EE 457 HW 1
Digital Design Review
Redekopp • Puvvada

Name: _____

Due: See Website **Score: _____**

Please post any questions regarding HW problems on Piazza.

1. (25 pts.) Mealy machine Design

Design a simple (though inefficient) DIVIDER to divide X by Y to obtain quotient Q and remainder R. All are 4-bit unsigned numbers. Y is a non-zero number.

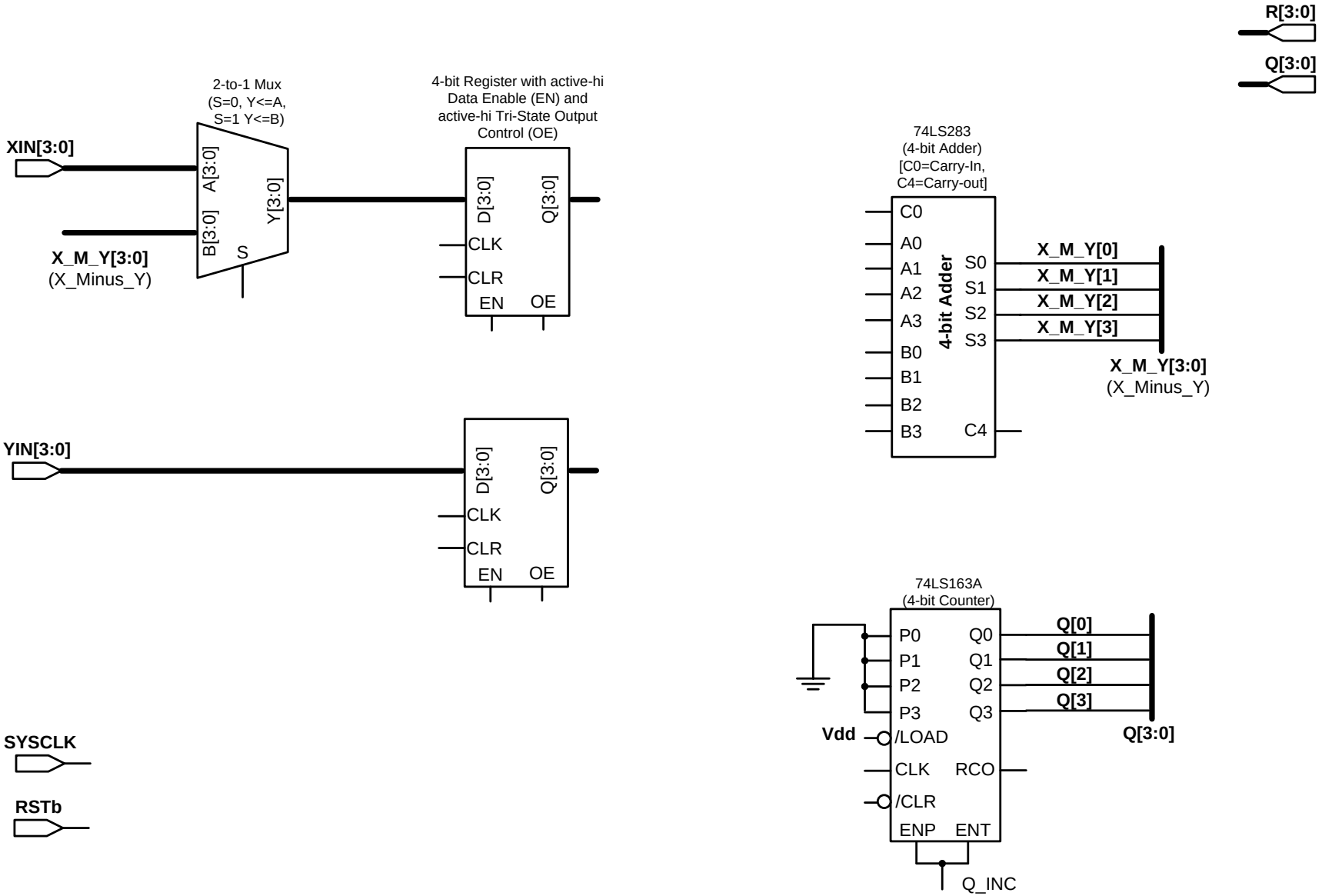
Method: Subtract Y from X repetitively until Y does not go through X anymore. There shall be an INITIAL state I, COMPUTE state C, and a DONE state D. Remain in the I state until the START command S is received. In the initial state, clear the Q register. In the compute state, perform the subtraction (X - Y), and based on borrow output from the subtractor, which indicates whether Y went through X or not, update X with (X - Y) and increment Q. If Y did not go through X, you should exit the compute state (without updating X or incrementing Q) and go to the Done state. Remain in the done state while the END command is 0. Once END is 1 then go back to the initial state. While in Done state, the register X shall contain the remainder R and the register Q shall contain the quotient.

