**ECE 3544: Digital Design I**

**Project 2: Modeling the Timing of a Device**

Student Name: Alex Orlov

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

**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**

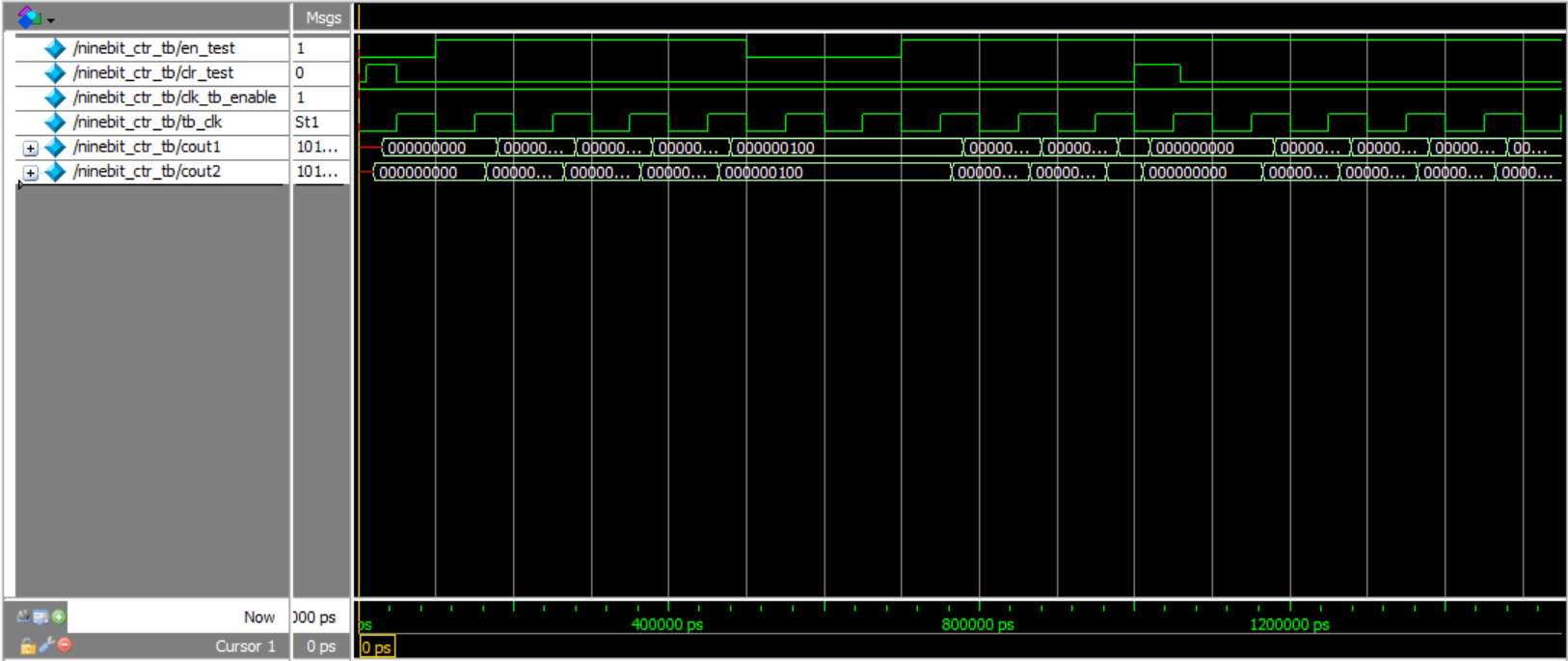
**Purpose**

The purpose of this project was to create a parity checker, and implement it in a system. This involved

