**Lab Project #1**

**Alex Kasiniak**

**Joseph Shieh**

**TABLE OF CONTENTS**

2. Abstract

2. Introduction

2. Design Specifications

3. Hardware Implementation

3. Software Implementation

4. Design Testing

5. Data and Error Analysis

6. Project Development Issues and Analysis

6. Summary and Conclusion

7. Appendix

9. Contributions

**ABSTRACT**

In this project, we familiarized ourselves with both the Verilog and C programming languages. On the hardware side, different types of counters were developed while using different models of description. To ensure accuracy, we wrote testbenches, and then viewed the waveforms in the GTKWave wave viewer. The code was then loaded onto the DE1-SoC board, and the hardware was tested further. The counters performed according to the specifications. Three failure mode analysis on the counters was conducted. There were project development issues, and failure analysis was conducted.

On the software side, a currency conversion calculator was developed wherein the user can convert dollars to some other currency and the other way around. The calculator performed according to the specifications.

Important lessons are embedded design often involves a top-down approach and bottom-up implementation and testing, and implementation level can vary, from gate level to entry-schematic level.

**INTRODUCTION**

This project involved familiarizing ourselves with the Verilog and C programming languages, Verilog and C compilation in the terminal using iVerilog and gcc respectively, and debugging hardware with GTKWave wave viewer.

On the hardware side, we used these tools to implement four 4-bit down counters at the gate (hardware design with gates), dataflow (hardware design by controlling data flow), behavioral (hardware design by specifying hardware behavior), and schematic-entry (hardware design through schematics) design levels. We then downloaded these designs onto the DE1-SoC FPGA development board. Three failure mode analysis details the ramifications should any input or output signal be stuck at 0 or 1. There were issues using Quartus SignalTap; reasons discussed in the section.

On the software side, we used the C programming language to implement a currency conversion calculator that uses input currency conversion rate to convert from U.S. dollars to a foreign currency and vice versa.

**DISCUSSION OF PROJECT**

**Design Specifications**

The design specifications were the following:

* Design a 4-bit ripple down counter at the gate design level using the D-flip flop model provided. Refer to Figure 1 for the D-flip flop model.
* Design a 4-bit synchronous down counter at the dataflow design level using the D-flip flop model provided.
* Design a 4-bit Johnson down counter at the behavioral design level.
* Design a 4-bit synchronous down counter at the schematic-entry design level.
* Design a currency conversion calculator that takes in USD to local currency conversion rate, asks for conversion direction, asks how many dollars (USD or local) that the user has, and converts to local or USD.

**System Description**

There was no overall system for this project. The counters all had to work independently, and they each integrated only the flip flop module given to us. Block diagrams of each of the hardware counters that needed to be designed can be found in the Appendix.

**Hardware Implementation**

**General 4-bit Down Counter**

All 4-bit down counters consist of four connected D-flip flops. Refer to Figure 2 for the D-flip flop model. Note that the D-flip flop has an active low reset. All down counters have their resets assigned to switch pins and outputs assigned to LED pins.

**4-bit Ripple Down Counter**

We used the ripple down counter schematic from the *Learn About Electronics* website for a 4-bit ripple down counter and implemented the design using gate-level Verilog. Refer to Figure 3 for the 4-bit ripple down counter schematic.

**4-bit Synchronous Down Counter**

We drew a boolean truth table and determined the boolean equations for the 4-bit synchronous down counter and implemented the design using dataflow-level (RTL) Verilog; another version of the 4-bit synchronous down counter was implemented at the schematic-entry design level using Quartus II IDE. Refer to Equation 1 and Equation 2 for the boolean equations. Refer to Figure 4 for the 4-bit synchronous down counter schematic.

Equation 1. Boolean equation for the ith bit, excluding the 0th bit (LSB), in the 4-bit synchronous down counter.

Equation 2. Boolean equation for the 0th bit in the 4-bit synchronous down counter.

After getting this design, we then implemented this design using Quartus II’s block diagram capabilities which can be seen in Figure 7.

**4-bit Johnson Down Counter using Behavioral Design Level**

The Johnson counter was new to us, and we had to look up the behavior of it first to understand how to design it. We found that the counter should follow a specific pattern, and noticed that it is simply a shifter that first sends 1’s through each bit of the count, and then all 0’s through each bit. This is achieved by hooking the Q ports of each flip flop to the D port of the subsequent flip flop except for the last one, where the Q port is inverted, and then hooked back to the D port. The block diagram for our Johnson counter can be seen in Figure 9.

