Laboratory Exercise A

Counters

TCES 330 Digital Systems Design

Spring 2016

**Authors:**

Zeeshan Karin \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rand Almaroof \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Brian Wolk \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submission Date: April 27, 2016

**Table of Contents**

[Laboratory Assignment 1 1](#_Toc224977355)

[Requirements: 1](#_Toc224977356)

[Part I 1](#_Toc224977357)

[Part II 1](#_Toc224977358)

[Part III 2](#_Toc224977359)

[Part IV 2](#_Toc224977360)

[Part V 2](#_Toc224977361)

[Design 2](#_Toc224977363)

[Part I 2](#_Toc224977364)

[Part II 2](#_Toc224977365)

[Part III 3](#_Toc224977366)

[Part IV 3](#_Toc224977367)

[Part V 3](#_Toc224977368)

[Test Procedures 4](#_Toc224977370)

[Part I 4](#_Toc224977371)

[Part II 4](#_Toc224977372)

[Part III 4](#_Toc224977373)

[Part IV 5](#_Toc224977374)

[Part V 5](#_Toc224977375)

[Test Results 5](#_Toc224977377)

[Part I 5](#_Toc224977378)

[Part II 5](#_Toc224977379)

[Part III 7](#_Toc224977380)

[Part IV 7](#_Toc224977381)

[Part V 8](#_Toc224977382)

[Observations 8](#_Toc224977384)

[Part I 8](#_Toc224977385)

[Part II 8](#_Toc224977386)

[Part III 8](#_Toc224977387)

[Part IV 8](#_Toc224977388)

[Part V 8](#_Toc224977389)

[Conclusion 8](#_Toc224977391)

[Apendix A 8](#_Toc224977392)

# Laboratory Assignment A

The purpose of this exercise is to learn how to use counters with verilog and implement circuits that use counters into the DE2-115 board. We will use the switches *SW*1*−*0, *KEY*0 and the built in 50-MHz clock on the DE2 board as inputs to the circuit and 7-segment displays as output devices.

## Requirements:

The objectives of each part of the lab assignment are described in this section of the report.

### Part I

Consider the circuit in Figure 1. It is a 4-bit synchronous counter which uses four T-type flip-flops. The counter increments its count on each positive edge of the clock if the Enable signal is asserted. The counter is reset to 0 by using the Reset signal. You are to implement a 16-bit counter of this type.

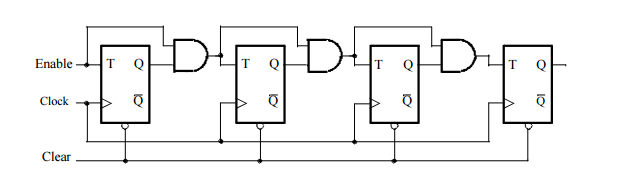


Figure . A 4-bit counter.

### 

1. Write a Verilog file that defines a 16-bit counter by using the structure depicted in Figure 1. Your code should include a T flip-flop module that is instantiated 16 times to create the counter. Compile the circuit. How many logic elements (LEs) are used to implement your circuit? What is the maximum frequency, Fmax, at which your circuit can be operated?
2. Simulate your circuit to verify its correctness.
3. Augment your Verilog file to use the pushbutton KEY0as the Clock input, switches SW1and SW0as Enable and Reset inputs, and 7-segment displays HEX3-0 to display the hexadecimal count as your circuit operates. Make the necessary pin assignments needed to implement the circuit on the DE2-115 board, and compile the circuit.
4. Download your circuit into the FPGA chip and test its functionality by operating the implemented switches.
5. Implement a 4-bit version of your circuit and use the Quartus II RTL Viewer to see how Quartus II software synthesized your circuit. What are the differences in comparison with Figure 1?

### Part II

Simplify your Verilog code so that the counter specification is based on the Verilog statement

Q <= Q + 1;

Compile a 16-bit version of this counter and compare the number of LEs needed and the Fmax that is attainable. Use the RTL Viewer to see the structure of this implementation and comment on the differences with the design from Part I.

### Part III

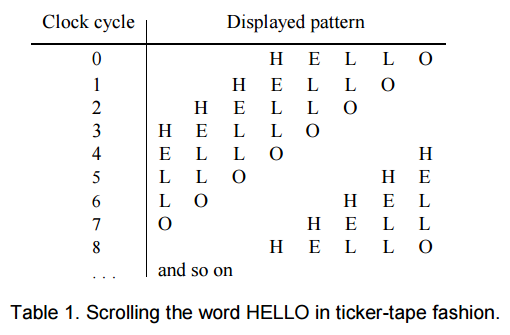
Use an LPM from the Library of Parameterized modules to implement a 16-bit counter. Choose the LPM options to be consistent with the above design, i.e. with enable and synchronous clear. How does this version compare with the previous designs?

### Part IV

Design and implement a circuit that successively flashes digits 0 through 9 on the 7-segment display HEX 0. Each digit should be displayed for about one second. Use a counter to determine the one-second intervals. The counter should be incremented by the 50-MHz clock signal provided on the DE2-115 board. Do not derive any other clock signals in your design–make sure that all flip-flops in your circuit are clocked directly by the 50 MHz clock signal.

### Part V

Design and implement a circuit that displays the word HELLO, in ticker tape fashion, on the eight 7-segment displays HEX 7 − 0. Make the letters move from right to left in intervals of about one second. The patterns that should be displayed in successive clock intervals are given in Table 1.



## Design

This section of the report describes our analysis of the requirements for this laboratory exercise and the resulting project design.

### Part I

The first sub-task is to write a Verilog module that works as a Toggle flip-flop which will have the following properties based on the definition of a T flip-flop.

1. Has three inputs, a clock input Clk, reset input ClrN, and the control input T
2. Produces two outputs Q and QN, in which QN is the negation of Q
3. On the positive edge of each clock cycle, if T is asserted the output Q is negated to QN for the next clock cycle. If ClrN is asserted, Q is set to 0.

The module that accomplishes this task is provided in Appendix A, figure 2.

This module is stored in the file TFFx.v, which can be found in our Part1 folder.

The next sub-task is to create a module that creates the 16-bit counter in figure 1 using the module for the T flip-flop. The following properties describe the 16-bit counter.