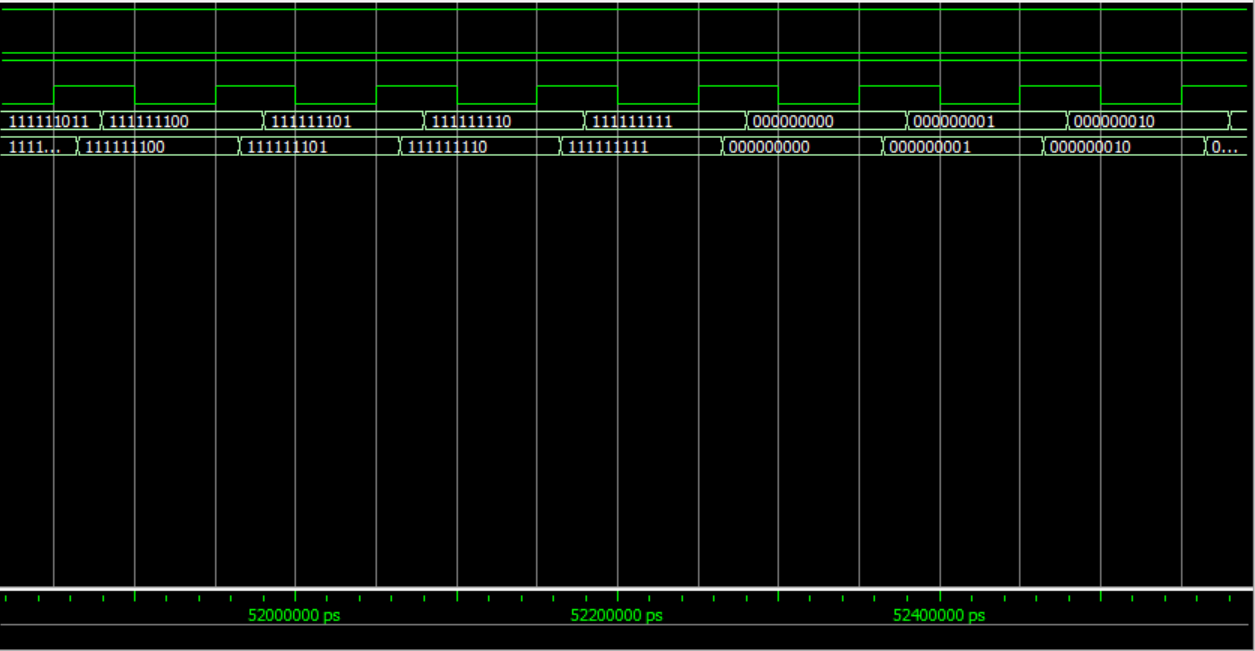
writing several modules and consolidating them into a single system. As I designed the system, I tested incrementally by writing a test bench for each module.