Use two 4-bit registers described in the datapath below for X and Y. Use 74LS163A counter for Q. Use one-hot state assignment and three D-FFs for your state memory. If you do not know one-hot method for designing a control unit please get help from your TA, the grader, or the instructor. State machine design using one-hot state assignment will be covered in the first week of classes in EE457.

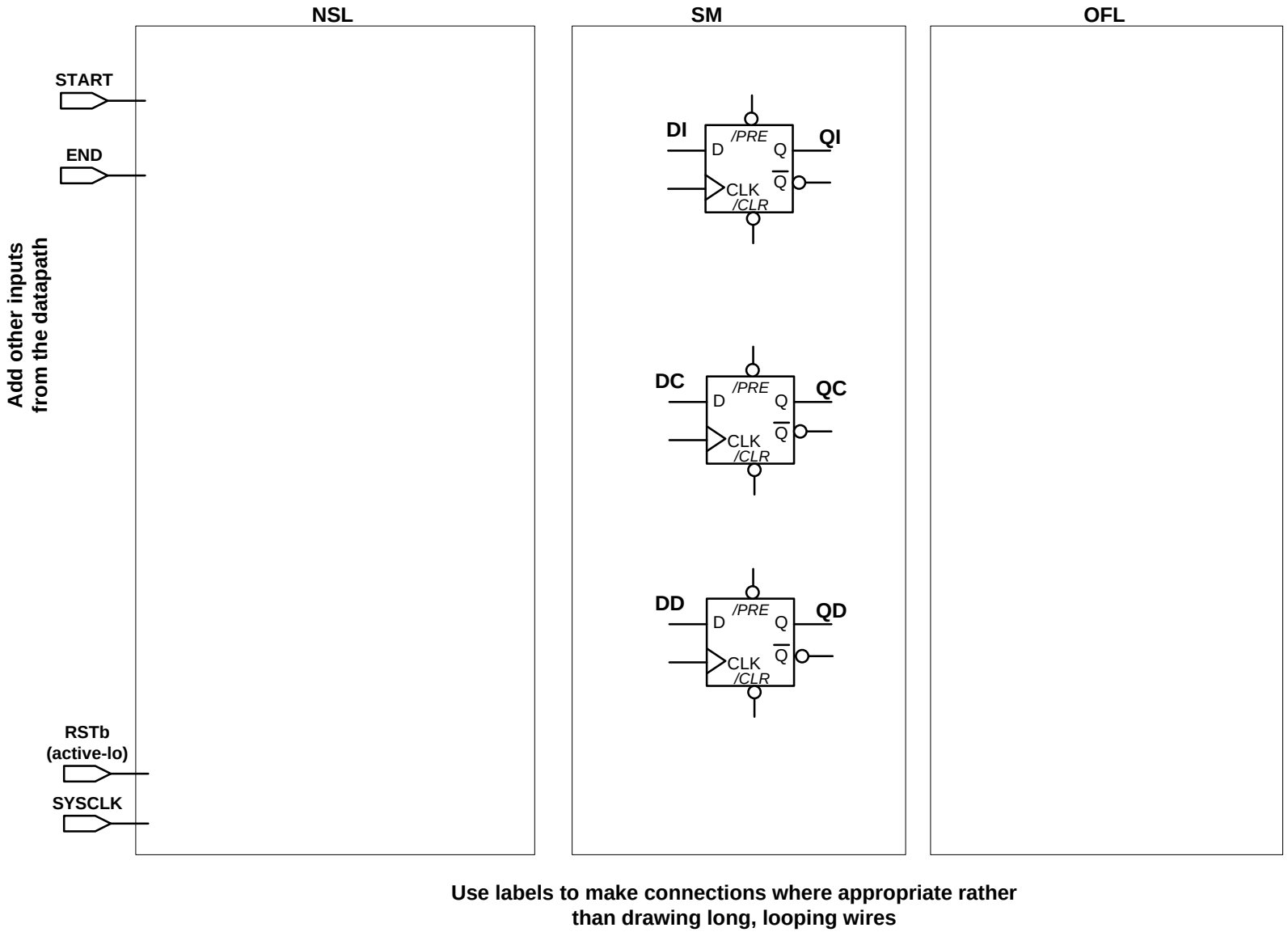
Incomplete designs of the DPU (Datapath unit) and the CU (Control unit) are provided on the next two pages. Complete the designs. All inputs to logic blocks must be in terms of the primary inputs: XIN[3:0], YIN[3:0], START, END, RSTb and the SYSCLK or other intermediate signals that you generate. You may tie unused inputs to appropriate constants GND or VDD if desired. You should define two intermediate signals: X_LOAD, Y_LOAD (data enables for the respective registers), Q_INC (count enable), Q_CLR (counter reset), and BORROW.

