ECE 3544: Digital Design I
Project 2: Modeling the Timing of a Device

Student Name: Sefunmi Ashiru

Honor Code Pledge: I have neither given nor received unauthorized assistance on this assignment.

Sefunmi Ashiru

Grading: The design project will be graded on a 100 point basis, as shown below:

Manner of Presentation (30 points)

_____ Completed cover sheet included with report (5 points)

_____ Organization: Clear, concise presentation of content; Use of appropriate, well-organized sections (15 points)

_____ Mechanics: Spelling and grammar (10 points)

Technical Merit (70 points)

_____ General discussion: *Did you describe the objectives in your own words? Did you discuss your conclusions and the lessons you learned from the assignment?* (5 points)

_____ Design discussion: *Did you discuss your design approach, and the design decisions that you made as a part of implementing your modules?* (10 points)

_____ Timing analysis discussion: *Did you determine the minimum clock period that allows correct operation of the system?* (5 points)

_____ Testing discussion: *What was your approach to formulating your test benches? How did you verify the correctness of the modules you designed?* (10 points)

_____ Supporting figures: *Waveforms showing the correct operation of the various modules, Waveforms demonstrating valid and invalid behavior of the system.* (20 points)

_____ Supporting files: *Do the modules pass any tests applied by the grading staff? Modules whose declarations do not conform to the requirements of the project specification cannot be tested, and will receive no credit.* (20 points)

_____ **Project Grade**

Purpose

In this project, I will use Verilog as a means for modelling the timing of a module. I will be simulating two pre-designed circuits. I will be modelling the 74HC/HCT280 9-bit odd/even parity generator/checker and using a delay model to analyse the timing of a system that utilizes the component. I must also design simple digital logic circuits for a 9-bit counter and a 10-bit register. Using these components, I will create a transmitter and receiver module.

Technical Approach

For my design process I utilized the continuous assignment prebuilt functions and procedural assignments for the synchronous modules. For the 9-bit counter, 10-bit register, and receiver modules I used procedural assignment. For my hc280 module I followed continuous assignment. For my transmitter module I only used structural assignment to call other modules. Using these methods, I aimed to replicate the structural designs provided by the data sheet for the parity checker/generator and the 10-bit register.

Results

Testbench

The waveforms from testbench can be found in the Appendix 1-4

For my 9-bit counter test (Appendix 1) I followed the same testbench procedure as the 4-bit counter provided to us. This approach worked by turning on the clock cycle and at future steps activating and deactivating all necessary parameters to insure it's working. The clock cycle then continues until manually stopped. This allows me to check all possible outputs of the counter without figuring out the delay required before stopping for arbitrary large counters like the 9-bit counter. For my hc280 test (Appendix 2) I modelled my test bench like that of the 9-bit counter as I aimed use the 9-bit counter to provide all possible inputs for the hc280 module, however I did edit the procedural timing so that it only turns on the counter as I know the other functionality of the counter are working from my previous test. IO used the same testbench to test for without (Appendix 2.1) and with delay (Appendix 2.2). For my 10 bits register test (Appendix 3), I implemented the same test module design as my hc280 and included the hc280as my 10th bit for the register. This was not necessary for testing the register, but I found benefits as by doing so I also got to practice designing the transmitter. For my transmitter and receiver test (Appendix 4) AKA tb_channel1 and tb_channel2, I set the modules up using structural design and the same procedural activation of the clock and enable functions. So far, I have been unable to get the simulation to run due to an error in my code/logic somewhere.

I have however considered when the PERIOD value for the clock cycles may affect the parity checking during transmission and receival. I believe this would be valued if the cock cycle is shorter than the propagation delay of the components. For example, my 9-bit counter has a total propagation delay of about 50 for each call to the module. If the clock cycle can go from high to low and back high again. The count would not be able to register this as it has not finished executing the first instruction set activated by the clock's first high value. This would cause the counter to be working on the first instruction while other components with short enough delays will be working on the 2nd instruction, which leads to miss match in the data and those errors in the code due to it not being synchronous.

