

EXPERIMENT 3. Flip Flops and Sequential Circuits

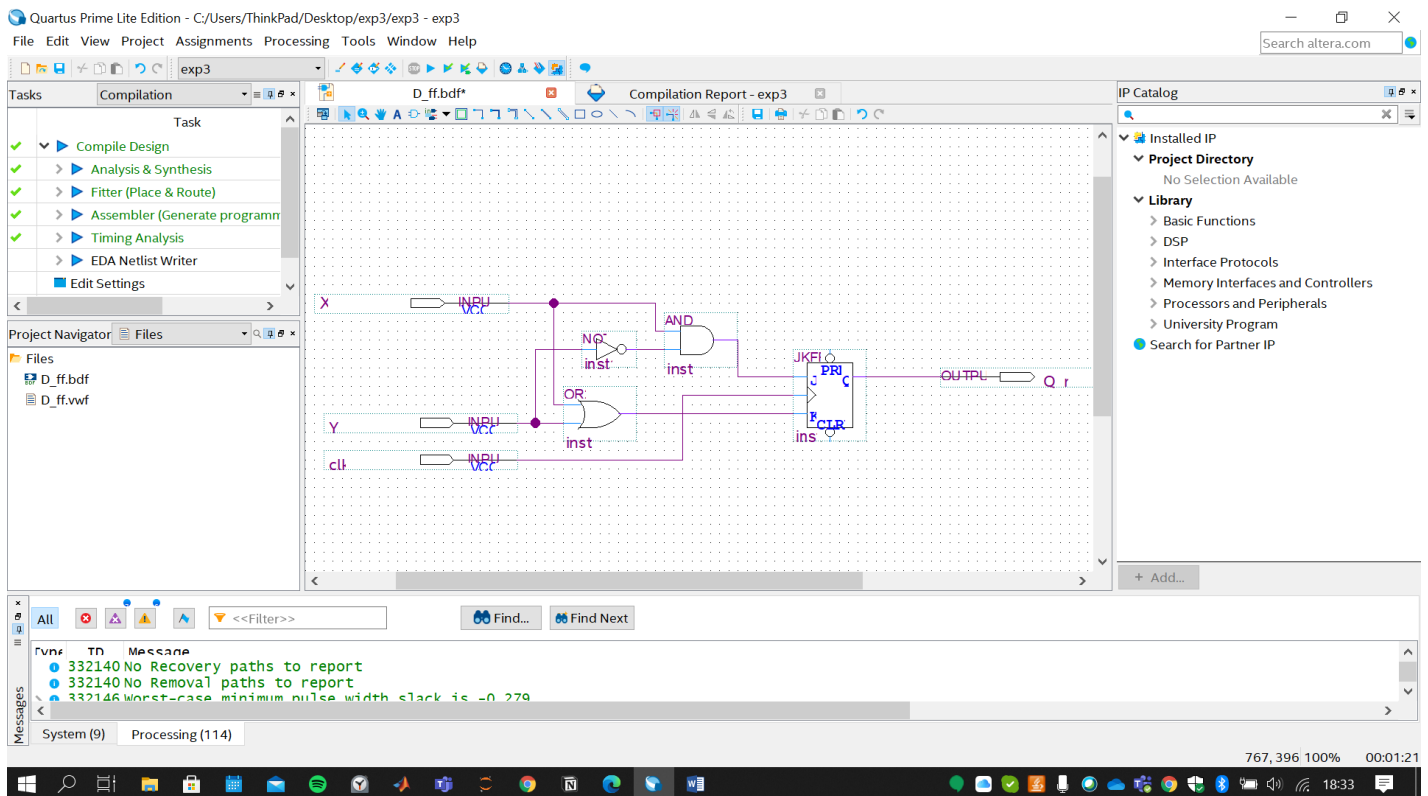
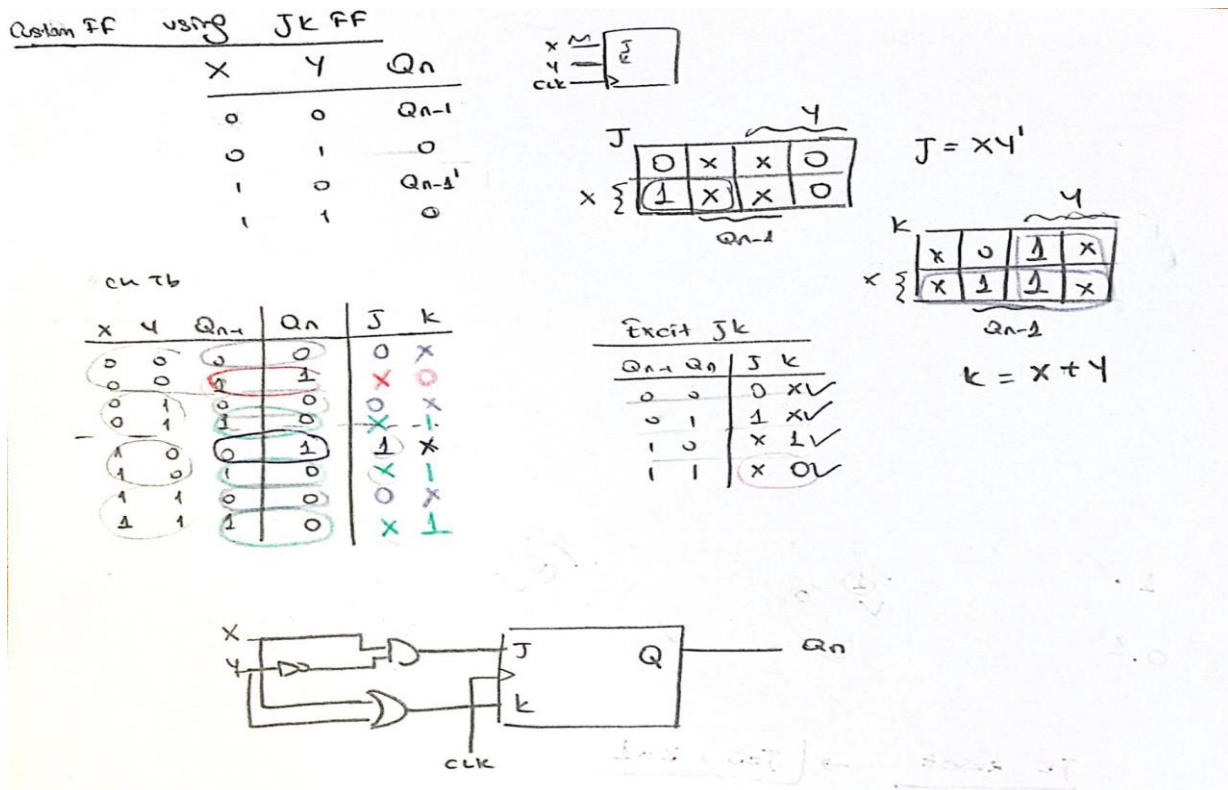
NAME : Zeynepnur ŞAHİNEL
ID : 2305399

Important Notes:

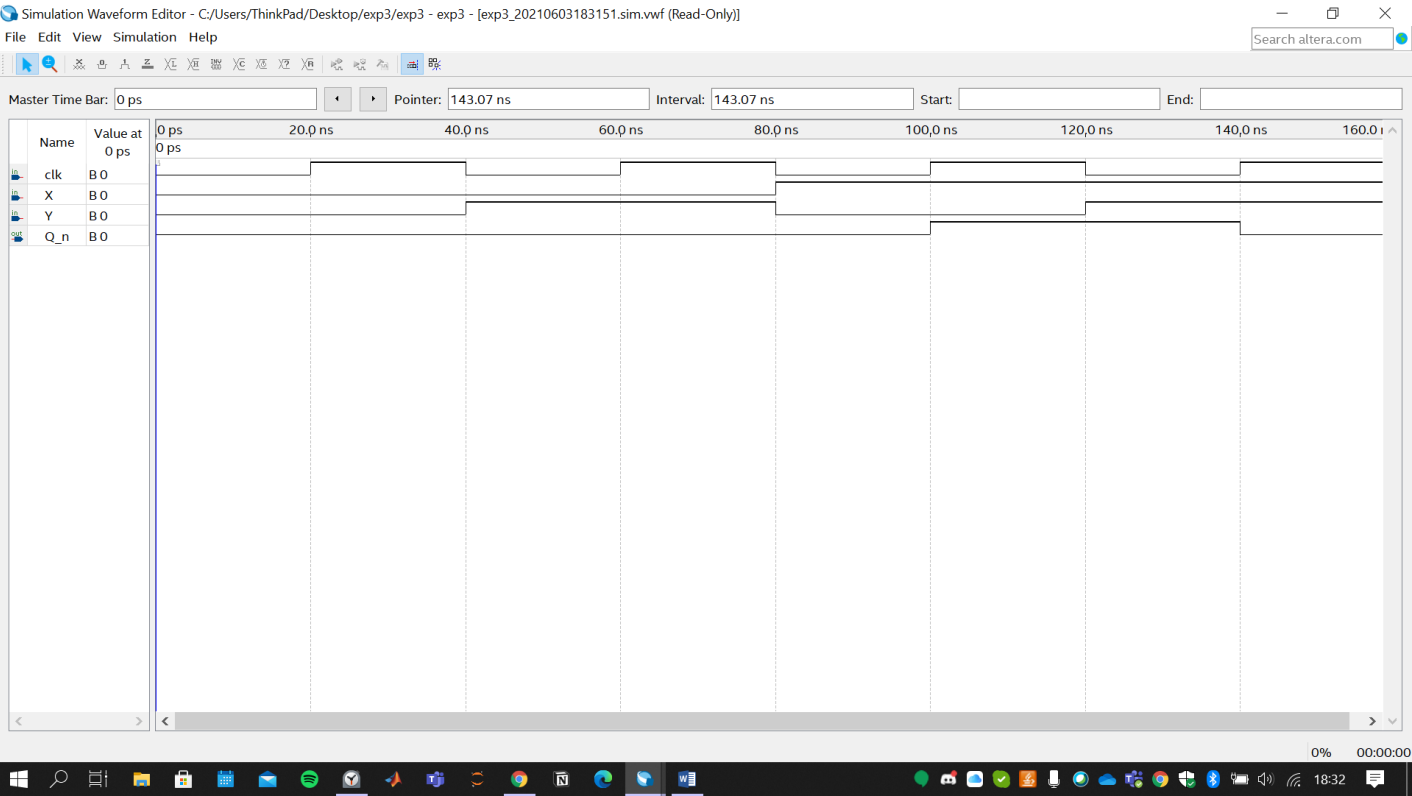
- You will be expected to provide screenshots of your designs and simulation results. You have to take full screenshots for all of them. You can use the full-screen mode of the Snipping Tool for this. **DO NOT crop** any image.
- If the image is too dense or too small that makes it difficult to view, then add zoomed-in images as EXTRAS (again full screenshots).
- You will name your report as “**Exp# _ Report_StudentID.pdf**”
- You may make comments if any discrepancy occurs between your results and your expectations. You can write your expectations, possible reasons of that discrepancy, etc. below related section. **In addition to this report, DO NOT FORGET to upload your project files at the end of each part.**

A. Custom Flip Flop

A1) Please post the screenshot of the block diagram of the custom flip flop that you have designed. Do not forget to mention which custom flip flop you are designing. Make sure the blocks and the wiring are clearly visible. Briefly explain the methodology behind your design.

