**Experiment Report of Digital Logic Circuit with Verilog**



Experiment Information:

|  |  |
| --- | --- |
| Index | B |
| Name | Comprehensive Design |

Student Information:

|  |  |
| --- | --- |
| Student No.： | 2018380039 |
| Student Name： | Dikshya Kafle |
| Grade No.: |  |
| Experiment Time： | 29th November 2019 Friday |
| Experiment Place： | Computer Science Building Room 118 |

**Northwestern Polytechnical University**

**School of Computer Science**

**Fall 2019**

*Attention: R****eplace name of this file with your name-student number-report-experiment Index, like XXX-XX-Report-A.docx***

# 1: Object and Requirement

*Tip: introduce the aim of the experiment task. It may cover the function requirement/adopted algorithm/ protocol analyzing/timing requirement/ area requirement*

1: learn how to design a state machine with Verilog.

2: design the reaction time detector in Chapter 5 of your text book

3: Implement the reaction time detector on FPGA board

**ModelSim:** A multi-language HDL simulation environment by Mentor Graphics for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartos Prime, Xilinx ISE or Xilinx Vivado.Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

**Altera Quartus II**:The **Altera Quartus II** design software provides a complete, multiplatform design environment that easily adapts to your specific design needs. It is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The **Quartus II** software includes solutions for all phases of FPGA and CPLD design. It is the programmable logic device design software produced by Intel. It includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation. It enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

# 2: Environment and Tools

*Tip: simply introduce the EDA tools you need to complete the experiment.*

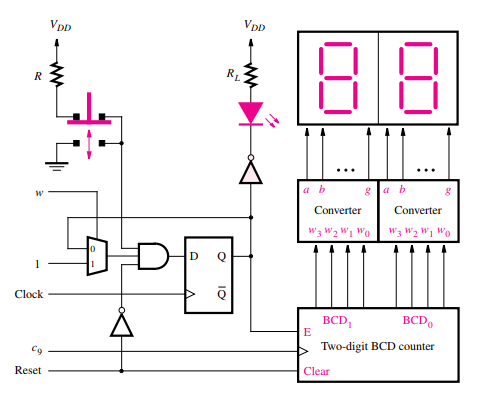
**ModelSim:** A multi-language HDL simulation environment by Mentor Graphics for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartos Prime, Xilinx ISE or Xilinx Vivado.Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

**Altera Quartus II**:The **Altera Quartus II** design software provides a complete, multiplatform design environment that easily adapts to your specific design needs. It is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The **Quartus II** software includes solutions for all phases of FPGA and CPLD design. It is the programmable logic device design software produced by Intel. It includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation. It enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer

# 3: Procedures

*Tip: show the detailed circuit you designed. It may cover:*

*1: the TOP schematic and explanation*



*Explanation:*

The figure shown above is a circuit used to count the reaction time for a user. Initially, an *light-emitting diode* (LED) is required to be off and as the counting begins in the circuit the LED will turn on and will not stop until the button is pushed by a user. Therefore, a reset port is required to work on the counter and the D flip-flop. Furthermore, when depressed, the push button induces the D flip-flop to be synchronously reset.

Whether the LED is on or off is determined by the output of this flip-flop; it also provides the count enable input to a two-digit BCD counter. From the explanation of the book Section 5.11, each digit in a BCD counter has four bits that take the values 0000 to 1001. Thence the counting sequence can be viewed as decimal numbers from 00 to 99.

*2:the interface between sub-modules*

The following models have been used for our reaction timer circuit:

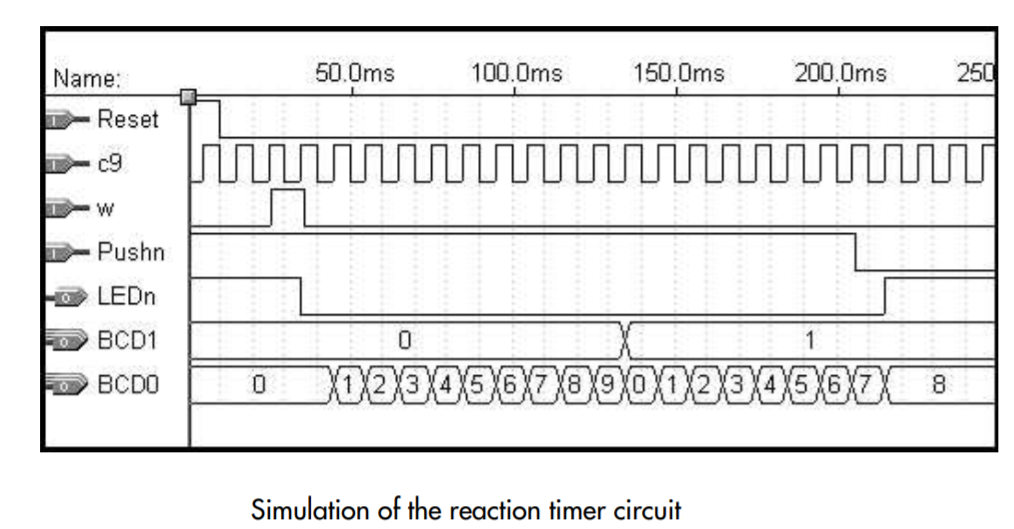
-counter: It is the clock divider;