1. Has three inputs; a clock input Clock, an enable input Enable, reset input Clear
2. Has a 16 bit output Q that will count to Q + 1 on every positive Clock edge when Enable is asserted, and clear to zero if Clear is asserted.

The module that accomplishes this task is provided in Appendix A, figure 3.

This module is stored in the file CountNG.v, which can be found in our Part1 folder.

The next sub-task is to create a Verilog module that decodes a 4-bit input and allows output to 7-segment hex displays in hexadecimal format 0-F.

1. 4-bit input s, representing a binary numerical input 0000-1111
2. 7 bit output hex that displays proper character based on input value

The module that accomplishes this task is provided in Appendix A, figure 4.

This module is stored in the file BCDto7Seg.v, which can be found in our Part1 folder.

Finally, we need a module that interfaces to the DE2-115 board using the switch and output specifications below and implements the 16-bit counter using the previous modules.

1. Use KEY0 as the Clock input;
2. SW1 as the Enable input
3. SW0 as the Reset input
4. 7-segment displays HEX3-0 as outputs

The module that accomplishes this task is provided in Appendix A, figure 5.

The top level module is part1.v and is included in the Part1 folder.

All of the above is in the Quartus II project named Part1 contained in a folder of the same name.

### Part II

The modules from Part I are reused with the exception of CountNG, which is simplified using the Verilog statement

Q <= Q + 1;

The module that accomplishes this is called count16b.v and is included in the Part2 folder and figure 6 in the appendix A.

All of the above is in the Quartus II project named Part2 contained in a folder of the same name.

### Part III

The modules from Part I are reused with the exception of CountNG, which is replaced with a 16-bit counter module generated using the Library of Parameterized Modules. This generates a black box module that meets the requirements for the counter module, the corresponding files are included in the Part3 folder.

### Part IV

The module for the 4-bit binary to Hex display from part I is reused. In addition, we needed to create a module capable of using the 50MHz built in clock signal of the DE2-115 to determine one second intervals for flashing the display. The following characteristics were met:

1. An enable input En and clock input Clk
2. An output that asserts every one second or 50,000 cycles Pulse

The module that accomplishes this is called clocktopulse.v and is included in the Part4 folder and the appendix figure 7.

A 4 bit counter from 0-9 is implemented as well in the file count4b.v in the Part4 folder. Figure 8.

Finally a module for interfacing with the DE2-115 board was designed with the following specifications.

1. CLOCK\_50 as the built in clock input, SW[0] as an enable signal and SW[1] as a clear signal
2. HEX0 used for output of the digits 0-9

All of the above is in the Quartus II project named Part4 contained in a folder of the same name.

### Part V

We were able to repurpose many of the modules for this lab from previous Altera lab assignments, including the clocktopulse.v module from part IV and the modules for a 3 bit wide 8 to 1 multiplexor Mux3w\_8to1.v, Mux8\_1.v, Mux2\_1.v and HexHELO.v. We needed a module for counting from 0-7 which is included in the appendix as count3b.v, and our interface module met the following specifications:

1. Inputs of CLOCK\_50 for timing, SW[16] for clear and SW[15] for enable, and SW[14:0] for setting the proper character inputs.
2. Outputs of Hex displays HEX0, HEX1…HEX6, HEX7.

The module meeting these requirements is named part5.v and is in the Part5 folder and the appendix figure 9.

All of the above is in the Quartus II project named Part5 contained in a folder of the same name

## Test Procedures

The following test procedures will be used to verify that each part of this laboratory exercise satisfies the requirements given in the Requirements section, above.

### Part I

Testing and verification of the Part1 project will be accomplished in two parts. First the module CountNG.v will be simulated using ModelSim. Then the full Quartus project, Part1, will be compiled and uploaded to the DE2-115 board.

ModelSim

The CountNG.v module will be tested using ModelSim and the testbench shown in Figure 10, Appendix A.

1. Open the project in ModelSim and begin the simulation.
2. The testbench generates a 50MHz clock cycle and increments a counter, and compares the counter to the output of CountNG for 66000 iterations.
3. If the test is successful, the display will show “Successfully checked 66000 iterations”. If an error occurs the simulation will stop and display “Error at [count]” where count is the current iteration.

DE2-115

1. Open the project Part1 in Quartus and verify that compilation produces no errors or un-allowed warnings.
2. Load the project onto the DE2-115 board without errors.
3. Verify that the counter operates as expected by repeatedly asserting the KEY0 switch
4. Test the enable and reset inputs at SW[1] and SW[0].
5. Observe whether the hex displays and the circuit behaves as expected.

### Part II

Testing and verification of the Part2 project will be accomplished in two parts. First the module Count16b.v will be simulated using ModelSim. Then the full Quartus project, Part2, will be compiled and uploaded to the DE2-115 board.

ModelSim

The Count16b.v module will be tested using ModelSim and the testbench shown in Figure 11, Appendix A.

1. Open the project in ModelSim and begin the simulation.
2. The testbench generates a 50MHz clock cycle and increments a counter, and compares the counter to the output of Count16b for 66000 iterations.
3. If the test is successful, the display will show “Successfully checked 66000 iterations”. If an error occurs the simulation will stop and display “Error at [count]” where count is the current iteration.

DE2-115

1. Open the project Part2 in Quartus and verify that compilation produces no errors or un-allowed warnings.
2. Load the project onto the DE2-115 board without errors.
3. Verify that the counter operates as expected by repeatedly asserting the KEY0 switch
4. Test the enable and reset inputs at SW[1] and SW[0].
5. Observe whether the hex displays and the circuit behaves as expected.

### Part III

The part3 project will be tested by compilation and uploading to the DE2-115 board.

DE2-115

1. Open the project Part3 in Quartus and verify that compilation produces no errors or un-allowed warnings.
2. Load the project onto the DE2-115 board without errors.
3. Verify that the counter operates as expected by repeatedly asserting the KEY0 switch
4. Test the enable and reset inputs at SW[1] and SW[0].
5. Observe whether the hex displays and the circuit behaves as expected.

### Part IV

The Part4 project will be tested by compilation and uploading to the DE2-115 board.