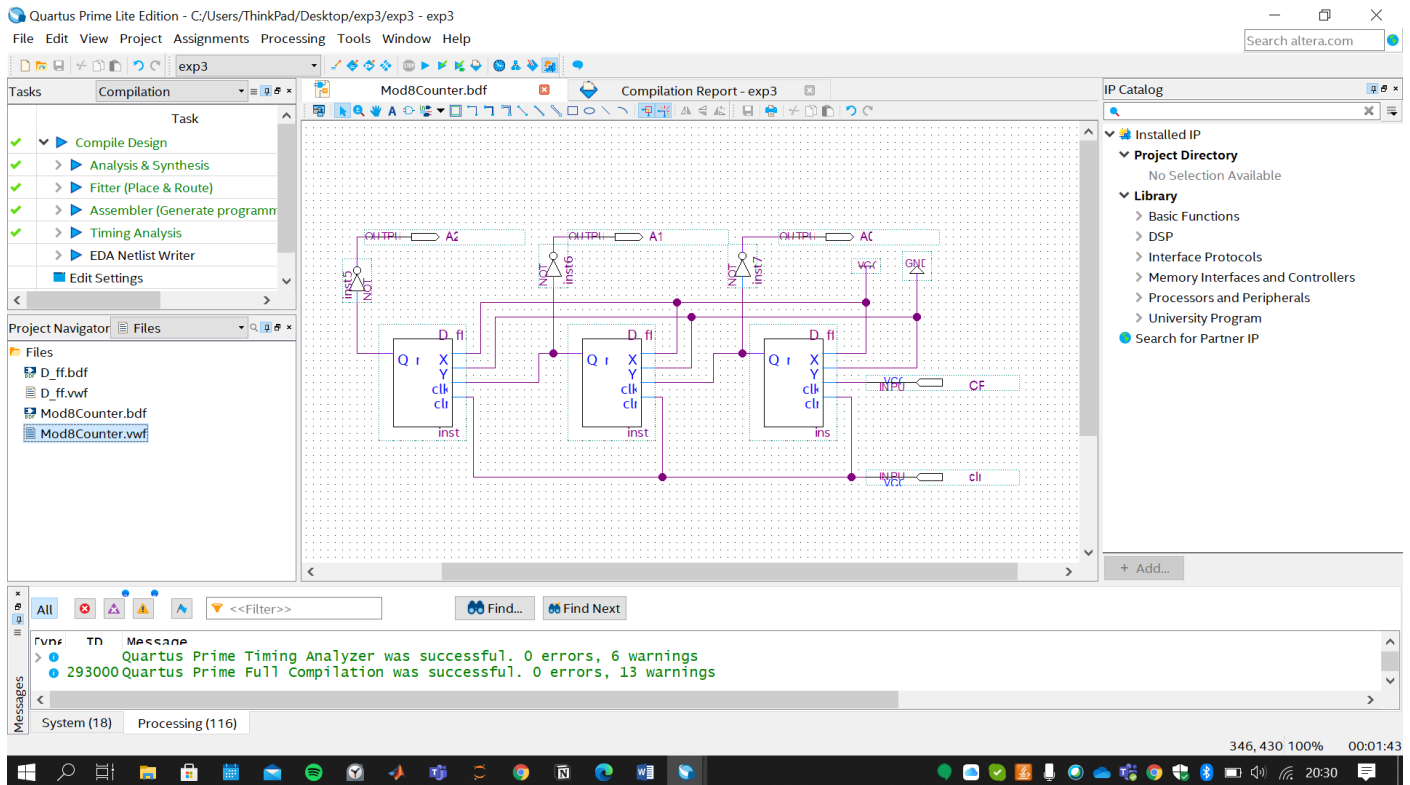


A2) Put the screenshot of the simulation results of the custom flip flop that you have designed. Please make sure there are enough examples of each condition in the sample set, which are clearly visible in the screenshot.



B. Design of Mod 8 Counter

B1) Please put the screenshot of the mod 8 counter that you have implemented via the custom flip flop. Pay attention to the clear demonstration of the blocks and the wiring between the blocks. Briefly explain your approach on the design task.



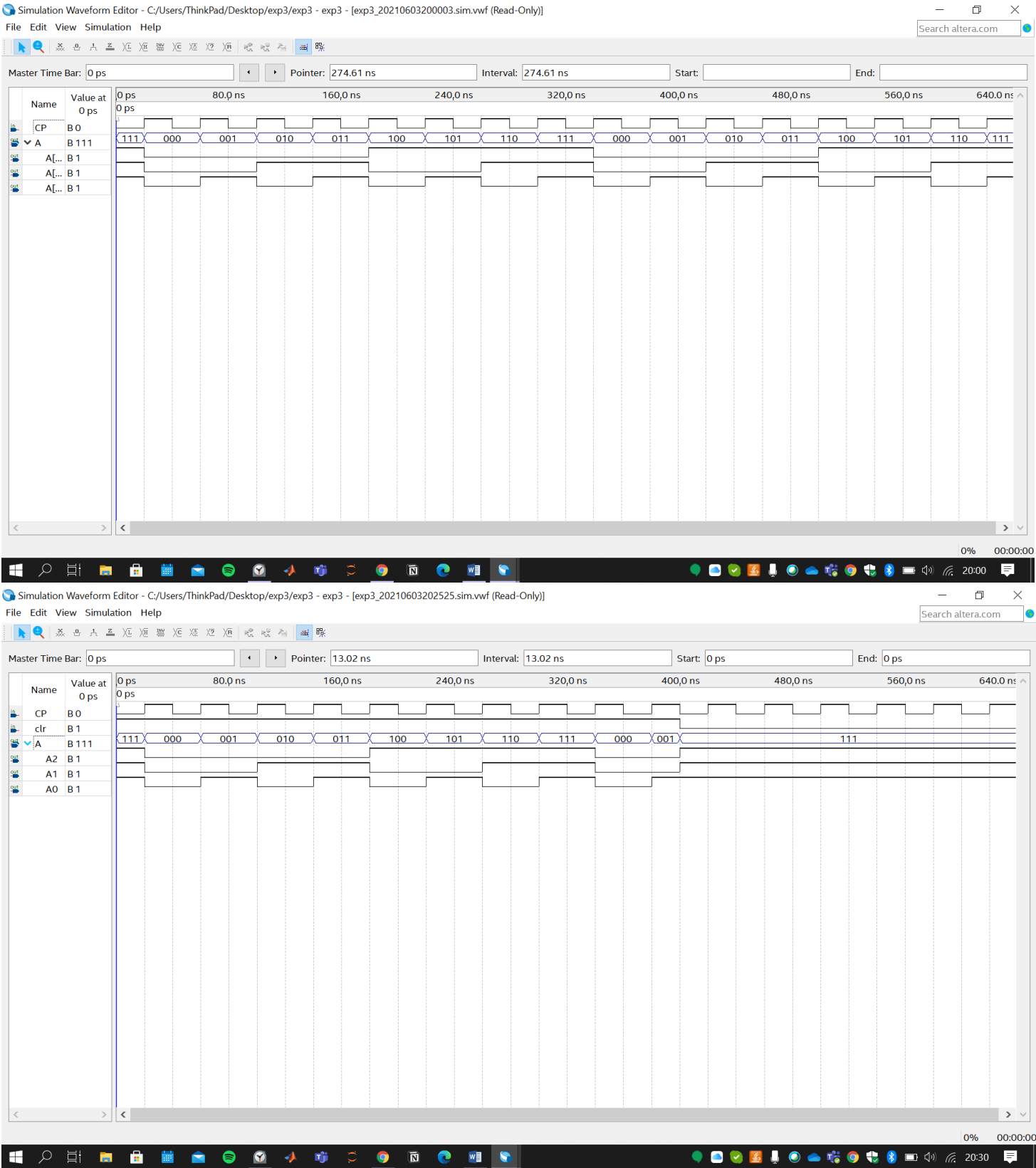
Since I had designed 8mod counter, there is no need to imply clr input. However, question asked to do. Thus, I also added the clr input.

Moreover, before constraining counter, I used asynchronous clear input of the JK FFs to achieve XY Custom FF with clr input.

