Project 1

Introduction to the Lab Environment

EE 371 AC

January 15, 2016

NAME	**STUDENT ID**
Denny Ly	1231185
Minhhue H. Khuu	1329349
Ruchira Kulkarni	1234324

Table of Contents:

1. Abstract

The goal of this lab was to understand different developing environments and tools such as iVerilog and gtkwave and Quartus II IDE for developing and testing Verilog code. This was achieved by making four 4-bit down counters, namely the ripple-down counter, a four stage synchronous down counter and a Johnson down counter and testing each implementation using a variety of tools to understand each ones input and output relationship at varying situations. Additionally, a currency converter was implemented in C to calculate different values given an initial amount and an exchange rate.

2. Introduction

In this lab, EE 271 material will be reviewed and new material will be investigated further. The purpose of the lab is extending the working knowledge of the Altera DE1-SoC board, Quartus II development environment and tools used in previous embedded courses. Three different counters are designed using Verilog HDL and the schematic entry feature in Quartus. Each of the counters will be created using different modeling levels that the modelling language supports. The difference of each of the models will be compared and contrasted together. Finally, the end of the lab focuses on a little programming in C. The program is a simple currency calculator that will be used to buy stuff faster.

3. Tools

The lab requires the use of iVerilog and gtkwave to produce waveforms of our designs instead of the previous simulation tool, ModelSim, used in previous classes. ModelSim and iVerilog/gtkwave are functionally equivalent and will just give a wider perspective of tools that can be used to analyze future projects. Quartus II Signal Tap Logic Analyzer is also another tool that can be used to see the output waveforms from the various structures that were built in the lab. The RTL Viewer will be used to examine the differences of the different levels of the modelling language. The gcc compiler will be used to compile C to be ran as an executable on a given computer machine. Finally, to generate the Verilog and C code, Notepad++ was used as the text editor. The tools used in this lab is shown in Table 1.

Table 1. Table of Tools Used for this Lab

Tool	Purpose
iVerilog	Synthesize Verilog files
gtkwave	Output waveform of Verilog files
Quartus II Waveform Viewer	Output waveform of schematic files
Quartus II Signal Tap	Output waveform of hardware result
Quartus II RTL Viewer	Synthesizes a structural model of Verilog file
Cyclone V FPGA (Model 5CSEMA5F31C6)	Platform to implement project design
GCC	C compiler for C files
Notepad++	Text editor

4. Discussion of the Project

This section will explain (1) Design Specification, (2) Design Procedure, (3) System Description, (4) Software Implementation, and (5) Hardware Implementation. The design specification, design procedure, and system description, will describe design for the counters used in this lab that will be tested and analyzed throughout the lab. The software and hardware implementation will explain how the design for the counters will be implemented.

The C program will be explained in detail in Appendix located at Section 9.

4.1 Design Specification

In this project, four counters were implemented in Verilog. The first three counters were implemented at different levels, gate, dataflow, and behavioral, respectively. The fourth counter was implemented using a schematic design derived from the synchronous 4-bit down counter. All the counters had an asynchronous active low reset to 0000. The 4-bit down counters were displayed on the 4 LEDs (LED[3:0]) on the DE1-SoC board. The active low reset is implemented by one of the DE1-SoC's keys (KEY[0]).

The 4-bit counters are as follows:

1) Ripple Down Counter (gate level/ structural level), an asynchronous 4-bit down counter with an active low reset.
2) Synchronous Down Counter (dataflow model), a synchronous down 4-bit counter with an active low reset.
3) Synchronous Johnson Counter (behavioral model), a synchronous down 4-bit counter with an active low reset.
4) Synchronous Down Counter (schematic design), a synchronous down 4-bit counter with an active low reset.

4.1.1 Four Bit Ripple Down Counter (Structural Model)

A ripple counter is an asynchronous counter where only the first D Flip Flop is clocked by an external clock. The design required a gate or structural model with active low reset. Gate level or structural design implies interconnecting logical gates with wires (such as and, nor, or gates) only. All subsequent flip-flops are clocked by the output of the preceding D Flip Flop. The ripple counter ranges from 0000 to 1111. On reset, the counter starts at 0000. Otherwise, the outputs decrements to 1111, 1110, 1101, until 0000 and repeats.

4.1.2 Synchronous Down Counter (Dataflow Model)

In a synchronous down counter, all the output bits change state simultaneously in the next positive clock edge signal. The synchronous down counter was implemented using the data flow model using D Flip Flop with an active low reset and bitwise operations. The dataflow model is similar to the structural model, but instead of wiring modules with logical gates, assign statements and bitwise operations are used instead to model the flow of data across the system. On reset, the counter starts at 0000. Otherwise, the outputs decrements to 1111, 1110, 1101, until 0000 and repeats.

4.1.3 Synchronous Johnson Counter (Behavioral Model)

The down Johnson Counter is a shift register with feedback with inverted output Q of the last flip-flop is now connected back to the input D of the first flip-flop. The down Johnson counter behaves similar to a right shift register, with the difference that the right most bit will be inverted and push into the left most bit and repeats. On reset, the counter starts at 0000. Otherwise, the output “decrements” or right shifts to 1000, 1100, 1110, 1111, 0111, 0011, 0001, until 0000 and repeats.

4.1.4 Synchronous Down Counter (Schematic Entry)

The schematic entry of the synchronous down counter is a schematic implementation of the structural implementation of the synchronous down counter. The design is completely based on the dataflow model using logic gates and wiring as oppose to bitwise operations and assign statements. The schematic entry should behave identical to the dataflow model. On reset, the counter starts at 0000. Otherwise, the outputs decrements to 1111, 1110, 1101, until 0000 and repeats.

4.2 Design Procedure

4.2.1 D Flip Flop

An important component used in this lab is the D Flip Flop. The D Flip Flop is used to store the current state of the output until the next positive clock edge. On a positive clock edge, the D Flip Flop will update the current state of the output to the new state of the output. The D Flip Flop is reset on an active low reset signal, meaning at low the D Flip Flop will output 0 asynchronously. On a high reset signal, the D Flip Flop will continue to operate every positive clock edge. For convenience, the D Flip Flop module contains an inverted current state of the output. The Verilog code for the D Flip Flop module used in this lab is shown in Figure 1.

```
module DFlipFlop(q, qBar, D, clk, rst);
  input D, clk, rst;
  output q, qBar;
  reg q;
  not n1(qBar, q)
  always@ (negedge rst or posedge clk) begin
    if (!rst)
      q = 0;
    else
      q = D;
  end
endmodule
```

Figure 1. Verilog code for DFlipFlop module given by James K. Peckol.

4.2.2 Four Bit Ripple Down Counter (Structural Model)

The design of the four bit ripple down counter was a modification of the four bit ripple up counter that was found by research. The four bit ripple down counter uses four D Flip Flops, the new state output is the inverted current state output. Each of the four D Flip Flop’s clock signal, with the exception of the least significant bit (0th bit), is controlled by the current state output of the previous bit. The 0th bit’s D Flip Flop is clocked normally with a clock signal.

The schematic for the four bit ripple down counter is shown in Figure 2.

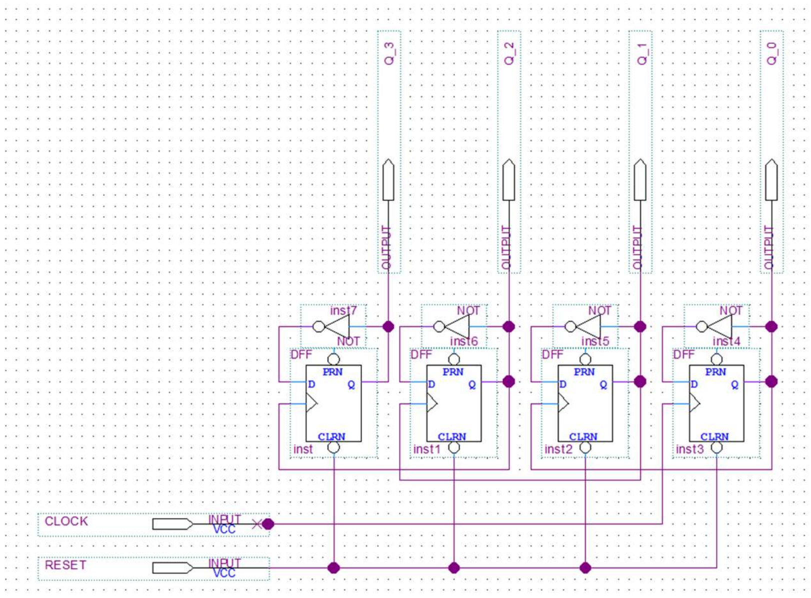


Figure 2. Ripple Down Counter Structural Diagram designed in Quartus.

