**Project 1**

**Introduction to the Lab Environment**

**EE 371 AC**

**January 08, 2016**

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

Table of Contents:

[1. Abstract 3](#_Toc440458142)

[2. Introduction 3](#_Toc440458143)

[3. Discussion of the Project 3](#_Toc440458144)

[3.1 Design Specification 3](#_Toc440458145)

[3.1.1 Four Bit Ripple Down Counter 3](#_Toc440458146)

[3.1.2 Synchronous Down Counter (Dataflow Model) 3](#_Toc440458147)

[3.1.3 Synchronous Johnson Counter (Behavioral Model) 4](#_Toc440458148)

[3.1.4 Synchronous Down Counter (Schematic Entry) 4](#_Toc440458149)

[3.2 Design Procedure 5](#_Toc440458150)

[3.2.1 Four Bit Ripple Down Counter 5](#_Toc440458151)

[3.2.2 Synchronous Down Counter (Dataflow Model) 6](#_Toc440458152)

[3.2.3 Synchronous Johnson Counter (Behavioral Model) 7](#_Toc440458153)

[3.2.4 Synchronous Down Counter (Schematic Entry) 8](#_Toc440458154)

[3.3 System Description 8](#_Toc440458155)

[3.4 Software Implementation 8](#_Toc440458156)

[3.5 Hardware Implementation 8](#_Toc440458157)

[4. Testing 8](#_Toc440458158)

[4.1 Test Plan 8](#_Toc440458159)

[4.1.1 iVerilog and gtkwave 9](#_Toc440458160)

[4.1.2 Quartas II and Signal Tap 9](#_Toc440458161)

[4.1.3 Quartas II and Waveform tool 9](#_Toc440458162)

[4.2 Test Specification 9](#_Toc440458163)

[4.2.1 Reset Signal is set low 10](#_Toc440458164)

[4.2.2 Reset Signal is set high 10](#_Toc440458165)

[4.3 Test Cases 11](#_Toc440458166)

[4.3.1 iVerilog and gtkwave 11](#_Toc440458167)

[4.3.2 Quartas II Signal Tap 12](#_Toc440458168)

[4.3.3 Quartas II Waveform tool 12](#_Toc440458169)

[5. Presentation, Discussion, and Analysis of the Results 12](#_Toc440458170)

[6. Analysis 12](#_Toc440458171)

[6.1 Analysis of any errors 12](#_Toc440458172)

[6.2 Analysis of possible errors 12](#_Toc440458173)

[6.3 Feedback upon the analysis 12](#_Toc440458174)

[7. Summary and Conclusion 12](#_Toc440458175)

[8. Appendices 13](#_Toc440458176)

# Abstract

The goal of this lab was to understand different developing environments such as iVerilog and gtkwave and Signal Tap Tool in the Quartus II IDE for developing Verilog code. We achieved this by making four 4-bit down counters, namely the ripple-down counter, a four stage synchronous down counter and a Johnson down counter. We also familiarized ourselves with the basics of C programming by writing a simple currency conversion calculator.

# Introduction

In this lab, our main focus was to analyze the process of model design and testing process by implementing different down counters in Verilog. To understand this process,

The tools we used in this lab are:

* iVerilog
* gtkwave
* Quartus II Signal Tap
* Cyclone V FPGA
* Command line tool
* GCC C Compiler
* CodeBlocks

# Discussion of the Project

## Design Specification

### Four Bit Ripple Down Counter

A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. The ripple counter ranges from 0000 to 1111. On reset, the counter starts at 0000, then goes to 1111, 1110, 1101, … till 0000 again.

According to the design specifications given to us, we made 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. The advantages of this design is that it is has a lot of implementation details.

In our design, we drew out the required bit patter results for the ripple counter and formed a Boolean equation from it. We used this Boolean equation as a reference for designing and constructing or circuit at the gate level in Verilog. This table, equations and circuit is shown in the design procedure for ripple down counter. We also used the code snippet provided to us for the D flip flop in the spec.