DE2-115

1. Open the project Part4 in Quartus and verify that compilation produces no errors or un-allowed warnings.
2. Load the project onto the DE2-115 board without errors.
3. Observe whether the hex displays and the circuit behaves as expected.

### Part V

The part5 project will be tested by compilation and uploading to the DE2-115 board.

DE2-115

1. Open the project Part5 in Quartus and verify that compilation produces no errors or un-allowed warnings.
2. Load the project onto the DE2-115 board without errors.
3. Observe whether the hex displays and the circuit behaves as expected.

## Test Results

### Part I

ModelSim

1. The output of the simulation indicated that the CountNG module behaved as intended. Output was:

"Successfully checked 66000 iterations"

DE2-115

1. Compilation by Quartus was successful.
2. The RTL View produced the result shown in Figure 12. This matches the specification.
3. The project was uploaded to the DE2-115 board without errors.
4. The hex displays show the expected outputs and increment along with the clock input.
5. The enable and clear inputs behave as expected.

Thus all parts of the Test Procedure were executed and correct results were produced. Thus our design passes our test.

### Part II

ModelSim

1. The output of the simulation indicated that the count16b module behaved as intended. Output was:

"Successfully checked 66000 iterations"

DE2-115

1. Compilation by Quartus was successful.
2. The RTL View produced the result shown in Figure 13. This matches the specification.
3. The project was uploaded to the DE2-115 board without errors.
4. The hex displays show the expected outputs and increment along with the clock input.
5. The enable and clear inputs behave as expected.

Thus all parts of the Test Procedure were executed and correct results were produced. Thus our design passes our test.

### 

### Part III

DE2-115

1. Compilation by Quartus was successful.
2. The project was uploaded to the DE2-115 board without errors.
3. The enable and reset inputs at SW[1] and SW[0] behave as expected.
4. Asserting the KEY0 button switch resulted in the expected counting behavior output in the hex displays.
5. The RTL View produced the result shown in Figure 14.

Thus all parts of the Test Procedure were executed and correct results were produced. Our design passes our test.

### Part IV

DE2-115

1. Compilation by Quartus was successful.
2. The project was uploaded to the DE2-115 board without errors.
3. The hex displays and the circuit behave as expected.
4. The RTL View produced the result shown in Figure 15.

Thus all parts of the Test Procedure were executed and correct results were produced. Thus our design passes our test.

### Part V

DE2-115

1. Compilation by Quartus was successful.
2. The project was uploaded to the DE2-115 board without errors.
3. The hex displays and the circuit behave as expected.
4. The RTL View produced the result shown in Figure 16

Thus all parts of the Test Procedure were executed and correct results were produced. Thus our design passes our test.

## Observations

### Part I

The module CountNG and the project part1 passed all of the tests conducted in accordance with our Test Procedures in both ModelSim and the implementation in the DE2-115. The number of logic elements used were 50 and we found that the circuit could not operate beyond the frequency Fmax = 362.06MHz. The difference between the Quartus II synthesized circuit and the circuit in Figure 1 are:

### Part II

The module count16b and the project part2 passed all of the tests conducted in accordance with our Test Procedures in both ModelSim and the implementation in the DE2-115. The number of logic elements used were 45, less than for part1 and we found that the circuit could not operate beyond the frequency Fmax = 386.1MHz. This part differs from Part 1 in that the logic operations are defined in an always loop, rather than using individual flip flops to perform the counting.

### Part III

The project part3 was successfully compiled and uploaded to the DE2-115 without errors and passed all of the tests conducted in the Test Procedures section. This version used 46 logic elements which is just one more than used in part2 but four less than Part1.

### Part IV

The project part4 was successfully compiled and uploaded to the DE2-115 without errors and passed all of the tests conducted in the Test Procedures section.

### Part V

The project part5 was successfully compiled and uploaded to the DE2-115 without errors and passed all of the tests conducted in the Test Procedures section.

## Conclusion

We were able to successfully complete the objectives of the lab. We found that the most efficient implementation of the counter circuit from parts I-III was the circuit derived in part II.

## Appendix A

This Appendix contains listings of the files used in this laboratory exercise.