-BCDcount: The BCD counter;

-seg7: The 7 segments converter;

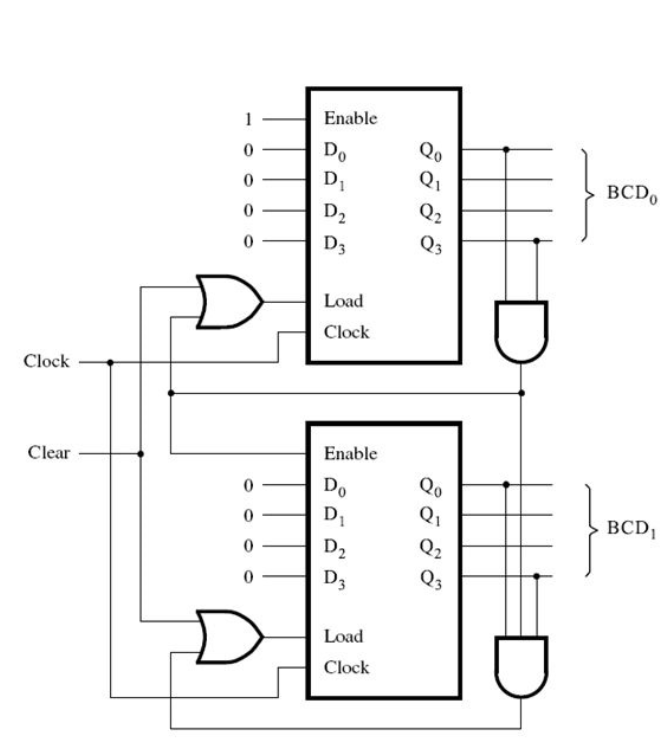
-reaction timer: The main module that calls the past modules to get the final circuit.

*3: the critical timing diagram expected*

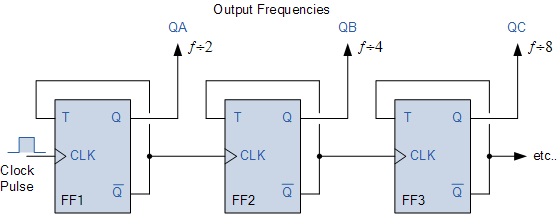


*4: the detailed circuit diagram of each submodule*

The BCD counter +the 7 segment converter:



The Clock Divider:



*Fig: Frequency Division using Divide-by-2 Toggle Flip-flops*

*5: the verification strategy and the testbench design*

The verification of the circuit implementation is done by simulating the implemented circuit. At first: Reset is at 1, Clock is at 0, rst is at 0, w is at 0 and Pushn is at 1. Afterwards, rst is kept back at 1 and Reset at 0; then w is set to 1 and after waiting it is set to 0 again. In this way the circuit starts counting and the user waits for the button to be pushed. Furthermore, the user will set a time for Pushn to be 0 which would be the reaction time resulted from the time of one cycle times the number that user got from the counter.

*The original clock toggles every 10 unit times.*

# 4: Result and Discussion

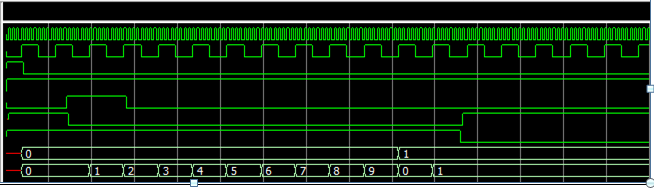
*Tip: what about the result, it may cover:*

*1: whether you have simulated all possible cases*

Considering the circuit there are not much cases however what is being prepared is to be considered before further steps i.e. being able to counting and wait for the button to be pushed which was done by reset port and w port. In addition it is noted that Pushn is 0 after a specific time which was also simulated.

*2: whether the wave generated by the Modelsim is exactly the same as you expected.*

*You could paste some wave pictures to prove your work.*



*3: whether you have detected some bugs in your code and how to fix them.*

During the coding process, some bugs were surely encountered:

-**Syntax Error**: While running the code I faced problems with a lot of syntax in the beginning. I had forgotten some brackets and simple commas and sometime I messed up with letters as well. Eg: initially defined Pushn was written as pushn and after realizing the error I corrected to Pushn and the error was eliminated after that. Similarly, this kind of errors happened in other cases as well and after finding out the error it was changed to the correct syntax and the code was corrected. While calling the module, a typing error was made when calling a module, so simulation could not be designed and the EDA launched an “*Error design”.* Afterwards the code was debugged with the correct module name and no error was found later. The code was successfully compiled with no errors or warnings.