For sample values of X = 1101 (thirteen) and Y = 0011 (three), draw typical waveforms for this divider showing the clock, the inputs S and E, the control signals from the controller to the datapath unit such as X_LOAD, Q_INC (increment counter), the values of X and Q over time, the state Flip-flop outputs, QI, QC, and QD over time, etc.

Q1 Datapath:

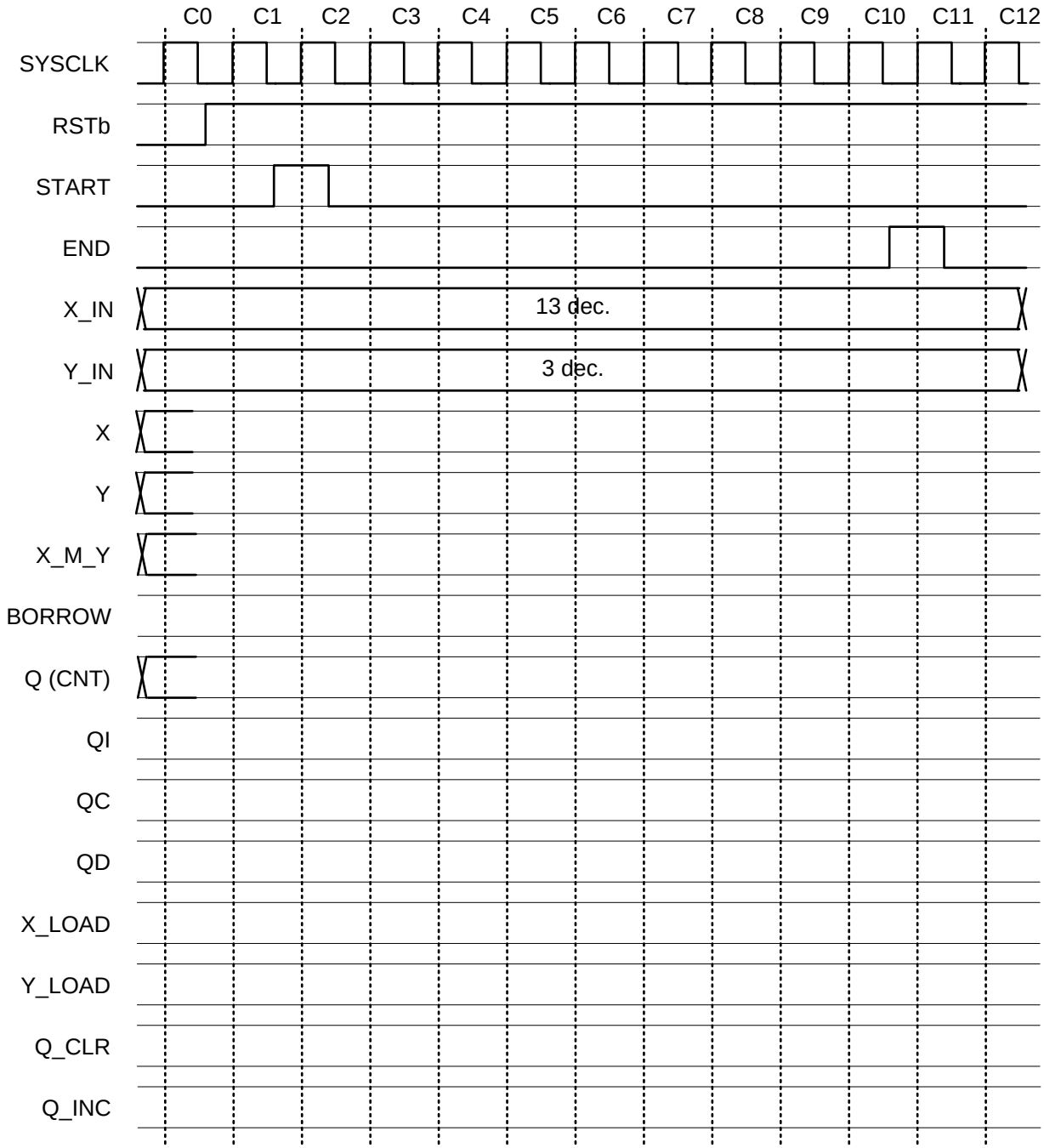


Q1 Control Unit:



Use labels to make connections where appropriate rather than drawing long, looping wires

Q1 Sample Waveform:

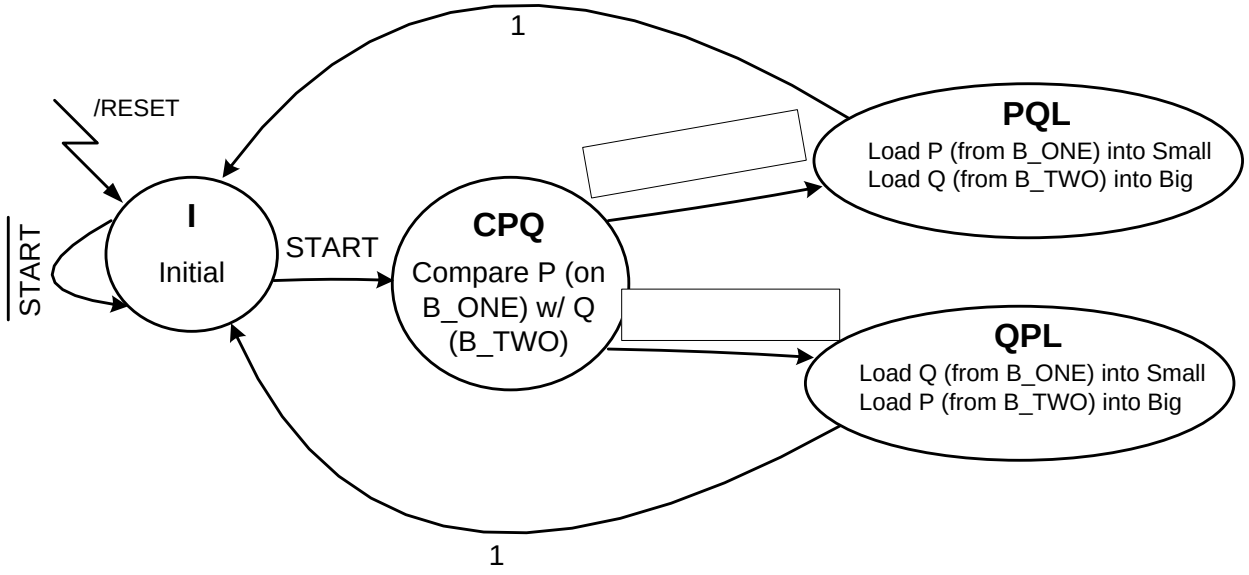


2. (25 Pts.) Datapath and Control Design

You are given two 4-bit unsigned numbers, **P** and **Q**. You need to compare them and deposit the *smaller* in **SMALL_REG** and the *bigger* **BIG_REG**. The next page contains a complete data path. Notice that you can bring either P or Q on bus #1 (B_ONE) or bus #2 (B_TWO). SMALL_REG is only tied (i.e. connected) to B_ONE where as BIG_REG is only tied to B_TWO.

2.1. 4-state State Machine

2.1.1. Complete the state diagram below by writing the state transition conditions.



2.1.2. Complete the one-hot implementation of the above 4-state state machine on page 7. Before you produce the outputs, answer the following questions.

