CMPE 260 Laboratory Exercise 3

Arithmetic Logic Unit

Benjamin Maitland
Performed: February 15th, 2018
Submitted: February 22nd, 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 exercise, a variable-size Arithmetic Logic Unit (ALU) with 10 operations was designed in VHDL. Generics were used to change the size of the operands. The ALU contained three functional units, doing basic arithmetic, boolean logic, and shifting. Its inputs were the two operands and a four-bit control signal. The output was the result of the operation chosen by the control signal. The ALU was tested successfully with behavioral and post route simulations, and then on a physical Field Programmable Gate Array (FPGA).

2 Design Methodology

For this laboratory exercise, the goal is to implement an ALU. This ALU has two inputs and one output that are the same generic width and a 4 bit control input. A block diagram of the ALU is shown in Figure 1.

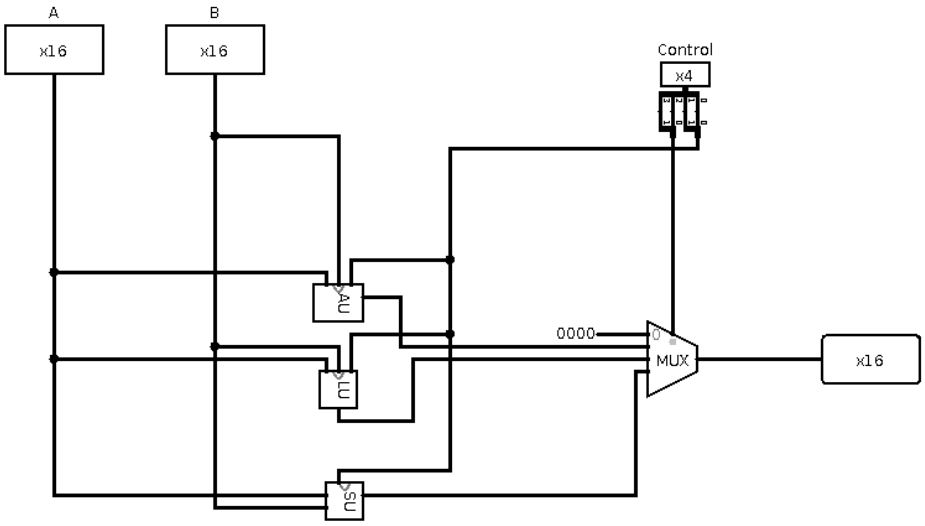

Figure 1: ALU block diagram

The ALU was composed of an arithmetic unit, a logic unit, and a shift unit. Each unit is connected to the lower two control bits as an additional input to the two operands. The upper two bits are used to choose which unit's output is the ALU's output.

An arithmetic unit is composed of a ripple carry full adder, a subtracter, and a multiplier. The correct output is chosen based on a two-bit control signal. A block diagram for the arithmetic unit is shown in Figure 2.