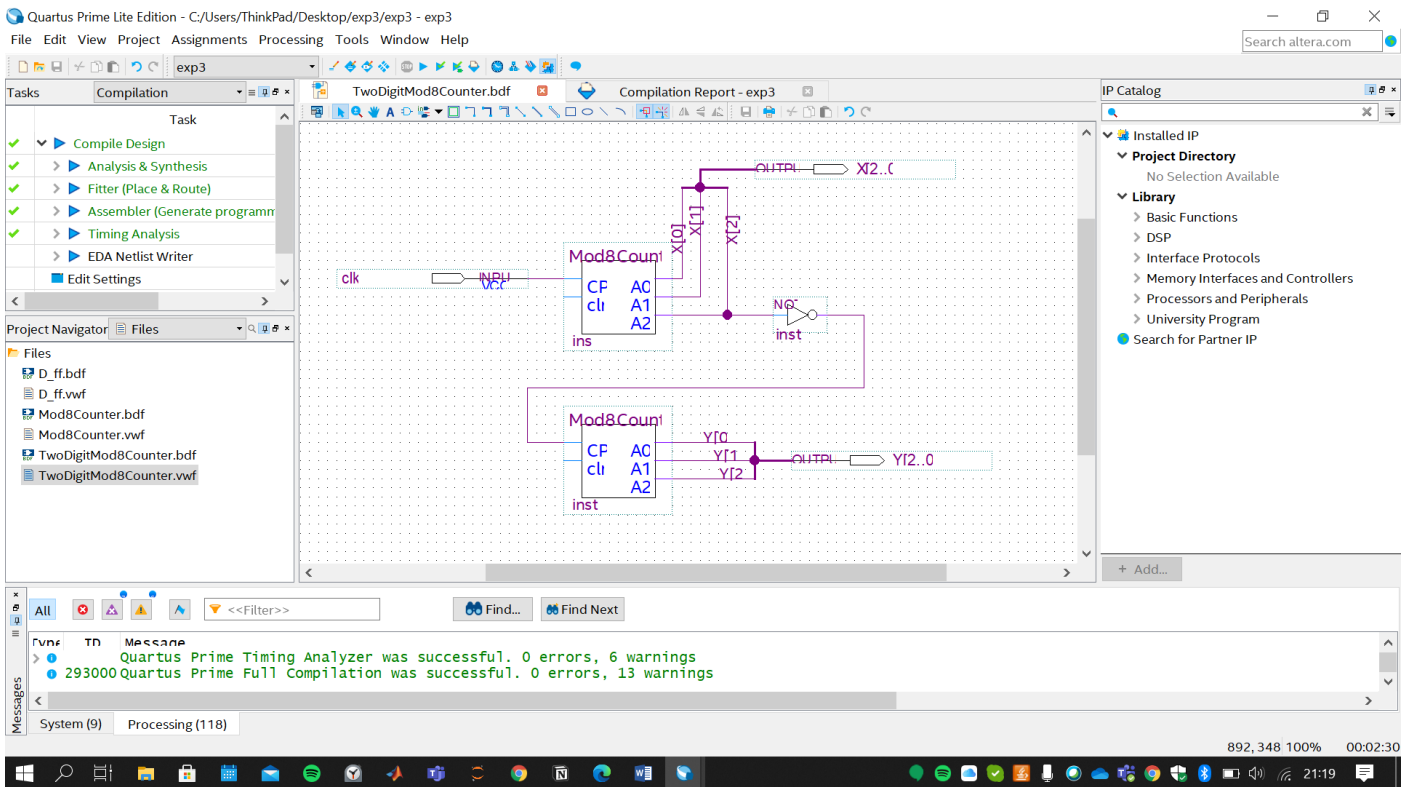
In this counter, I used the ripple(asynchronous) counter. Meaning, outputs of the XY FF were connected to the clk of the after XY FF. To achieve mod8, three custom FF were enough.

The question asks positive triggered FF and up counter at the same time; therefore, **complements** of the Qs (output of the XY FF) were used as output.

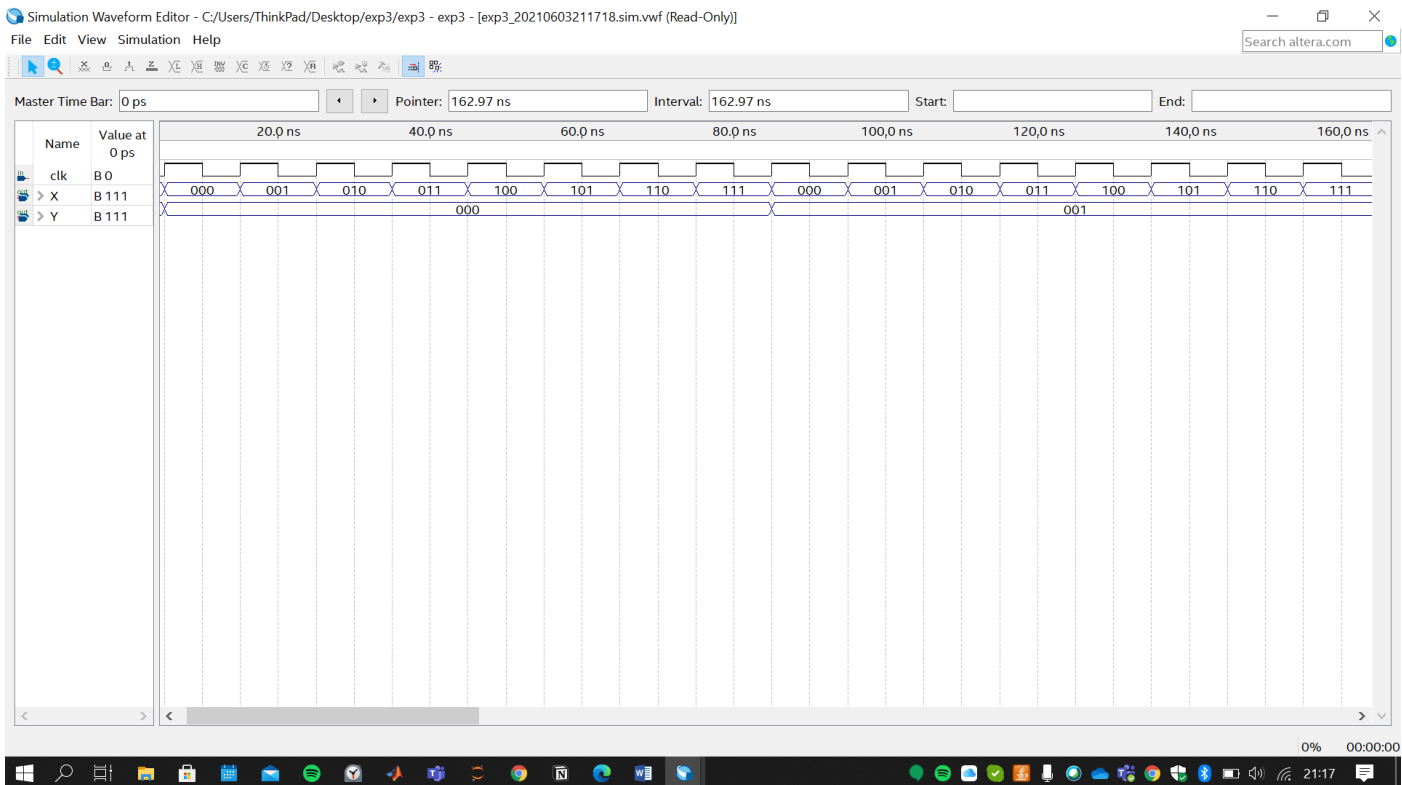
B2) Put the screenshot of the functional simulation results. Make sure that you have provided the proper operation for each output configuration.