### Synchronous Down Counter (Dataflow Model)

In a synchronous down counter, all the output bits change state simultaneously. We designed the counter using the data flow model as mentioned in the spec with an active low reset. In a dataflow design, a module is implemented by specifying the movement of the data. This allowed us to use assign statements instead of using AND gates in our module for the counter. We also used the code snippet provided to us for the D flip flop in the spec.

In our design, we drew out the bit pattern which ranged from 0000 to 1111. On reset, the counter starts at 0000, then goes to 1111, 1110, 1101, … till 0000 again. We also made use of K-maps to determine the new state value using combinational logic and present values.

### Synchronous Johnson Counter (Behavioral Model)

The 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.

We designed the Johnson Counter using a behavioral model with reset on active low. In behavioral modelling, the underlying hardware is abstracted away based upon an algorithmic description. The Johnson 4-bit counter only takes 8 different values. At reset the counter is set to 0000. It follows the pattern of 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001.

### Synchronous Down Counter (Schematic Entry)

The syn

## Design Procedure

### Four Bit Ripple Down Counter

Table 1. Four Bit Ripple and Synchronous Down Counter Truth Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Q[3]** | **Q[2]** | **Q[1]** | **Q[0]** |
| **0** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **0** |
| **1** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** |
| **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** |
| **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **0** |
| **0** | **0** | **0** | **1** |
| **0** | **0** | **0** | **0** |

**Boolean Equation:**

|  |
| --- |
| 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.

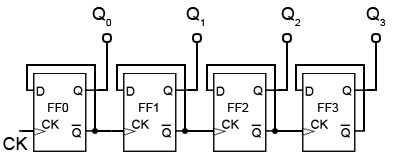


Figure **2.** Ripple Down Counter Structural Diagram from Quartas.

### Synchronous Down Counter (Dataflow Model)

#### Truth Table

Truth table that we used for the synchronous counter is the same one as the Table 1.

#### Boolean Equations

By observation, the Boolean equations are as follows:

(1)

Q [0]

Q [1]

|  |  |
| --- | --- |
| 1 | 0 |
| 0 | 1 |

(2)

Q [1]

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |

(3)

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |

(4)

#### Diagram

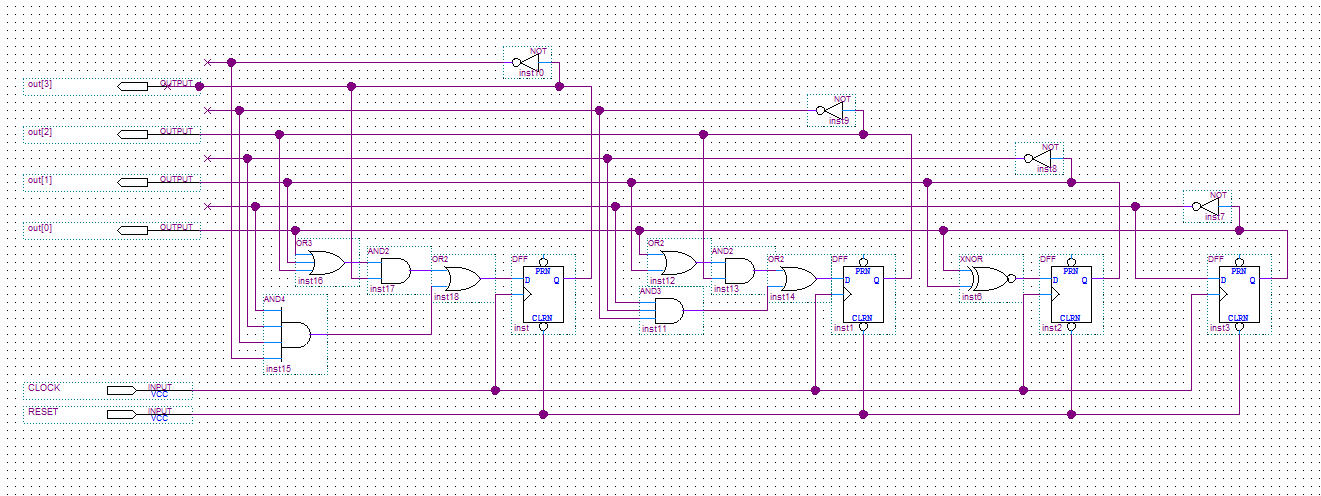


Figure **3.** Synchronous Down Counter Structural Diagram Implementation from Quartas.

