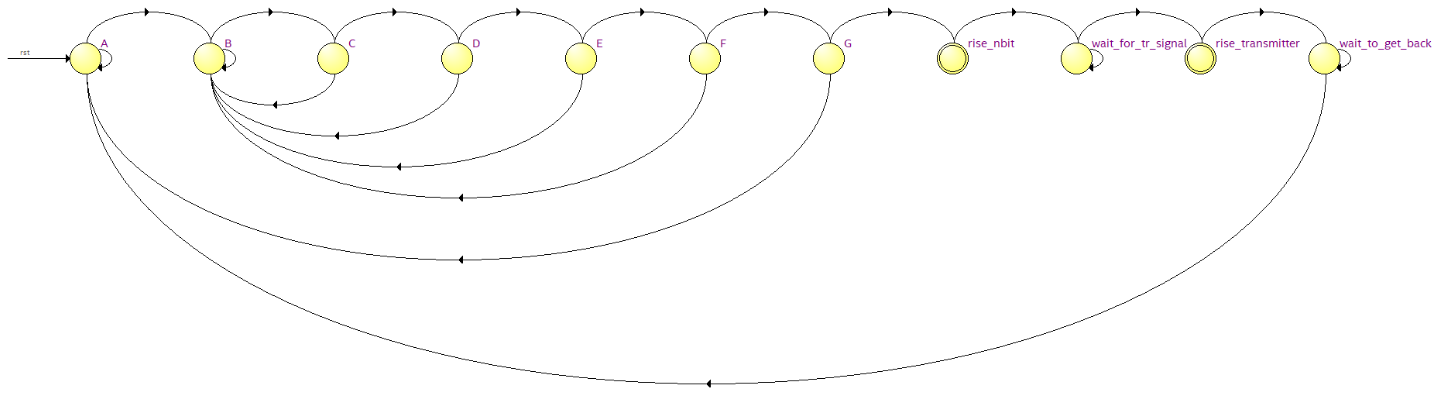
Question a)

Part 1)

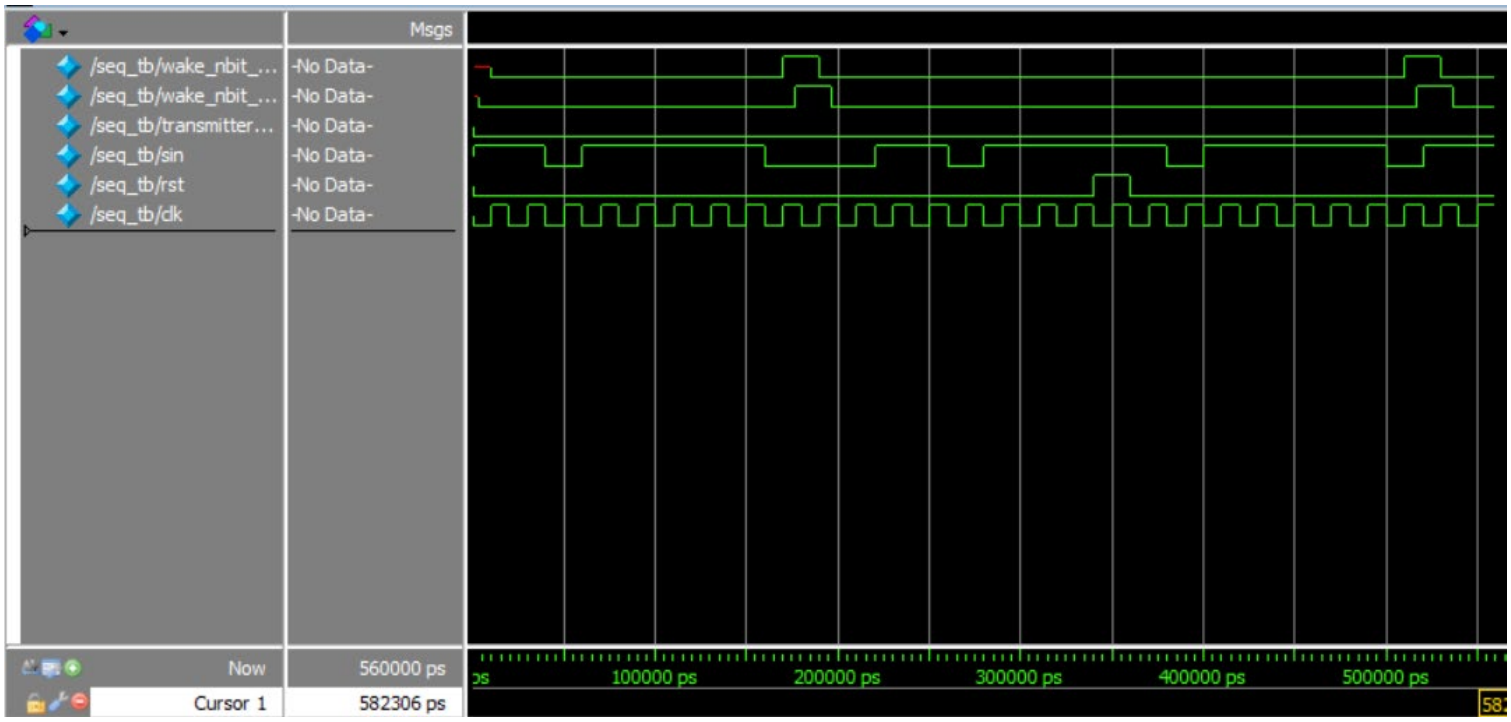
The Verilog code for this part is in a>1. Let's take a look at the state machine for this part:



There is 6 states for the sequence detector part, then one state is a combination of issuing the signal of our nbit counter, one state to wait for the signal coming back from the nbit counter, next state is issuing the signal for transmitter and final states waits for a signal coming from transmitter to do this all over again.

Part 2 and 3)

The codes for this part is in folder 2 and 3. After synthesizing using quartus, here is the testbench of this part:

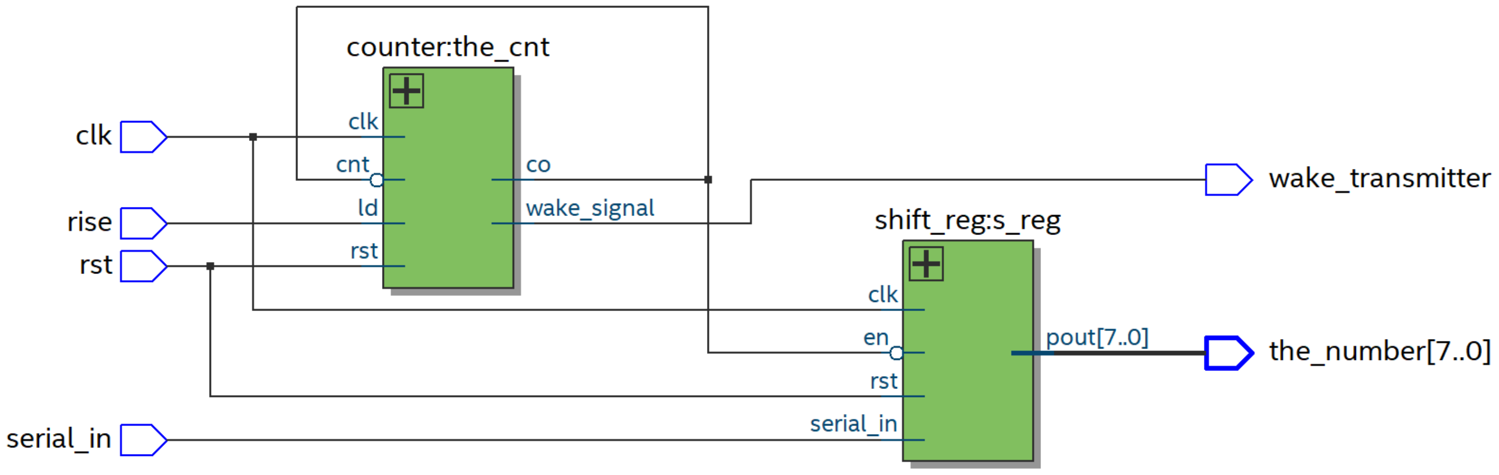


Note that all of the operations is not tested here and it will be tested in question d. as you can see the outputs of pre synthesis is ahead of the post synthesis and it has delay.