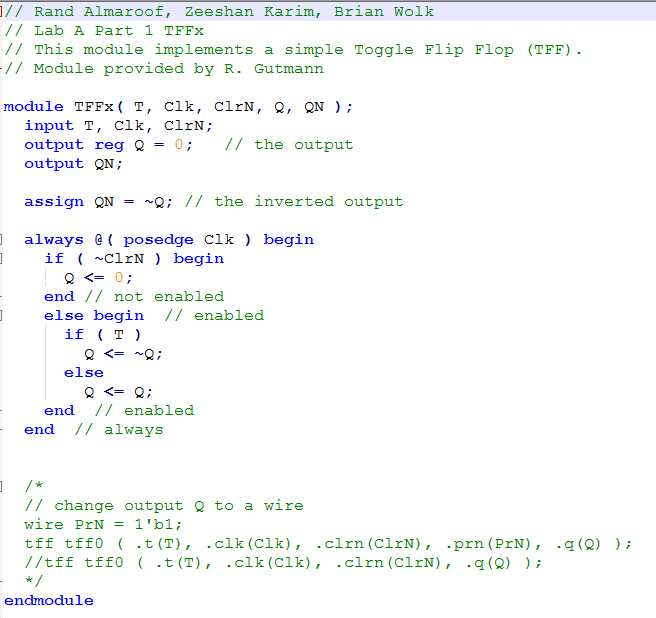


Figure : T flip flop module

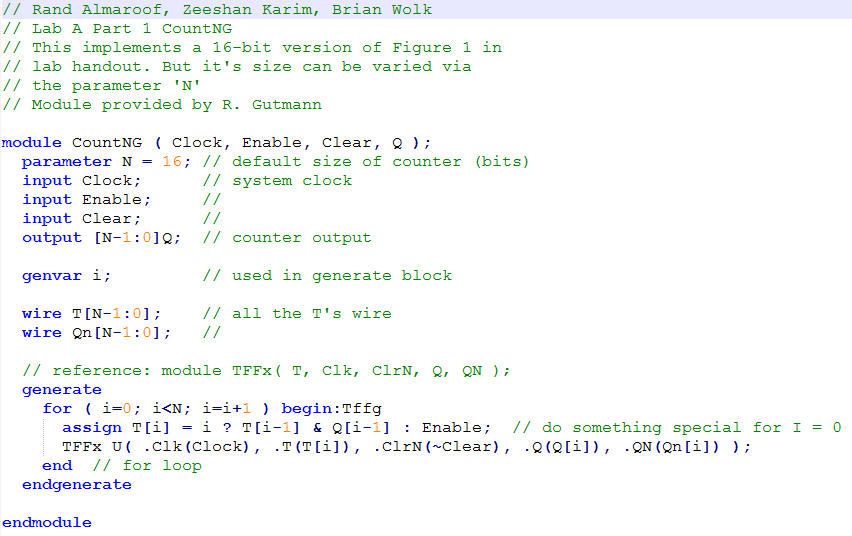


Figure : CountNG module

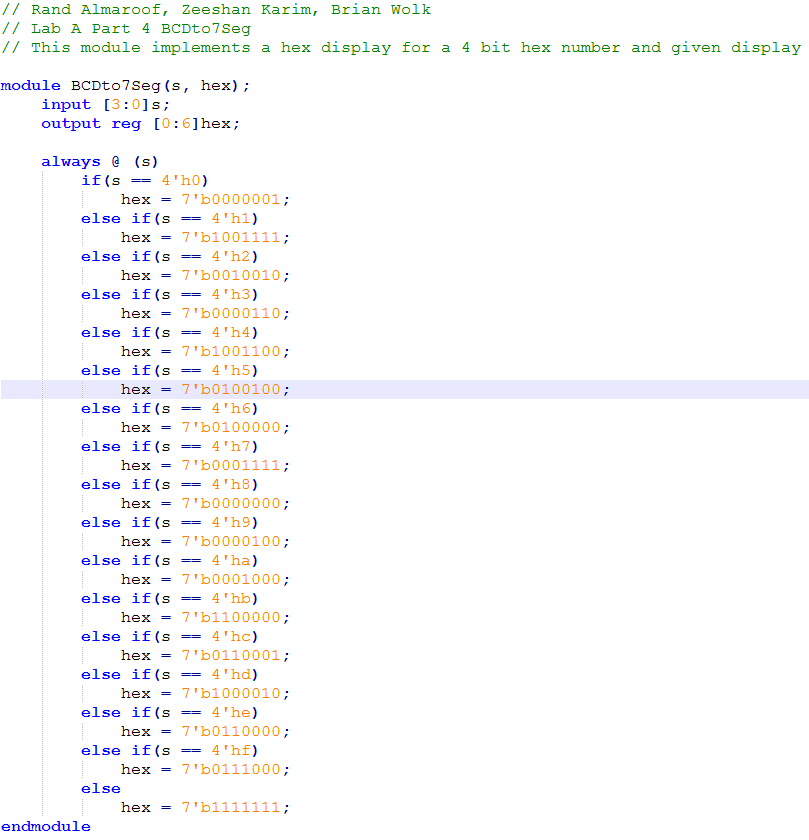


Figure : BCDto7Seg module for decoding binary to hex display