*-****Clock Divider****:* During the implementation of the clock divider, the resulting signal showed up very late because of very low frequency. Therefore, the output was changed to the earlier T flip-flop considering the frequency would be higher. As expected it can now be seen in the simulation.

*4: what about the critical path and the area cost.*

The critical path is from the clock divider through the BCD counter ang going through 7 segment convert.

*5: how to optimize your design*

In order to optimize the circuit, transmission gate can be implemented. Transmission gate is simple and it also can implement certain structures with less transistors. This saves the area of the circuit and may offset speed penalties in some cases.

*6: whether you have follow some basic coding styles.*

*-****Naming Varaible:*** Assigning meaningful names to variables.

***-Commenting****:* For the further explanation of some codes comments were used which was denoted by //.

**-*Separate variables declaration:*** Declaring each variable alone.

-**Creating Project:** After the completion of all the codes a project named “Task 4” was created.

**-Compiling**: All the codes were compiled and the errors were corrected.

-**Simulation:** At the final stage simulation was carried which allowed us to view the waves of all the codes of the project.

# 5: Conclusion

*Tip:Give a brief conclusion of your work and evaluate your work by yourself.*

The simulation of reaction timer is done in the given experiment. After the completion of the code writing the simulation of functionality of reaction timer was done. It can also be seen from the simulation process in EDS that the signals performed exactly as it was expected theoretically. Therefore, this shows that the codes were correct which lead us to reach the goal of succeeding experiment.

# Performance Evaluation

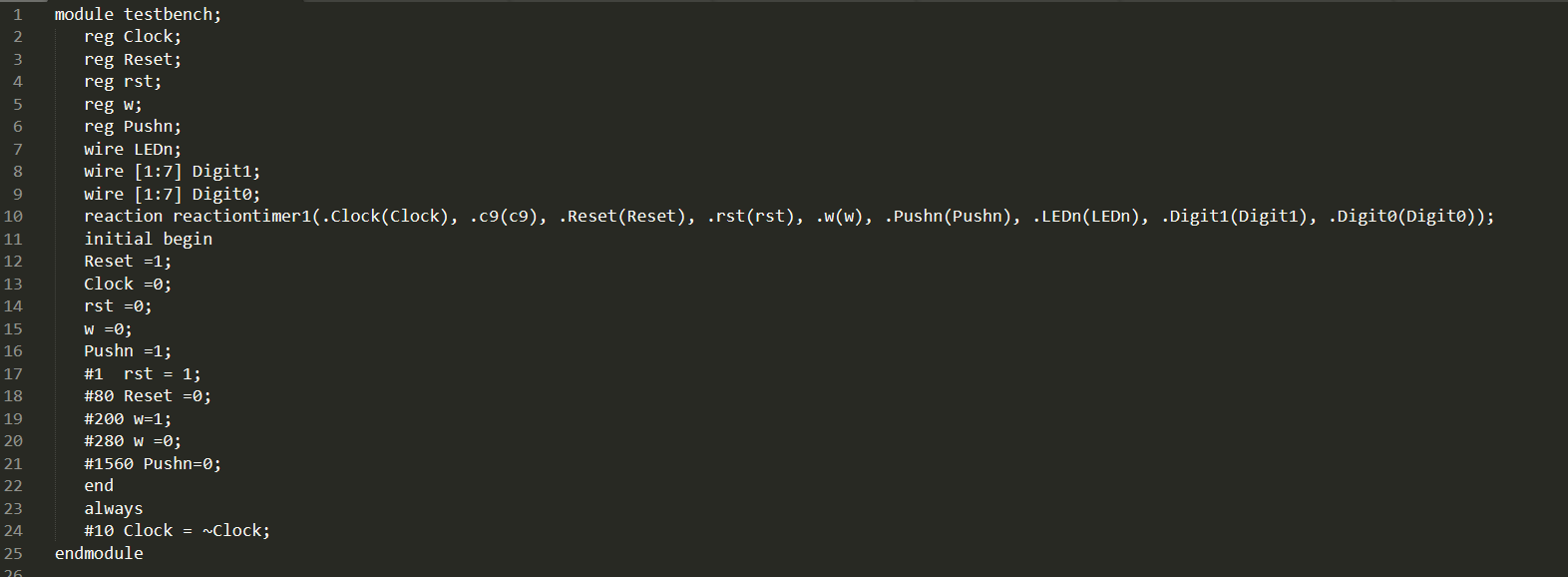
(This part is left only for the instructor)