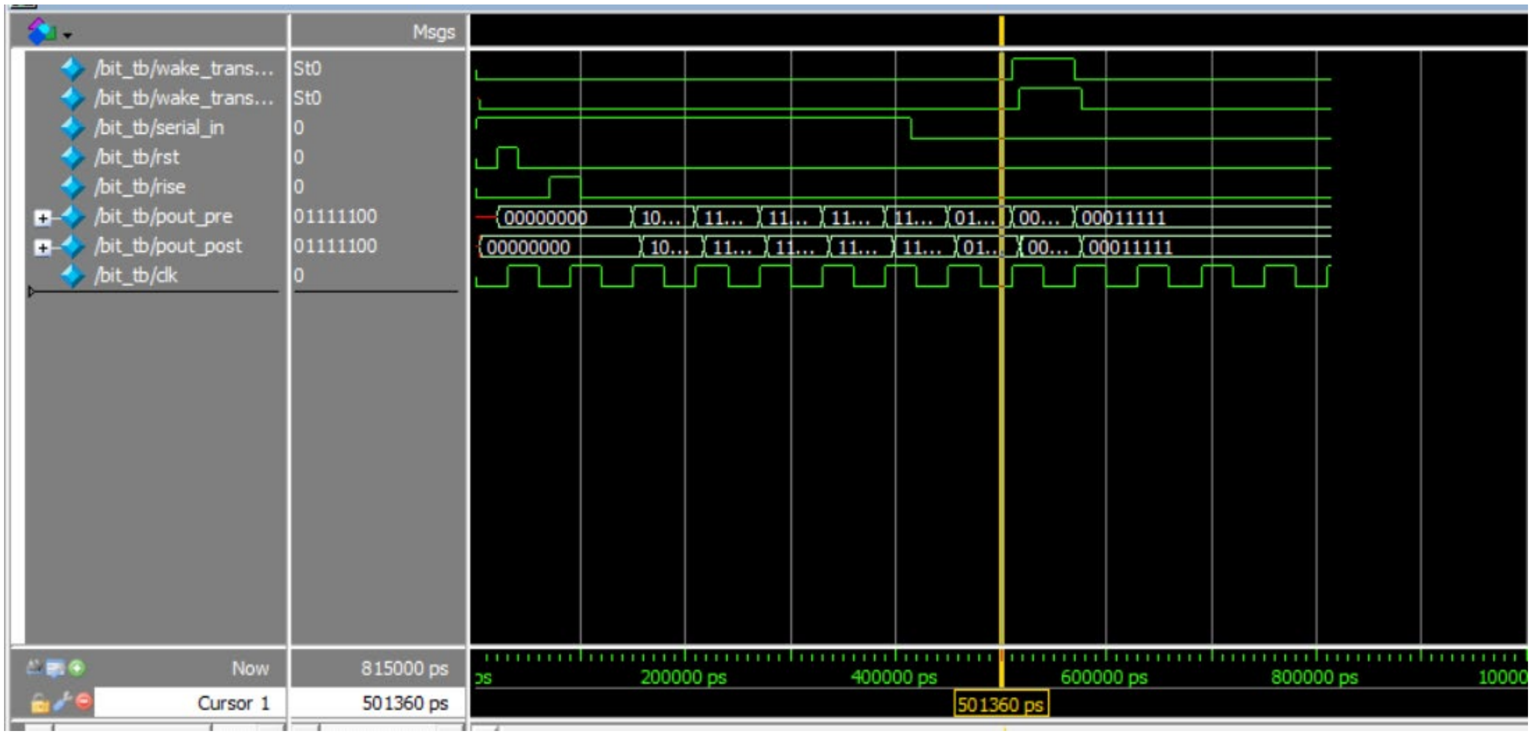
Question b)

Part 1 and 2)

In this part we must note some important things: first Is that we cannot lose any clocks, not before nor after. So the coming clock should start to shift si to register. And the other way, we cannot lose any clocks after the 8th clock. As we set co as the shen, we will lose the last shift if we use a 0 to 7 counter, because when it reaches to 7, it's co is 1 and it cannot shift the last number. So we will use a 0 to 8 counter so that the hold state won't effect anything. On the other hand, the signal that raises the transmitter must wake one clock before the shifting Is over, though when the job of shift register is over, the transmitter will start to put si on output. So based on these explanations we build our circuit. After that, we synthesis our circuit:



And after that, we test our circuits:

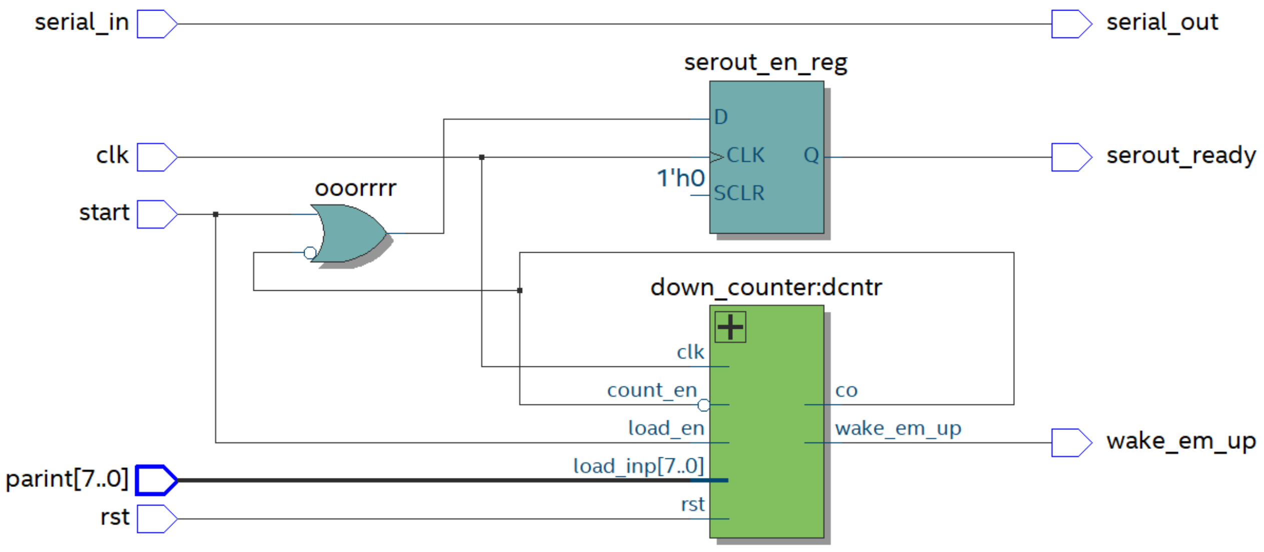


As you can see, we first reset the circuit and then we give it the rise signal. Without losing any clocks, it will start to shift the values into the register, and one clock before the shifting is over, it will note the outer blocks. After 8 shifts it’s job will be over.

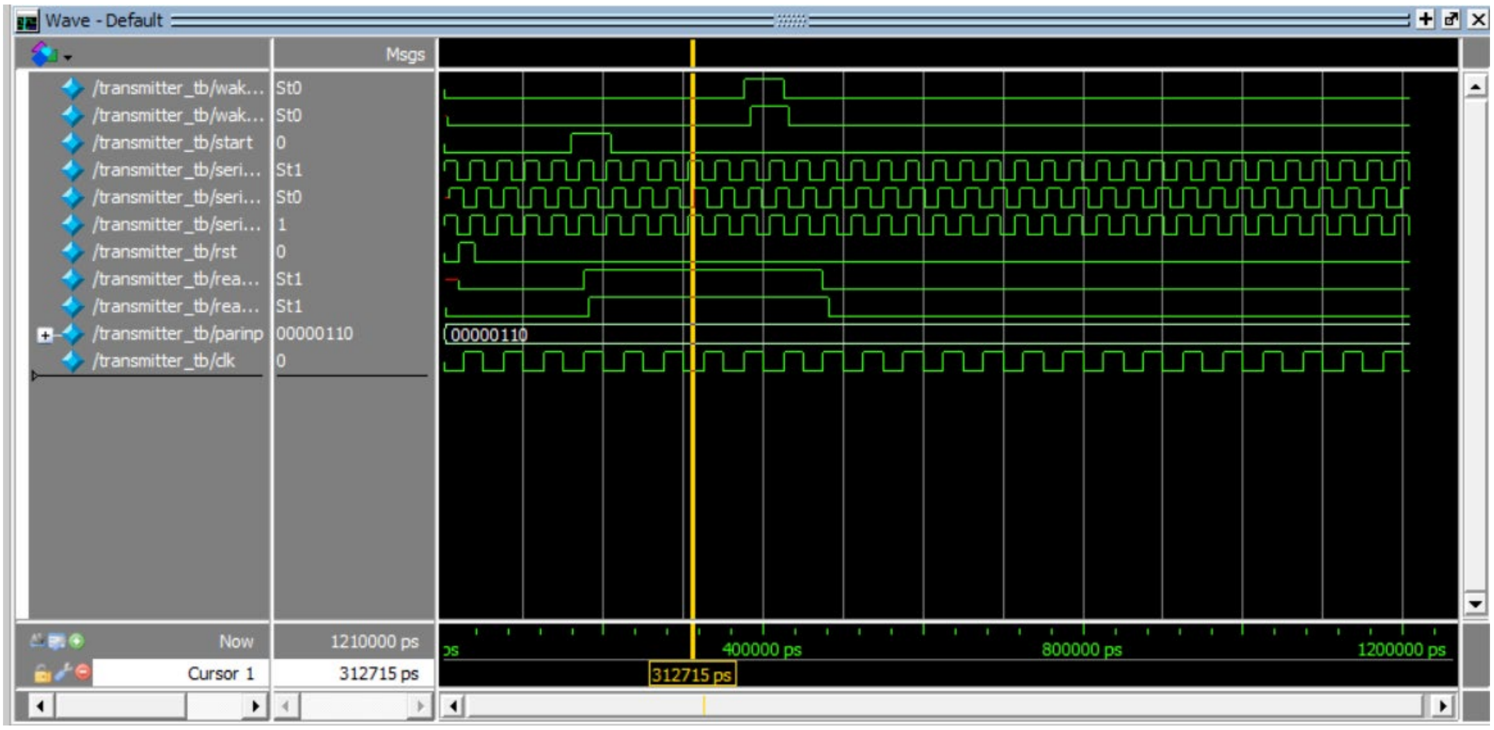
Question c)