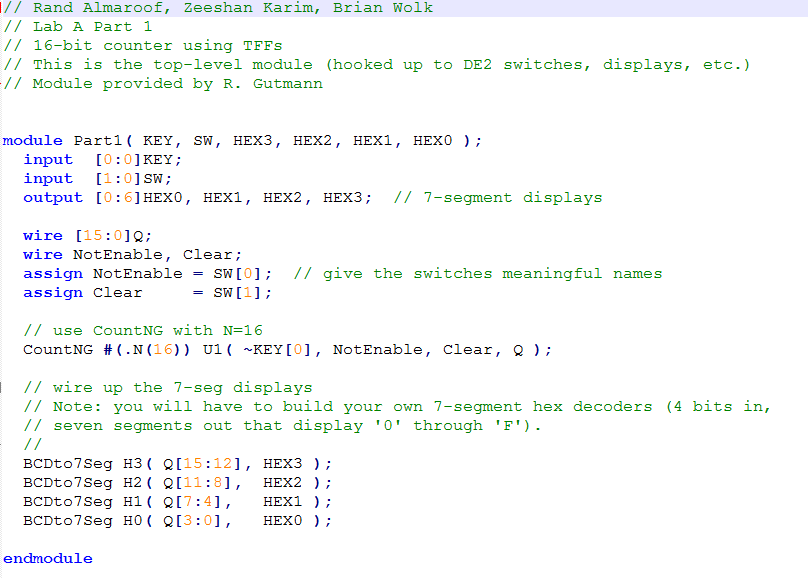


Figure : Top level module for part1

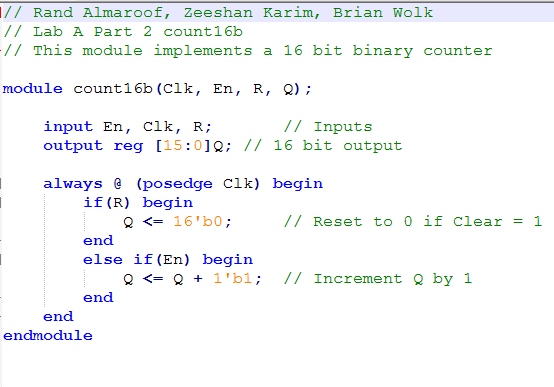


Figure : The 16 bit counter using the statement Q <= Q + 1’b1