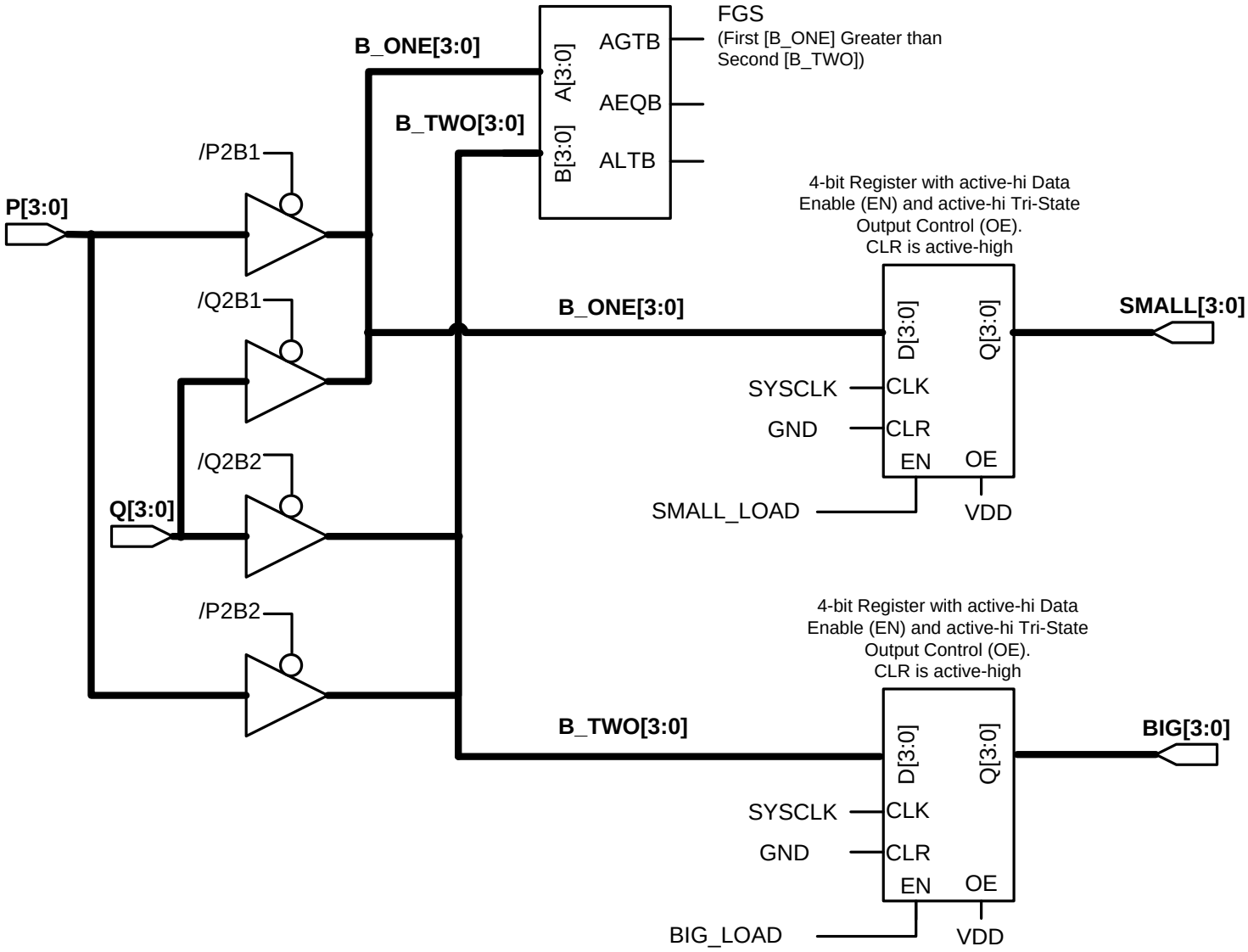
2.1.2.1. **[Yes / No]** (Circle one) Can we say that whenever we put P or Q on one of the two buses, we ***may*** put the other on the other bus?

2.1.2.2. **[Yes / No]** (Circle one) Can we say that, in the initial state, we ***may*** drive the buses even though it is not necessary?

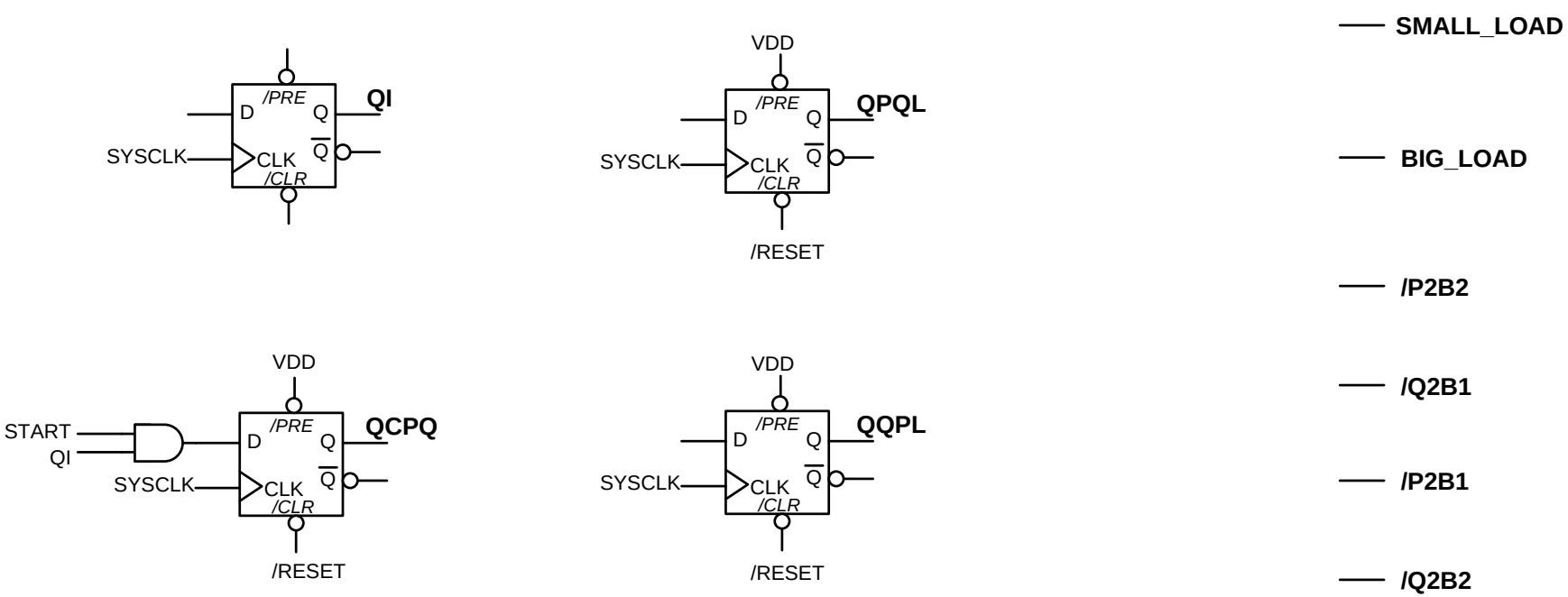
2.1.2.3. **[Yes / No]** (Circle one) Can we say that we either load ***both*** SMALL_REG and BIG_REG or load ***neither***?