Using the schematic and individually analyzing each positive clock edge on an active high reset signal, a state table was generated as shown in Table 2.

Table 2. Four Bit Ripple and Synchronous Down Counter State Table

State	Q[3]	Q[2]	Q[1]	Q[0]
0	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
0	0	0	0	0

By inspection, the four bit ripple down counter is counting from decimal value 15 to decimal value 0 and repeats after the reset signal is active high.

4.2.3 Synchronous Down Counter (Dataflow Model)

The implementation of the synchronous down counter required an analysis of the state table shown in Table 2. The state table for the synchronous down counter is identical to the state table of the ripple down counter. By analyzing the relationship of the new state with the current state of each of the individual output bits, a Karnaugh Map is formed and a Boolean equation was derived from the Karnaugh map.

The new state for the 0th bit (D[0]) is an inversion of the current state of the 0th bit (Q[0]). This was simple enough that a Karnaugh map is not necessary.

For the new state for the 1st, 2nd, and 3rd (D[1], D[2], and D[3]) required a Karnaugh map to analyze the Boolean equation to model the relationship of each of the current state bits. The Karnaugh maps for D[1], D[2], and D[3] is shown in Table 3, Table 4, and Table 5, respectively.

Table 3. Karnaugh Map for new value D[1] for the Synchronous Down Counter

	$\overline{\boldsymbol{Q[0]}}$	$\boldsymbol{Q[0]}$
$\overline{\boldsymbol{Q[1]}}$	0	1
$\boldsymbol{Q[1]}$	1	0

Table 4. Karnaugh Map for new value D[2] for the Synchronous Down Counter

	$\overline{\boldsymbol{Q[1]}}\,\overline{\boldsymbol{Q[0]}}$	$\overline{\boldsymbol{Q[1]}}\,\boldsymbol{Q[0]}$	$\boldsymbol{Q[1]}\,\boldsymbol{Q[0]}$	$\boldsymbol{Q[1]}\,\overline{\boldsymbol{Q[0]}}$
$\overline{\boldsymbol{Q[2]}}$	1	0	0	0
$\boldsymbol{Q[2]}$	0	1	1	1

Table 5. Karnaugh Map for new value D[3] for the Synchronous Down Counter

	$\overline{\boldsymbol{Q[1]}}\,\overline{\boldsymbol{Q[0]}}$	$\overline{\boldsymbol{Q[1]}}\,\boldsymbol{Q[0]}$	$\boldsymbol{Q[1]}\,\boldsymbol{Q[0]}$	$\boldsymbol{Q[1]}\,\overline{\boldsymbol{Q[0]}}$
$\overline{\boldsymbol{Q[3]}}\,\overline{\boldsymbol{Q[2]}}$	1	0	0	0
$\overline{\boldsymbol{Q[3]}}\,\boldsymbol{Q[2]}$	0	0	0	0
$\boldsymbol{Q[3]}\,\boldsymbol{Q[2]}$	1	1	1	1
$\boldsymbol{Q[3]}\,\overline{\boldsymbol{Q[2]}}$	0	1	1	1

By using the Karnaugh maps, the Boolean equations was derived for their respective new state in equations 1, 2, 3, and 4. Where D[0], D[1], D[2], and D[3] represents the new state value for the corresponding indexed bit and Q[0], Q[1], Q[2], and Q[3] represents the current state value for the corresponding index bit.

$$D[0] = \overline{Q[0]} \tag{1}$$

$$D[1] = \overline{Q[1] \oplus Q[0]} \qquad (2)$$

$$D[2] = \overline{Q[2]}\ \overline{Q[1]}\ \overline{Q[0]} + Q[2](Q[0] + Q[1]) \qquad (3)$$

$$D[3] = \overline{Q[3]}\ \left(\overline{Q[2]}\,\overline{Q[1]}\,\overline{Q[0]}\right) + Q[3](Q[2] + Q[1] + Q[0]) \qquad (4)$$

By using the derived Boolean equations as mentioned, assign statements and bitwise operations was used to completely define the four bit synchronous down counter.

4.2.4 Synchronous Johnson Counter (Behavioral Model)

The implementation of synchronous Johnson down counter was a modification of the synchronous Johnson up counter found by research. The modification was changing the direction of which the Johnson up counter operated. Typically, the Johnson up counter inverts the left most bit and pushes it to the right most bit, which simulates a shift left register. A Johnson down counter does the opposite, which inverts the right most bit and pushes it to the left most bit. This simulates a shift right register.

The schematic for the Johnson down counter is shown in Figure 3.

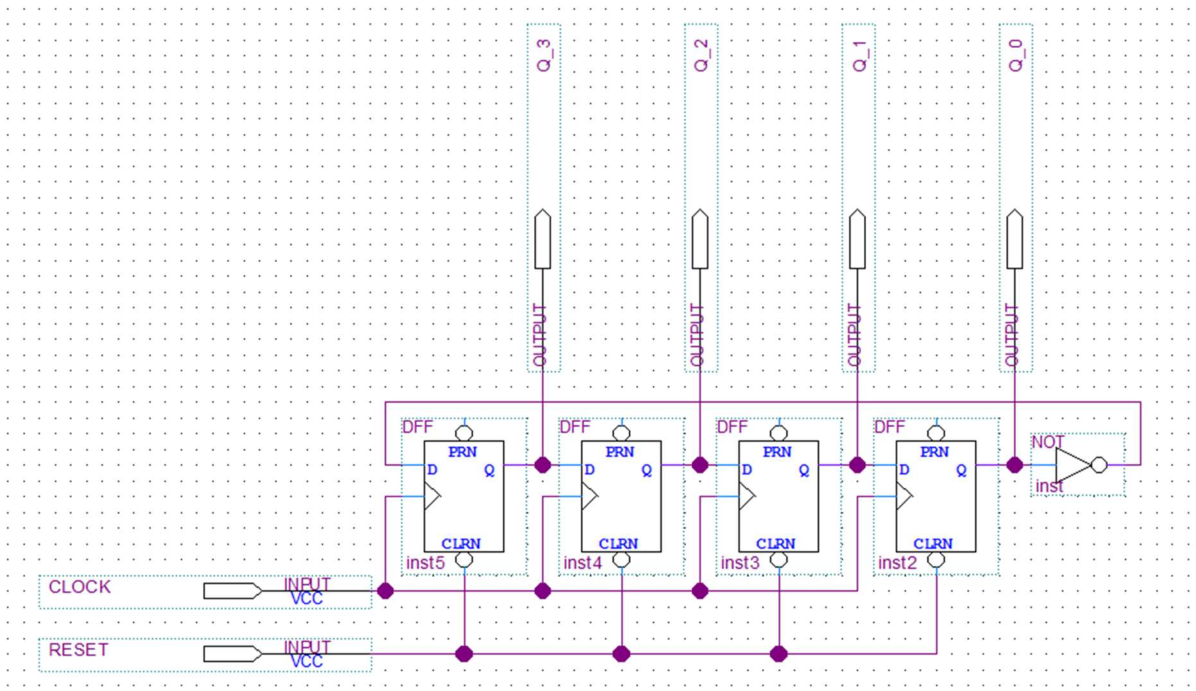


Figure 3. Johnson Down Counter Structural Diagram designed in Quartus.

Using the schematic and individually analyzing each positive clock edge on an active high reset signal, a state table was generated as shown in Table 6.

Table 6. Four Bit Johnson Down Counter State Table

State	Q[3]	Q[2]	Q[1]	Q[0]
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
0	0	0	0	0

4.2.5 Synchronous Down Counter (Schematic Entry)

The implementation of the schematic entry of the synchronous down counter was based on the implementation of the dataflow model of the synchronous down counter. By interpreting the assign statements and bitwise operations with wires and logic gates, we constructed a structural model of the synchronous down counter. By using the structural model, a schematic can be created by dictating the flow of data using wires and logic gates. This is the lowest level implementation that can be derived without going into the circuitry of each logic component.

The schematic for the synchronous down counter is shown in Figure 4.