Conclusions

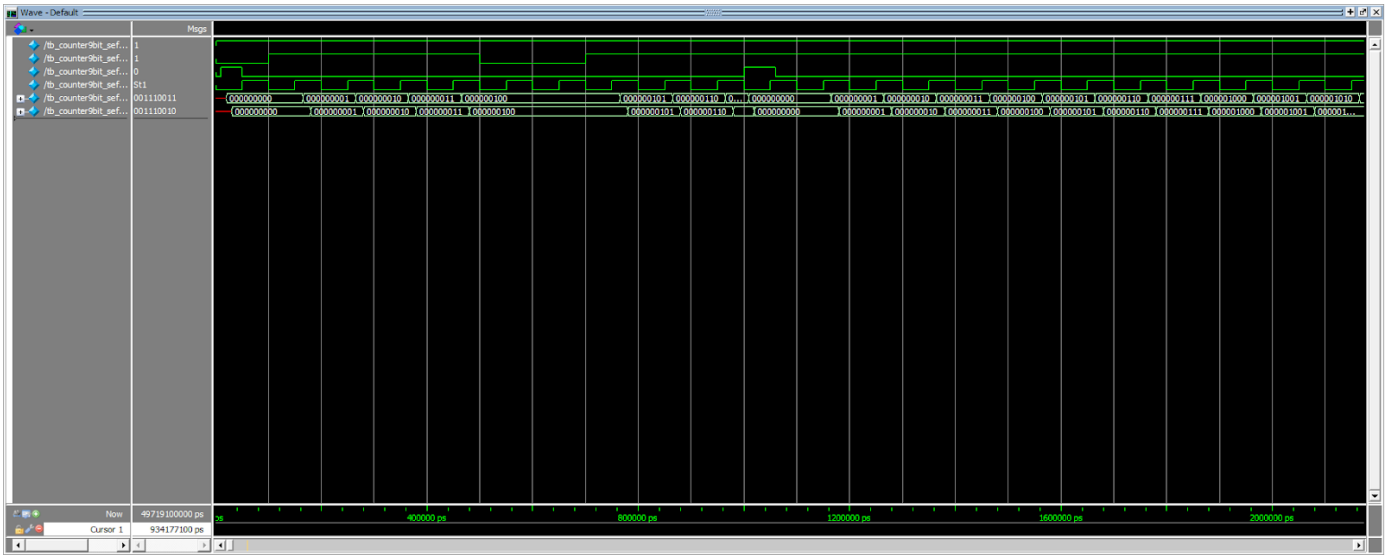
In Conclusion I feel that I have learned a lot about Verilog structural, procedural & continuous design. I have also gained a lot of understanding for synchronous behaving and spotting when errors could occur due to

delay values. If I were to proceed again, I would attend office hours earlier to solve issues I am unable to do on my own.

Appendices

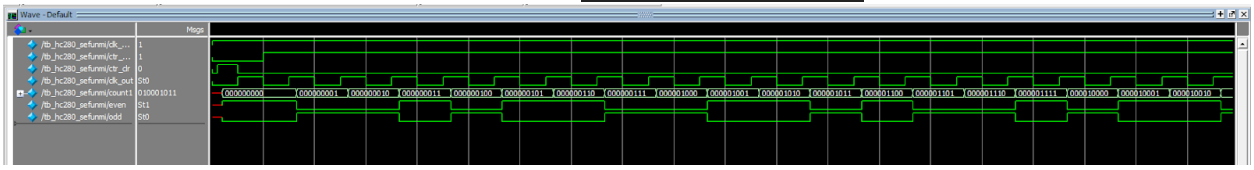
Appendix 1 - 9-bit counter

9-bit counter test waveform



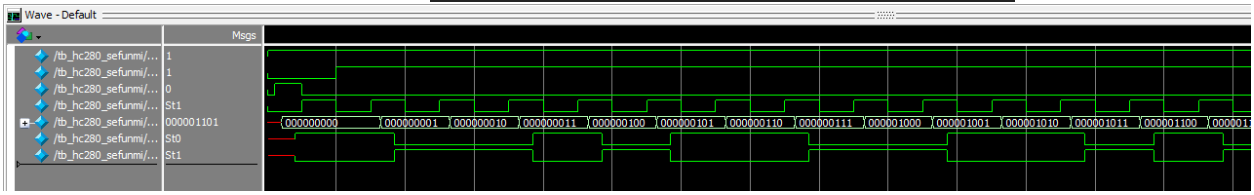
Appendix 2.1 - hc280 parity checker/generator

hc280 test waveform



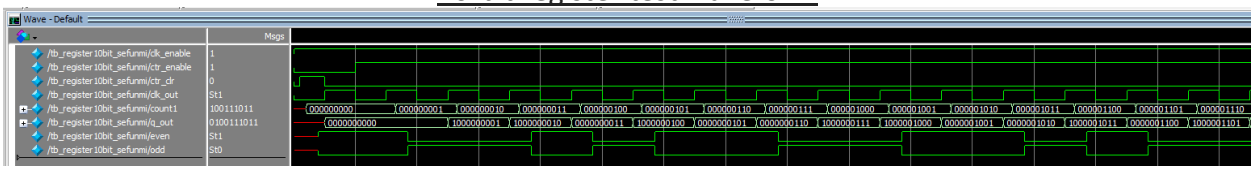
Appendix 2.2 - hc280 parity checker/generator with propagation delay

hc280 with propagation delay test waveform



Appendix 3 - 10-bit register

10-bit register test waveform



Appendix 4.1 - Channel 1 transmitter to receiver

Appendix 4.2 - Channel 2 transmitter to receiver