2.1.3. Complete the waveform on page 8.

Q2 Datapath. Note: This datapath is complete and is for your reference.

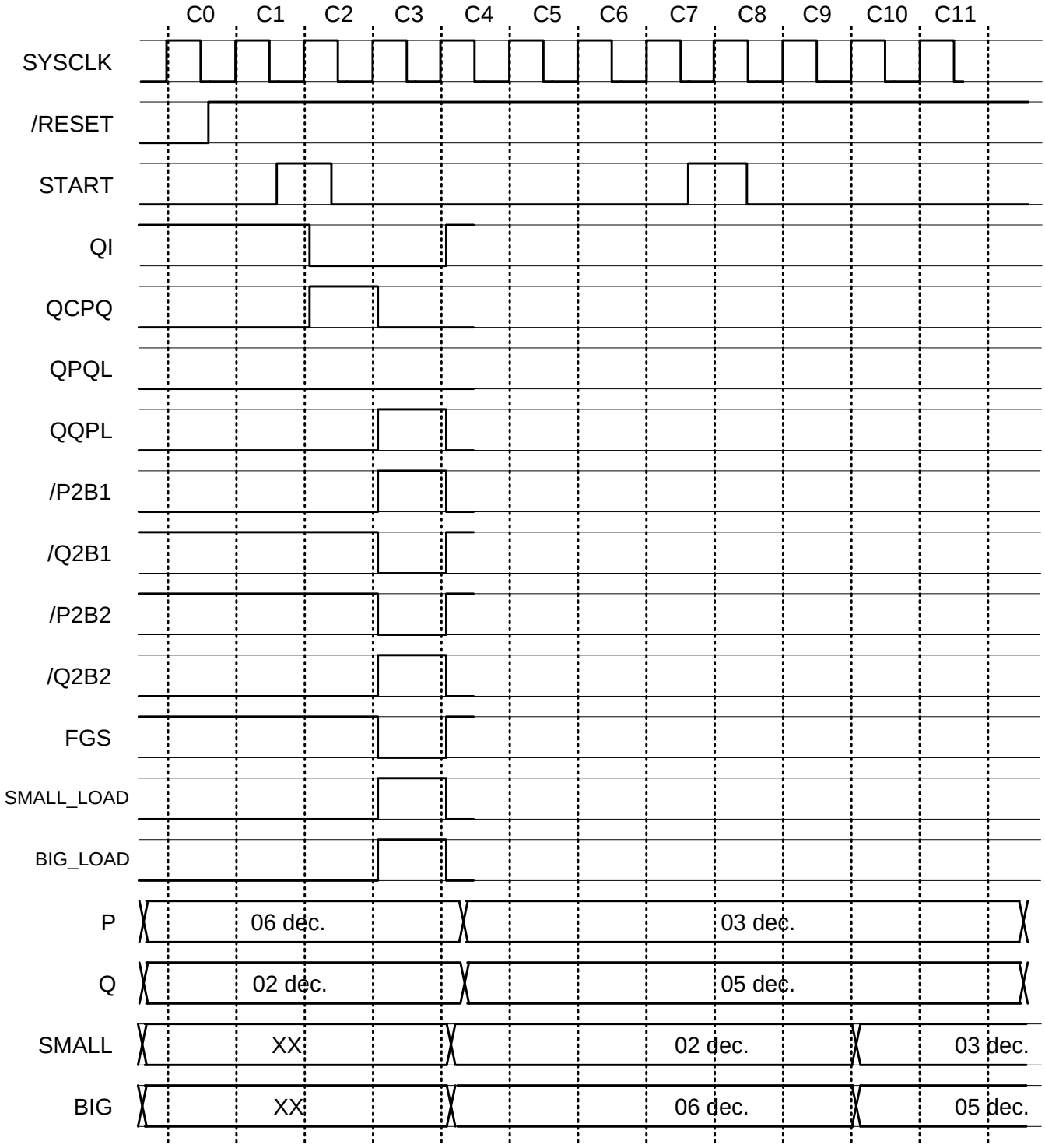


Q2.1 Control Unit. Complete this.