**Software Implementation**

The currency conversion calculator executes the following steps:

1. Prompts user for conversion rate from USD to local.

2. Prompts user for conversion direction (from USD to local or vice versa).

3. Prompts user for amount of cash they have (USD if converting to local or vice versa).

4. Perform currency conversion.

1. If from USD to local, uses given conversion rate.
2. If from local to USD, uses inverse of given conversion rate.

5. Outputs results of currency conversion.

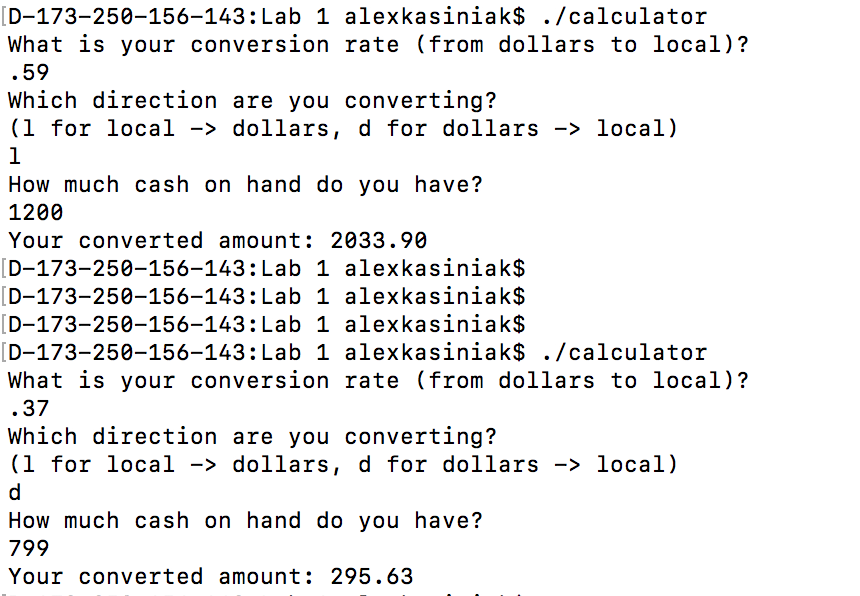
****

Figure 1.Example conversions in both directions

**DESIGN TESTING**

**Test Plan**

Because the counters were relatively simple, testing did not need to be too rigorous. After we implemented each one, we would check to make sure that the counter could repeat the pattern that we wanted. This would be done through the GTKWave viewer. If it did not, then we would need to start looking at the signals between each flip flop to make sure that the logic was doing what we intended. However, we never ran into this case, because the designs were so simple, and the counters all worked on the first try.

**Hardware Test Cases**

The Verilog implementations of the down counters should be tested using the GTKWave wave viewer. The 0th-bit (LSB) D-flip flop should switch from 0-1 or 1-0 every clock cycle. The 1st-bit D-flip flop should switch every two clock cycles. The 2nd-bit D-flip flop should switch every four clock cycles. The 3rd-bit (MSB) D-flip flop should switch every eight clock cycles. The overall result should have the 4-bit down counters counting down in hexadecimal: F => E => D => C => B => A => 9 => 8 => 7 => 6 => 5 => 4 => 3 => 2 => 1 => 0. The Johnson counter will behave differently because it will follow a different pattern. The pattern should be as follows: 0000 => 0001 => 0011 => 0111 => 1111 => 1110 => 1100 => 1000 => 0000.

The Verilog implementations of the down counters should be further tested by downloading the design on the DE1-SoC FPGA board and using Quartus SignalTap to observe down counter output waveforms. When the negative edge of the reset is triggered, the output waveforms should be 0000. When the third state of the down counter is triggered, the output waveforms should be the third state of the down counter (e.g. ripple counter third state trigger and output is 1101; same for synchronous counters but different for Johnson down counter, which is 0011.

The Verilog implementations of the down counters should be further tested by downloading the FPGA board and observing the LED configurations. When active-low reset is applied, all LEDs should be off, representing 0000. When reset is not applied, LEDs should output countdown states; for example 1101 should be represented by ON, ON, OFF, ON. Refer to first paragraph in this section on expected LED output states for down counters.

**Software Test Cases**

The calculator was tested by inputting multiple currency conversion rates and cash amounts and checking that the output currency amount was correct. Since the procedure to get from one currency to another is very simple, again we did not need very rigorous testing. There also was no reasonable lower or upper bound on the number of dollars that someone could convert, since we were using doubles to keep track of dollar values, and the upper bounds on each of those far exceed any amount of money that any person has.