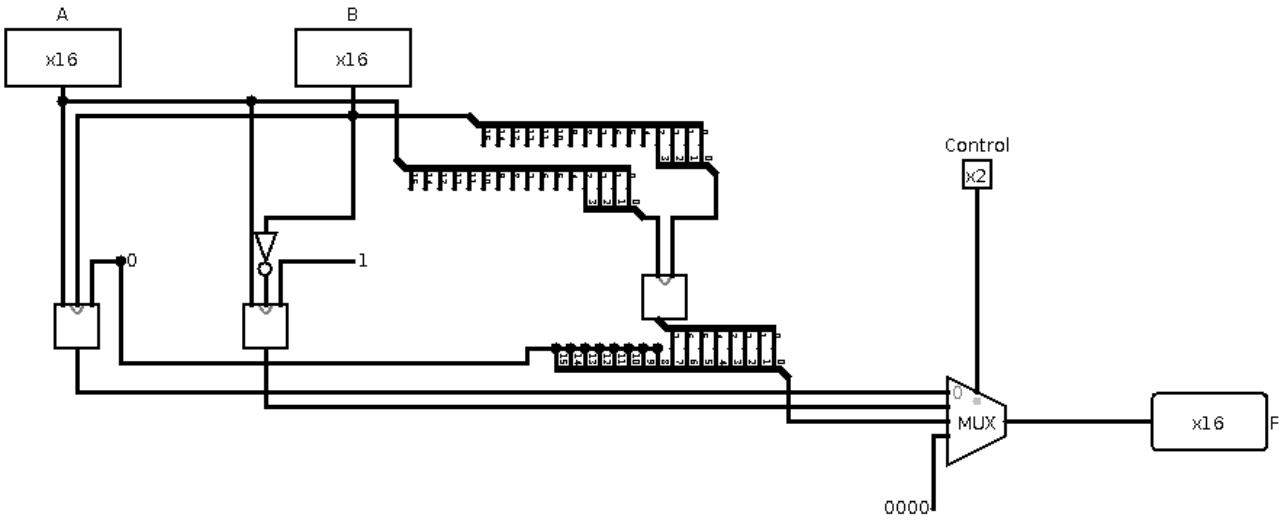

Figure 2: Arithmetic unit block diagram

A ripple carry full adder is composed of full adders, with the carry out of each full adder connected to the carry in of the next. A block diagram for the ripple carry full adder is shown in Figure 3.

A subtracter is a ripple carry full adder with one input inverted and the carry in set to one.

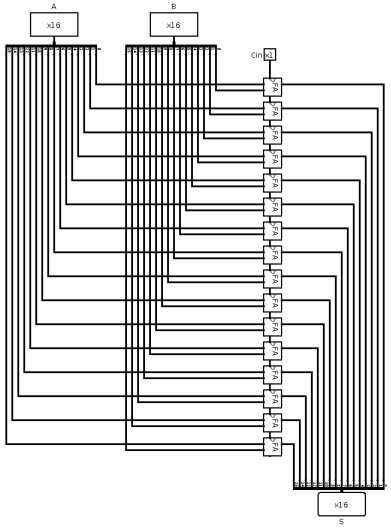

Figure 3: Ripple Carry Full Adder block diagram

A multiplier takes two n bit numbers and multiplies them, creating a 2n output. It is constructed from full adders and and gates. A block diagram for the multiplier is shown in Figure 4.

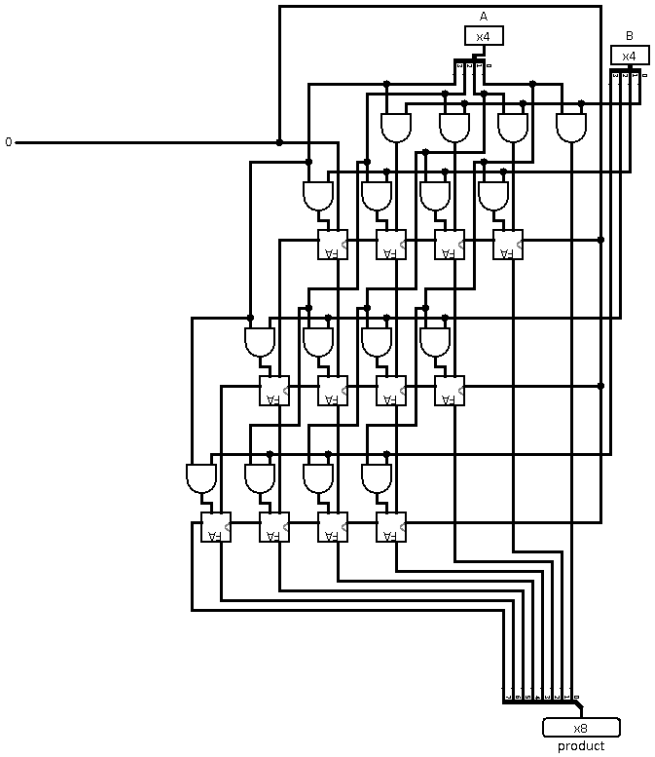

Figure 4: Multiplier block diagram

A logic unit is composed of logic blocks. A block diagram for the logic unit is shown in Figure 5.