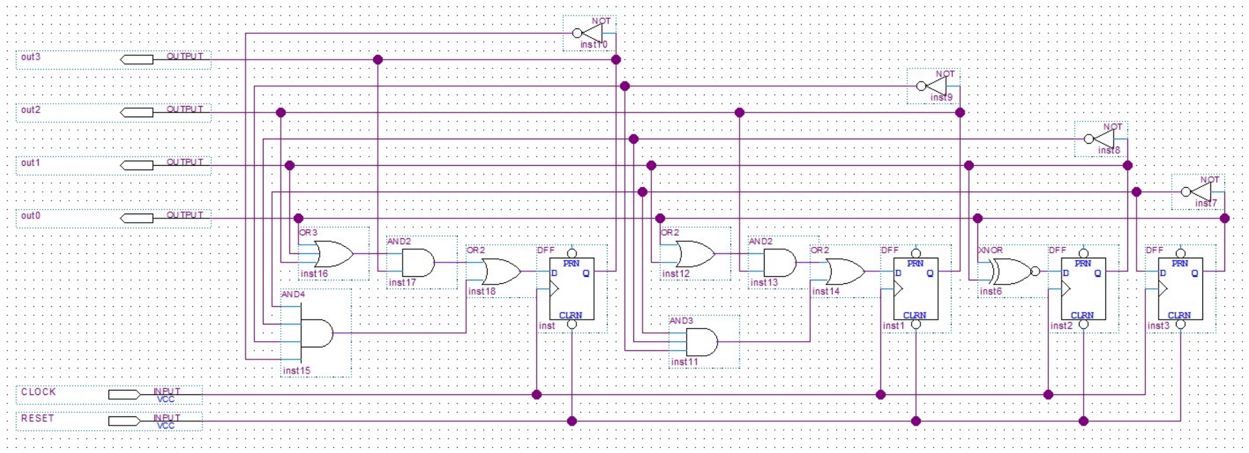


Figure 4. Synchronous Down Counter Structural Diagram Implementation from Quartus.

4.3 System Description

Four different counting circuits were built and tested in this lab. Each of the counters takes in a reset signal from a key and a clock as their inputs. The output of the counters is displayed onto 4 LEDs. The reset for the counters are designed to be an active low reset. When the reset signal is on, the counters will begin with no LED lights and then increment to their next state at each rising edge of the clock cycle. When the reset is off, the counter will return to the state with no LED lights once the clock hits the next rising edge and will freeze at that state. The counters also use a clock divider so the pattern of each counter can be seen on the board.

Each of the counters consist of a top level module and a counter module programmed in Verilog HDL. The Johnson down counter is designed using the behavioral model. The synchronous down counter was implemented using the dataflow model and also schematic entry. The ripple down counter was designed using the gate/structural model.

4.4 Software Implementation

The code for this lab can be found in the included zip file submitted on canvas. Table 7 shows the filename of implemented code after extraction in the ./code folder and its purpose.

Table 7. File name of code written for this lab

File name of Code	**Purpose**
ripple_counter.v	Four bit ripple down counter (structural)
count_down.v	Four bit synchronous down counter (dataflow)
count_down_johnson.v	Four bit Johnson down counter (behavioral)
top_count_down_dataflow.v	Top level model for dataflow
top_counter_down_schematic_struct.v	Top level file for structural
Top_johnson_counter_behavioral.v	Top level file for behavioral
./code/c_code/calc.c	The C program for the currency exchange program

The primary method of implementing the four bit down counters is to create Verilog code that models the system of the different types of the down counters. To do this, the code was written in Notepad++ using tab settings set to two spaces. After writing the code that models a down counter, a test bench and tester module was written to generate a .vcd file for gtkwave to display the wave form. Finally, the Verilog files are compiled using iVerilog, ran to produce a .vcd file, and ran on gtkwave to display a waveform that displays the input and output relationship of the device under test.

The schematic entry was created using Quartas II schematic builder. This geerated a .bdf file that model the lowest level implementation of the down counter, the hardware implementation using logic gates and wires. The schematic was tested using the Quartas II Wavefrom tool to generate a waveform describing the input and output relationship of the device under test.

4.5 Hardware Implementation

We used the Quartus IDE to load our Verilog modules on to the DE1_SOC FPGA board. For our hardware, we used simple D- Flip Flops to achieve the functionality of the ripple down, Johnson and synchronous counters. We assigned the required pins on the FPGA board such as LEDS[3:0], KEY[0], system CLOCK (50 MHz) using a pin planner tool in Quartus. The pin assignment layout was imported from an excel spreadsheet we created by studying the DE1_SOC datasheet for FPGA pin assignments.

Table 8. Signal to FPGA Pin Pairing

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY[0]	PIN_AA14	Push-button[0]	3.3V
LEDR[0]	PIN_V16	LED[0]	3.3V
LEDR[1]	PIN_W16	LED[1]	3.3V
LEDR[2]	PIN_V17	LED[2]	3.3V
LEDR[3]	PIN_V18	LED[3]	3.3V
CLOCK_50	PIN_AF14	50 MHz clock input	3.3V

5. Testing

The testing of the design was broken up into three segments, all of which contributed to the testing process: (1) Test Plan, (2) Test Specifications, and (3) Test Cases. The test cases are developed based on the specifications of the test for the general test plan for each testing tool. The results of the test is shown in Section 6.

5.1 Test Plan

A down counter is a system and a system is purely defined by its output and input relationship. Therefore, to test each of the counters implemented in this lab, each counter of the three 4-bit down counters: (1) Ripple, (2) Synchronous, and (3) Johnson were analyzed for their corresponding output signal given a varying input signal. By using the truth tables derived from Table 2 for the Ripple and Synchronous down counter and Table 6 for the Johnson down counter, their behaviors can be predicted and analyzed for any similarities and irregularities.

In this lab, the followings tools were used to test each of the three counters:

- iVerilog and gtkwave
- Quartus II and Signal Tap
- Quartus II and Waveform tool for schematic files

5.1.1 iVerilog and gtkwave

The process of testing in iVerilog and gtkwave consists of the following steps:

1) Design a tester module in Verilog to generate the input signals for the unit under test.
2) Design a testbench module in Verilog to generate a .vcd file to view the waveform generated from the unit under test.
3) Synthesize associated Verilog files in iVerilog and run the synthesized Verilog program to generate a .vcd file and run gtkwave to view the waveform using the batch commands as shown in Figure 5.

```
1. iverilog –o out <insert associated Verilog .v files>
2. vvp out
3. gtkwave <insert associated gtkwave .vcd file>
```

Figure 5. Command Prompt commands to display waveform from Verilog file.

4) Analyze the waveform to see if the waveform has any similarities or irregularities from predicted results from the test specifications.

5.1.2 Quartus II and Signal Tap

The process of testing in signal tap consists of the following steps:

1) Design a top level module in Verilog under the Quartus II IDE.
2) Assign the output to the appropriate LED on the DE1-SoC board to display the 4-bit number.
3) Assign the input to a KEY on the DE1-SoC board for a reset signal.
4) Synthesize and open up Signal Tap and set input, output, and clock ports for testing.
5) Set the trigger conditions for test condition.
6) Set clock and sample depth to tBase[19] and 512 samples, respectively.
7) Load the .sof file into the DE1-SoC board.
8) Run the Signal Tap and utilize the input signal to activate the trigger condition.
9) Analyze the waveform to see if the waveform has any similarities or irregularities from predicted results from the test specifications.

5.1.3 Quartus II and Waveform tool

The process of testing in signal tap consists of the following steps:

1) Create the schematic file using Quartus.
2) Synthesize the schematic file.
3) Open up the waveform tool.
4) Locate the output, input pins for the schematic, and insert them into the waveform tool.
5) Set the input signals wave form.
6) Run the analyzing tool to view the output waveform.
7) Analyze the waveform to see if the waveform has any similarities or irregularities from predicted results from the test specifications.

5.2 Test Specification

The test specifications were similar for all of the tools used for testing which includes two cases: (1) reset signal is low and (2) the reset signal is high. For the record, the clock should always behave independently from the reset signal and continue alternating from high and low regardless of the state of the reset signal.

5.2.1 Reset Signal is set low

- Reset signal is initially set low.
  - Output is 0000 (0 across all four bits).
  - Output remains 0000 while the reset signal is set low.
- Reset signal is set low after being set high.
  - Output is instantly changed from its current state to 0000.
  - The reset is asynchronous, and is independent of the clock cycle.

5.2.2 Reset Signal is set high

The output behavior of the down counters differs between the Ripple and Synchronous down counter to the Johnson down counter. The Ripple and Synchronous down counter counts from decimal value 15 to decimal value 0 then jumps to decimal value 15 and repeat. The Johnson down counter behaves like a right shift register while having the least significant bit inverted and push back to the most significant bit.

5.2.2.1 Ripple and Synchronous Down Counter

- Reset signal is initially set high.
    - Current output is set to the next output at the next positive (rising edge) of the clock signal.
    - The next output is dependent on the current output.
        - If the current output is 0000, then the next output will jump to 1111.
        - Otherwise, the next output will decrement the current output value, meaning 1111 to 1110 to 1101 until the current output is 0000.
- Reset signal is set high after being set low
    - Current output is set to the next output at the next positive (rising edge) of the clock signal.
    - The next output will always be 1111, then decrements and repeats.

