CMPE 260 Laboratory Exercise 1

Implementation Techniques

Benjamin Maitland
Performed: February 1st, 2018
Submitted: February 1st, 2018

Lab Section: 4
Instructor: Sayed Ashraf Mamun
TA: Piers Kwan

Lecture Section: 01
Professor: Richard Cliver

By submitting this report, you attest that you neither have given nor have received any assistance (including writing, collecting data, plotting figures, tables or graphs, or using previous student reports as a reference), and you further acknowledge that giving or receiving such assistance will result in a failing grade for this course.

Your Signature: ____________________

Table of Contents

1 Abstract

In this lab, we are introduced to using VHDL code to program a Xilinx FPGA with Xilinx's ISE tool. This is done by having us design an Arithmetic Logic Unit (ALU) circuit in VHDL, simulate it, and then test it on the physical hardware.

2 Design Methodology

For this exercise, the goal is to implement an ALU with the operations shown in Table 1.

Table 1: This is a Table title. Also do not need to number this.

Operation	ISO ALPHA 2 Code
Bit-wise or	OR
Bit-wise not	NOT
Bit-wise and	AND
Bit-wise xor	XOR
Shift Left Logical	SLL
Shift Right Logical	SRL

The first step was to visualize the shift register. A block diagram was designed in Logisim. When i = 4 and N = 8, the shift register is as shown in Figure 1.

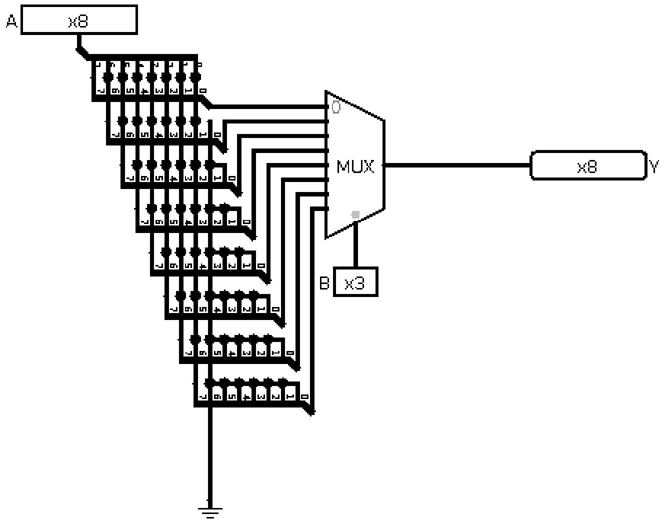

Figure 1: Left Shift with i = 4 and N = 8

After the shift register was designed, it and the rest of the operations needed to be implemented in VHDL. These were written with the text editor Vim. After the individual operations were implemented, the final ALU was implemented, and a VHDL testbench was made to test it. This testbench was not autogenerated and just gave the ALU a series of inputs. This testbench was used to simulate the ALU in GHDL, a free and open source VHDL simulator.

After the code was successfully simulated, the code was synthesized in Xilinx ISE Project Navigator 14.7. A Post-Route simulation was attempted, but even with help from TAs and instructors, the simulation was unable to run.

After synthesis, inputs and outputs were mapped to the switches on the Nexys 3 FPGA Development Board for manual testing. This was done using Plan Ahead. The first 4 switches were mapped to A, the last 4 mapped to B, and the LEDs were mapped to XOR and Logical Shift Left.

When I/O was done being mapped, ISE was used to generate a programming file. The board was then programmed using the Digilent Adept command line tools for Linux, using the .bit file generated by ISE.

3 Results

The results of the initial simulation are shown in Figure 2.

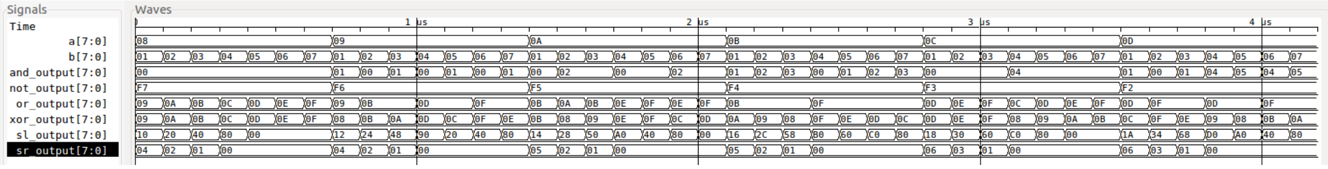
Signals
Waves
Time 0 1 us 2 us 3 us 4 us
a[7:0] 08 09 0A 0B 0C 0D
b[7:0] 01 02 03 04 05 06 07 01 02 03 04 05 06 07 01 02 03 04 05 06 07 01 02 03 04 05 06 07 01 02 03 04 05 06 07 01 02 03 04 05 06 07
and_output[7:0] 00 01 00 01 00 01 00 01 00 02 00 02 01 02 03 00 01 02 03 00 04 01 00 01 04 05 04 05
not_output[7:0] F7 F6 F5 F4 F3 F2
or_output[7:0] 09 0A 0B 0C 0D 0E 0F 09 0B 0D 0F 0B 0A 0B 0E 0F 0E 0F 0B 0F 0D 0E 0F 0C 0D 0E 0F 0D 0F 0D 0F
xor_output[7:0] 09 0A 0B 0C 0D 0E 0F 08 0B 0A 0D 0C 0F 0E 0B 08 09 0E 0F 0C 0D 0A 09 08 0F 0E 0D 0C 0D 0E 0F 08 09 0A 0B 0C 0F 0E 09 08 0B 0A
sl_output[7:0] 10 20 40 80 00 12 24 48 90 20 40 80 14 28 50 A0 40 80 00 16 2C 58 B0 60 C0 80 18 30 60 C0 80 00 1A 34 68 D0 A0 40 80
sr_output[7:0] 04 02 01 00 04 02 01 00 05 02 01 00 05 02 01 00 06 03 01 00 06 03 01 00

Figure 2: Testbench simulation

After synthesis, the total number of slices used was 7, with 11 LUTs used.

4 Conclusion

In this laboratory exercise, we learned how to write VHDL code that runs on FPGAs. This is the most likely use of VHDL in the wild, so it is important to learn it now.