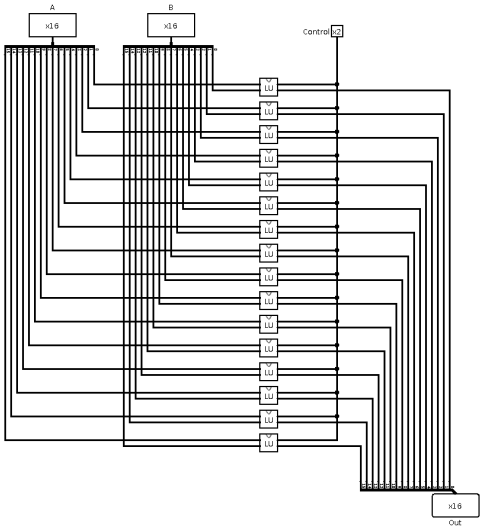

Figure 5: Logic unit block diagram

A logic block is composed of logic gates and a multiplexer to choose the correct output. A block diagram for the logic block is shown in Figure 6.

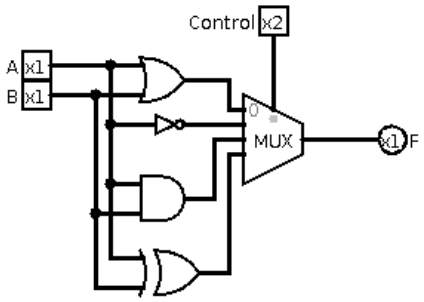

Figure 6: Logic Block block diagram

A shift unit is composed of a left shift and two right shifts. One right shift has its mode bit set to one, so that arithmetic right shift is done. A block diagram for the shift unit is shown in Figure 7.

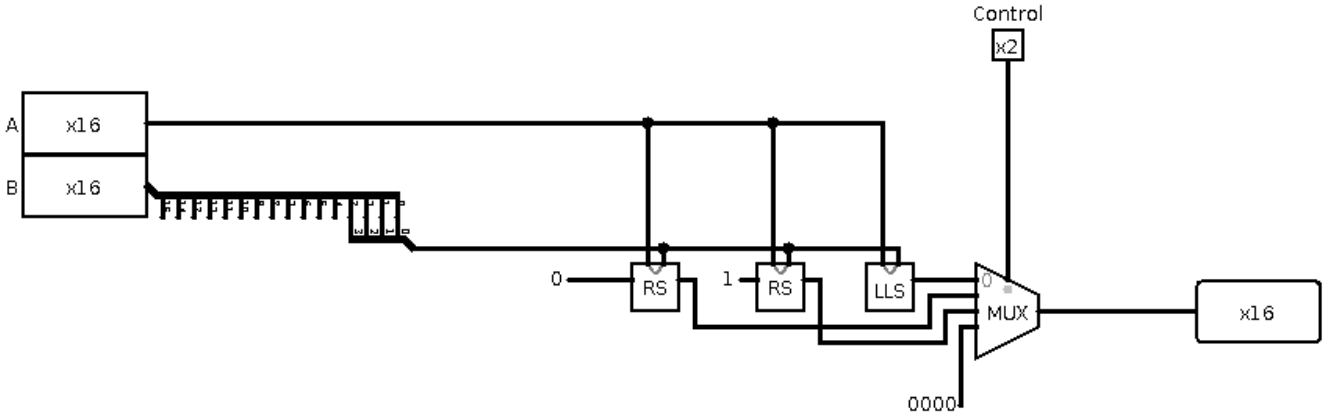

Figure 7: Shift Unit block diagram

A left shift shifts the bits of the first input left by the amount specified in the second input. A block diagram for the left shift is shown in Figure 8.

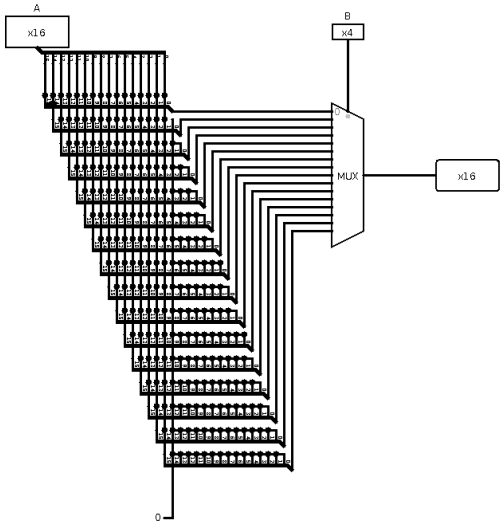

Figure 8: Left Shift block diagram

A right shift shifts the bits of the first input right by the amount specified in the second input. Depending on

the mode, it will fill the gap with zeros or ones depending on the most significant bit in the first input. A block diagram for the left shift is shown in Figure 9.