5.2.2.2 Johnson Down Counter

- Reset signal is initially set high
    - Current output is set to the next output at the next positive (rising edge) of the clock signal.
    - The next output is dependent on the current output
        - If the current output is 0000, then the next output will jump to 1000.
        - Otherwise, the next output will "decrement" by shifting to the right.
            - The least significant bit of the current output will be inverted and pushed into the most significant bit of the next output.
            - The other bits will shift towards the right.
            - Example: 0000 to 1000 to 1100 to 1110 to 1111 to 0111 until the current output is 0000 and repeat.
- Reset signal is set high after being set low
    - Current output is set to the next output at the next positive (rising edge) of the clock signal.
    - The next output will always be 1000, then "decrements" by shifting to the right.

5.3 Test Case

The test cases were conducted using the procedures described in Section 5.1 for each of the corresponding testing methods. Specifically, the test cases involve testing the counters in two states: (1) reset is set low and (2) reset is set high. Additionally, the transition between low and high and vice versa will be examined heavily.

To make use of time, the general test case is to have the reset set low initially for a couple clock cycles, then set the reset high for a couple clock cycles, then set the rest low for a couple clock cycles, and finally set the reset low for a couple clock cycles. This test case allows for the examination of the behavior of the counter in the two possible states of the counters. Furthermore, by having the reset signal alternate multiple times, the transition period can be examined.

6. Presentation, Discussion, and Analysis of the Results

The following results are obtained through the methods explained in Section 5 regarding testing. The tests are conducted using the following three tools: (1) iVerilog and gtkwave for the structural, data path, and behavioral level counters, (2) Quartus II Waveform Tool for the schematic entry, and (3) Quartas II Signal Tap for real world analysis of the hardware implementation of the four counters.

6.1 iVerilog and gtkwave

In this first part of the lab, Verilog HDL was used to design and build the four counters. After the counters were built, the modules were compiled and simulated to produce waveforms for the inputs and outputs. Iverilog was used to compile the code and gtkwave was used to see the output of the four counters.

6.1.1 Four Bit Ripple Down Counter (Structural Model)

Figure 6 shows the simulation results of the 4-bit ripple down counter. The simulation displays some special cases of the counter. When the simulation begins, the reset signal is low and the counter circuit will continuously output 0 for all of the LEDs. When the reset signal is 1, the counter will increment down at every rising edge of the clock. Once the counter reaches the value 0, the counter will wrap back around to 15 and continue decreasing down. Further along the simulation, the reset signal reverts back to 0. The change in the reset signal also switches the output to continuously show 0 on all of the LEDs again.

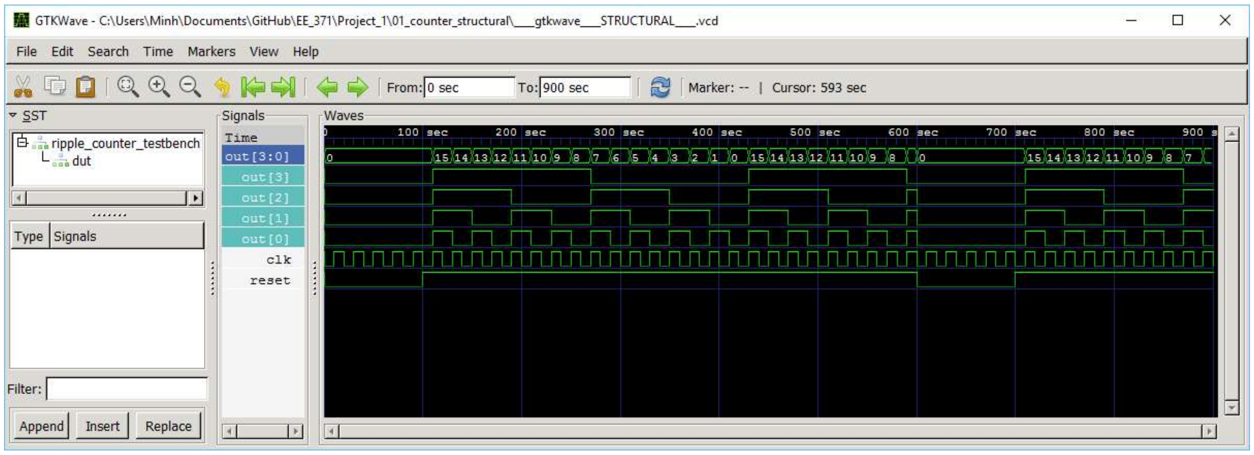

GTKWave - C:\Users\Minh\Documents\GitHub\EE_371\Project_1\01_counter_structural___gtkwave___STRUCTURAL___.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 900 sec
Marker: -- | Cursor: 593 sec
SST
ripple_counter_testbench
dut
Type Signals
Filter:
Append Insert Replace
Signals
Time
out[3:0]
out[3]
out[2]
out[1]
out[0]
clk
reset
Waves
100 sec 200 sec 300 sec 400 sec 500 sec 600 sec 700 sec 800 sec 900 s
0 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 0 15 14 13 12 11 10 9 8 7

Figure 6. Four Bit Ripple Down Counter Waveform using gtkwave.

6.1.2 Synchronous Down Counter (Dataflow Model)

Figure 7 shows the simulation results of the 4-bit synchronous down counter. This counters simulation looks like the ripple counters simulation. The reset signal has the same effect on the output. When the reset signal is 0, the simulation continuously outputs 0. When the reset signal is initially set to 1, the output will change into the value of 15 at the next rising edge of the clock. The counter also does wrap around from 0 to 15.

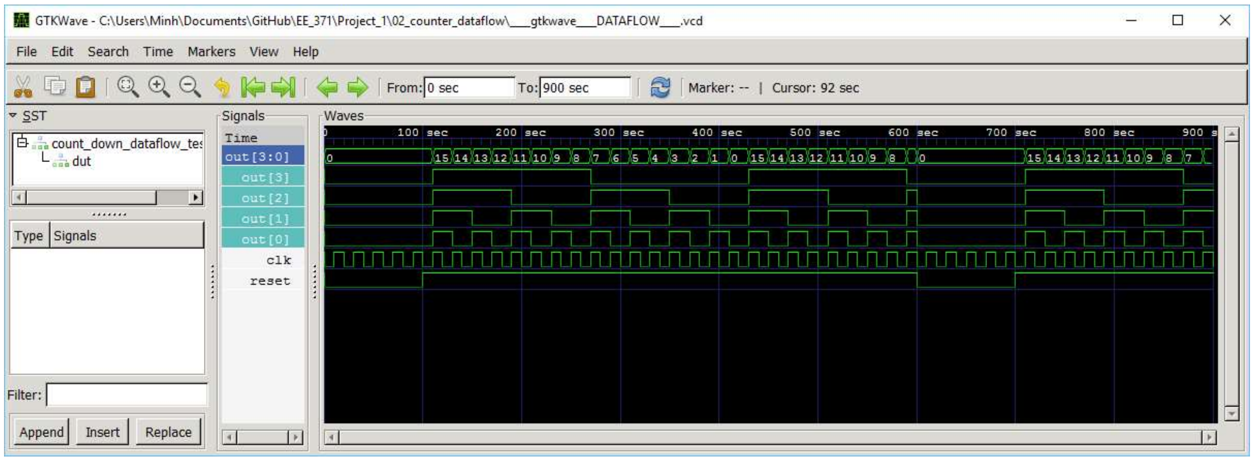


Figure 7. Four Bit Synchronous Behavioral Down Counter Waveform using gtkwave.

6.1.3 Synchronous Johnson Counter (Behavioral Model)

Figure 8 shows the simulation results of the 4-bit Johnson down counter. The Johnson counter has a different bit wise pattern than the other counters. The counter takes the right-most bit and moves the inversion of the signal into the left-most bit. The bit pattern for this counter will go through the pattern: 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000, etc. Some similarities between the counters is that they reset to 0 and bit patterns will repeat once it reaches its lowest value. The wrap around can be seen when the bit pattern is at 0000 and then changes back to 1000 in the simulation.

