Fibonacci Check and Sequence Generator

ECE327: PROJECT 1

September 29, 2019

Bradley Selee
Clemson University
Department of Electrical and Computer Engineering
bselee@g.clemson.edu

Abstract

Lab 1 was designed for an introduction of VHDL and to give a basic understanding of the Quartus simulator. Within this lab, several components were created. Through this, various port modes were declared, concurrent statements were created, and, in part 5, all components coalesced through signals. In part 5, the end of the experiment, the FPGA I/O: switches, LEDs, and 7-segment display all correctly displayed the output when given a specific input. When the select bit was 0 and the 4 bit input was a Fibonacci number, one LED was a logic high while the 7-segment display showed the Fibonacci number. Next, when the select bit was 1 and the 4 bit input was a Fibonacci number, one LED was a logic high while the 7-segment display showed the NEXT Fibonacci number number. However, regardless of the select bit, whenever the 4 bit input was not a Fibonacci number, the 7-segment display showed an 'E' for error [1].

1 Introduction

Lab 1 provides a basic understanding of simple input and output devices to an FPGA. To accomplish this, simple circuits were created, simulated, and implemented. The DE1-SoC, the lab FPGA, consists of 10 switches for input, and 10 LEDs and 6 7-segment displays for output; this lab uses these I/O devices to test the board implementation of each circuit. Every circuit was simulated through a test bench, where the output waveform was monitored.

For this lab, a basic Fibonacci Sequence Generator and Checker was created. Parts 1-4 are designed to create the basic components needed for for part 5, which implements the entire circuit. In part 5, signals were used to connect all the components. This circuit was able to receive a four-bit input, distinguish between a Fibonacci and a non-Fibonacci number, and display the Fibonacci number, or next Fibonacci number, or an error on the hex display.

In order for the FPGA to be programmed, the board had to be set-up properly in Quartus, along with the board file being correct. Then, because the DE1-series has hardwired connections between the chip and its I/O, the pin assignment file *DE1.qsf* was included in the project. Finally, once all parts were completed and tested for correctness, the final board file was compiled and tested on the board.

2 8-bit 2-to-1 Multiplexer

Part 1 designs the architecture for a 2-to-1 multiplexer. This takes two 8-bit inputs, x and y, and a 1-bit select input, s. This multiplexer outputs an additional 8 bits, m:

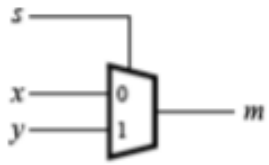

Figure 2.1: 2-to-1 multiplexer [1]

2.1 Architecture Design and Logic

First, the circuit was defined in a basic VHDL file, then the logic was written. The standard logic flow for this multiplexer in VHDL is

```
m <= (NOT (s) AND x) OR (s AND y);
```

However, this implementation can become confusing and over complicated very quickly. To counter this, using a with select statement makes the code more semantic and leaves less room for error.

2.2 Test Bench and Simulation

Once the architecture was created, the design needed to be simulated. This was done through test bench. A test bench allows the creation of test inputs and, in response, Quartus simulates the architecture by displaying the resulting input and output wave forms. For part 1, the test bench includes inputs for x, y, and s. After simulating, the waveform shows the inputs x, y, s and the output m at specific time intervals

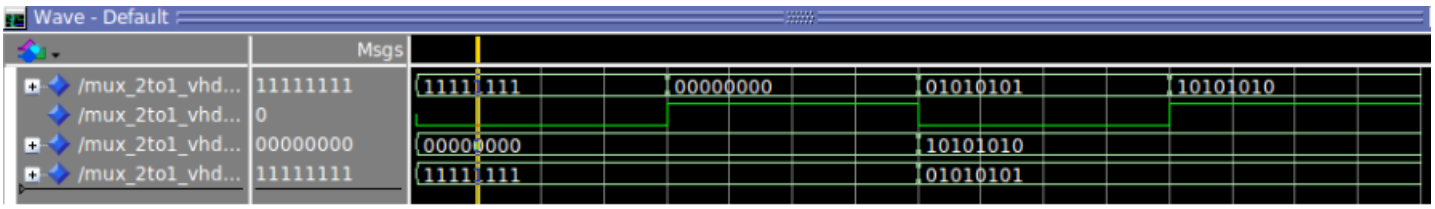

Figure 2.2: Part 1 waveform of test bench simulation

This output shows the multiplexer working correctly because it outputs the selected input.