### Synchronous Johnson Counter (Behavioral Model)

Table 2. Four Bit Johnson Down Counter Truth Table

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **0** |
| **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** |
| **0** | **0** | **0** | **1** |

**Diagram**

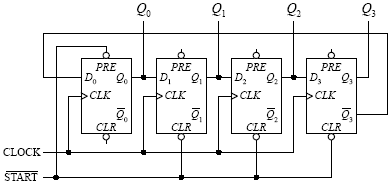


Figure **4.** Johnson Down Counter Structural Diagram Implementation from Quartas.

### Synchronous Down Counter (Schematic Entry)

## System Description

DENNY LY

## Software Implementation

Please refer to the code submitted via canvas dropbox.

## Hardware Implementation

We used the DE1\_SOC FPGA board.

# Testing

## 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 1 for the Ripple and Synchronous down counter and Table 2 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
* Quartas II and Signal Tap
* Quartas II and Waveform tool for .bdf files

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

### Quartas 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 Quartas 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. Load the .sof file into the DE1-SoC board.
7. Run the Signal Tap and utilize the input signal to activate the trigger condition.
8. Analyze the waveform to see if the waveform has any similarities or irregularities from predicted results from the test specifications.

### Quartas II and Waveform tool

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

1. Create the schematic file using Quartas.
2. Synthesize the schematic file.
3. Open up the waveform tool.
4. Locate the output and 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.

## 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.

### 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
  + Independent of the clock cycle meaning the output can be change to 0000 during any stage of the clock cycle.

### 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.

#### 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.

#### 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 1000.
    - Otherwise, the next output will “decrement” “Johnson style”
      * 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.

## Test Cases

The test cases were conducted using the procedures described in Section 4.1 for each of the corresponding testing methods.

### iVerilog and gtkwave

#### Four Bit Ripple Down Counter

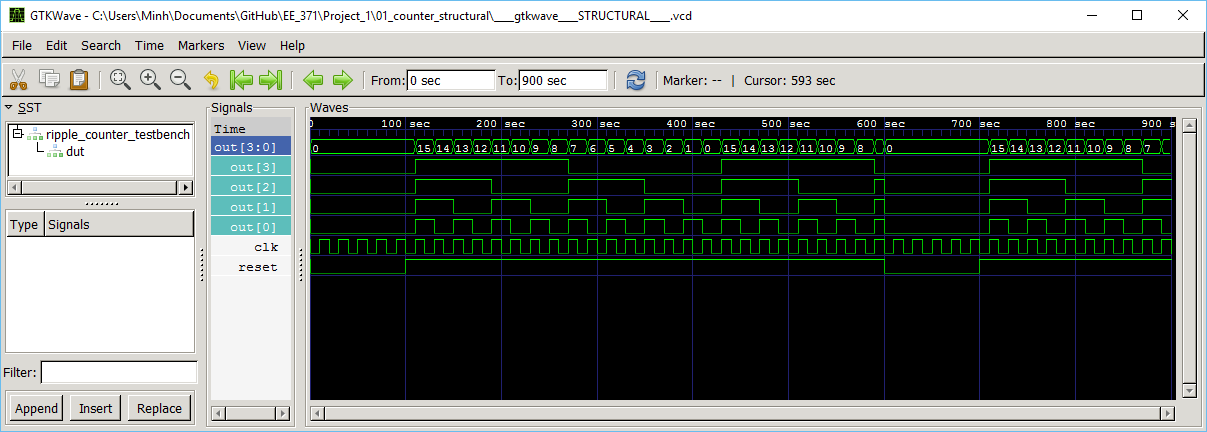


Figure **5.** Four Bit Ripple Down Counter Waveform using gtkwave.