Use labels to make connections where appropriate rather than drawing long, looping wires

Q2.1 Waveform. Complete this based on your design.



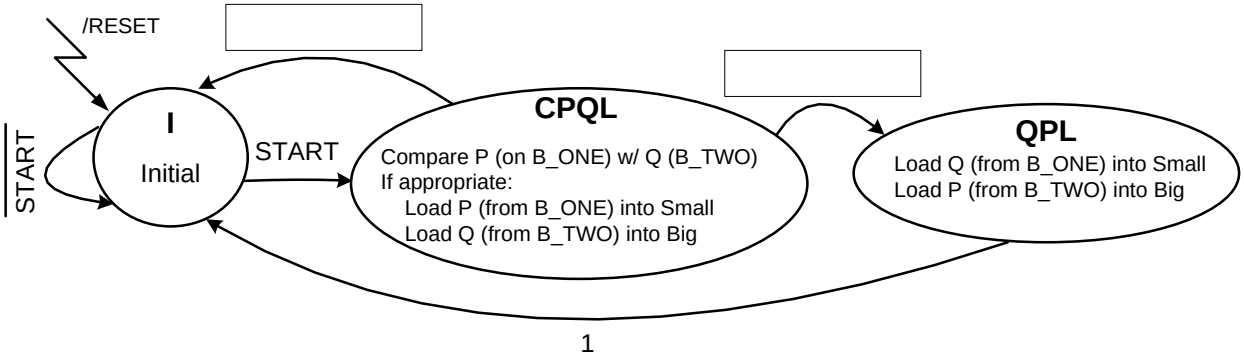
2.2. 3-state State Machine

2.2.1. The previous 4-state state machine is a _____ (**Mealy / Moore)** as the outputs generated are not influenced by the current inputs. The outputs are completely determined by the current state.

Let us now reduce the states by combining CPQ and PQL into CPQL "compare and load". The load operation is conditional in the CPQL state as can be seen below.

2.2.2. This 3-state machine is a _____ (**Mealy / Moore).**

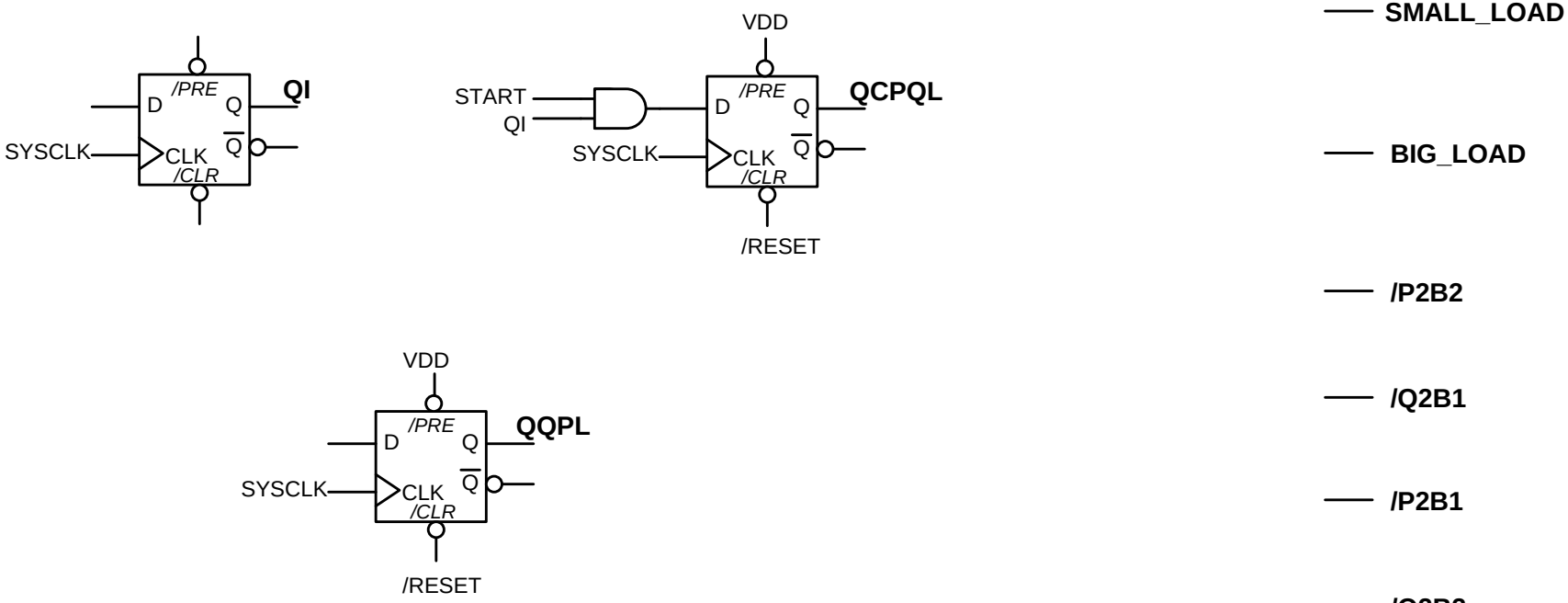
Complete the state diagram below by correctly labelling the transitions.