3 Fibonacci Circuit

Part 2 expands the scope of VHDL's capabilities. Part 2 is very similar to part 1, but this time a wrapper was created and tested on the FPGA. This circuit will recognize the Fibonacci numbers, these form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones. Typically, the Fibonacci sequence begins at 0 and 1, then goes to ∞. In this experiment, the Fibonacci set focused on is

$$F_n = \{0, 1, 1, 2, 3, 5, 8, 13\}$$

The circuit created will output a logic high if the input is within the Fibonacci set, F_n, or a logic low otherwise.

3.1 Architecture Design and Logic

For part 2, the first step was creating the architecture of the circuit. The ports were defined and the logic was implemented using a with select statement for clarity and simplicity.

3.2 Test Bench and Simulation

Next, a test bench was created for the architecture. Values 0-15 were simulated as inputs and the output was monitored

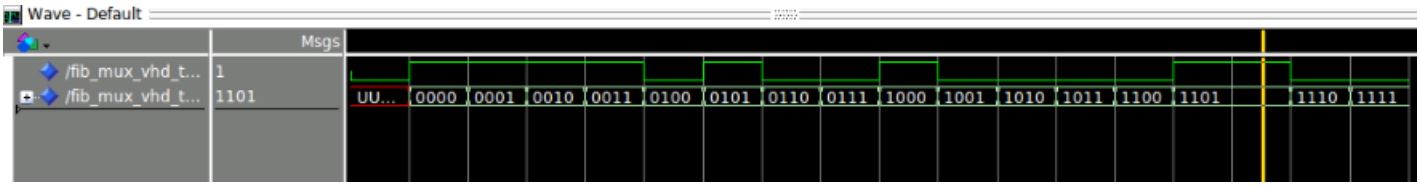

Figure 3.1: Part 2 waveform of test bench simulation

Figure 3.2 demonstrates the circuits correctness because when given a Fibonacci input, the circuit outputs a logic high. When given a non-Fibonacci number, the output is a logic low.

3.3 Hardware Implementation

Once the circuit was verified through a simulation, the circuit was tested on the FPGA. To do this, a wrapper, board file, was created and Quartus was setup to program the board. Finally, the board file was compiled and the board was tested.

When testing, switches 0-3 were used for inputs and LED 3 was used for the output. When given a an input of Fibonacci number, LED 3 lit up. When given a non-Fibonacci number, no LEDs were lit up.

4 Next Fibonacci Number

Part 3 creates a similar circuit to part 2; however, instead of outputting a logic high, the circuit will output the next Fibonacci number. More so, there is still a 4-bit input, but now there is a 5-bit output. If the input is not a valid Fibonacci number, the circuit will output 11111.

4.1 Architecture Design and Logic

Similar to the previous sections, this circuit implements another with select statement. However, because this circuit outputs 5 bits, later on the output will need to be converted to 4 bits when connecting each component through signals.

4.2 Test Bench and Simulation

Once again, a test bench for part 3 was created. Inputs 0-15 were tested and the output waveform was monitored for correctness

Here, the output shows a 5-bit output which corresponds to the next Fibonacci number of the input. Or, the output shows 11111 if the input is not a Fibonacci number.

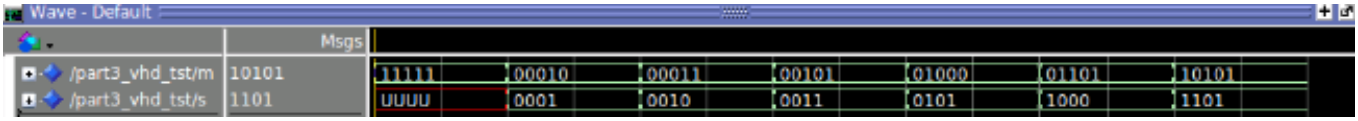

Figure 4.1: Part 3 waveform of test bench simulation

4.3 Hardware Implementation

After verifying the circuit with a test bench and simulation, the circuit was implemented on the FPGA. When compiled to the board, the circuit input was controlled with 4 switches, and the output was displayed by 5 LEDs. When the switches formed a binary Fibonacci number, the board LEDs responded with the next Fibonacci number. For example, if the switches formed a 0101, the LEDs displayed a 01000. If an invalid Fibonacci number was given, the LEDs formed 11111.

5 Displaying Fibonacci Number

Part 4 is the final component of lab 1. The objective of this circuit is to display the Fibonacci number on one 7-segment decoder. If the number is not part of the Fibonacci sequence, or 13 because only one digit can be displayed. The hex display is activated by writing a 7-bit binary number to the display. The segments are activated corresponding to the following figure

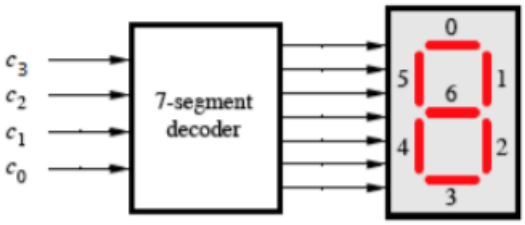

Figure 5.1: 7-segment display layout [1]

The display runs on active low, therefore, to activate a segment, a 0 needs to be written to in the binary position of figure 5.1.