**DATA AND ERROR ANALYSIS**

**4-bit Ripple Down Counter Results and Three Failure Mode Analysis**

The 4-bit ripple down counter worked per expectations outlined in the previous section.

Should the “D” input for a D-flip flop be stuck at 0 (SA0) or stuck at 1 (SA1), the output would be stuck at 1 or 0 respectively. The clocks for downstream D-flip flops would be stuck at 0 or 1. When the clocks are stuck at 0 or 1, the D-flip flops cannot switch between 0 and 1. The D-flip flops would constantly output the same bit. The overall effect is the ripple down counter would be stuck at a number and unable to count down. The same argument applies when the clock input for the LSB is stuck at 0 or 1.

Should the reset be stuck at 0, the outputs of all D-flip flops would constantly reset to 1 and be unable to switch to 0. The overall effect is the counter remains stuck at 16. Should the reset be stuck at 1 and the counter is unable to initiate an initial reset, the D-flip flops default to random values, resulting in random initial state (e.g. could be 12, 11, 9, 3, 2, 1, etc.). However, the counter would count down correctly from that initial number.

**4-bit Synchronous Down Counter Results and Three Failure Mode Analysis**

The 4-bit synchronous down counter worked per expectations outlined in the previous section.

Should the “D” input for a D-flip flop be SA0 or SA1, the output remains SA1 or SA0 respectively. All downstream D-flip flops would be affected because their inputs are the OR and/or XOR of previous D-flip flop outputs; OR and XOR outputs and thus downstream D-flip flops inputs remain the same. The D-flip flops cannot switch between 0 and 1. The D-flip flop would constantly output the same bit. The overall effect is the synchronous down counter would be stuck at a number and unable to count down. A similar argument applies when the clock inputs for the D-flip flops are SA0 or SA1.

Should the reset be SA0, the D-flip flops would constantly reset to 0. The counter would always be stuck at 16. Should the reset be SA1 and the counter is unable to perform initial reset, the D-flip flops would initiate to random values; the counter would initiate to a random number. However, the counter would count down correctly from that initial number.

**4-bit Johnson counter results and failure mode analysis**

The Johnson counter worked per expectations and the results can be seen in Figure 8. The count went through the proper sequence, and can be reset to the all zero’s state.

The failure analysis for the various signals is as follows:

reset:

SA0: The count will be stuck at 0.

SA1: The count will never be defined.

clk:

SA0: The count will never be reset because it needs at least one clock edge to do that

SA1: The same will happen as in the SA0 case

count[0]:

SA0: Eventually the count will get stuck at 0000

SA1: Eventually the count will get stuck at 1111

count[1]:

SA0: Eventually the count will get stuck at 0001

SA1: Eventually the count will get stuck at 1110

count[2]:

SA0: Eventually the count will get stuck at 0011

SA1: Eventually the count will get stuck at 1100

count[3]:

SA0: Eventually the count will get stuck at 0111

SA1: Eventually the count will get stuck at 1000

**PROJECT DEVELOPMENT ISSUES AND ANALYSIS**

We were unable to use Quartus SignalTap and obtain waveforms. Triggering events failed in SignalTap, preventing us from obtaining any pre-trigger and post-trigger data for unknown reasons. Attempts to resolve this issue was unsuccessful. However, it should be noted that the designs were successful on the GTKWave wave viewer and on the DE1-SoC FPGA board output LED displays.

**SUMMARY AND CONCLUSION**

This lab focused on familiarizing ourselves with the Verilog and C programming language and compiling them with iVerilog and gcc respectively, implementing designs at the gate, dataflow, behavioral, and schematic-entry level, testing designs using GTKWave wave viewer and on the DE1-SoC FPGA board, and performing three failure mode analysis. We used these tools to develop ripple, synchronous, and Johnson down counters and a currency conversion calculator. By using these tools, we were able to successfully develop the counters and calculator according to design specifications. However, we were unable to successfully utilize Quartus SignalTap because triggering events failed.

It appears that embedded design often involves top-down modeling (functional decomposition) but bottom-up implementation and testing (expediting the debugging process). It should be noted that developing the skill sets outlined in the previous paragraph is necessary in implementing increasingly complex embedded projects.