Part 1 and 2)

We use a down counter and some extra gates for this part. Loading will take one clock to be done, and the signal of load will propagate between two clocks, so to prevent serout_en to get active before the clock comes, we put a flip flop on his way. On the other hand, as flip flop’s memory will last after our final clock, we must dectreament the load value not to count one more extra clock. The down counter therefore will count one time less than what we want + one more extra clock which will be true after all. Based on this explanations here is the synthesized version of our circuit:



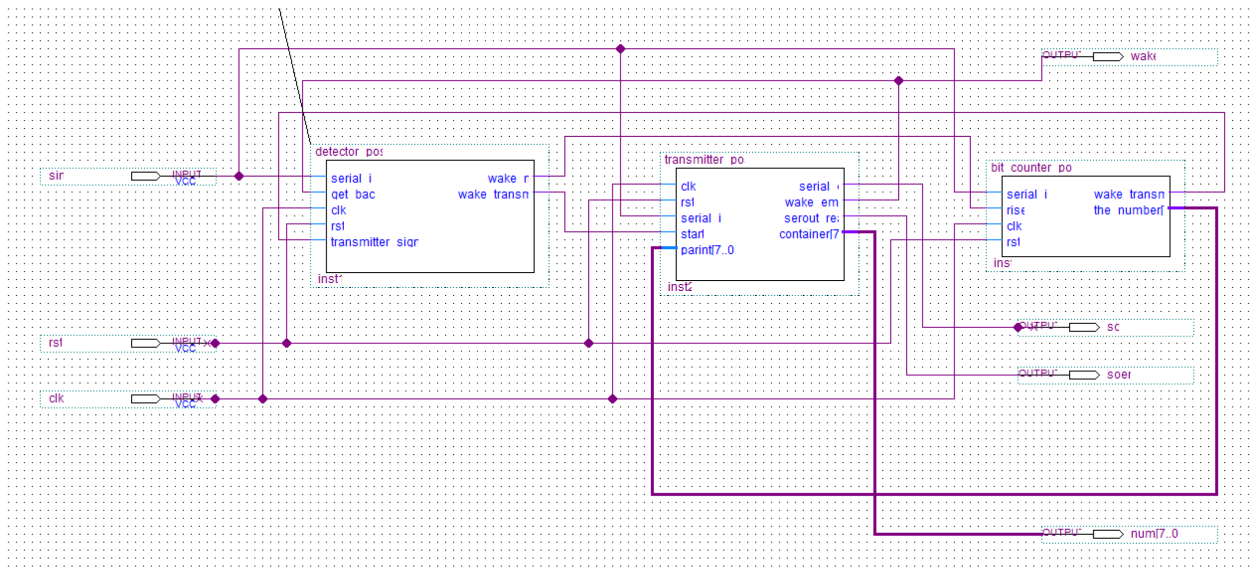
Wake em up’s duty is to wake the sequencer up and make it ready for the next input, though as we explained earlier it must raise them one clock before the job is done. Overall, here is the test bench:


Wave - Default
Msgs
/transmitter_tb/wak... St0
/transmitter_tb/wak... St0
/transmitter_tb/start 0
/transmitter_tb/seri... St1
/transmitter_tb/seri... St0
/transmitter_tb/seri... 1
/transmitter_tb/rst 0
/transmitter_tb/rea... St1
/transmitter_tb/rea... St1
/transmitter_tb/parinp 00000110
/transmitter_tb/clk 0
00000110
Now 1210000 ps
Cursor 1 312715 ps
400000 ps
800000 ps
1200000 ps
312715 ps

And as you can see ready will count 6 clocks right after the start turns to 1, and as we mentioned the start came between two clocks but it did not effect the output.

