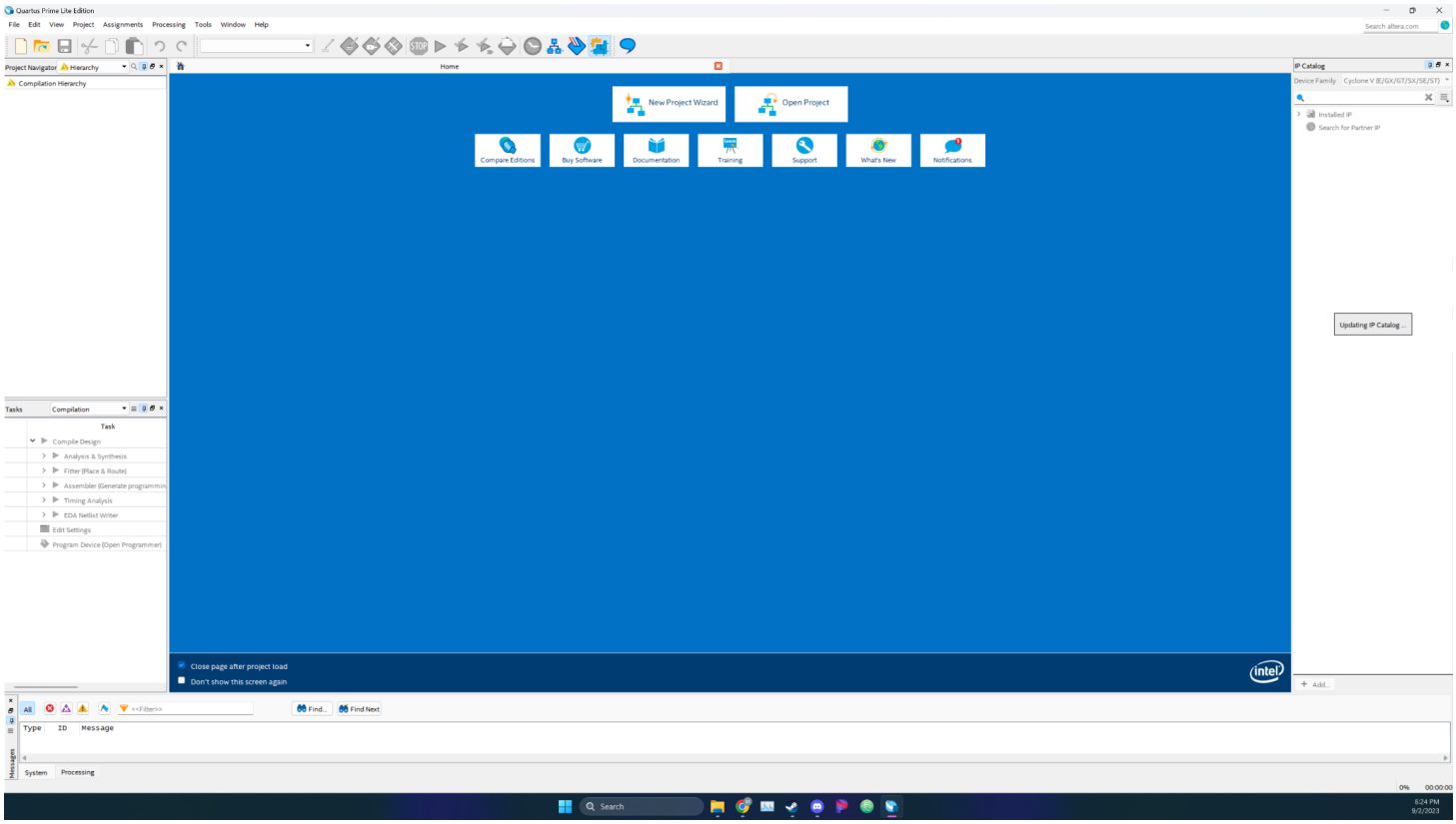
ECEN 5863 – Programmable Logic Embedded System Design Homework set 1

Due Date: 2023/09/09

(These problems should be done individually, not with a project partner)

1. Watch the videos and follow the directions for Quartus Prime Download, Installation, and Introduction. CAUTION: the download url has changed from the one in the video to https://drive.google.com/drive/folders/1k1vsgdW7rgoml3k5Bz07rZuPDcXqoLF0 or https://o365coloradoedu my.sharepoint.com/:f:/g/personal/tisc4629_colorado_edu/EsSrJTHGSjdFutqR3TmkiYwBCWOI4MTtd0rxpbMoykbutg?e=R1yoCW for version 16.1 and https://www.intel.com/content/www/us/en/software kit/665990/intel-quartus-prime-lite-edition-design-software-version-18-1-for-windows.html for version 18.1. You can also get version 16.1 from the 5863Resources folder on the OneDrive. Save and submit a screenshot of your working Quartus Prime project window.



2. Using only logic gates, design a 2-bit full adder with carry. Here is a partial truth table for the circuit.

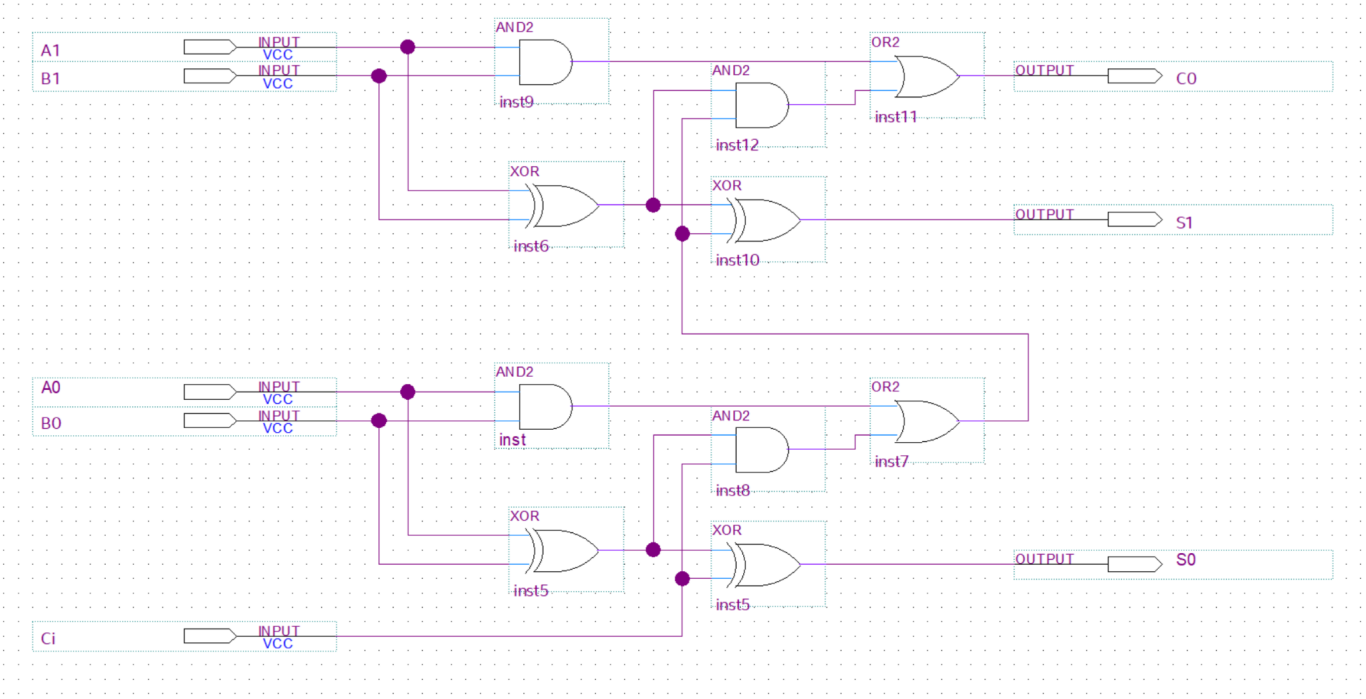
INPUTS					**OUTPUTS**		
A0	**A1**	**B0**	**B1**	**Ci**	**S0**	**S1**	**C0**
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0
1	0	1	0	0	0	1	0
0	1	0	1	0	0	0	1
0	1	0	0	1	1	1	0
1	0	1	0	1	1	1	0
1	1	1	1	1	1	1	1