#### Synchronous Down Counter (Dataflow Model)

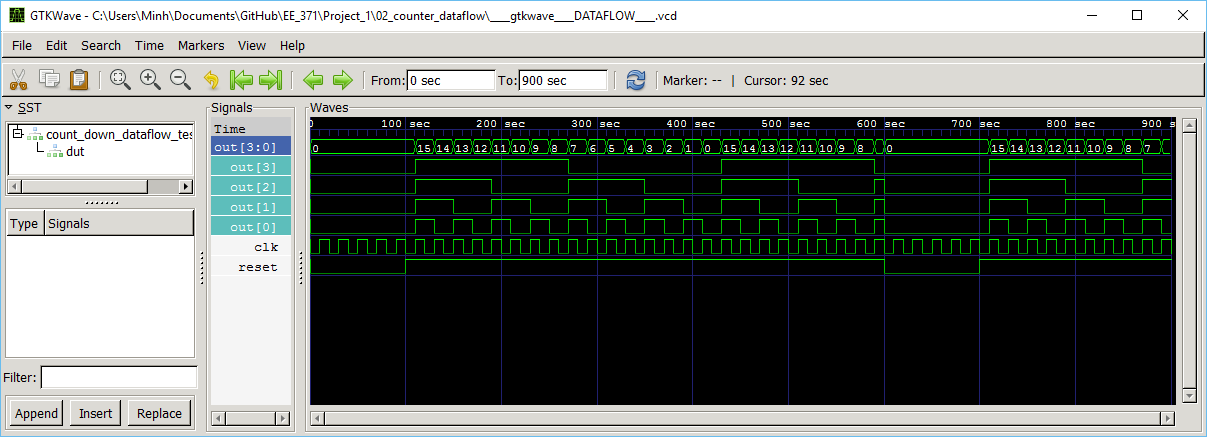


Figure **6.** Four Bit Synchronous Down Counter Waveform using gtkwave.

#### Synchronous Johnson Counter (Behavioral Model)

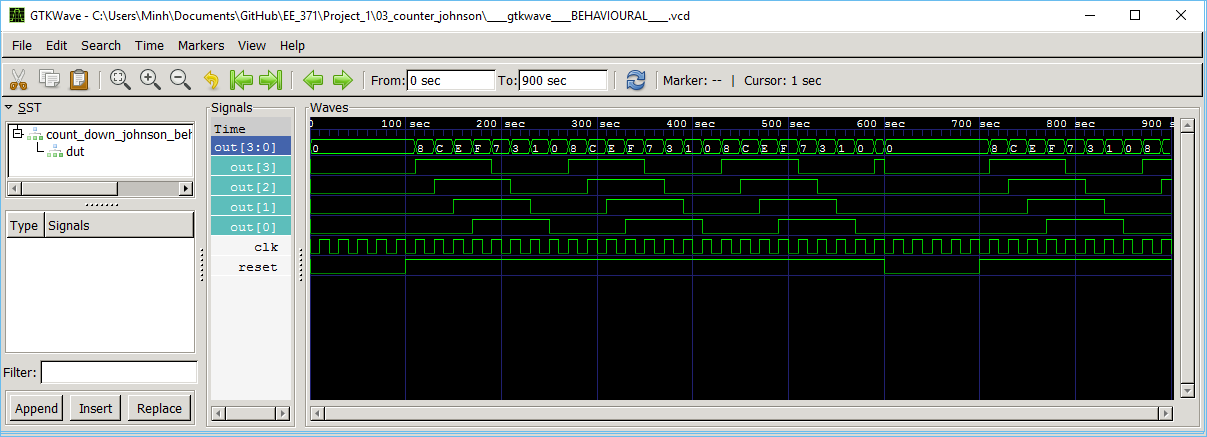


Figure **7.** Four Bit Johnson Down Counter Waveform using gtkwave.

### Quartas II Signal Tap

### Quartas II Waveform tool

# Presentation, Discussion, and Analysis of the Results

# Analysis

## Analysis of any errors

Software and tool problems. Iverilog not running because it is not included in windows path variable.

Problem wiring modules.

## Analysis of possible errors

One possible error is by selecting a clock speed that is too fast. This causes the gate delays to have a more noticeable effect.

Verilog syntax errors.

## Feedback upon the analysis

# Summary and Conclusion

# Appendices