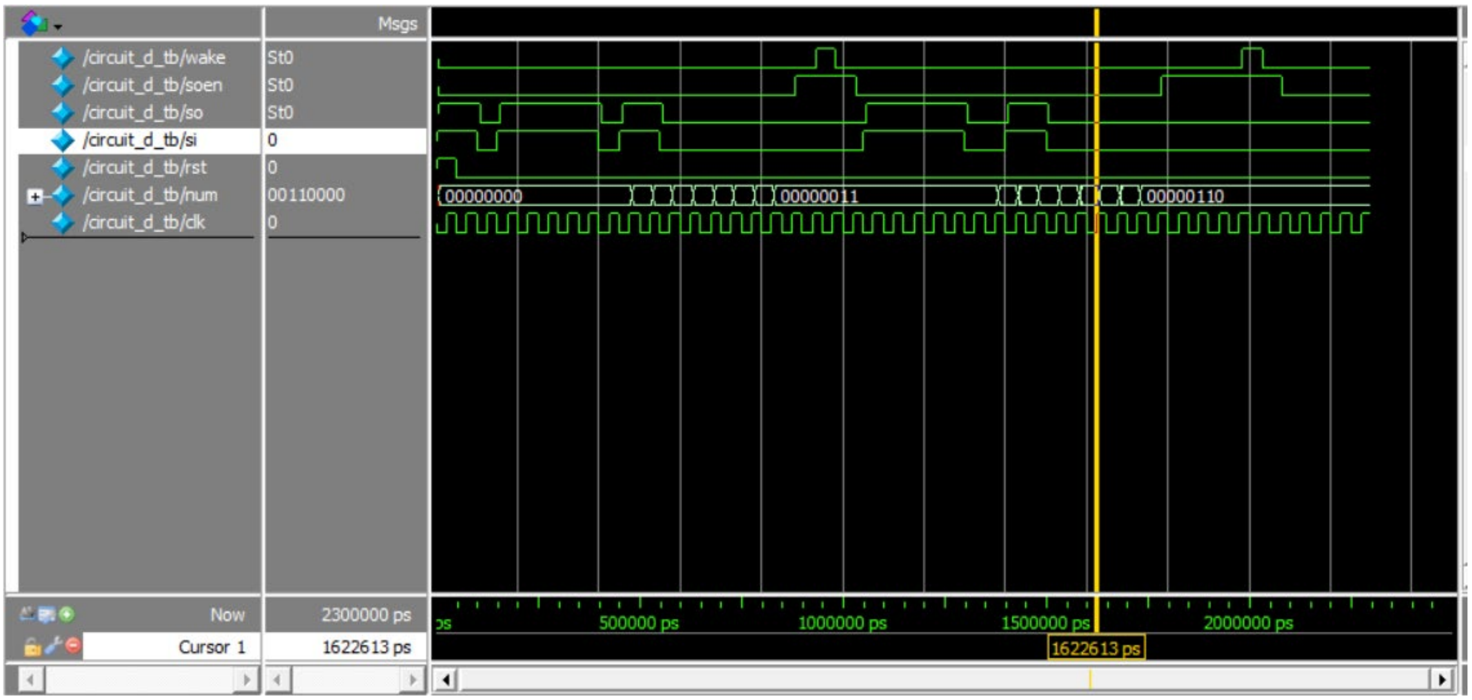
Question d)

In this part we combine all parts of this circuit. Let's take a look at our circuit first:



Outputs of num and wake are to see the shift contains and the wake timings.

The details have been discussed earlier. We will show the test bench now:



As you can see, the sequence will be detected and then immediately the shifting process will begin (as you can see for 2 first clocks we have 1 on si and they both are captured and interpreted as 3). After 8 shifts, with no losing in clock counts, the serout en will be ready and after 3 clocks it will be deactive again, just as we wanted. Again without losing any clock, it will start to detect the sequence coming after the last sequence (an immediate 0 and then 1 follows), and again all of this process will be repeated for 6. Finally as you can see, the circuit is working correctly without any loss in clock counts.