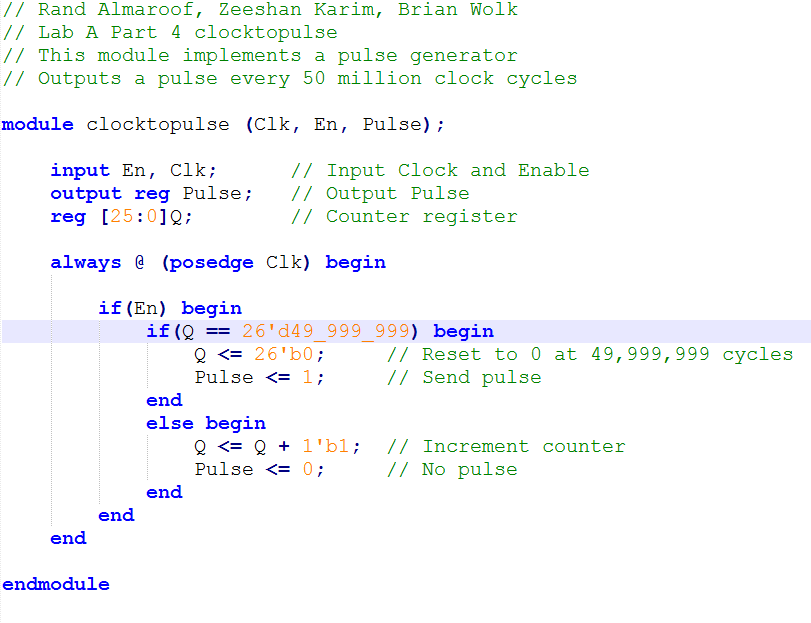


Figure : The clocktopulse module

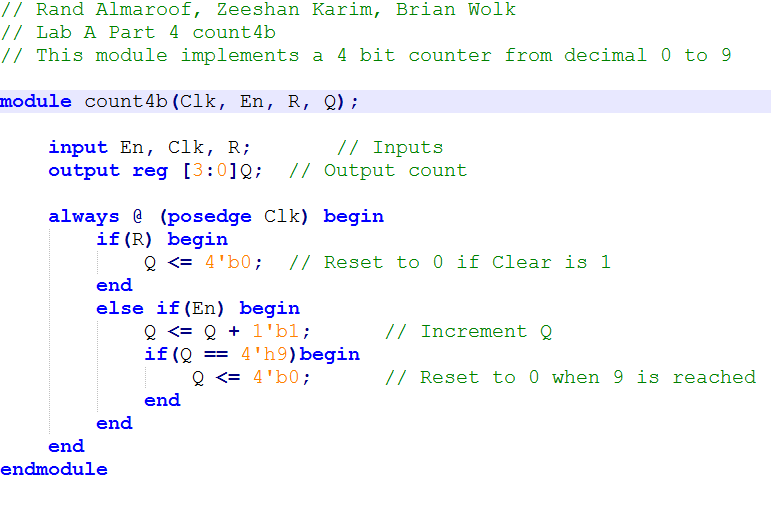


Figure : The count4b module

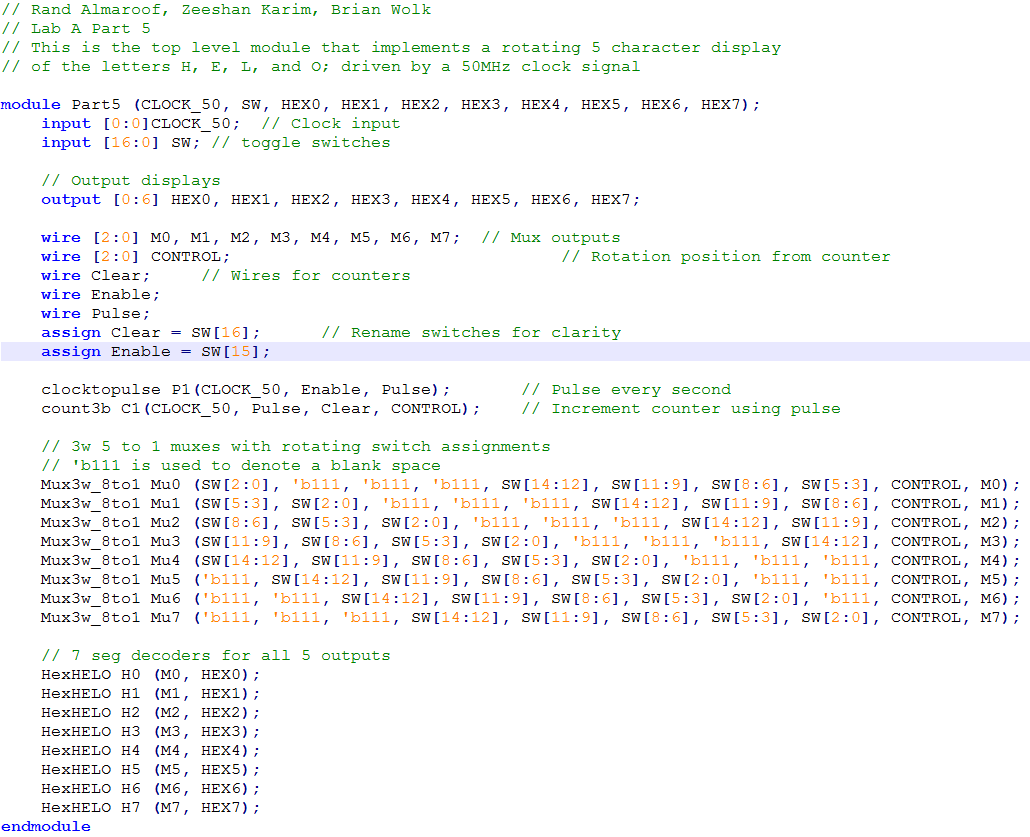


Figure : The top-level module for Part5