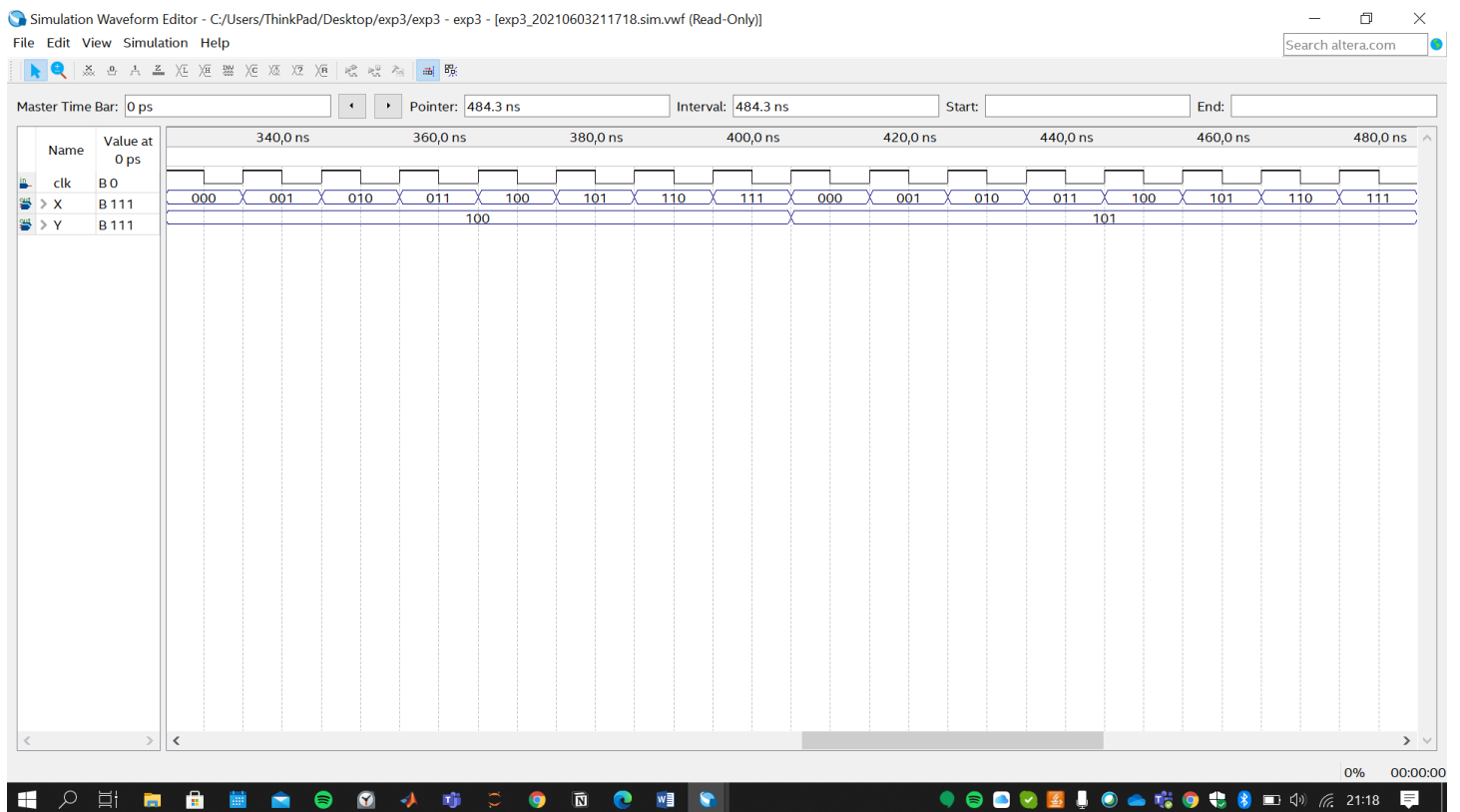
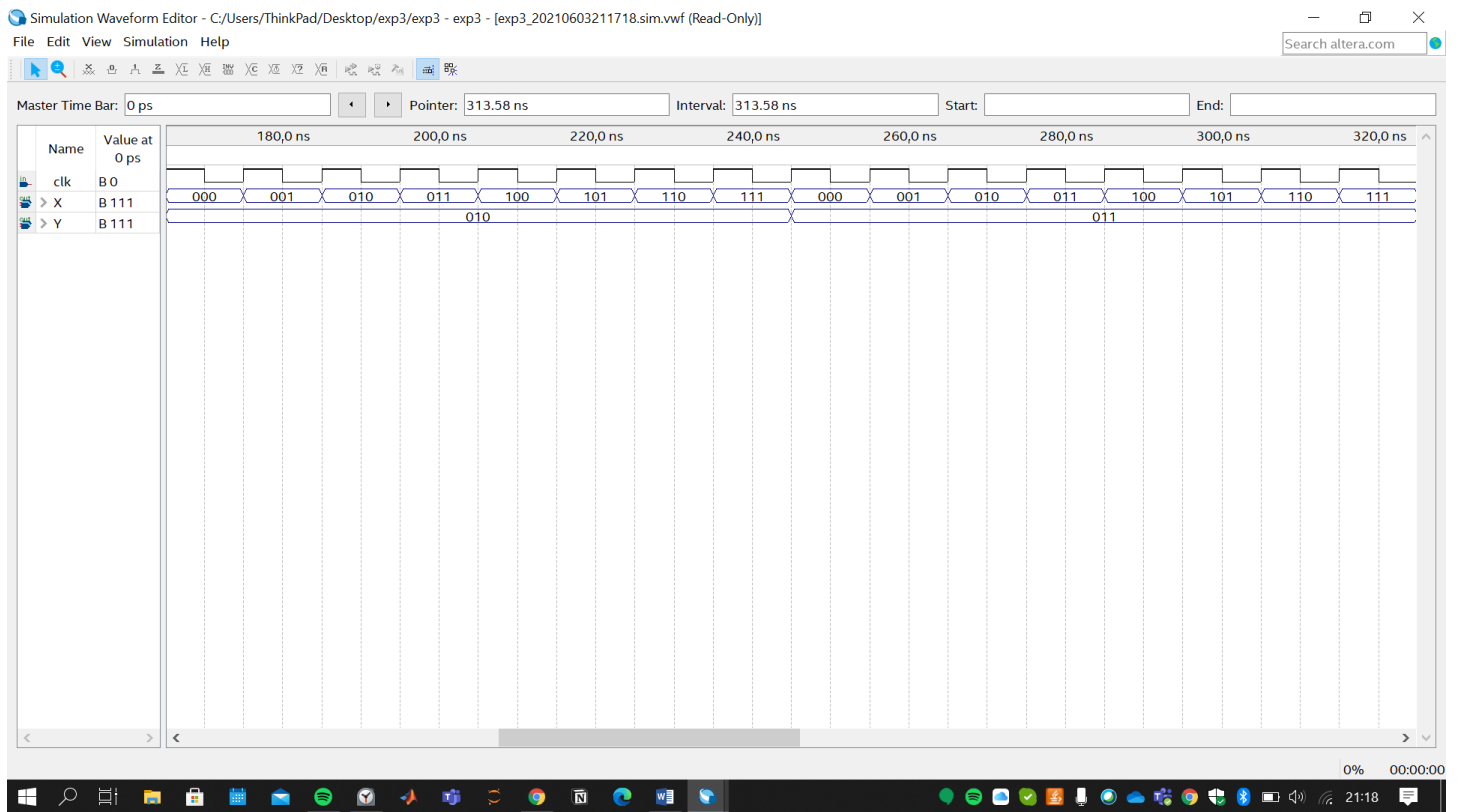