**APPENDICES**

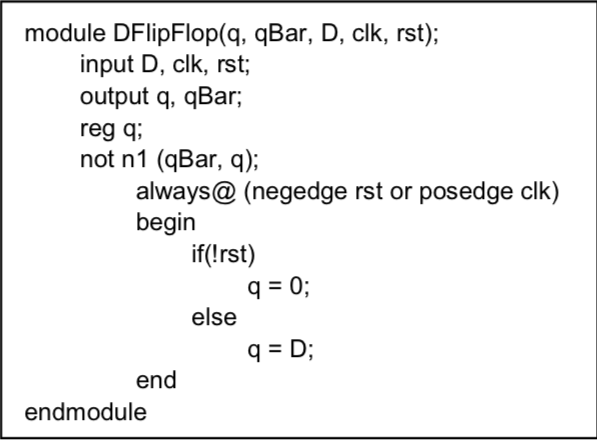


Figure 2. Verilog code for D-flip flop model with active low reset.

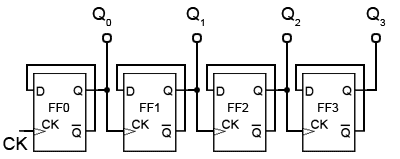


Figure 3. Schematic for a 4-bit ripple down counter.

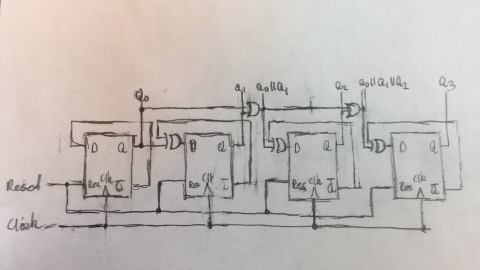
****

Figure 4. Schematic for a 4-bit synchronous down counter.