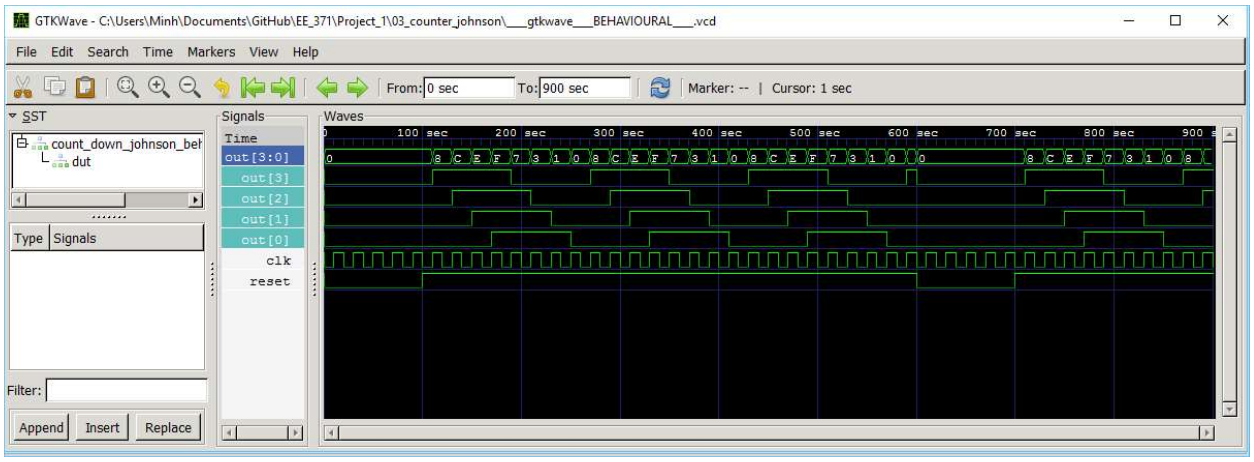

GTKWave - C:\Users\Minh\Documents\GitHub\EE_371\Project_1\03_counter_johnson\___gtkwave___BEHAVIOURAL___.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 900 sec
Marker: -- | Cursor: 1 sec
SST
count_down_johnson_beh
dut
Type Signals
Filter:
Append Insert Replace
Signals
Time
out[3:0]
out[3]
out[2]
out[1]
out[0]
clk
reset
Waves
100 sec
200 sec
300 sec
400 sec
500 sec
600 sec
700 sec
800 sec
900 s
0
8 C E F 7 3 1 0 8 C E F 7 3 1 0 8 C E F 7 3 1 0 0 8 C E F 7 3 1 0 8

Figure 8. Four Bit Johnson Down Counter Waveform using gtkwave.

6.2 Quartus II Waveform Tool

Figure 9 is the simulation results of the 4-bit synchronous down counter implemented using schematic entry. The inputs in this counter circuit affect the output the same exact way as the previous synchronous down counter. The simulation looks exactly like the results from the synchronous down counter.

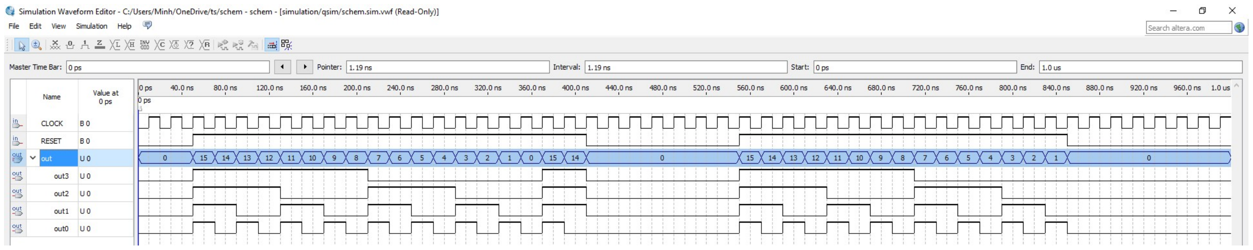


Figure 9. Four Bit Synchronous Schematic Down Counter Waveform using Quartus II Waveform Tool

6.3 Quartus II Signal Tap

The signal tap analysis allowed for real world analysis of the waveform on the hardware implementation of the four down counters. The sample depth was set to 1024 and the clock was set to the DE1_SoC board's 50 MHz internal clock. The test will analyze the waveforms of all of the four counters at two states: (1) Initial Reset and (2) 3rd stage.

6.3.1 DE1_SoC Board Testing at Initial Reset

6.3.1.1 Four Bit Ripple Down Counter (Structural Model)

Figure 10 show the Signal Tap II waveform of the output of the ripple counter. The counter will change to a 15 (1111 in binary) and increment down by one. As long as reset signal is high, the counter will increment down every clock cycle and wrap back around again. The change always occurs at the rising edge of the clock cycle. When the reset signal is 0, the counter changes to 0 (0000 in binary) and is stalled until the reset signal is 1 again. The counter will change to 0 independent of what the current bit pattern is displaying.

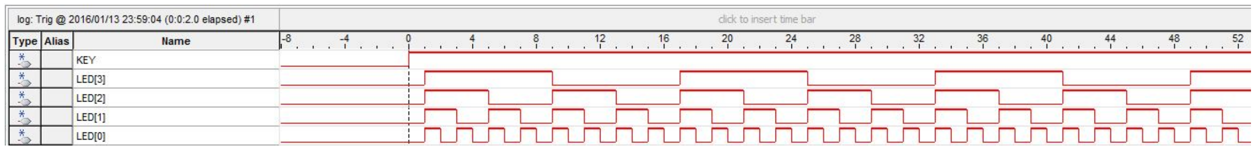


Figure 10. Four Bit Ripple Down Counter Waveform using Signal Tap II.

6.3.1.2 Synchronous Down Counter (Dataflow Model)

In Figure 11, the synchronous down counter follows the same output pattern as the ripple down counter. After the reset signal is high, the counter will change to a 15 (1111 in binary) and increment down by one. The counter will keep decreasing and will wrap back around again once it hits 0. If the reset signal is 0, the counter outputs a 0 and is stalled at that bit pattern until the reset changes to 1 again.

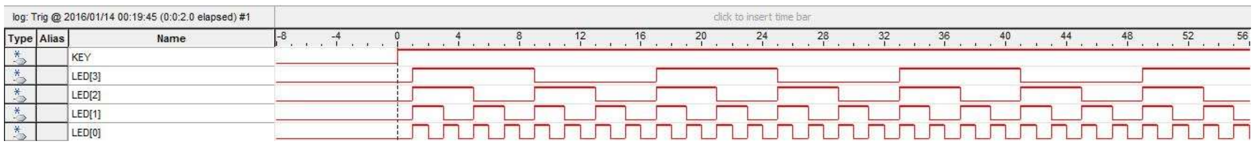


Figure 11. Four Bit Synchronous Behavioral Down Counter Waveform using Signal Tap II.

6.3.1.3 Synchronous Johnson Counter (Behavioral Model)

The Johnson down counter was implemented into the board. The counter has a different bit pattern than the ripple and synchronous down counters. In Figure 12 the bit pattern looks similar to a shift register. When the reset signal is 0, the counter changes to 0 independent of the current state and stalls at that bit pattern. Once the reset signal is 1, the counter begins to change its bit pattern. The pattern goes from 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000 and the repeats again as long as the reset signal is high.

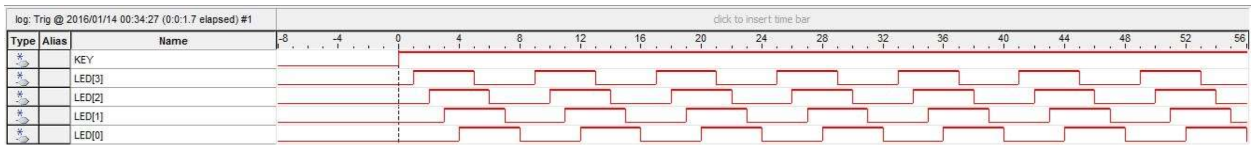


Figure 12. Four Bit Johnson Down Counter Waveform using Signal Tap II.

6.3.1.4 Four Bit Ripple Down Counter (Schematic Model)

Finally, the synchronous down counter was also designed using the schematic entry feature in Quartus. Using the given components in Quartus, the counter was built using D Flip Flops, inverters, AND, OR, and XOR gates. The reset signal and clock are still inputs into the system, and then the bit pattern is outputted onto the LEDs. The bit pattern that the counter follows is exactly like the previous implementation of the synchronous down counter and the ripple counter. Figure 13 is the output of the bit pattern displayed on the LEDs.