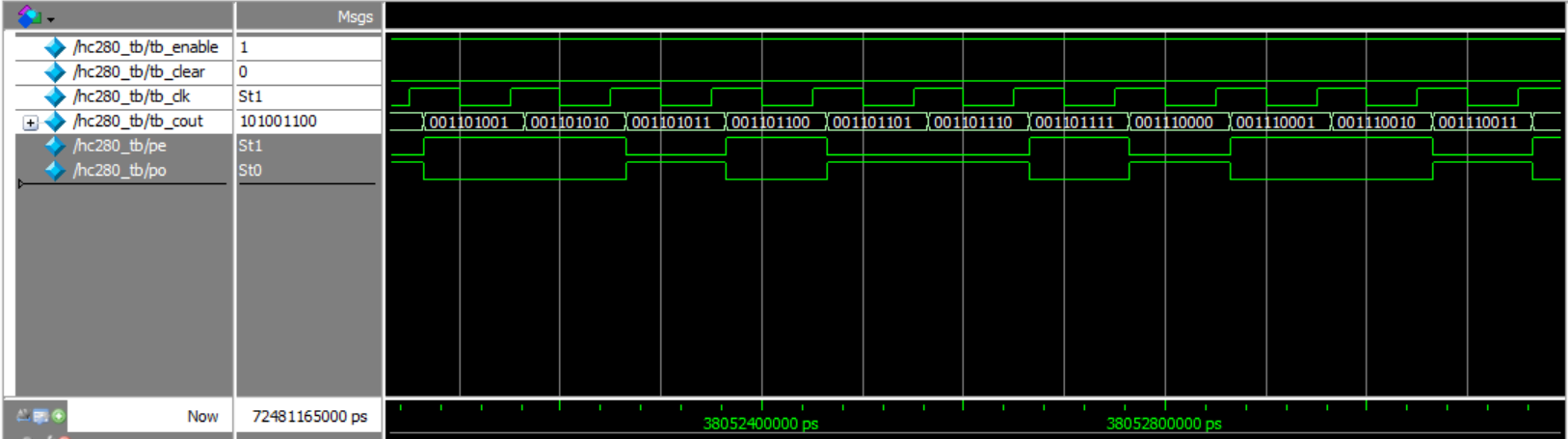
**Design and Testing**

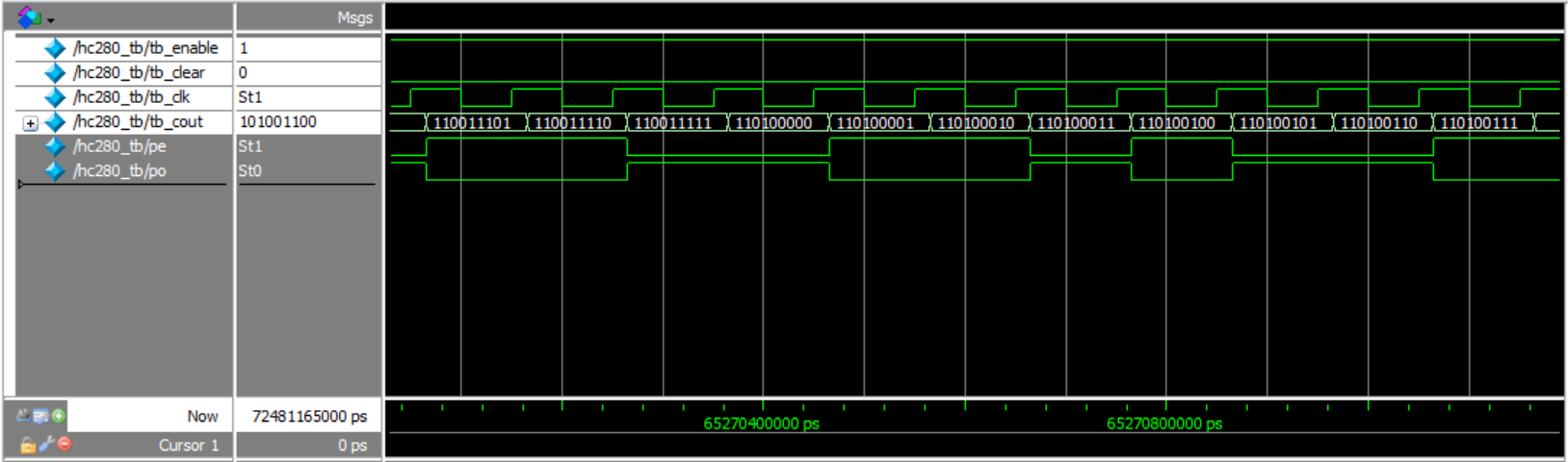
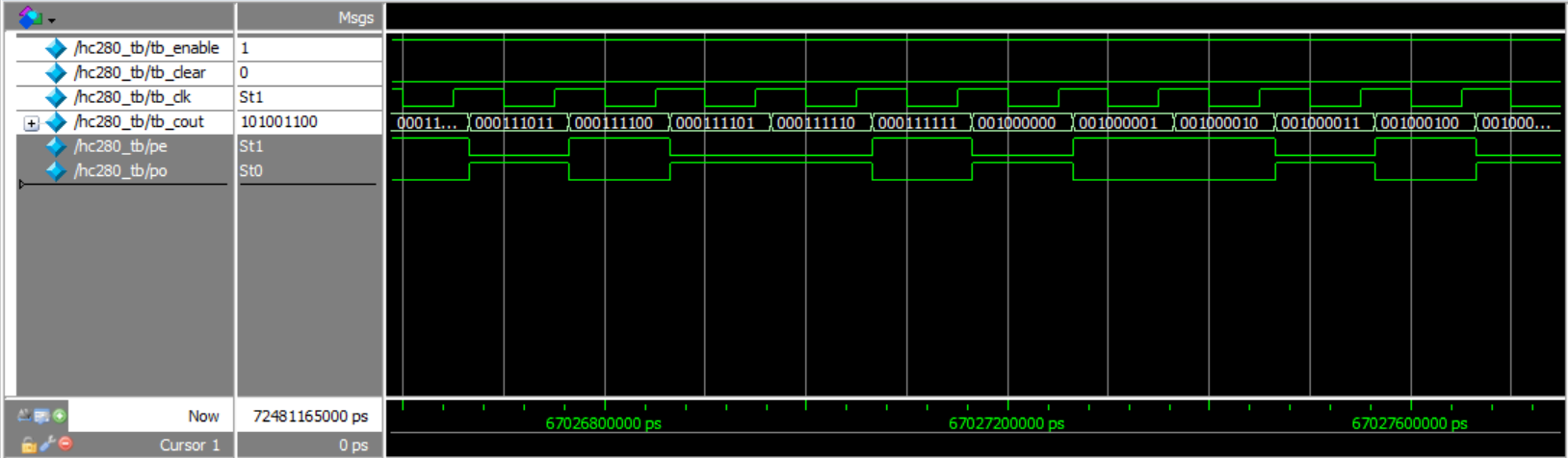
I began by modifying the provided 4 bit counter to create a 9 bit counter. I then verified my 9 bit counter

