but this is a bad idea because the circuit will tend to be slower and much harder to understand. It is better to think about a number of smaller FSMs that communicate.

2. Create the UCF file defined by Table 1.

In the Lab

1. Wire up the switches and LEDs on the digital board to act as the input sources and outputs for your circuit. Note that some of the control lines are already connected to the 7-segment display.

2. Demonstrate that your circuit works in this manner.

2 Interfacing to the Ultragizmo

In this part the goal is to use the PIT on the Ultragizmo as the means for inputting data and reading the results. The required connections to the PIT are shown in Table 1.

This part will be done in two steps.

2.1 Step 1

This step will first demonstrate that you have done the wiring correctly and that you understand how to program and use the PIT. This is a great technique that takes advantage of the reprogrammability of the FPGA. In this step, the FPGA is programmed to be a set of wires so that you can test the PIT interface.

Preparation

1. Write a simple VHDL program that simply makes the following connections: **D** to **Z**, $\overline{\textbf{Reset}}$ to **Done**, $\overline{\textbf{Load}}$ to **NV0**, **Compute** to **NV1**, and **Done** to **Ready**. Just to be sure, you should simulate this too!

2. Write a C program that will configure the PIT correctly and allow you to test that data placed on the input lines of the FPGA by the PIT can be read at the output lines by the PIT.

In the Lab

1. Test that your VHDL wire program works by loading it in the FPGA and using the switches and LEDs. You should not have to do any rewiring.

2. Rewire your board to use the PIT. You can leave the LEDs connected, but you will have to remove the switches. Test your C program.

2.2 Step 2

At this point you should have demonstrated that your computing circuit works and that you can manipulate the inputs and monitor the outputs using a C program. The final step is to use a C program to request input values and display the final result.

Preparation

Write a C program that will prompt the user for input data in decimal (base 10) format and display the results in decimal format. It should also ask you which computation to perform. The program should do all of the control sequencing required.

In the Lab

Demonstrate that your C program can correctly control your computing circuit.

Version of October 12, 1999.