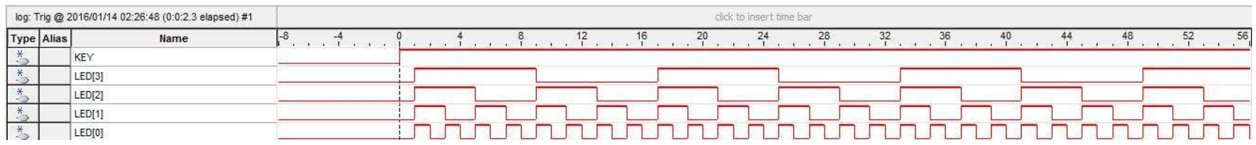


Figure 13. Four Bit Synchronous Schematic Down Counter Waveform using Signal Tap II.

6.3.2 Third State Analysis

The Figure 14, Figure 15, Figure 16, and Figure 17 displays the Signal Tap waveforms at the third state of the bit patterns for the four counters. Obtaining the waveforms beginning at the third state is beneficial because the user can gain experience using triggers. Using triggers can help debug if the time or location of the problem is known. As long as the conditions are known of the inputs and outputs, the waveform around the problem area can be found analyzed.

6.3.2.1 *Four Bit Ripple Down Counter (Structural Model)*

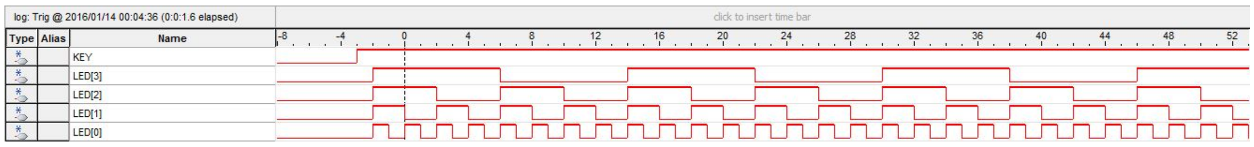


Figure 14. Four Bit Ripple Down Counter Waveform using Signal Tap II, 3rd stage.

6.3.2.2 *Synchronous Down Counter (Dataflow Model)*

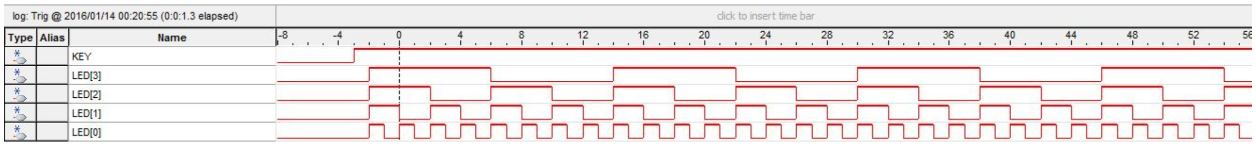


Figure 15. Four Bit Synchronous Behavioral Down Counter Waveform using Signal Tap II, 3rd stage.

6.3.2.3 *Synchronous Johnson Counter (Behavioral Model)*

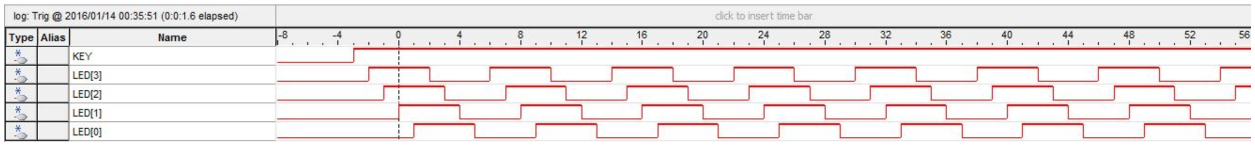


Figure 16. Four Bit Johnson Down Counter Waveform using Signal Tap II, 3rd stage.

6.3.2.4 *Four Bit Ripple Down Counter (Schematic Model)*

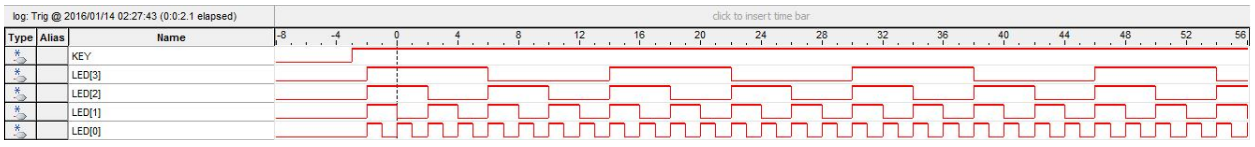


Figure 17. Four Bit Synchronous Schematic Down Counter Waveform using Signal Tap II, 3rd stage.

6.4 Original Structural Design vs RTL Viewer Design

Using the RTL viewer tool in Quartus, gate level implementations of the three counters were created. Some of the synthesized implementations of the counters were similar to our designs and some of the implementations had slight differences. Since the counter designs were simple, there won't be many differences between the Quartus version of the gate level implementations of the counters and gate level equivalents our group made.

6.4.1.1 *Four Bit Ripple Down Counter (Structural Model)*

In Figure 18, the structure of the counter is exactly the same as the ripple counter in Figure (). Both structures consist of four D Flip-Flops (DFF) and the wires are going into the same nodes on the drawings. The main characteristic of the ripple counter is that the output of the previous DFF is going into the current DFF's clock. Both of the structures have this characteristic.

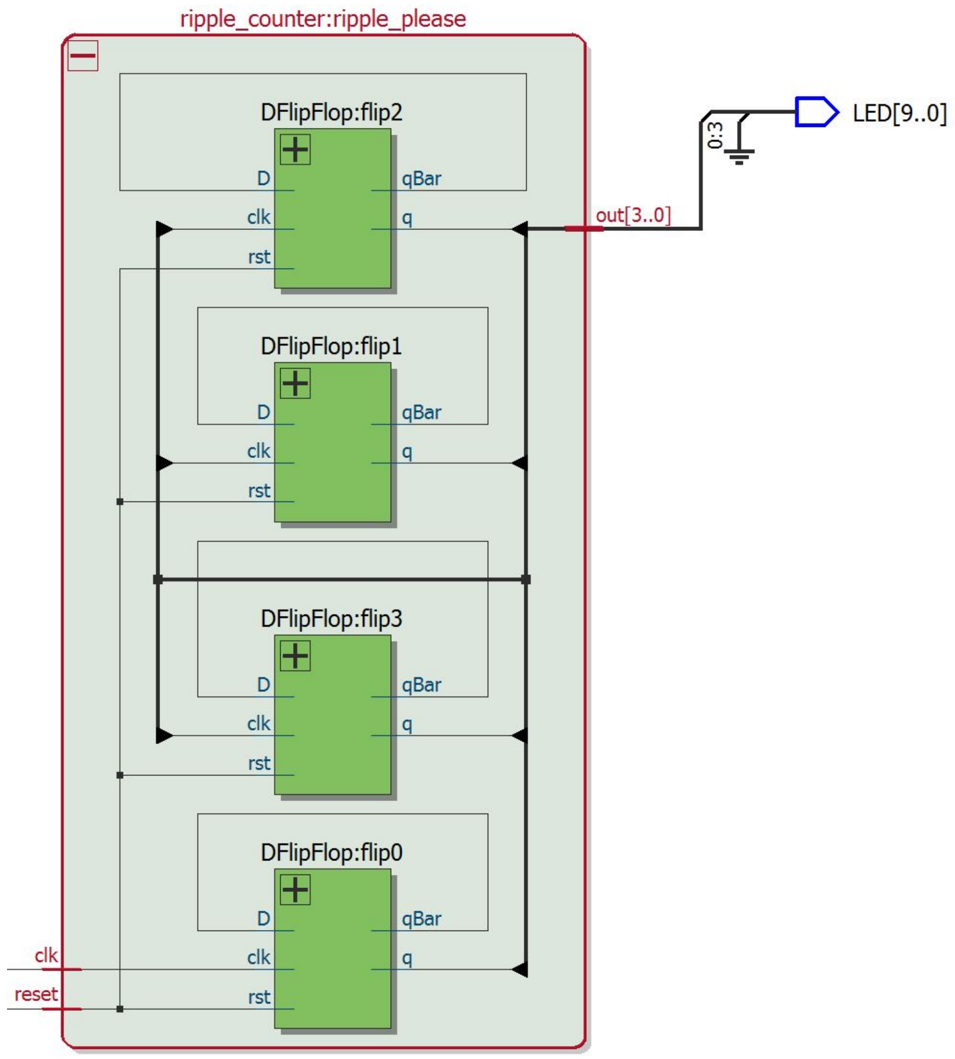


Figure 18. Four Bit Ripple Down Counter Schematic using RTL Viewer

6.4.1.2 Synchronous Down Counter (Dataflow Model)

In Figure 19, the structure of the synchronous down counter produced by the RTL Viewer tool is the same as our group's structural implementation. The inverter is switched to a bubble representation in the RTL Viewer, but are equivalent in how the component affects the circuit. Even though the dataflow model is higher level technique of modelling the counter, the difference between the Quartus version of the gate level implementation is the same our design of the synchronous down counter.