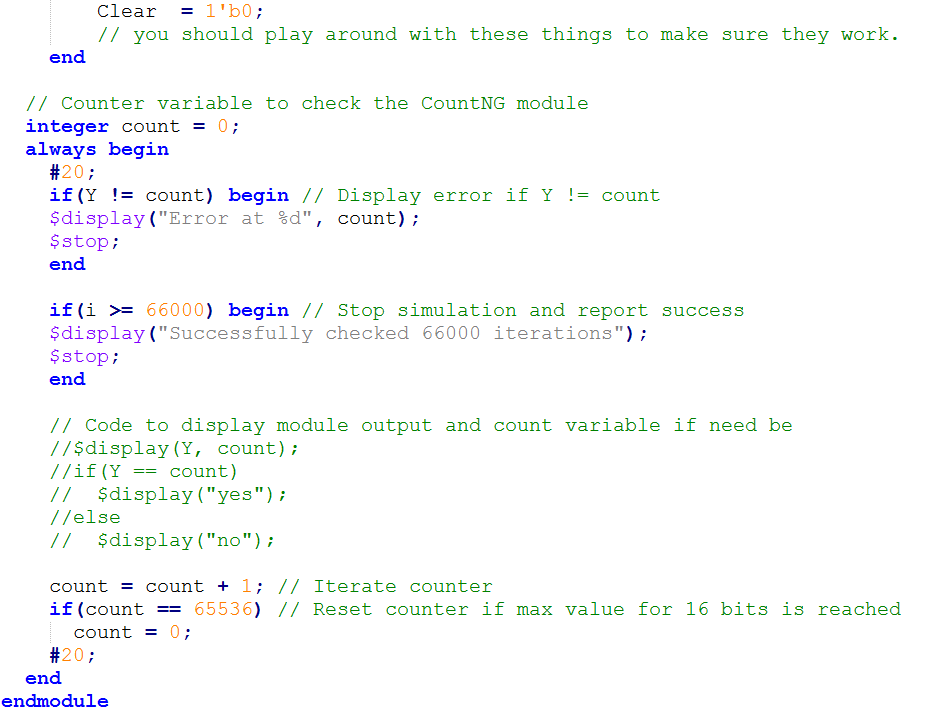
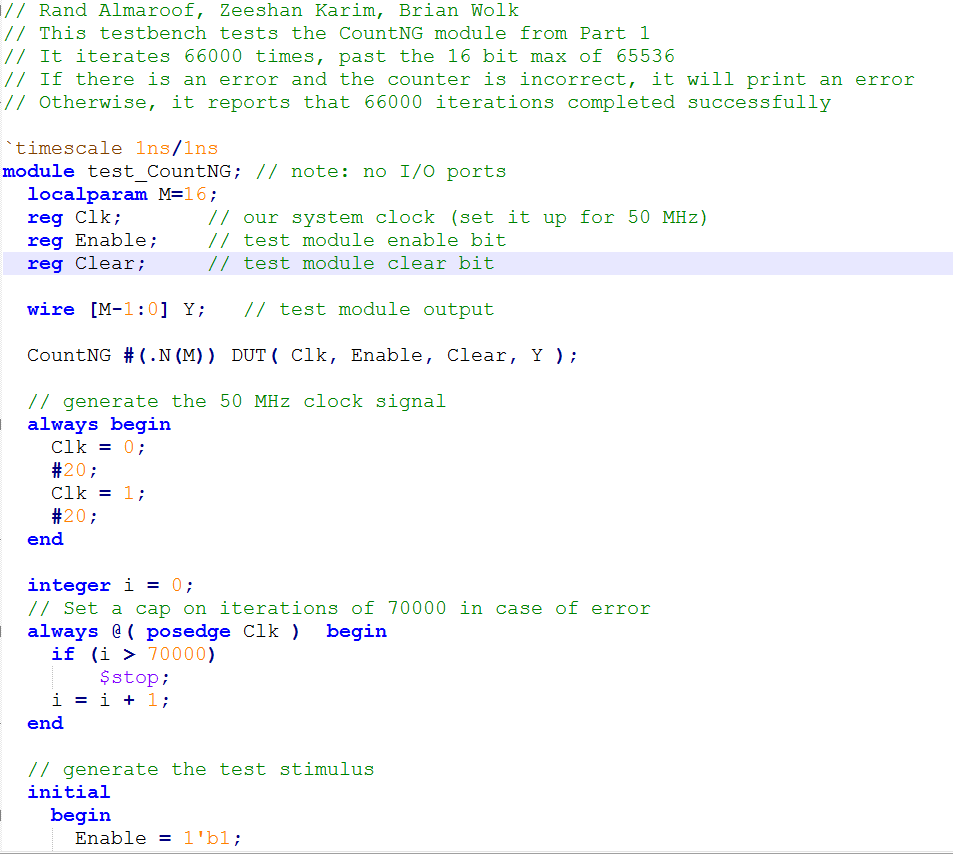


Figure : The test bench for part1

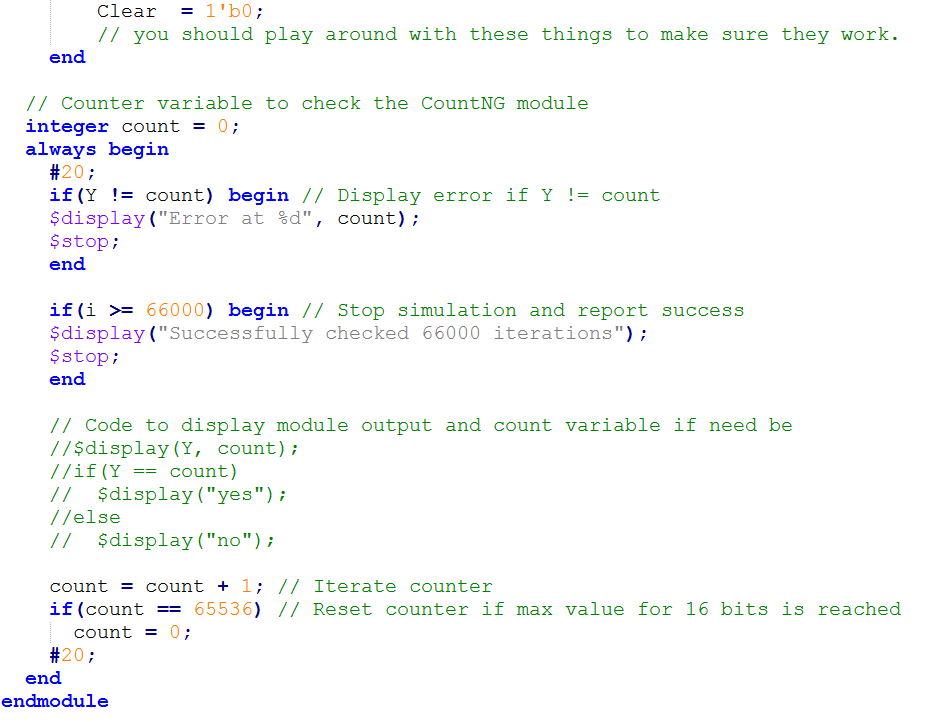
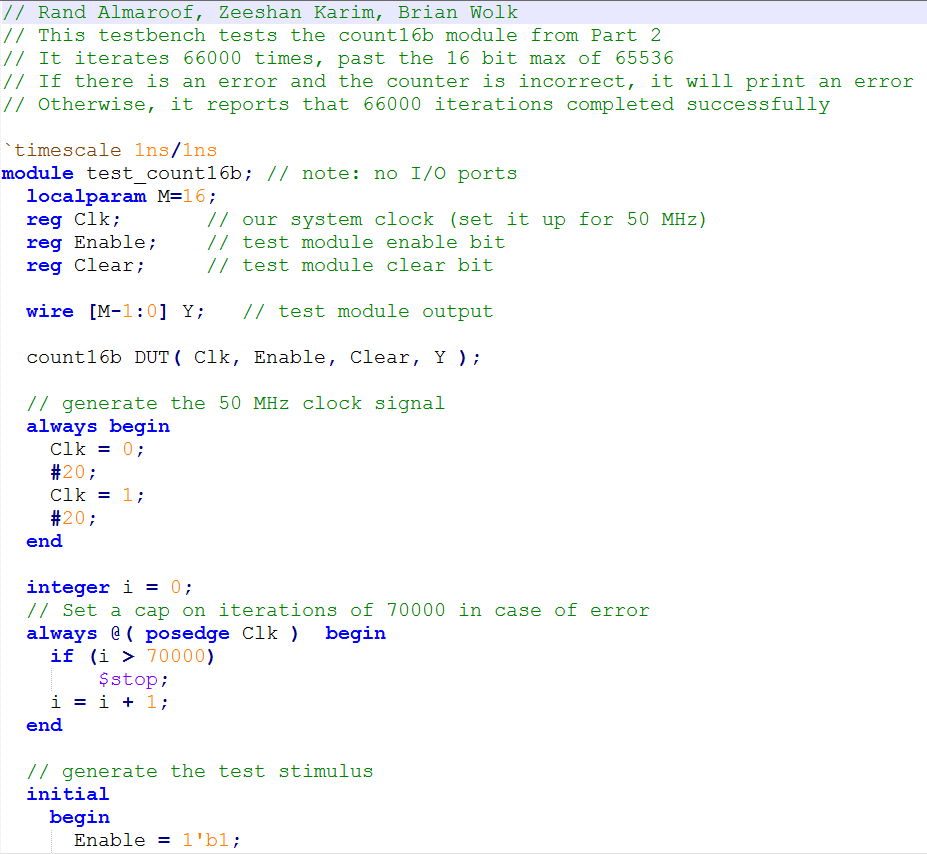