5.1 Architecture Design and Logic

For part 4, the component was defined to take a 4-bit input for the Fibonacci number and a 7-bit output to control the display. The binary values of the 7-segment display were selected using a with select statement. Any input that is not a Fibonacci number results in an error and should display an 'E', which corresponds to 0000110.

5.2 Test Bench and Simulation

Once the architecture was created, the component was simulated with a test bench. Numbers 0-15 were tested as inputs and the 7-bit output was monitored. The output is shown in figure 5.2.

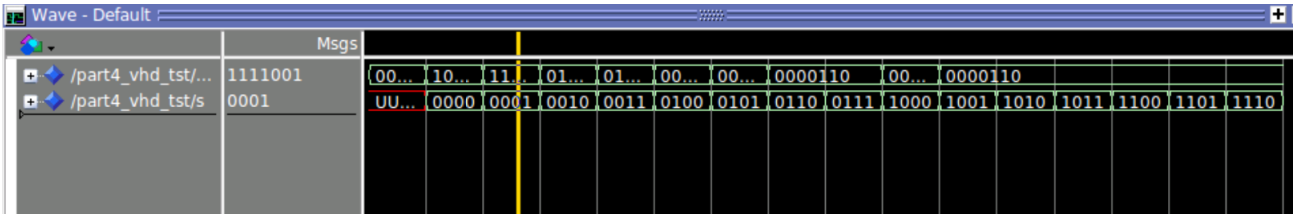

Figure 5.2: Part 4 output waveform

Figure 5.2 shows the correct output when a circuit is fed a Fibonacci input, other than 0 and 13. Other wise, the output displays an E, 0000110, for error.

5.3 Hardware Implementation

Again, the architecture was wrapped in a board file in order to test the circuit on the hardware.The output was mapped to the hex display with the command

$$hex => hex0$$

Once the circuit was compiled and transferred to the FPGA, the board properly displayed the Fibonacci number, and an 'E' when appropriate.

6 Connecting Components

In part 5, all the components from the previous sections were connected. No logic was written within the standard VHDL simulation file. Mentally visualizing the circuit diagram can be hard, therefore, it was very helpful to draw out the diagram of how the components will be connected as shown below

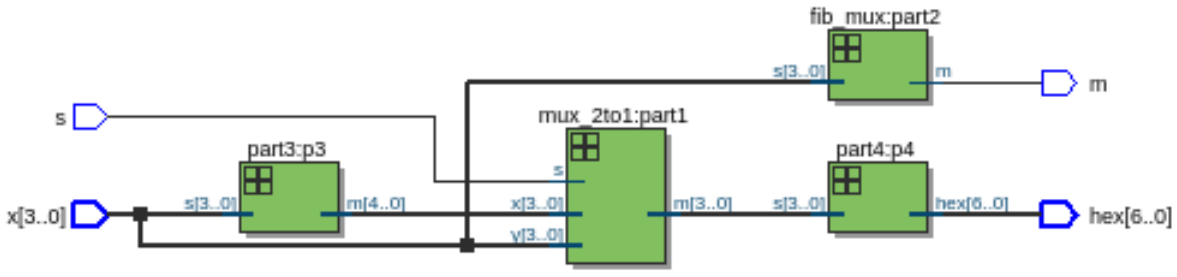

Figure 6.1: Part 5 block diagram generated by Quartus

The functionality of this schematic is explained in the Discussion and Results section.

6.1 Architecture

Unlike the previous sections, the standard VHDL simulation file serves only to instantiate the components and connect them together with signals via the signals keyword. Once, the signals and components were declared, they were mapped together using port map(). In the case of inputting to part 1's multiplexer, part 3 creates a 5-bit output, therefore the input to part 1 must be cast to a 4-bit vector. This was accomplished with the following implementation

```
part1: mux_2to1 PORT MAP (s, s1(3 downto 0), x, s2);
```

where the arguments are either signals or inputs/outputs.