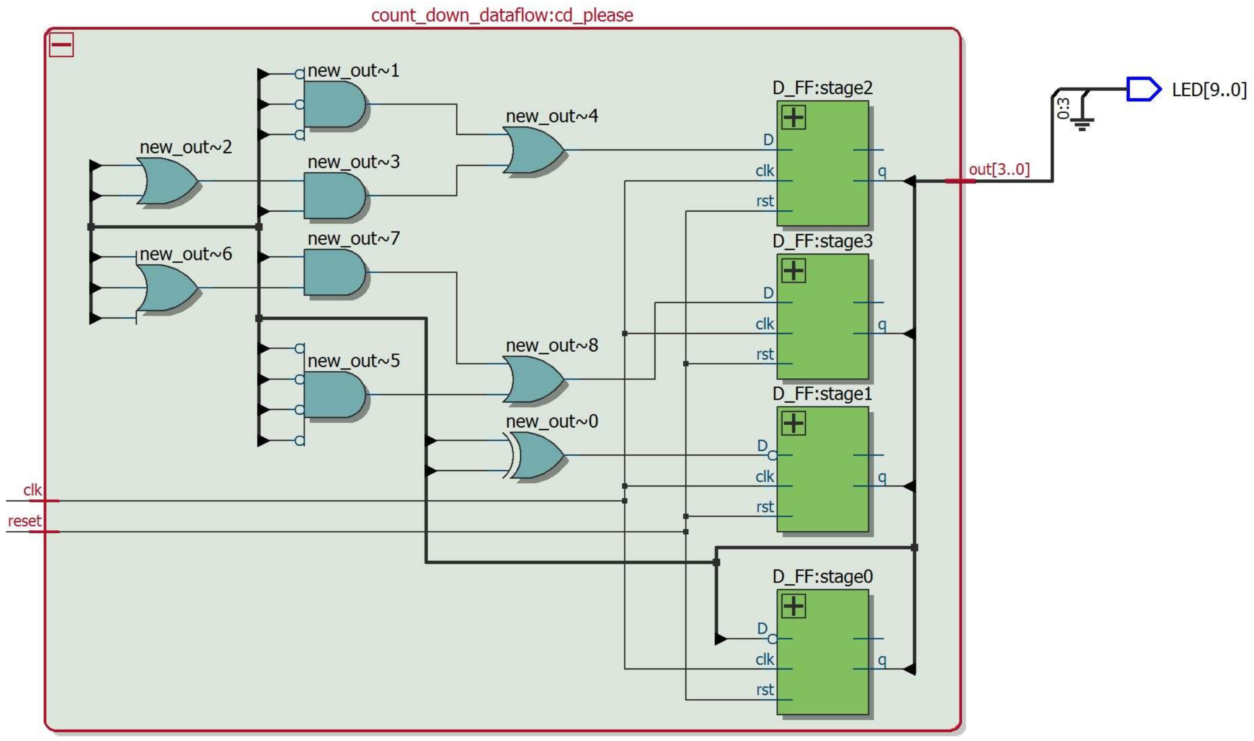


Figure 19. Four Bit Schematic Down Counter Schematic using RTL Viewer

6.4.1.3 Four Bit Johnson Down Counter (Schematic Model)

In Figure 20, the Johnson counter structure created by the Quartus RTL Viewer tool is slightly different than our original design. The difference between the two structures is that our group's design used four single DFFs but the Quartus version used a DFF that took in a bus of wires. The difference is expected since the behavioral model does not deal with any of the hardware and most of the time will produce a different gate level implementation.

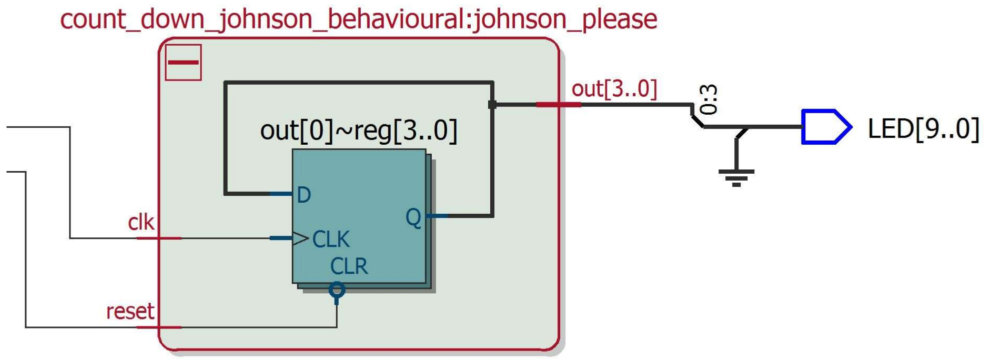


Figure 20. Four Bit Johnson Down Counter Schematic using RTL Viewer

6.5 Failure Mode Analysis

6.5.1 Four Bit Ripple Down Counter (Structural Model)

The first counter that will be examined is the ripple down counter. The counter has two inputs, reset and clock, that controls the system. The outputs go out to the LEDs. At SA0 for the outputs, the corresponding LEDs to those outputs will not light up and can possibly mess up the current bit pattern if the light is supposed to be on. At SA1 for the outputs, the corresponding LEDs to those outputs will light up and can possibly mess up the current bit pattern if the light is supposed to be off. If the reset signal is stuck at 1, the bit pattern will continue will decrement even if the key is being pressed and the signal should be a 0. If the reset signal is stuck at 0, the bit pattern will stall at 0 and not decrement even if the reset signal should be a 1. The last signal to look at is the clock signal. In both SA0 and SA1 mode, the counter will be stuck on the previous state since the components will not see the next rising edge to change its values.

6.5.2 Synchronous Down Counter (Dataflow Model)

The second counter that will be looked at is the synchronous down counter. First, reset will be looked at in SA0 mode. The reset will stall the counter at 0 when the reset is stuck at low. When the reset is in SA1 mode, the counter will continuously decrement down without nothing to stop the counter. The next signal is the clock. When the clock is stuck at 0 or 1, the counter never sees the positive edge of the clock so the counter will freeze and not be able to change. Finally, the output will be looked at in the different hazard modes. In SA0, the LEDs will just be off and could affect how the bit pattern should look at a current state. The same result can happen when the outputs are stuck at 1 because the LEDs will be on when might not need to be.

6.5.3 Four Bit Johnson Down Counter (Schematic Model)

The last counter that will be looked at is the Johnson down counter. Reset and the clock are the two inputs into the clock. When the clock is stuck at either high or low, the counter will freeze at the current stage and will not decrement. When the reset is stuck high, the counter will continuously decrement even if the key is being pressed to make the reset low. When the reset is stuck low, the counter will stall at 0. The outputs of the counter are the four LED lights. When any of the LED lights are stuck at 1, the bit pattern might be wrong at the current state because the LEDs could be on when it isn't supposed to. When any of the LED lights are stuck at 0, the bit pattern could be wrong at the current state because the LEDs could be off when they are meant to be on.

7. Analysis of Errors

7.1 Analysis of any errors

Throughout the lab, there was a couple of problems that caused a couple of errors. The first error is the reset signal is connected to the switch. Signal Tap waveform was inconsistent and kept oscillating on and off because the switch bounces when it is being moved. This instability was caused because of the transient state of the electricity at the moment of switch. This created an instable signal that caused the digital system to be unable to clearly define a high or low signal, the voltage on the switch was alternating above and below the voltage threshold of the system. As a result, instead of using the switch, the key was used in replacement for the reset signal. The key gave a more solid signal when it was either on or off and took a shorter time to settle at a value.

Another error was using the correct model number for the Quartus project. The model number affects the type of pins that can be assigned to the inputs and outputs. The number of pins that needed to be hooked up to the board was unavailable when the model number was chosen incorrectly. Searching through the user manuals of the DE1-SoC board showed the exact model number that were needed to create the project and program the board.

The final problem that was encountered was the schematic entry feature on Quartus. There was a bug when the 4-bit was set to an output bus rather than individual wires. When the schematic was compiled, the Verilog HDL file was created but the output was not wired up at all. The wire was just dangling and would not get a signal. The solutions to this problem was to either split the bus to four individual wires or name the wire connecting to the output. After doing one of these solutions, the created Verilog HDL file would have its output connected to an actual signal.