works.

In the range provided in the waveform, one can see the counter has the proper response to enable, clear, and clock:

The counter also demonstrated the proper transition from max to 0.

I then designed the hc280 parity checker, with no delays. My basic logic was to add the 9 bits of input using continuous assignment. Then, output parity based on the LSB of the sum. I created a test becnch and verified the parity checker works:

I made sure to check the waveform in multiple locations to build certainty of working model:

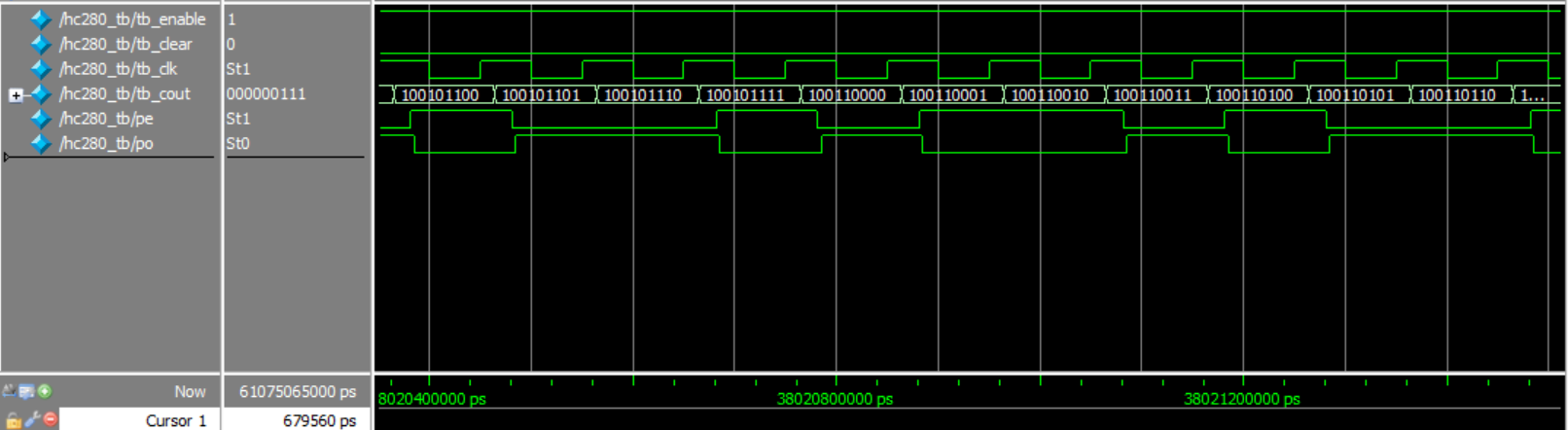
After designing the non-delay model of the hc280, I moved onto parity checker. Using the datasheet, I found propagation delays for the hc280.