Figure : The testbench for part 2

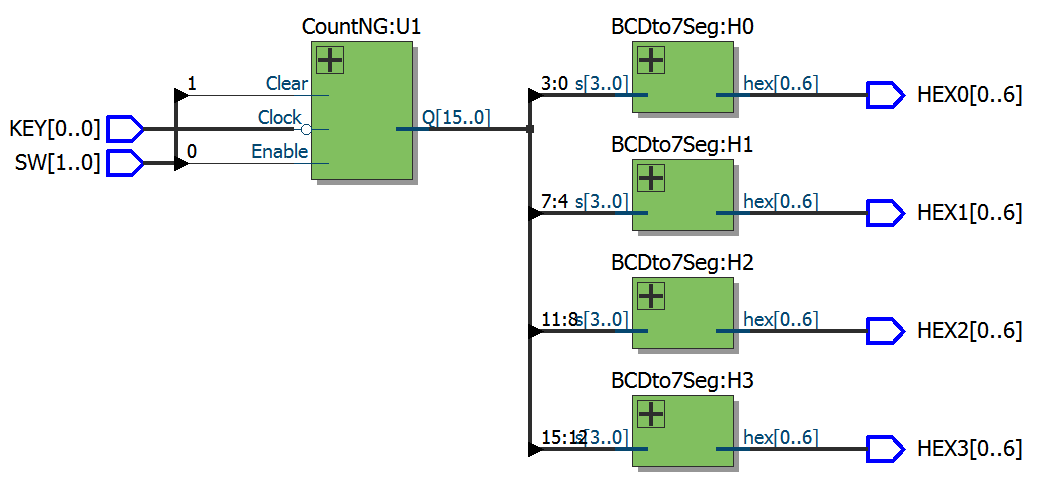


Figure 12: The RTL View for Part1

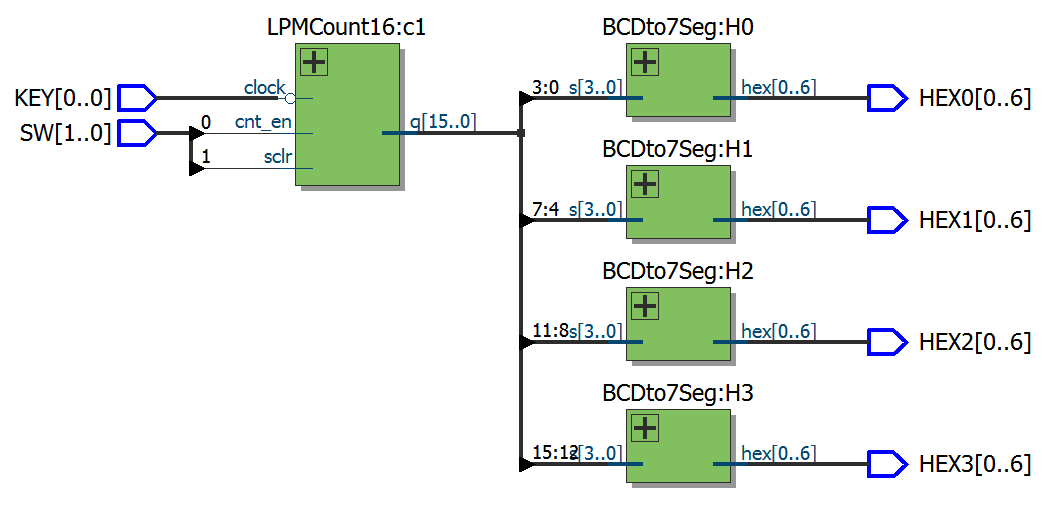


Figure 13: The RTL View for Part2

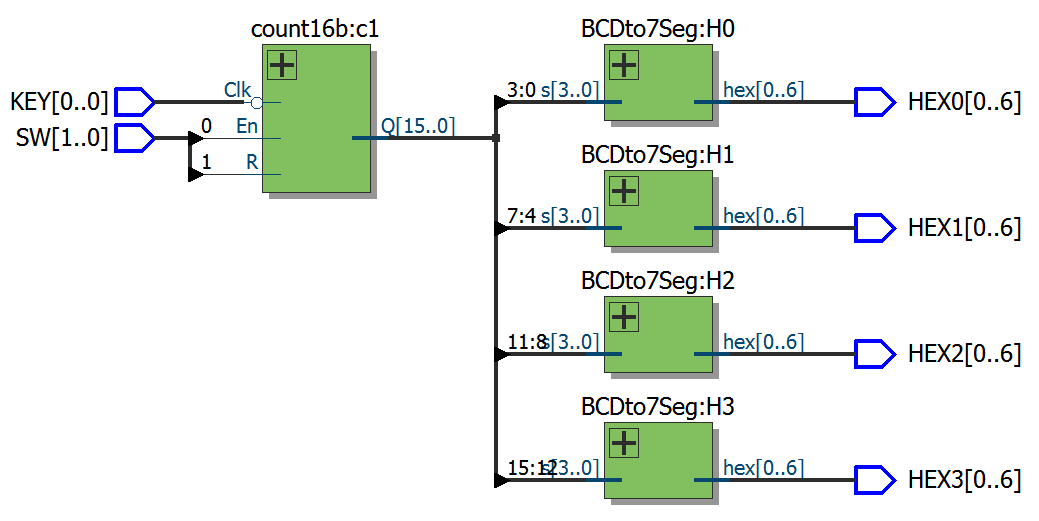


Figure 14: The RTL View for Part3

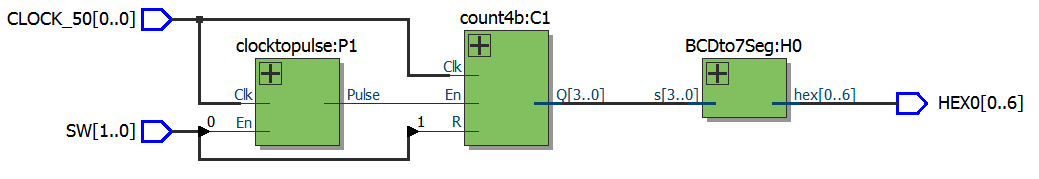


Figure 15: The RTL View for Part4

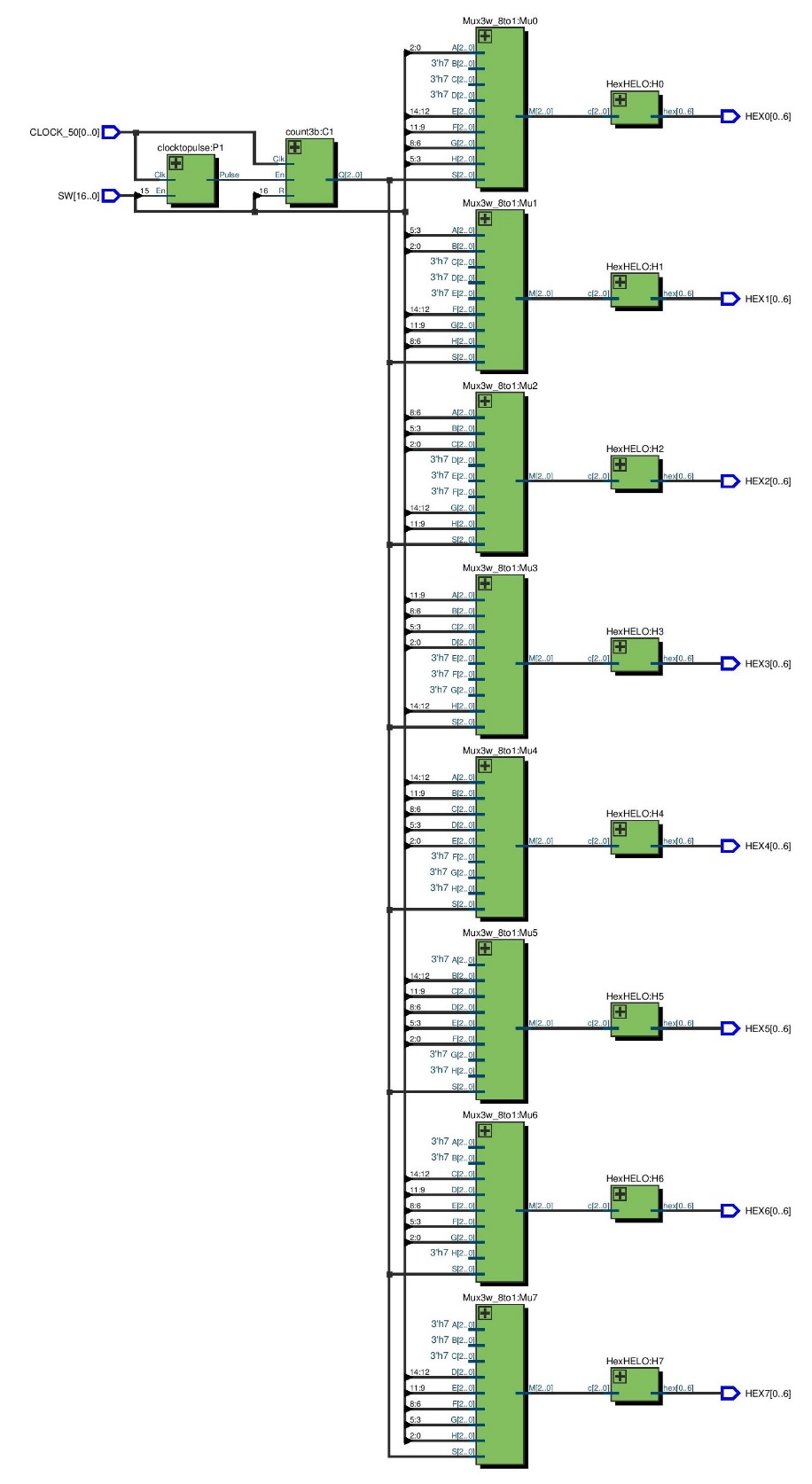


Figure 16: The RTL View for Part5