7.2 Analysis of possible errors

There are many problems that could have arisen from this lab that our group did not encounter. Some likely errors are if the clock signal from the clock divider is too fast or if there is miss wiring in the counter modules. Synchronous down counter is affected by the fast clock because there is a lot of logic gates which would accrue a larger gate delay. The ripple down counter will be affected as well since the D Flip Flops is wired up sequentially so the top most bit would accrue a higher gate delay. Miss wiring in the Verilog is always a big issue and can cause catastrophic issues. Wiring up to the wrong LED will cause the wrong bit pattern. Wiring up the reset key to the wrong pin will make the counter uncontrollable since that input is needed start and stop the counters.

8. Summary and Conclusion

Overall, the lab covered many things that have previously been covered and new things that are just being introduced. First, counters were built using different modelling levels in Verilog HDL. The counters were then compiled and then simulated using the gtkwave tool. The simulations gave a nice representation of the output waveforms that could be produced with the code. Then, the counters were programmed onto the board to display the counters on the LEDs. Using the Signal Tap II Analyzer tool, the actual output waveforms can be produced and compared to the simulated result from gtkwave. At the end of the lab, a c-program was developed to convert currency using a given exchange rate, amount, and the converted beer price in the foreign currency.

Tools were introduced in this lab that will be useful in future labs. Learning new tools will be essential in the industry since not all places will have familiar tools or some places will have a superior tool. The lab also demonstrated that even if simulations are producing the right output, the actual results on the board might differ. Using the Signal Tap tool, we determined that the key produces a better input signal than the switch. The counters were simple structures, but was also easier to compare the modeling levels with each other. The lab supplied a review of content from previous embedded courses and also gave an introduction to new tools and what future labs might hold.

9. Appendix (C Program)

This section will explain the currency exchange program for the lab.

9.1 Design Requirements

9.1.1 Abstract

In this part of the lab, we were introduced to the CodeBlocks learning environment for developing C code. We created a C program that functions as a currency exchange calculator.

9.1.2 Introduction

A currency conversion calculator is an important tool to have at hand with the increasing global economy coupled with lower airfare. To approach this problem, we created a C program with a simple user interface that allows to convert currency from US dollars to the desired foreign currency. In addition to that, it also lets the user know the price of a good old bottle of corona in the foreign currency.

9.1.3 Inputs to Calculator

The inputs to the calculator command line are as follows:

1. Amount in US Dollars/ Foreign Currency
2. The current exchange rate for the foreign currency
3. Number (1) if currency is converted from US Dollars to Foreign Currency
4. Number (2) if currency is converted from Foreign Currency to US Dollars

9.1.4 Outputs from Calculator

The outputs produced by the calculator are as follows:

1. Amount of corresponding Foreign Currency/or US Dollars (depending on the option the user chose)
2. The cost of a bottle of corona beer in foreign currency.

9.1.5 Major Functions

This program contains four major functions:

1. int main (int argc, char** argv) : Consists of lines of code to execute methods and initial argument interpretation.
2. void convertUStoForeign (float amt, float exhange) : converts and prints value from US Dollars to user specified foreign currency.
3. void convertForeigntoUS (float amt, float exhange) : converts and prints value from Foreign Currency to US Dollars.
4. void priceOfBeer(float exhange) : prints the value of a bottle of corona in foreign currency.

9.2 Design Specifications

9.2.1 Abstract

Our task comprised of using the Command-Line Tool Interface to send arguments to a C program file as well as compile and run the C program for calculating foreign currency and US dollars conversion rates.

9.2.2 Introduction

A currency conversion calculator is an important tool to have at hand with the increasing global economy coupled with lower airfare. To approach this problem, we created a C program with a simple user interface that allows to convert currency from US dollars to the desired foreign currency. In addition to that, it also lets the user know the price of a good old bottle of corona in the foreign currency.

9.2.3 Inputs to Calculator

The inputs to the calculator consisted of arguments passed through the program using the command line interface. The first argument passed as input was the amount of money (foreign or US dollars) that needed to be converted, which was represented as a float in C. The second value that was passed in is the float value for the exchange rate as appropriate for the conversion. The last value that is passed in is an integer (1 or 2 only) that determines whether conversion should be from US to Foreign Currency or vice versa.

9.2.4 Outputs from Calculator

The C program outputs to the stdout (command line terminal or IDE developing environment console). If the user chooses option 1, the calculator prints the value of the foreign currency after conversion from the US Dollars with the given exchange rate along with the price of beer in foreign currency. If the user chooses option 2, the calculator prints the value of US dollars after converting it from foreign currency. If the inputs to the program were invalid, the calculator prints an error message.

9.2.5 Major Functions

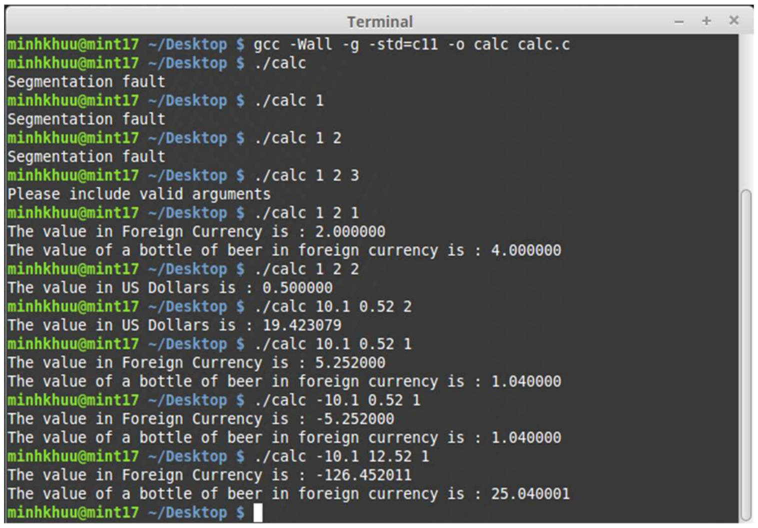
This program contains four major functions:

1. Main Function: This is the first method that is executing in any C program. It accepts the command line arguments which are specified in the inputs sections above. It uses the functions on atof and atoi to convert the command line arguments that are passed in as strings to floats and ints. The main program uses these arguments to call different methods to produce the results user needs. If the 3rd argument is 1, the method convertUStoForeign is called, if 2 is passed convertForeignToUS is called instead. If neither one of these ints are passed, an error method is printed out.
2. convertUStoForeign: Converts the US Dollars into the Foreign Currency. It takes in a float for currency and a float for exchange rate passed in as parameters. It then multiplies the two values and prints out the result as the resulting foreign currency value. This method also calls the priceOfBeer function which prints out the value of a beer in foreign currency along with this on the terminal.
3. convertForeigntoUS: Converts the Foreign currency to US Dollars. It takes in a float for currency and a float for exchange rate passed in as parameters. It then divides the two values and prints out the result as the resulting US Dollars value.

This section will discuss the C currency exchange program.

9.3 Results and Analysis

The results of the C program behaves as expected. When the user inputs valid arguments that follows the design specification, the program outputs a correct and corresponding result (see Figure 21).



```
minhkhuu@mint17 ~/Desktop $ gcc -Wall -g -std=c11 -o calc calc.c
minhkhuu@mint17 ~/Desktop $ ./calc
Segmentation fault
minhkhuu@mint17 ~/Desktop $ ./calc 1
Segmentation fault
minhkhuu@mint17 ~/Desktop $ ./calc 1 2
Segmentation fault
minhkhuu@mint17 ~/Desktop $ ./calc 1 2 3
Please include valid arguments
minhkhuu@mint17 ~/Desktop $ ./calc 1 2 1
The value in Foreign Currency is : 2.000000
The value of a bottle of beer in foreign currency is : 4.000000
minhkhuu@mint17 ~/Desktop $ ./calc 1 2 2
The value in US Dollars is : 0.500000
minhkhuu@mint17 ~/Desktop $ ./calc 10.1 0.52 2
The value in US Dollars is : 19.423079
minhkhuu@mint17 ~/Desktop $ ./calc 10.1 0.52 1
The value in Foreign Currency is : 5.252000
The value of a bottle of beer in foreign currency is : 1.040000
minhkhuu@mint17 ~/Desktop $ ./calc -10.1 0.52 1
The value in Foreign Currency is : -5.252000
The value of a bottle of beer in foreign currency is : 1.040000
minhkhuu@mint17 ~/Desktop $ ./calc -10.1 12.52 1
The value in Foreign Currency is : -126.452011
The value of a bottle of beer in foreign currency is : 25.040001
minhkhuu@mint17 ~/Desktop $
```

Figure 21. Demonstration of the C currency exchange program.