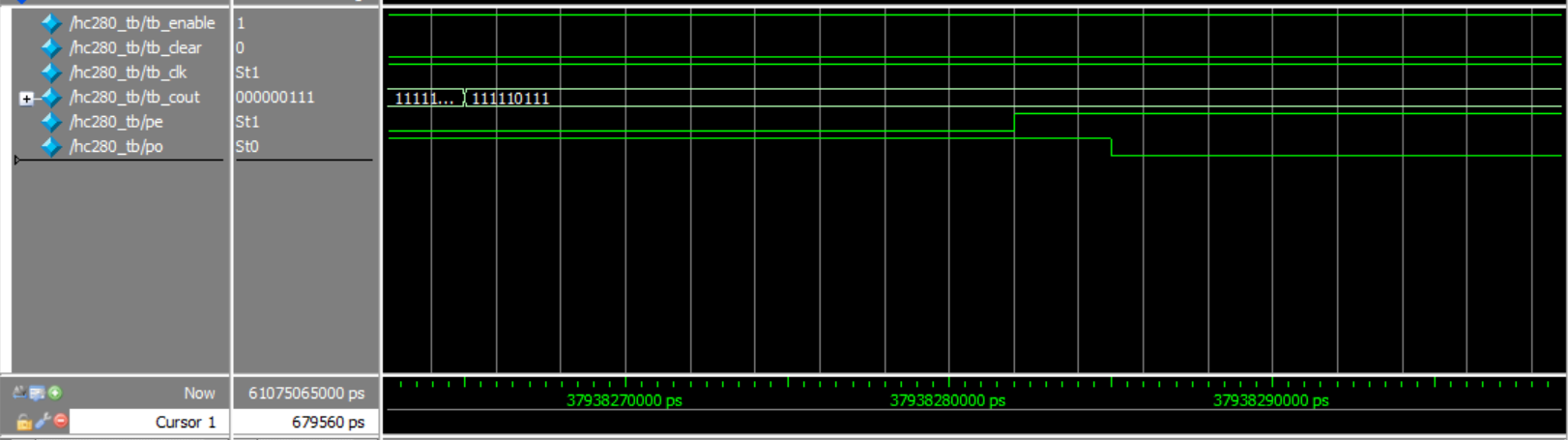
For the hc280, the typical propagation delays at V­cc = 5V and 25­° C are as follows:

PE propagation delay: 17 ns

PO propagation delay: 20 ns

Using specify blocks, I added these delays.

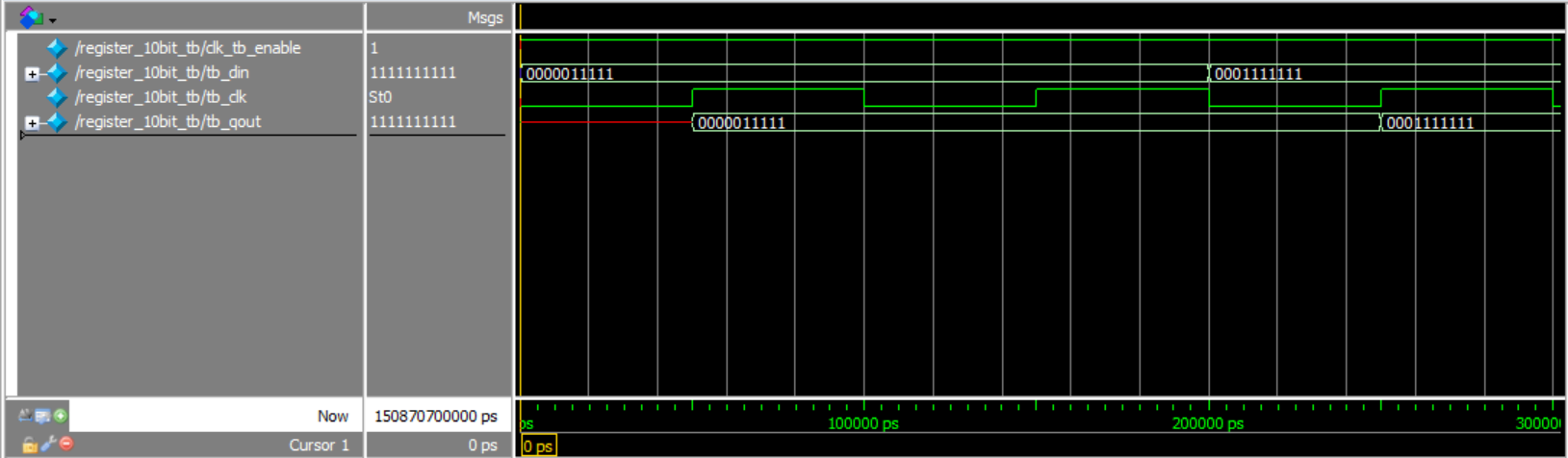
We can see from the waveform that correct operation is maintained.