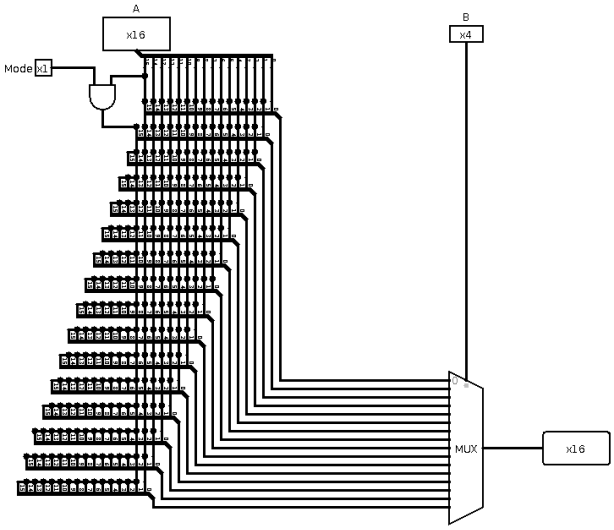

Figure 9: Right Shift block diagram

After the ALU was designed, it was implemented in VHDL. A VHDL testbench gave the ALU a series of inputs and compared the outputs to expected ones. A separate testbench tested only the multiplier using data read from a file. This testbench was then used to do a behavioral simulation in GHDL. This simulation was successful.

After the code was successfully simulated, the code was synthesized in Xilinx ISE Project Navigator 14.7. This allowed for a post-route simulation in ModelSim. This simulation was successful.

After synthesis and post-route testing, inputs and outputs were mapped to the switches and LEDS on the Nexys 3 FPGA Development Board for manual testing. This was done using Plan Ahead. Switches were mapped to the A and B inputs. Buttons were mapped to the control inputs. LEDs were mapped to the output.

3 Results

The results of the behavioral simulation are shown in Figure 5.

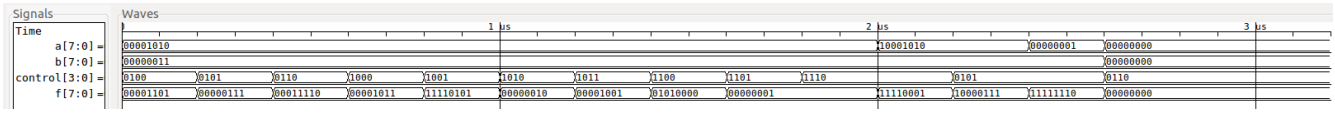

Figure 10: Behavioral simulation

The results of the post-route simulation are shown in Figure .

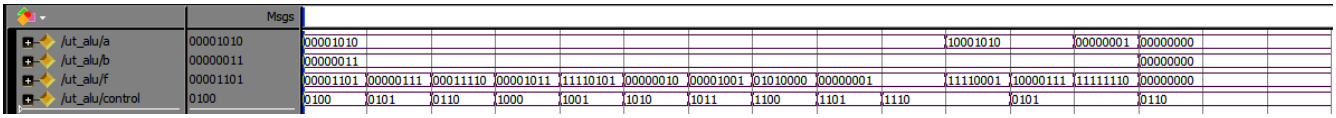

Figure 11: Post-Route simulation

After the FPGA was programmed, it worked successfully.

After synthesis, 25 slices were used, with 25 LUTs used.

4 Conclusion

This exercise was successful. A variable-size Arithmetic Logic Unit (ALU) with 10 operations was designed in VHDL. Generics were used to change the size of the operands. The ALU contained three functional units, doing basic arithmetic, boolean logic, and shifting. Its inputs were the two operands and a four-bit control signal. The output was the result of the operation chosen by the control signal. The ALU was tested successfully with behavioral and post route simulations, and then on a physical Field Programmable Gate Array (FPGA).