6.2 Test Bench and Simulation

Once the components were connected and functional, an extensive test bench was created. This test bench covers every possible input combination to ensure the correctness of the circuit. This can be seen by the following wave form

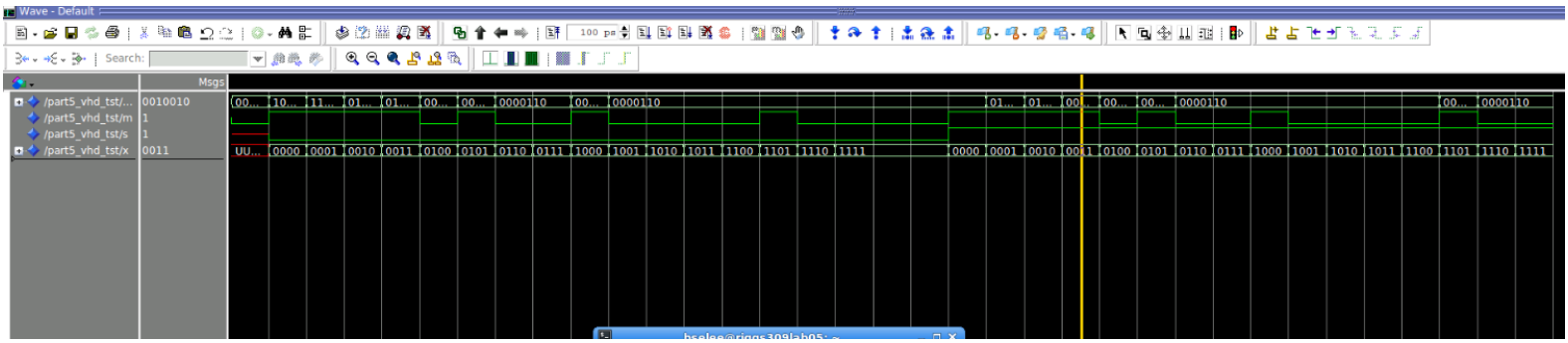

Figure 7.1: Part 5 full circuit waveform

An explanation of this waveform will be explained in the Discussion and Results section.

6.3 Hardware Implementation

Similar to the previous sections, the simulation file was wrapped in order to implement the final circuit on the FPGA. No special code was needed for the part 5 board file. The output of the hardware implementation is discussed in the Discussion and Results section.

7 Discussion and Results

Lab 1 will probably be the most simple lab that is assigned throughout the semester of Fall 2019; however, this lab was still very tedious because it presented very new concepts, languages, and development environments that were somewhat difficult to grasp the idea of. Even though tedious, lab 1 served its purpose and I feel much more confident with hardware description languages, testing, and Quartus. The ultimate goal of lab 1 was to design a circuit with multiple components that could recognize and display Fibonacci numbers through I/O of the FPGA. I encountered several difficulties throughout this project. First, I treated VHDL like any normal programming language, therefore, I was using sequential programming. Later, I found out this typically over complicates the task and logic gates used. To correct this issue, I switched to concurrent programming and sped up my design process. The other major difficulty I encountered was in part 5, when it came to linking all of the components. I misunderstood the instructions and, as a result, I completed part 5 without using any of the previous components. This took much more time and resulted in a longer project. Once I understood the basic concepts of signals, I was able to successfully connect each component and fully design the circuit, as shown in Figure 6.1. Before connecting the components from parts 1-4, each part was tested through a test bench. The test bench included most possible input, then, the circuit was simulated and the output waveform was monitored for correctness. Then, a wrapper file was created in order to test the circuit on the FPGA, using physical inputs, switches, and outputs, LEDs and 7-segments displays. After the all components were found to be correct, they were link together to form the final circuit. This final circuit was tested in the same fashion as the previous components. This final circuit outputs either the inputted Fibonacci number through the hex display, if the select is a 0, and outputs a logic high on an LED. Or, the circuit outputs the NEXT inputted Fibonacci number through the hex display, if the select is a 0, and outputs a logic high on an LED. If at any point the inputs are not part of the Fibonacci sequence, or an 8 or 0 when the select is a 1, the 7-segment display shows an 'E' for error and an LED is given a logic low.

8 Conclusion

VHDL is a powerful hardware description language (HDL), that provides a somewhat-simple, interface which helps users quickly and efficiently design, simulate, and test circuits. Because FPGAs are re-programmable and the user can design nearly any circuit, this makes these tools very expensive. A key concept when programming in an HDL is to focus on concurrent programming most of the time, this will help greatly simplify most circuits and speed up the run time. Another theme is that these projects can be quite long, not necessarily due to lack of understanding, but, more so, because compiling and testing can take a very long time. Overall, after over coming certain difficulties, lab 1 was completed successfully. If I could redo the experiment, I would have made more time to complete the extra credit.

References

[1] M. Smith. *Lab 1: Fibonacci Check and Sequence Generator*. Clemson University, 2019.

9 Appendix

No further references were needed for this project.