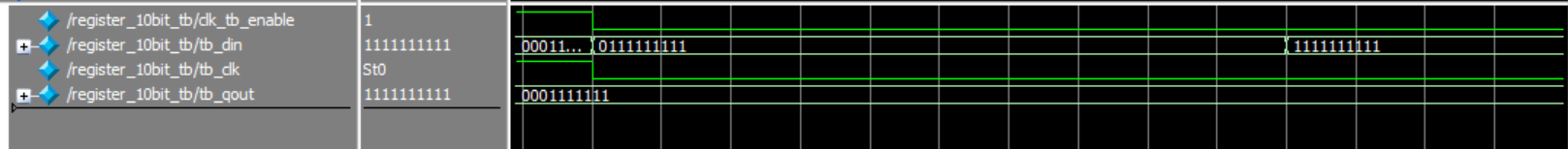
If we zoom in further, we can verify that the propagation delays are correct:

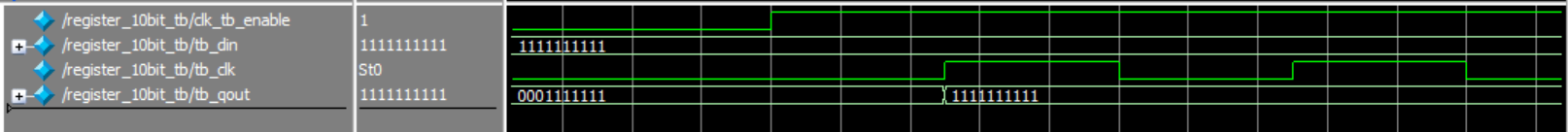
From the zoomed in image, we can make the observation that every 5 major vertical lines is equal to 10000 ps = 10 ns, thus the space between each line is 2 ns. There are 8.5 spaces from the counter change to the change in pe, which is equivalent to a 17 ns delay. Likewise, there are 10 spaces to the change in po, thus a 20 ns delay.

From here, I moved onto part 3.

I designed a 10 bit register which loads values in parallel on every rising edge. I then made a test bench for the register.

The simulation demonstrated that the register only updates on rising clock edges:

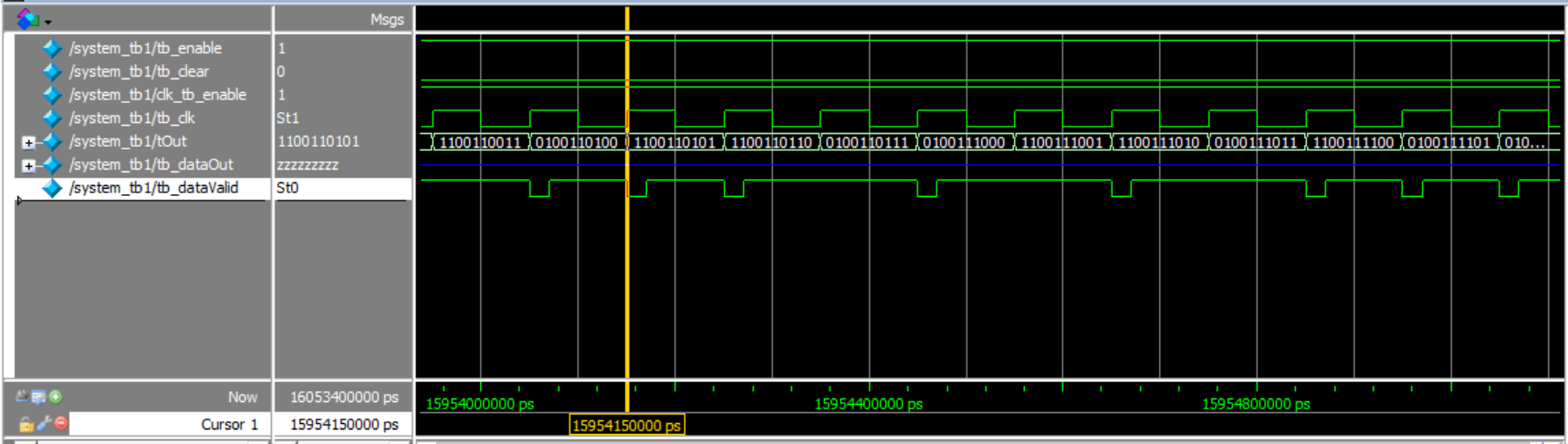
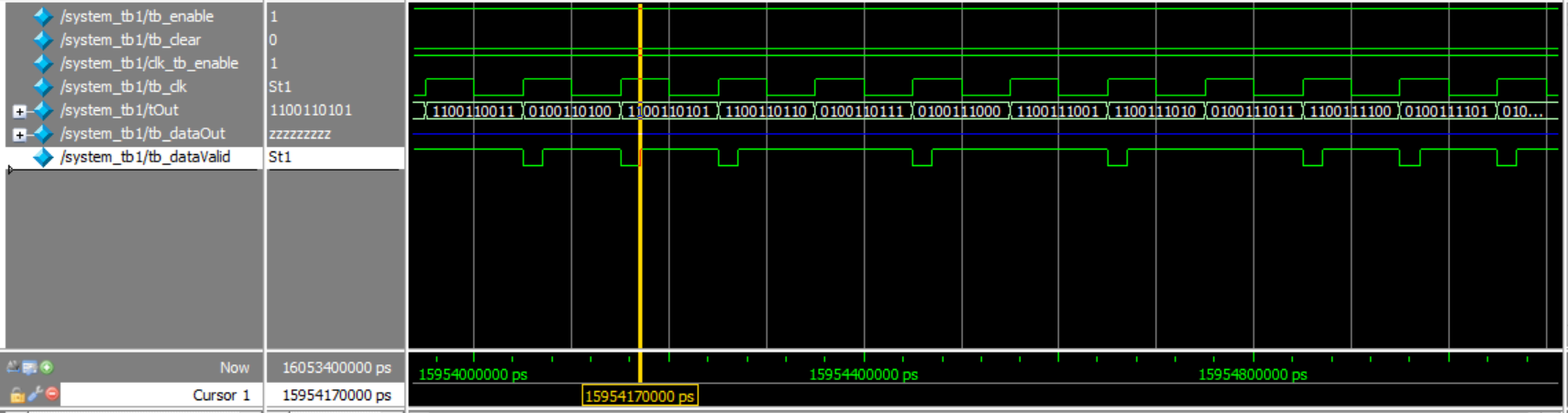
We can see that when we turn the clock off, the register does not update:

And when we turn it back on it begins to update again:

Therefore we verified that the 10 bit register was working.

Using the diagram provided in step 4, I created two modules for system\_tbX; a transmit module, and a receive module. Transmit involved linking up the modules counter\_9bit, hc280, and register\_10bit. Receive linked register\_10bit, hc280, and an xor gate. In system\_tb1 and system\_tb2, transmit was hooked up the receive module. The only difference between system\_tb1 and system\_tb2 was the clock provided.

Building the system\_tbX module, it was important to determine the minimum clk period that will result in valid output. The counter module has a 15 ns delay, thus we must wait 15 ns after posedge for the counter to update. Furthermore, the parity checker has a 20 ns delay to the odd bit. Thus we must wait at least 35 ns to update. Note that on the receive side, the XOR gate takes input from the odd output of a parity checker, which takes input from the register on the receive side. It also take input from the same register. Due to the 20 ns delay in the parity checker, there will always be a 20 ns period in which the data\_valid xor checker will have invalid output.

Choosing a 100 ns delay, the system\_tb1.v output the valid waveform below. As discussed above, there are 20 ns blips in data\_valid.

Setting the clock below the minimum threshold, to 20 ns, for system\_tb2 yields:

**Conclusions**