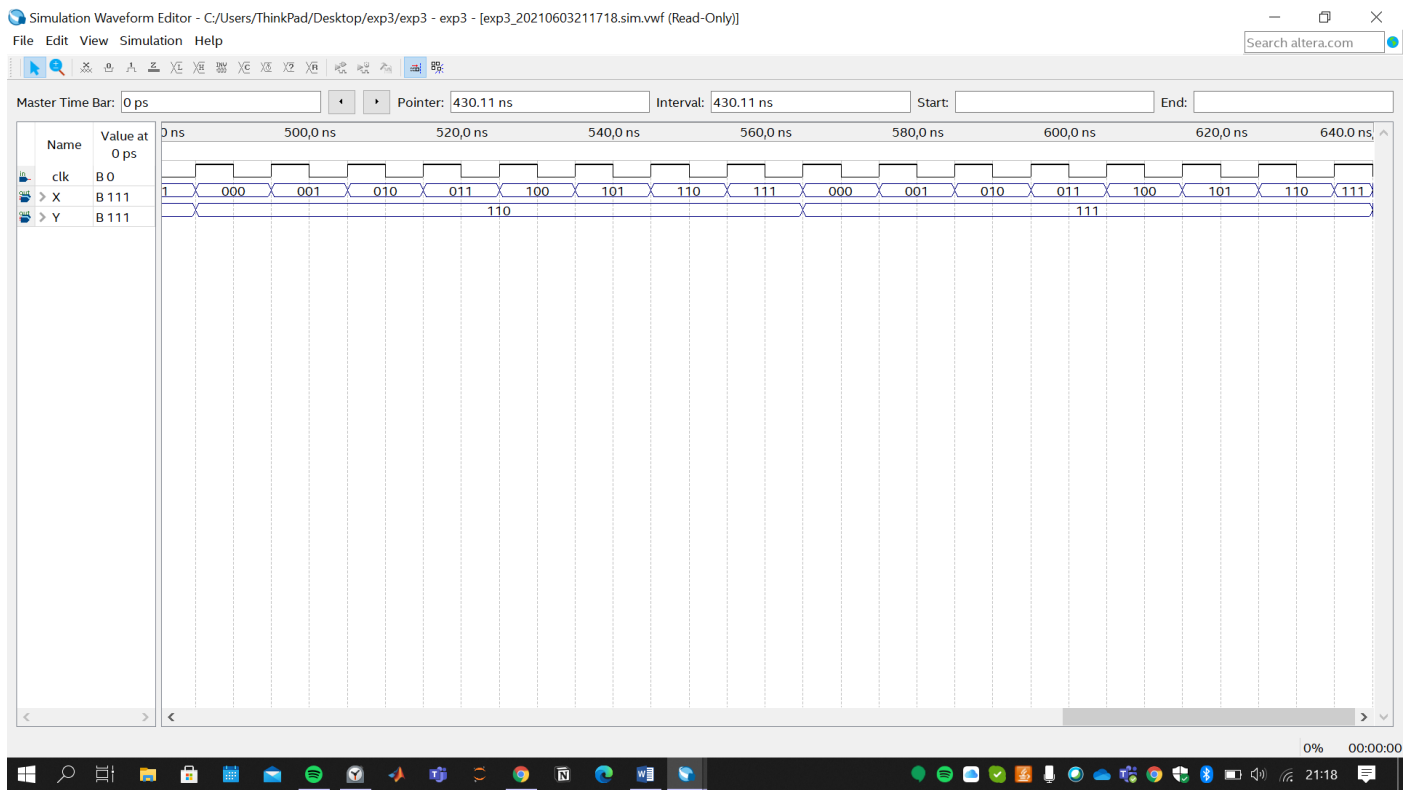
B3) Please put the screenshot of the block diagram for the 2-digit base 8 counter that you have designed via mod 8 counter in part B1)



B4) Please put the screenshot of the functional simulation results. Make sure that you have provided the proper operation for each output configuration.



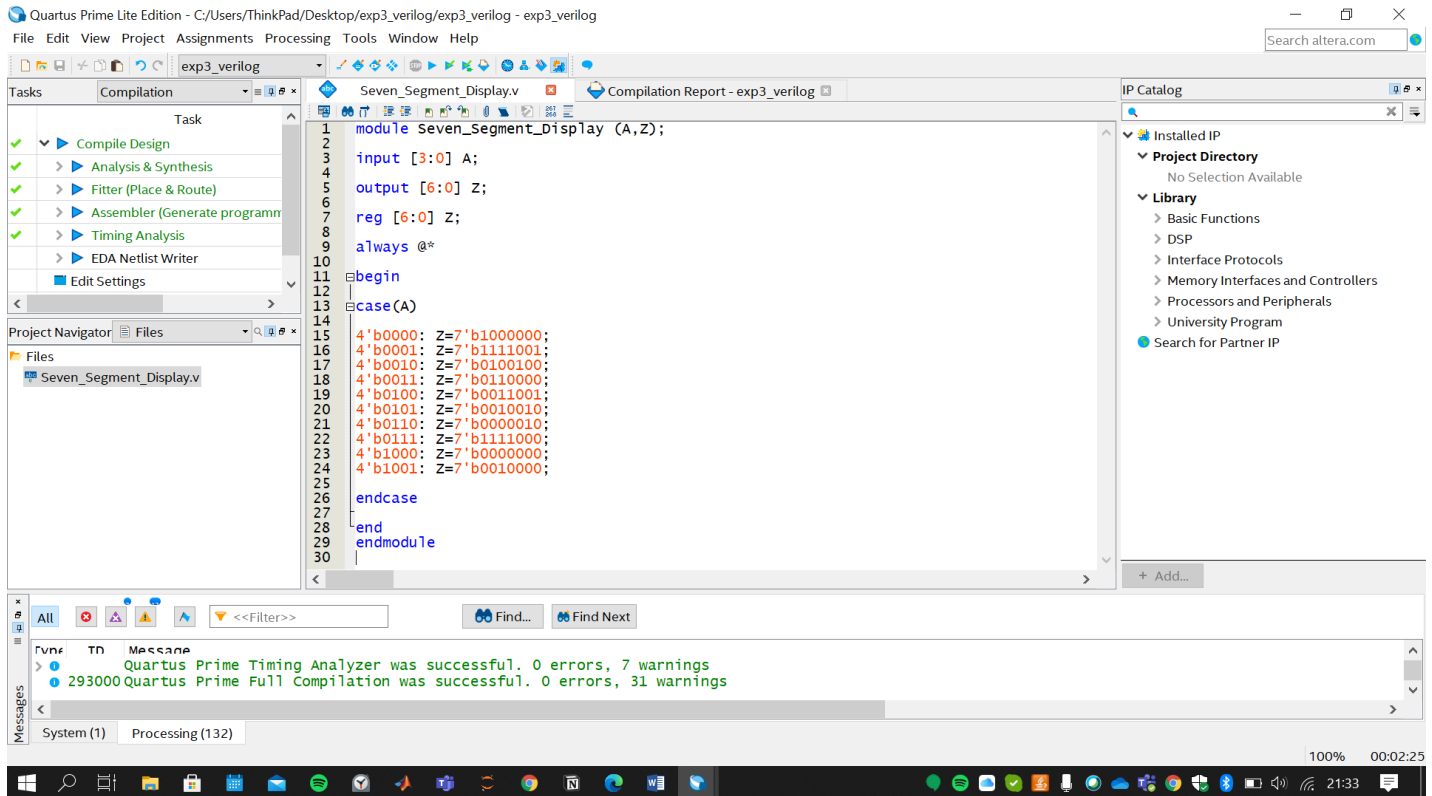




To achieve this pattern, complement of A3 (as negative triggered pulse) is connected as Clock Pulse of the second Mod8 Counter.