...

Where A and B are inputs, Ci is carry in, Si is output and Co is carry out.
Draw a schematic showing the gate interconnections. You may find the schematic editor in Quartus useful for drawing the schematic.

The equations for this problem is as follows:
S0 = (A0 XOR B0) XOR (Ci)
S1 = (A1 XOR B1) XOR ((A1 AND B1) OR ((A1 XOR B1) AND (((A0 AND B0) OR ((A0 XOR B0) AND Ci)))))
C0 = (A1 AND B1) OR ((A1 XOR B1) AND (((A0 AND B0) OR ((A0 XOR B0) AND Ci))))

Below is a screenshot of the diagram that represents the equations above.


A1
B1
INPUT
VCC
INPUT
VCC
AND2
inst9
XOR
inst6
AND2
inst12
OR2
inst11
OUTPUT
C0
XOR
inst10
OUTPUT
S1
A0
B0
INPUT
VCC
INPUT
VCC
AND2
inst
XOR
inst5
AND2
inst8
OR2
inst7
XOR
inst5
OUTPUT
S0
Ci
INPUT
VCC

3. In some applications, special functions must be implemented by logic in microprocessor systems to meet requirements. For example, a special multiply circuit may be required if the microprocessor multiply instruction is too slow. Design a combinational circuit that computes $y=x^2$ with x as a 4-bit input. You may use gates, or any higher level function IC's to implement the circuit.

Solution for bitwise multiplication of two 2 bit numbers:

(A1*2^1 + A0*2^0)*(B1*2^1 + B0*2^0)

=4*B1*A1 + 2*A1*B0 + 2*A0*B1 + A0*B0

If A and B are the same (when squaring):

=4*A1*A1 + 2*A1*A0 + 2*A0*A1 + A0*A0

=4*A1 + 4*A0*A1 + A0

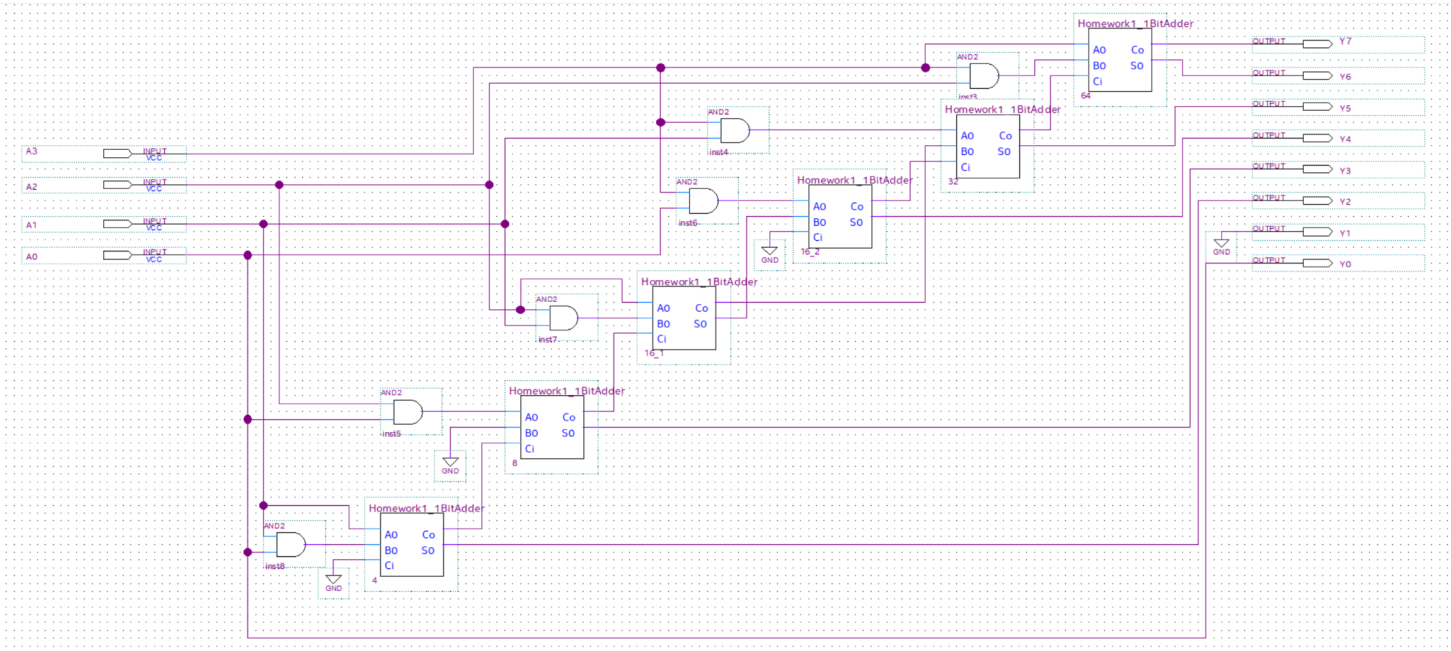
Squaring a 4 bit number:

(8*A3+4*A2+2*A1+A0)^2

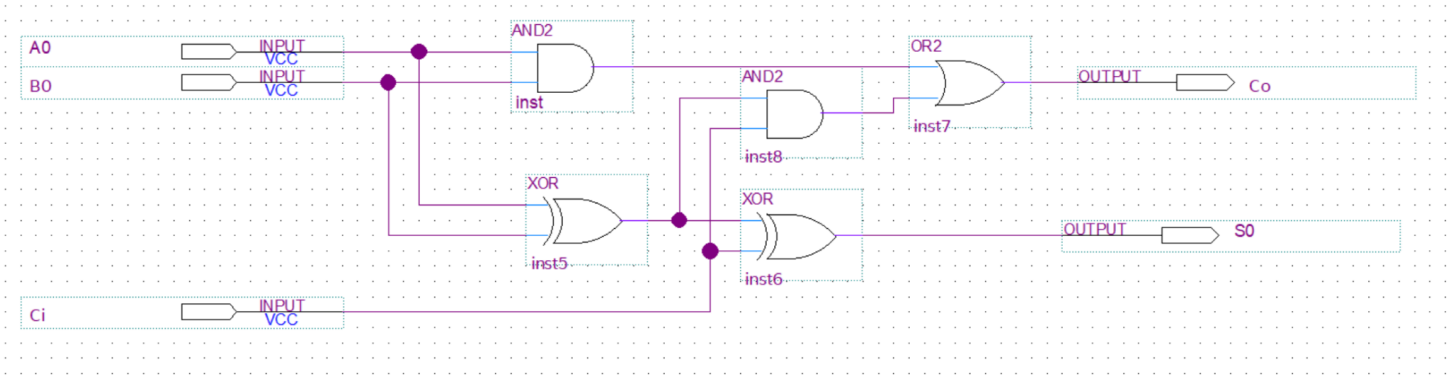
=(64*A3+32*A3*A2+16*A3*A1+8*A3*A0)+(32*A2*A3+16*A2+8*A2*A1+4*A2*A0)+(16*A1*A3+ 8*A1*A2+4*A1+2*A0*A1)+(8*A3*A0+4*A2*A0+2*A1*A0+A0)

=(64*A3+64*A2*A3)+(32*A3*A1)+(16*A3*A0+16*A2+16*A2*A1)+(8*A2*A0)+(4*A1+4*A1*A0)+ A0

The work above is the mathematical basis to my design that optimizes the multiplication since the design is doing x^2, not x*z. This math is reflected in the following design. I created a 1 bit adder symbol that I could reuse, and I used AND gates for any multiplication of two bits.



Here is the schematic for the 1 bit adder:



4. Solve this problem

Show how the logic equation (NOT(A) AND (B)) OR ((C) AND NOT (D)) can be implemented using the following:

A. The PLA in Figure 3.3 (You must change the fuse links)

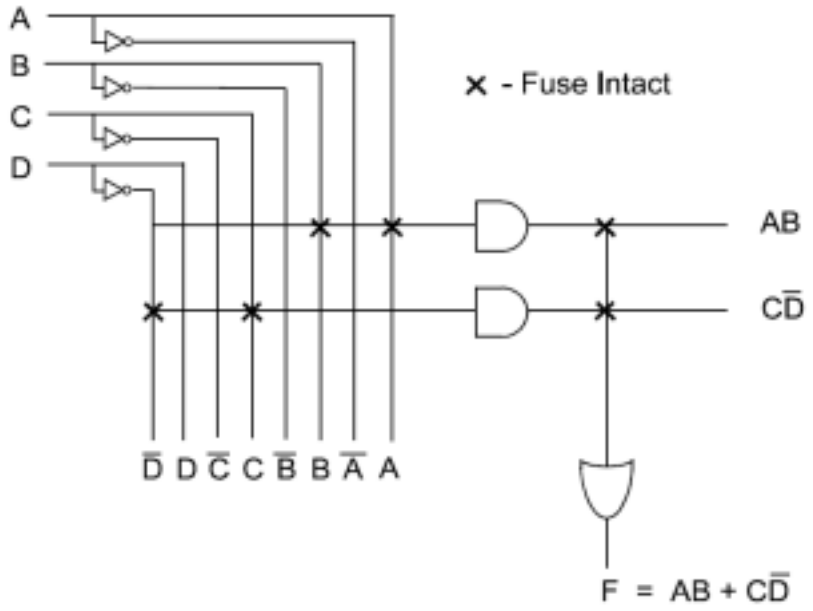
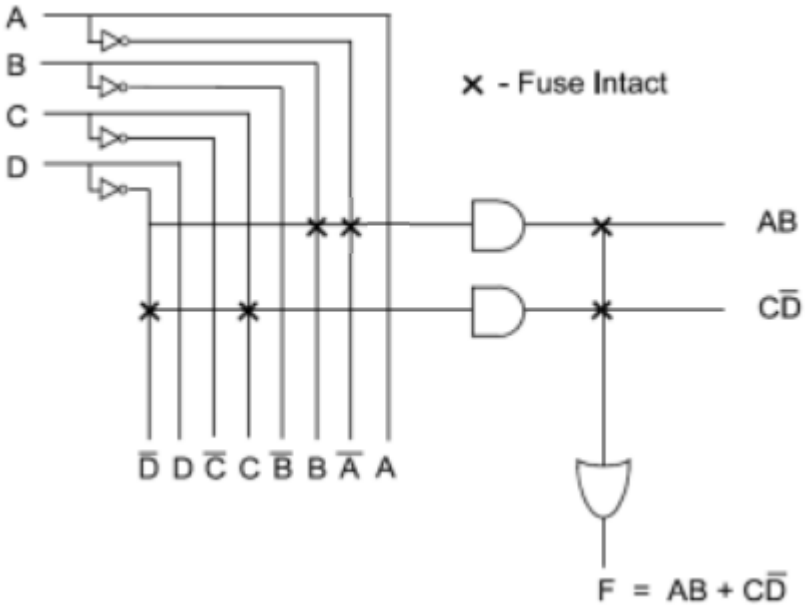


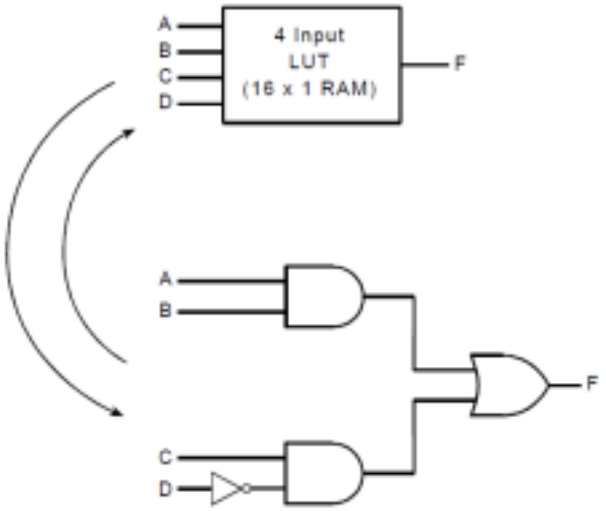
Figure 3.3 Using a PLA to implement a Sum of Products equation.

To get the equation above, you have to set the fuse that crosses the NOT A line and the AB line, as well as the B line and the AB line. Similarly, the C line and the CD line have to be set, as well as the NOT D and CD line. Both the AB and CD lines have to be set with the F line. The diagram below shows the fuse configuration. The only fuse that changes is the fuse from A to NOT A.



Note: the equation for F at the bottom doesn't reflect the fuse configuration.

B. The LUT in Figure 3.8 (You must change the Data contents)



RAM Contents				
Address				Data
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Figure 3.8 Using a look-up table (LUT) to model a gate network.