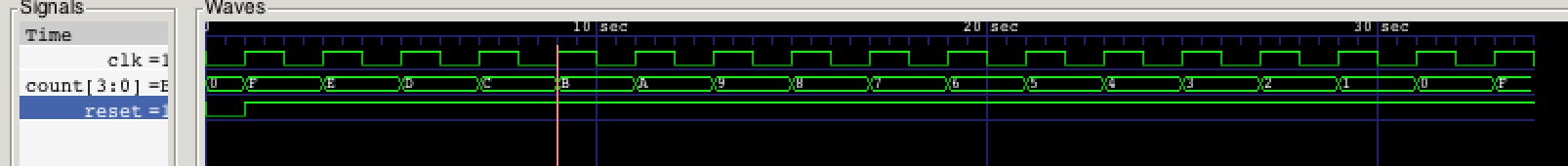


Figure 5. The ripple counter waveform

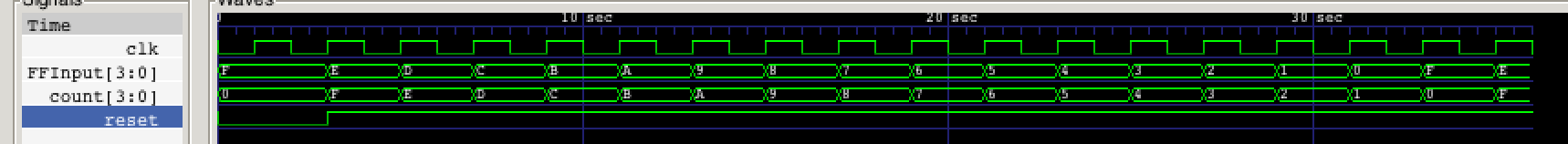


Figure 6. The synchronous counter waveform

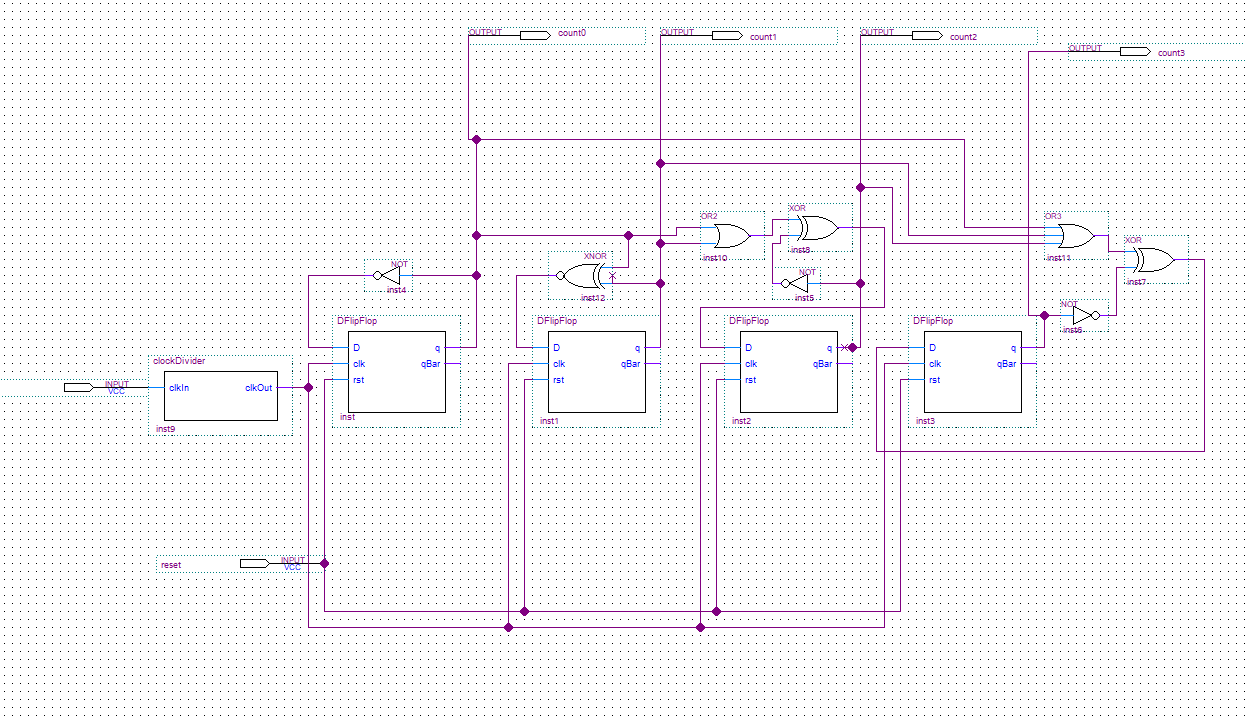


Figure 7. Block diagram for synchronous down counter

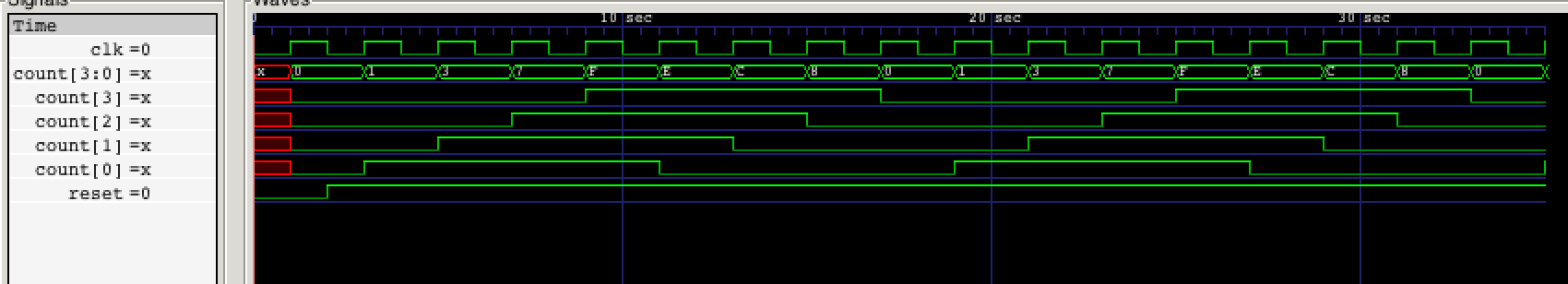


Figure 8. Waveform for Johnson counter

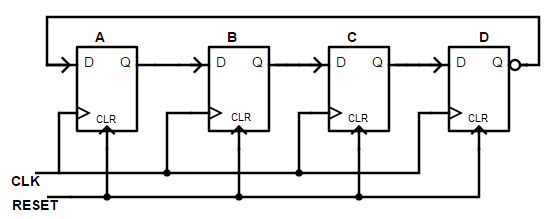


Figure 9. Block diagram for Johnson counter

**Contributions**

I, Alex Kasiniak, contributed the following:

Helped with design of all of the counters

Designed the calculator

Collaborated on the lab report

I, Joseph Shieh, contributed the following:

Helped with design of all of the counters

Worked on capturing the Signal Tap waveforms

Collaborated on the lab report