C. Implementation of the BCD to 7-Segment Decoder

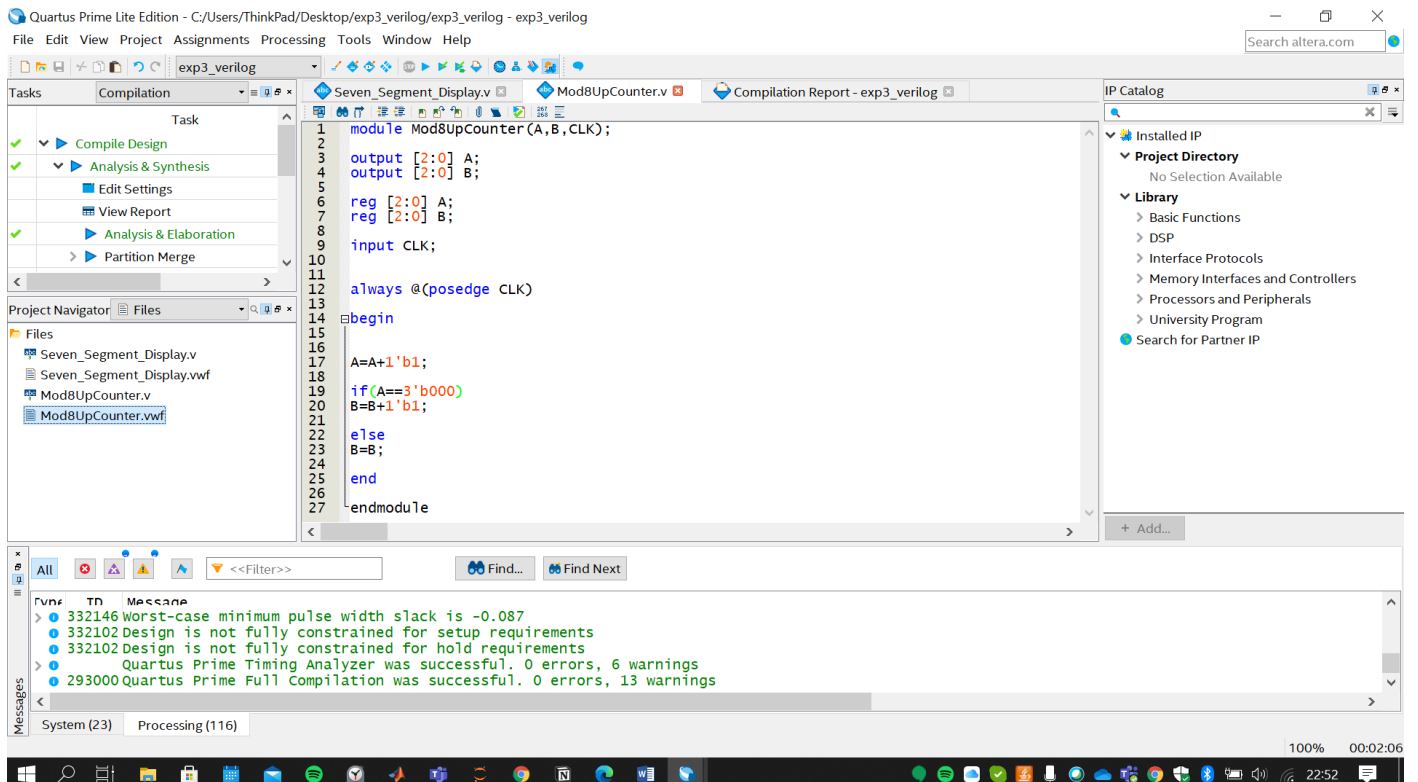
C1) Please put the screenshot of the Verilog code that you designed to implement the BCD to 7-Segment decoder.



The screenshot shows the Quartus Prime Lite Edition interface. The main window displays the Verilog code for a module named `Seven_Segment_Display`. The code implements a BCD to 7-segment decoder. The inputs are a 3-bit BCD value `A` (lines 3-4), and the output is a 7-bit vector `Z` (line 5). The code uses a `case` statement to map BCD values to 7-segment patterns (lines 13-24). The messages window at the bottom shows a successful compilation with 0 errors and 31 warnings.

```
1 module Seven_Segment_Display (A,Z);
2
3 input [3:0] A;
4
5 output [6:0] Z;
6
7 reg [6:0] Z;
8
9 always @*
10
11 begin
12
13 case(A)
14
15 4'b0000: Z=7'b1000000;
16 4'b0001: Z=7'b1111001;
17 4'b0010: Z=7'b0100100;
18 4'b0011: Z=7'b0110000;
19 4'b0100: Z=7'b0011001;
20 4'b0101: Z=7'b0010010;
21 4'b0110: Z=7'b0000010;
22 4'b0111: Z=7'b1111000;
23 4'b1000: Z=7'b0000000;
24 4'b1001: Z=7'b0010000;
25
26 endcase
27
28 end
29 endmodule
30
```

C2) Please put the screenshot of the Verilog code that you designed to implement the 2-digit-base-N counter (use the same N as in part-b).