Be sure to include the PLA fuse pattern and the contents of the LUT

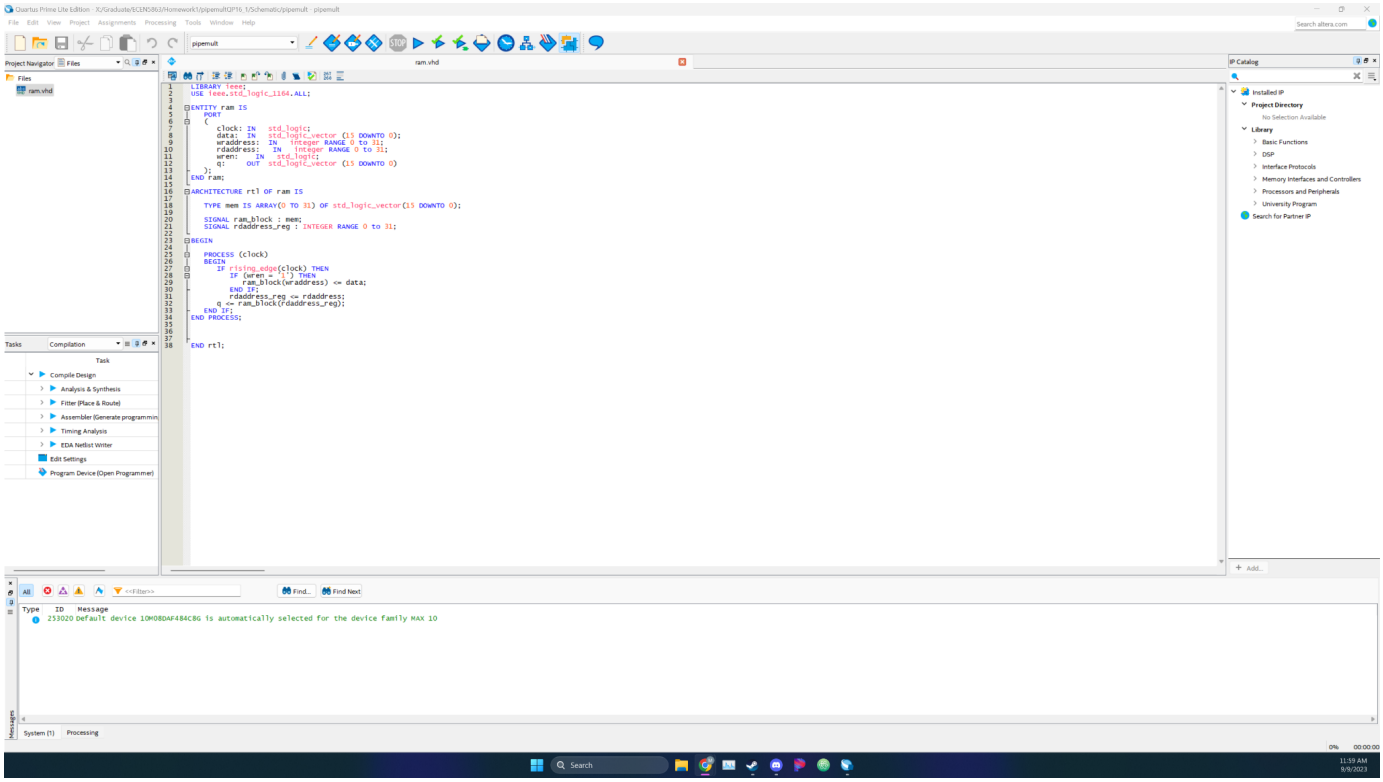
The LUT contents to achieve that equation are as follows:

RAM Contents				
Address				Data
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

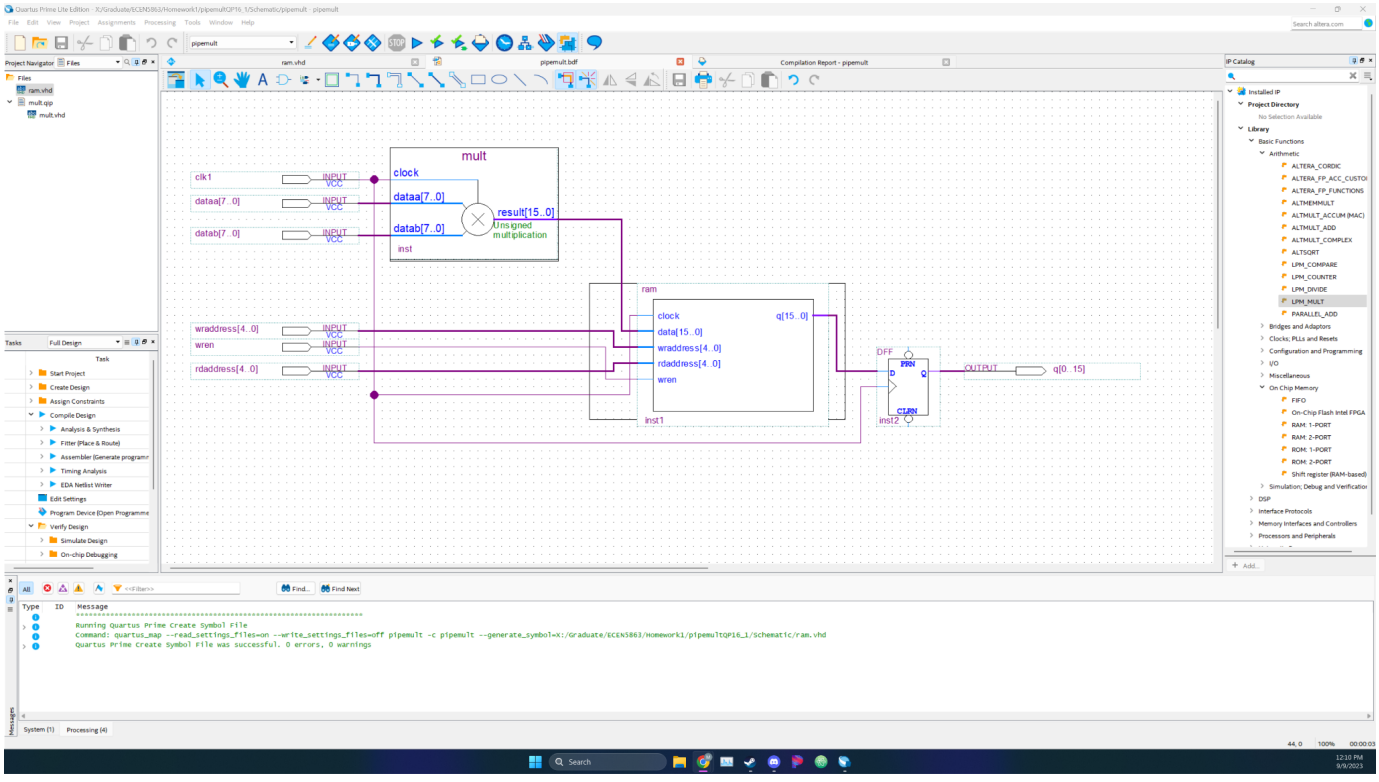
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

5. Watch the videos and follow the directions for Quartus Prime Project Creation, Design Entry, Compilation, and Timing Analysis. Save and submit screenshots of your working Quartus Prime project at the end of each video.