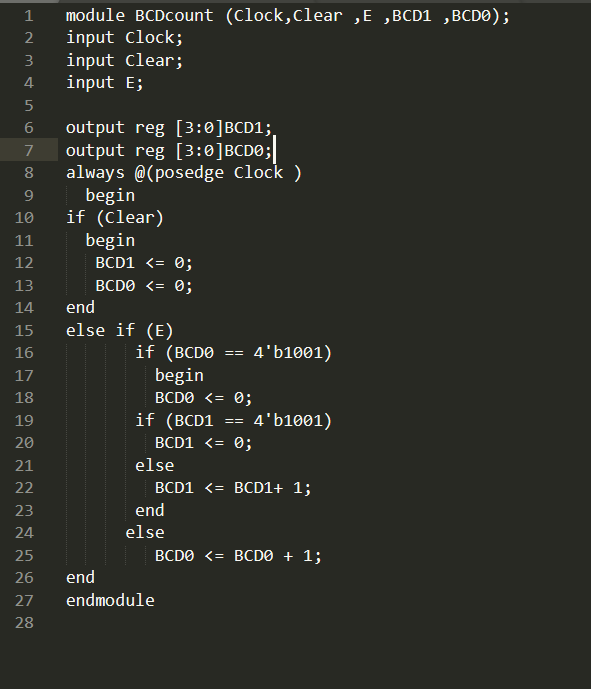
|  |  |
| --- | --- |
| Comment: | Score:  Signature:  Time: |

# Appendix:

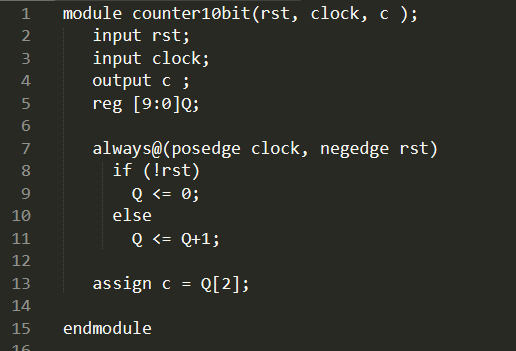
(Here, please attach key pieces of code of your design and testbench)

**Testbench:**

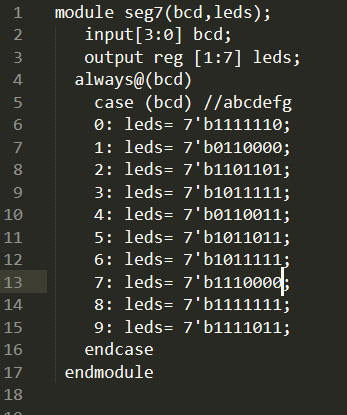
**BCDcount:**



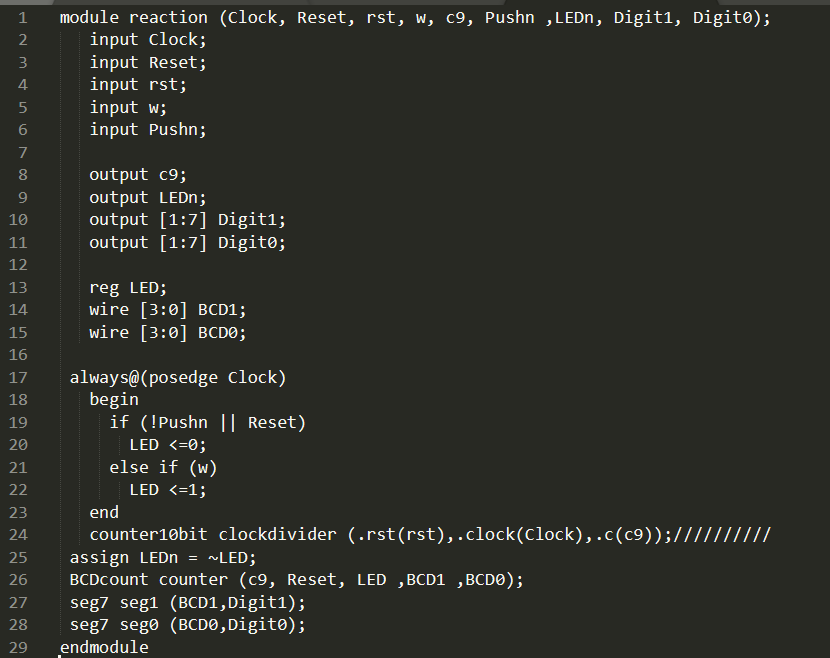
**Counterclock:**



**Seg7:**

****

**Reactiontimer:**

****