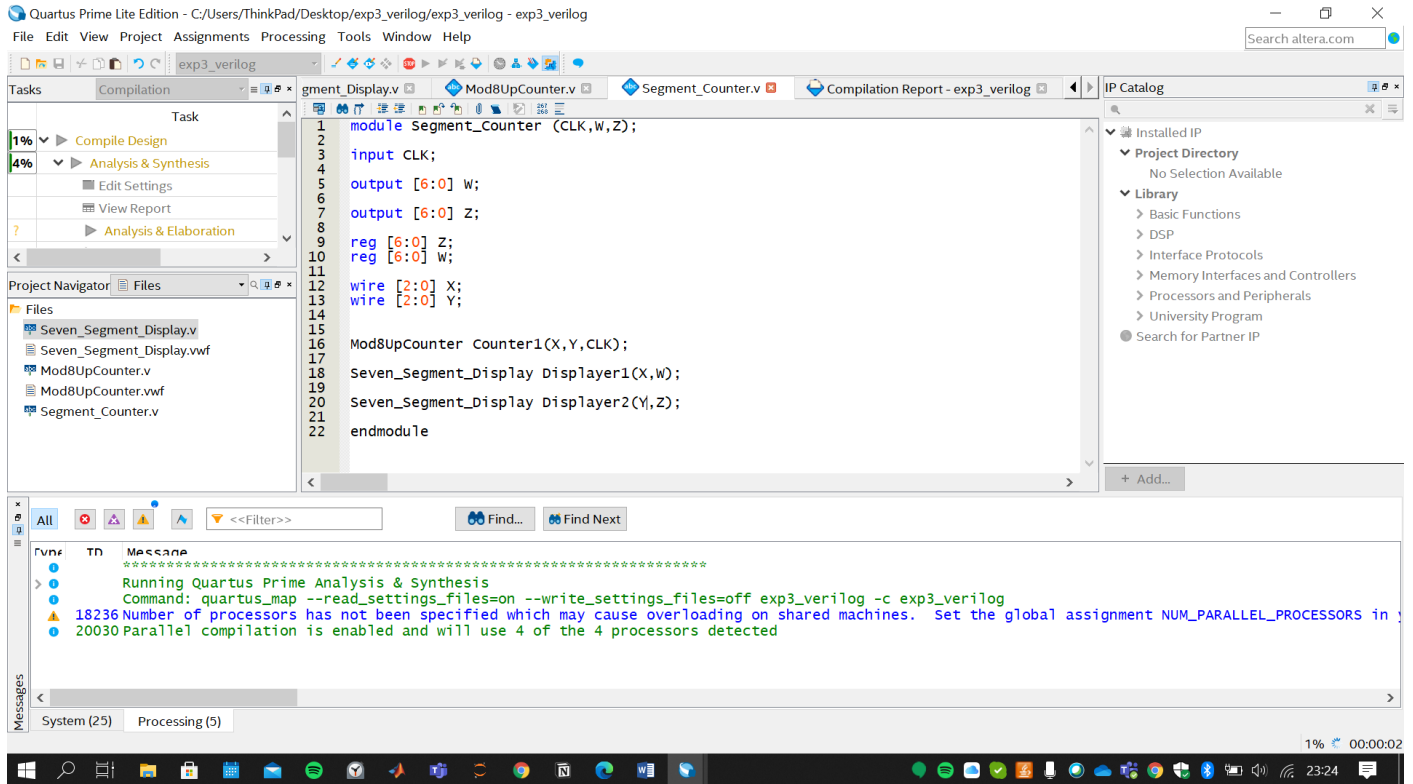


The screenshot shows the Quartus Prime Lite Edition interface. The main window displays the Verilog code for a module named `Mod8UpCounter`. The inputs are a 2-bit value `A` (line 3), a 2-bit value `B` (line 4), and a clock signal `CLK` (line 9). The code uses a `always` block to increment the counter on the rising edge of the clock (lines 12-13). The messages window at the bottom shows a successful compilation with 0 errors and 13 warnings.

```
1 module Mod8UpCounter (A,B,CLK);
2
3 output [2:0] A;
4 output [2:0] B;
5
6 reg [2:0] A;
7 reg [2:0] B;
8
9 input CLK;
10
11
12 always @(posedge CLK)
13
14 begin
15
16 A=A+1'b1;
17
18 if(A==3'b000)
19 B=B+1'b1;
20
21 else
22 B=B;
23
24 end
25
26 endmodule
27
```

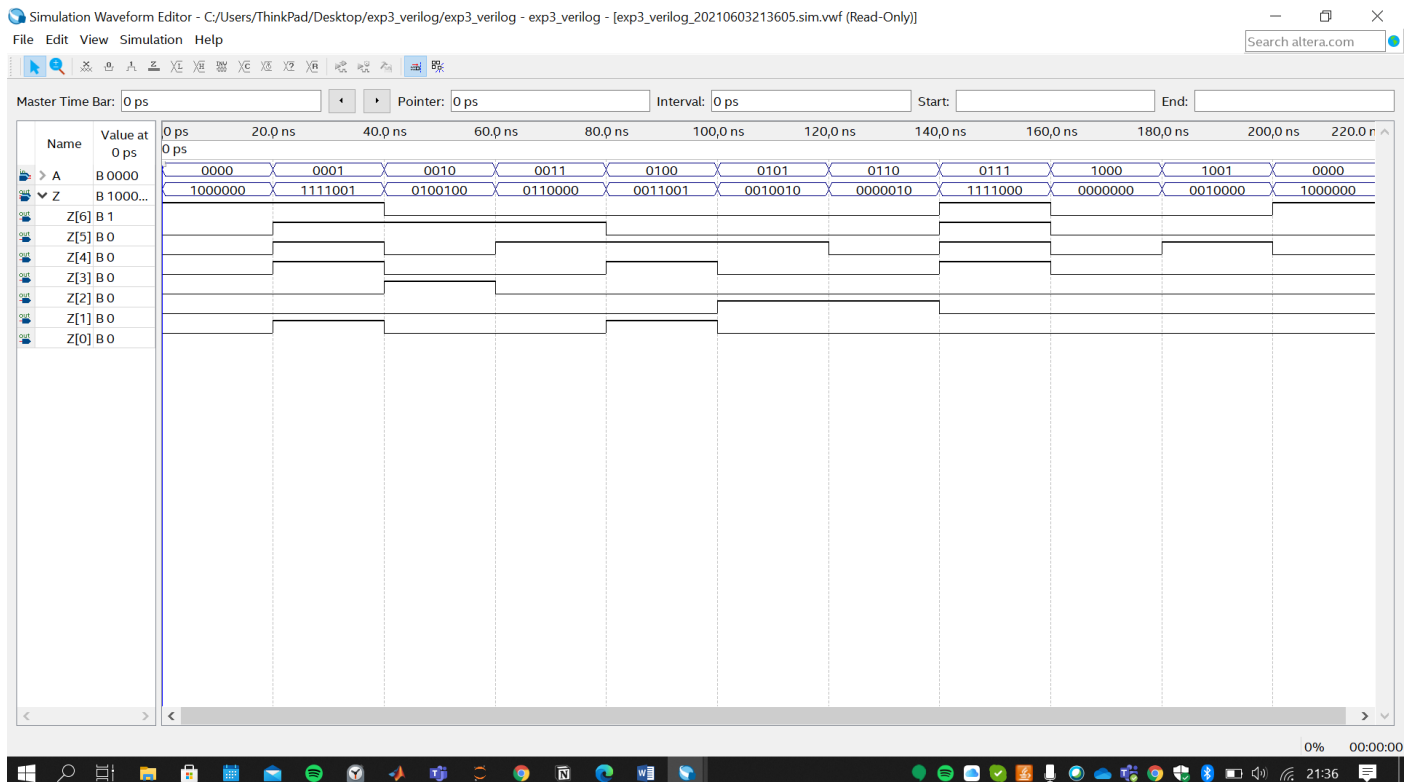
C3) Please post the screenshot of the final schematic of your design.

Segment Counter Code



C4) Please put the screenshots of the proper simulation results of your design.

BCD to 7-Segment decoder Simulation



2-digit-base-8 counter Simulations