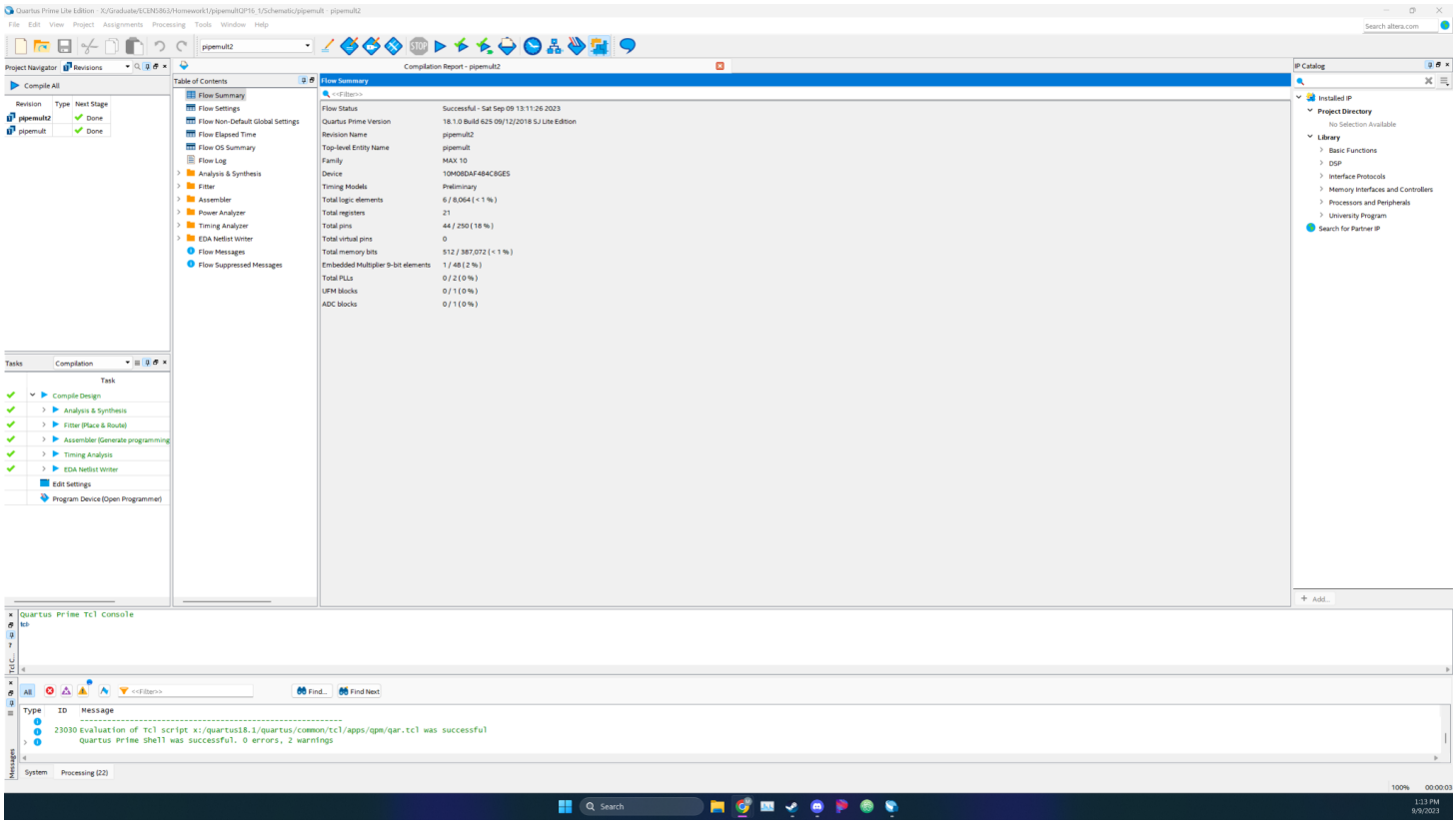
Video 1:



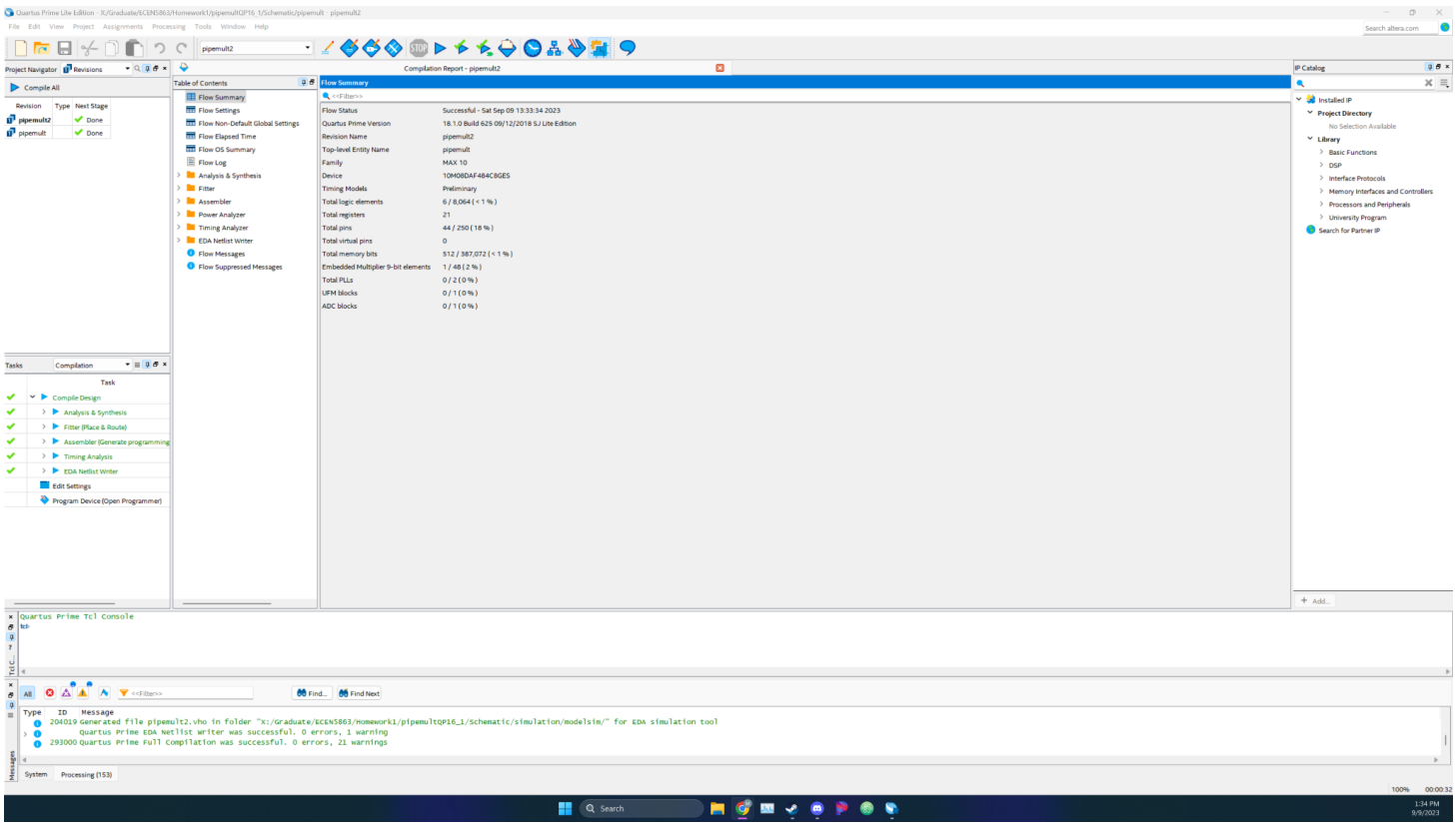
Video 2:



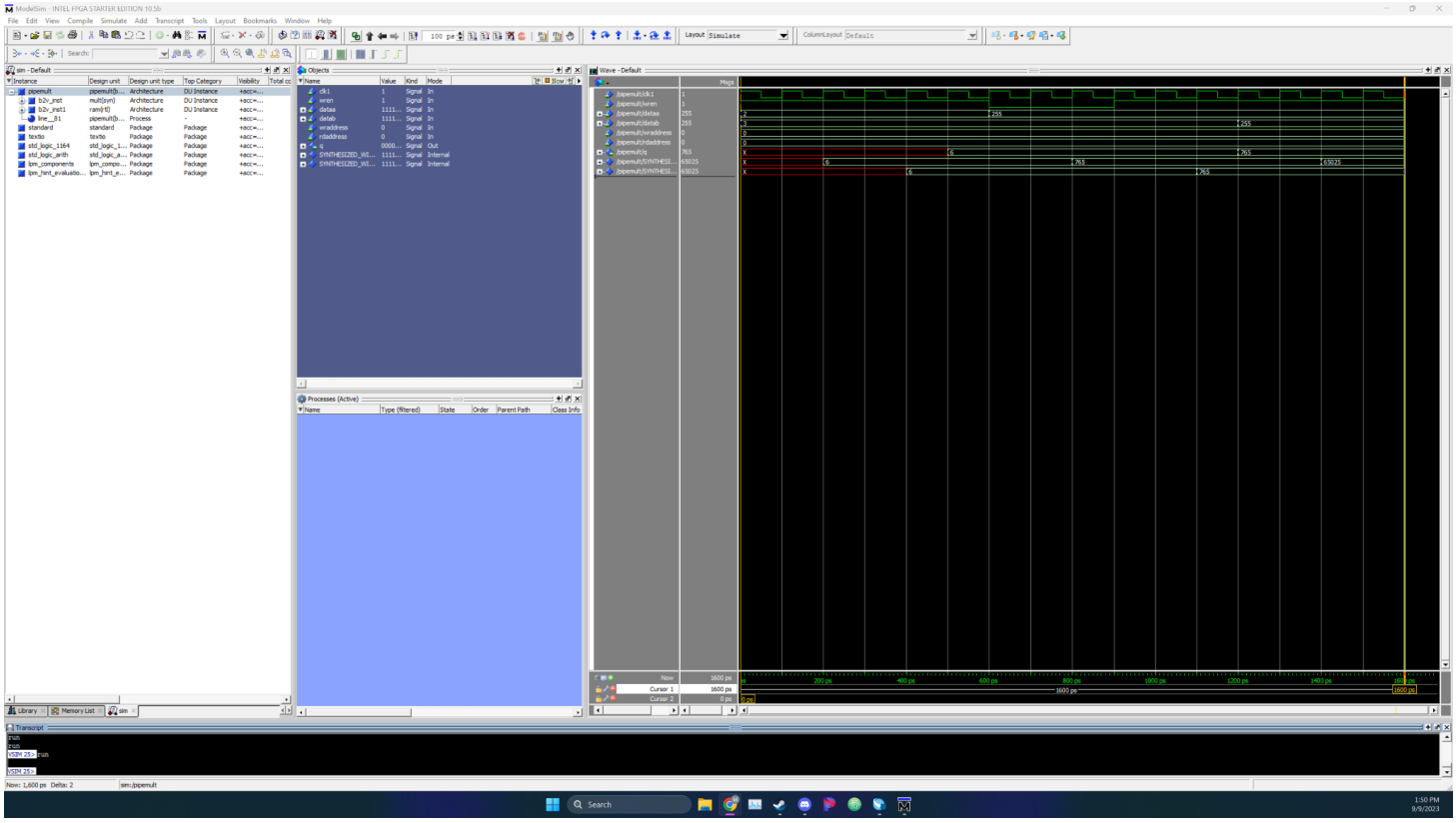
Video 3:



Video 4:

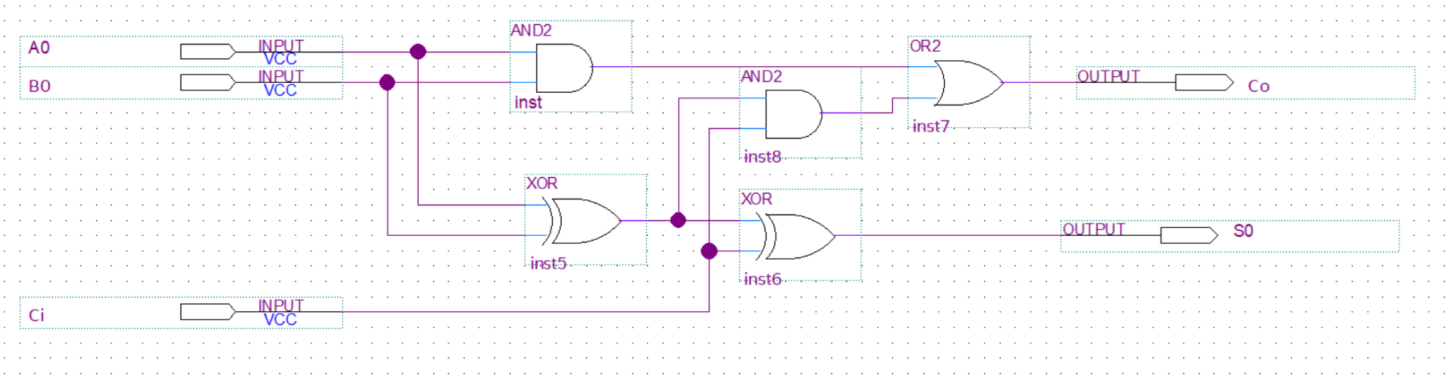


Video 5:

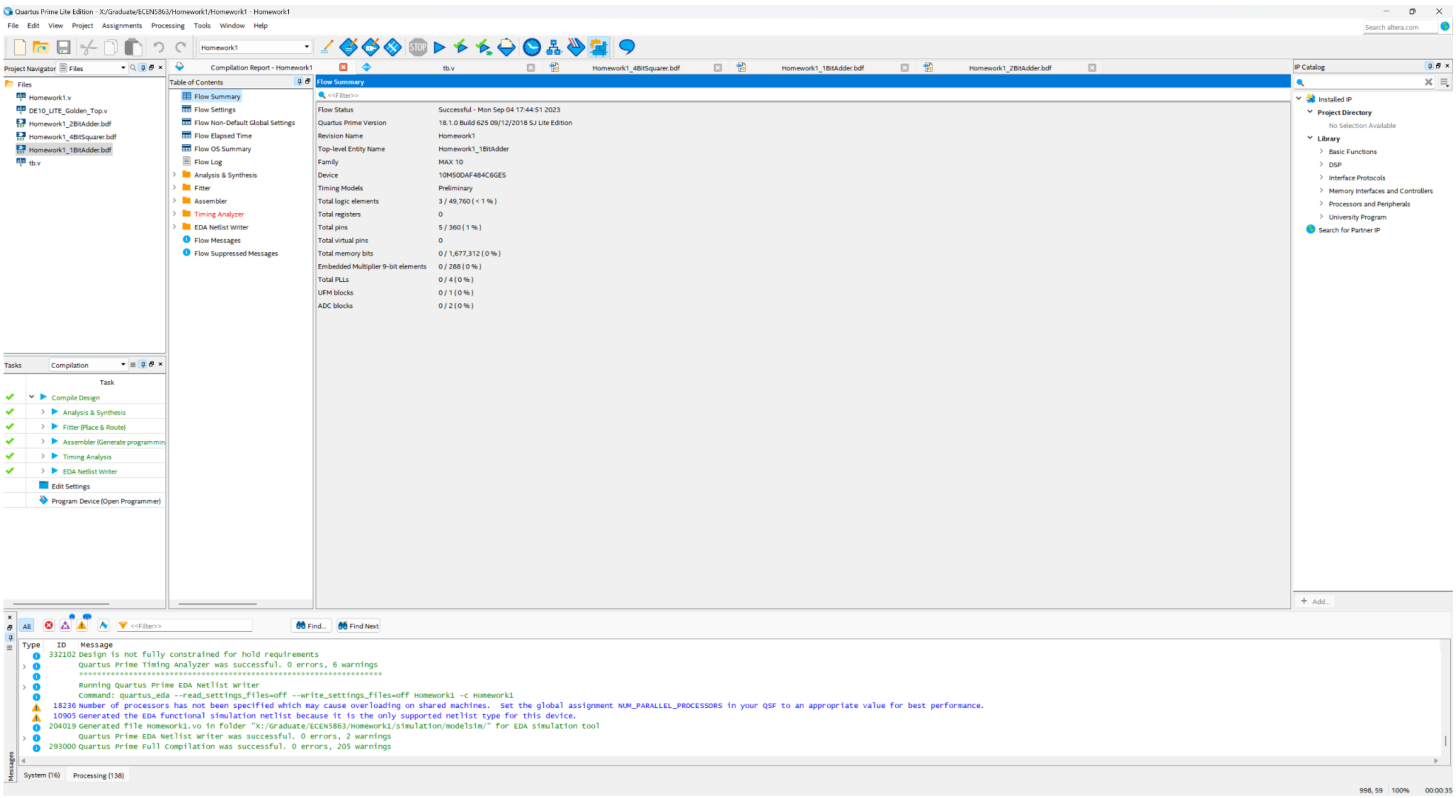


6. Design a 1-bit full adder circuit using schematic design tool in Quartus. Compile the design and submit the screenshot of both the schematic design and compilation report. (For guidance use the Quartus Prime tutorial video.)

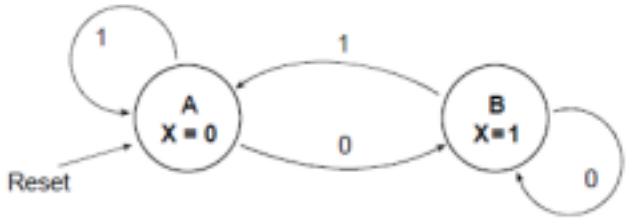
This is a screenshot of the 1-bit full adder:



I made this my top level design file and compiled. Here is a screenshot of Quartus showing the complete compilation of my 1 bit adder:

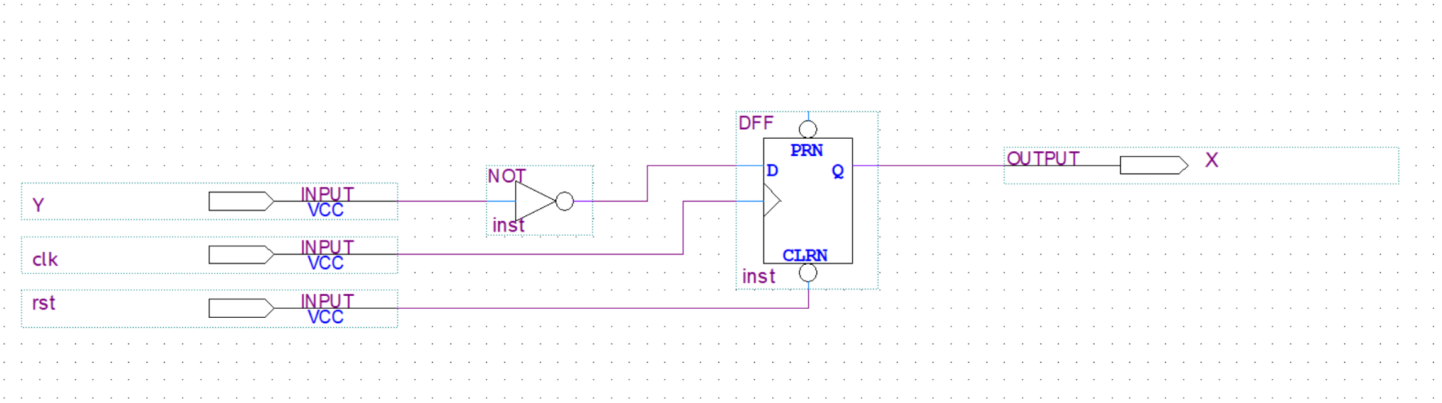


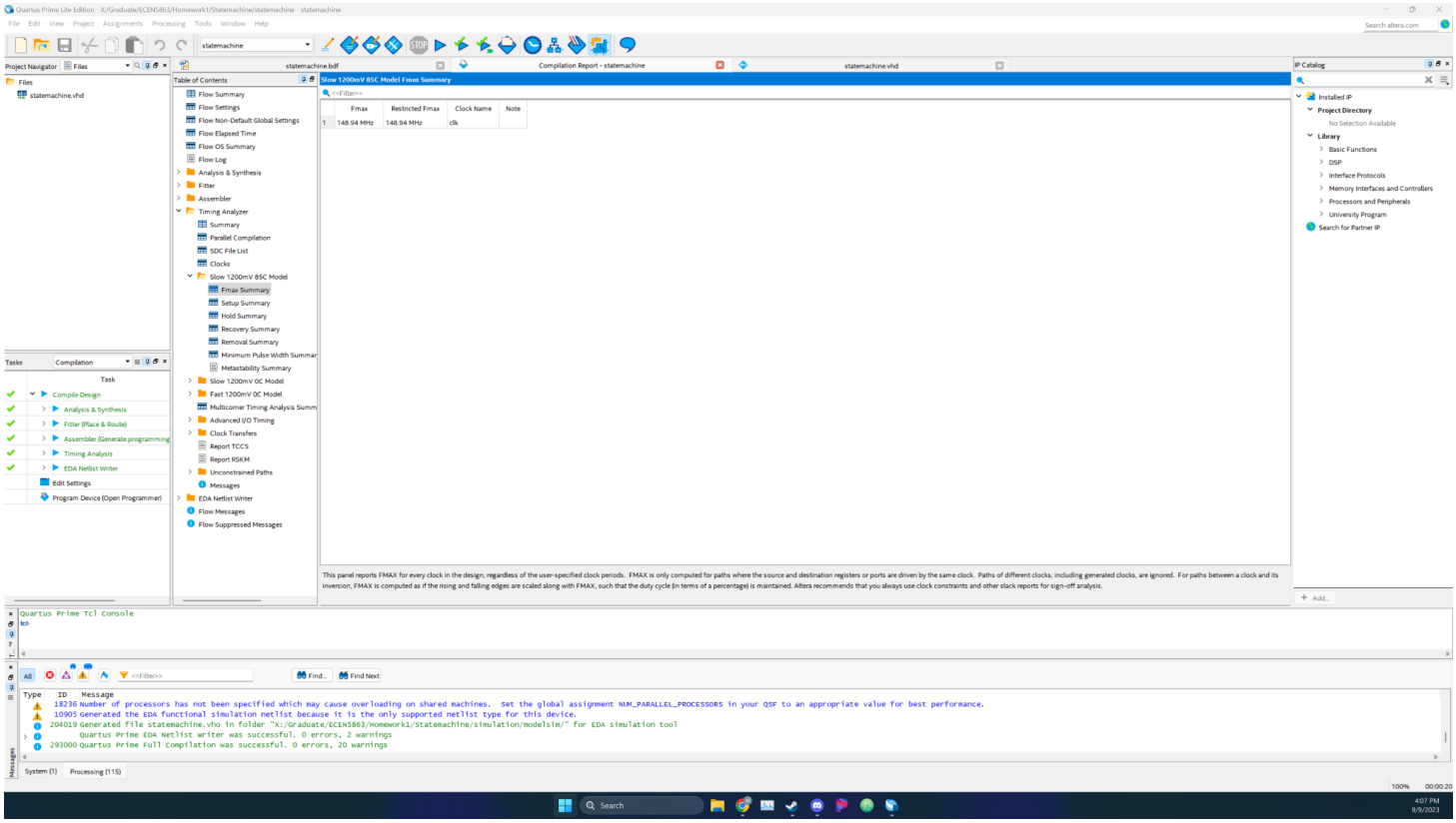
17. Use gates and a DFF part from the primitives storage library with graphical entry to implement the state machine shown in the following state diagram. Verify correct operation with a simulation using the Altera CAD tools. The simulation should exercise all arcs in the state diagram. A and B are the two states, X is the output and Y is the input. Use the timing analyzer's **Processing ⇨ Classic Timing Analyzer Tool ⇨ Registered performance** option tab to determine the maximum clock frequency on the Cyclone device. Reset is asynchronous and the DFF Q output should be high for state B.



7. Use gates and a DFF part from the primitives storage library with graphical schematic entry to implement the state machine shown in the following state diagram. Verify correct operation with a simulation using Altera CAD tools or Modelsim. The simulation should exercise all arcs in the state diagram. (Note that some simulators will not accept .bdf files as an input, so some conversion may need to be made). A and B are the two states, X is the output and Y is the input. Use the TimeQuest Timing Analyzer Tool => Slow 1200mV 85C Model => Fmax Summary option tab in the Compilation Report to determine the maximum clock frequency on the MAX10 device. Reset is asynchronous and the DFF Q output should be high for state B.

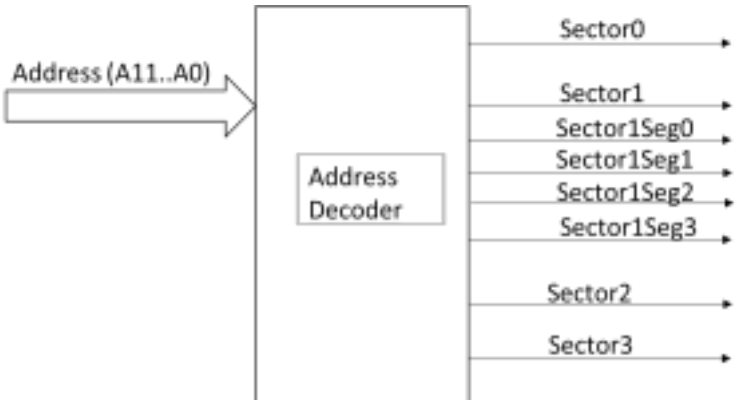
This state machine is relatively simple and can be accomplished with a single DFF. The only thing about this state machine that requires combinational logic is the input must be inverted so that a Y input of 1 sets the state (and the output X) to zero. Shown below is the schematic of the state machine as well as the TimeQuest output.





As shown above, the Fmax for this design is 148.94Mhz.

8. You have been assigned to create an address decoder like the one below for a system with 4096 locations. The inputs to the decoder are address lines, the outputs are 4 evenly divided sectors, except that 2nd sector starting at 0x400 has 4 segments of 16 locations each for peripherals; therefore 8 outputs in all.



Considering the architecture, speed, and the cost, from the list below, what device family would you use? **Circle** one, and justify your answer with facts and reasoning:

a. Microsemi IGLOO nano

b. Xilinx XC9500XL

c. Altera MAX10

d. Lattice MACHXO2

e. Altera Cyclone V

For a design like this, not much logic is required since this is functionally a LUT with 12 inputs and 8 outputs. This could be achieved using 3 4 input LUTs and a MUX to select the output. Considering the cost, the Lattice MACHXO2 is the cheapest. Looking at the MACHXO2 architecture, each slice contains 2 LUT4s and 2 registers. This means that this design would require at least 2 slices. The cheapest Lattice MACHXO2 is the LCMXO2-256HC-4SG32C, which has 256 LUTs. This means that it has enough logic to complete the design. Given the problem statement, it is difficult to determine the speed requirements for this design. If this design requires higher speeds, then it may require a different part, but the MACHXO2 will be able to complete this design, although it isn't feasible to determine the speed at which it can run. Given this lack of information, I would pick the MACHXO2 since it's architecture easily fits the requirements, it's the lowest cost option, and the design speed requirements are unknown.

Instructions for submission

1) All answers are to be submitted in a single word or pdf file.
2) Include Quartus and Modelsim screenshots whenever possible with appropriate annotations for a better presentation in a word or pdf file.
3) Submit individual zipped folders for each project question if there is a set of code for submission.

Grading Rubric

1) - [5 points] Screen shots for completion
2) - [5 points]
    [2 points] Equation
    [3 points] Schematic
3) - [5 points]
    [2 points] Design explanation
    [3 points] Solution
4) - [5 points]
    [2 points] PLA Schematic
    [3 points] LUT Table
5) – [5 points] 5 Final Screenshots
6) - [5 points]
    [3 points] Screen shots for completion of each step on Quartus Prime
    [2 points] Design steps and screen shot of schematic
7) – [5 points]
    [2 points] Equations or Tables
    [2 points] Schematic
    [1 points] Time Quest Analysis
8) – [5 points] Selection of best PLD alternative

** Commenting of code and presentation of material also carries weight*