2.2.3. Complete the one-hot implementation of the above 3-state state machine on page 10.

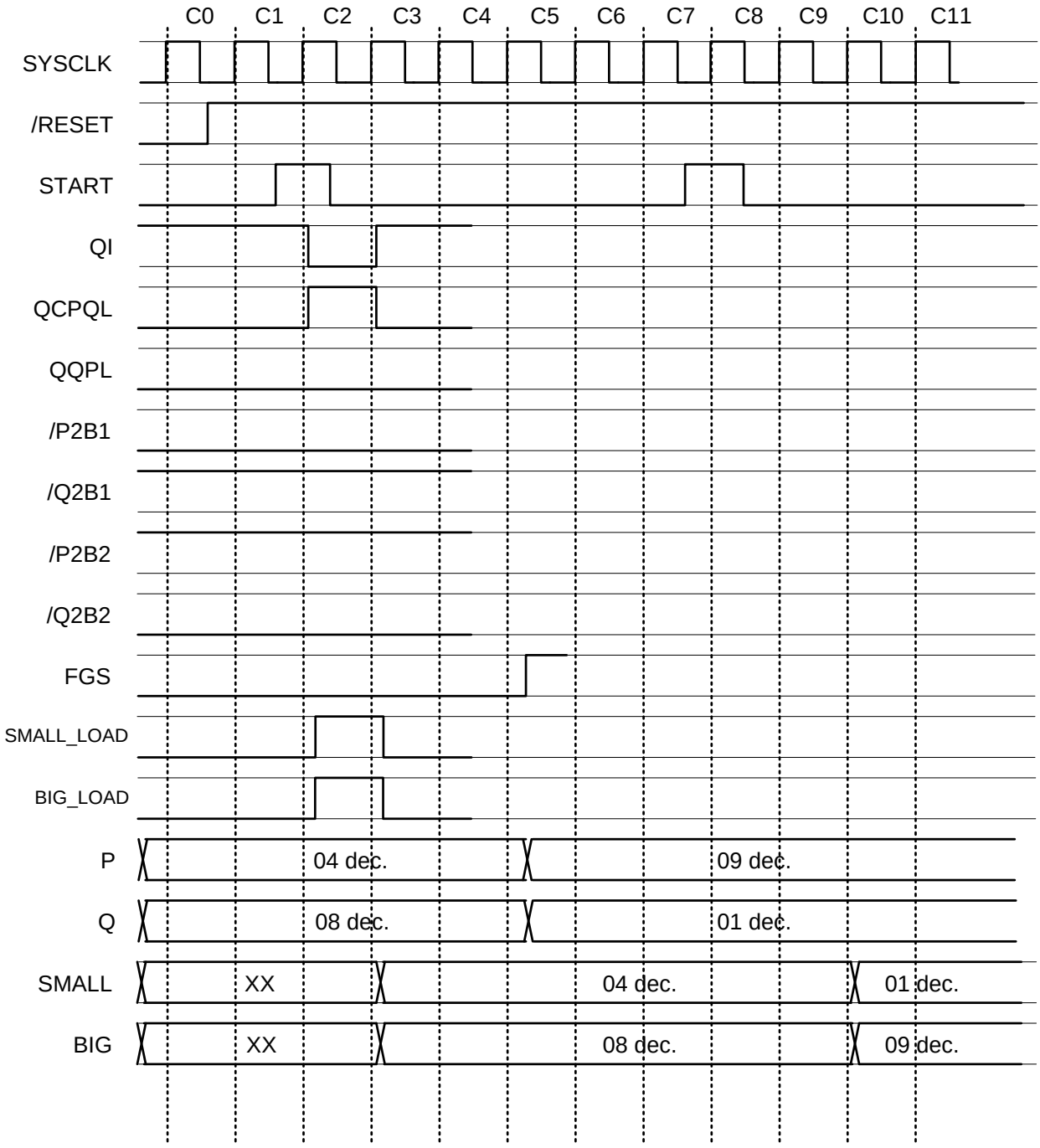
2.2.4. Complete the waveform on page 11.

Q2.2 Control Unit. Complete this.



Use labels to make connections where appropriate rather than drawing long, looping wires

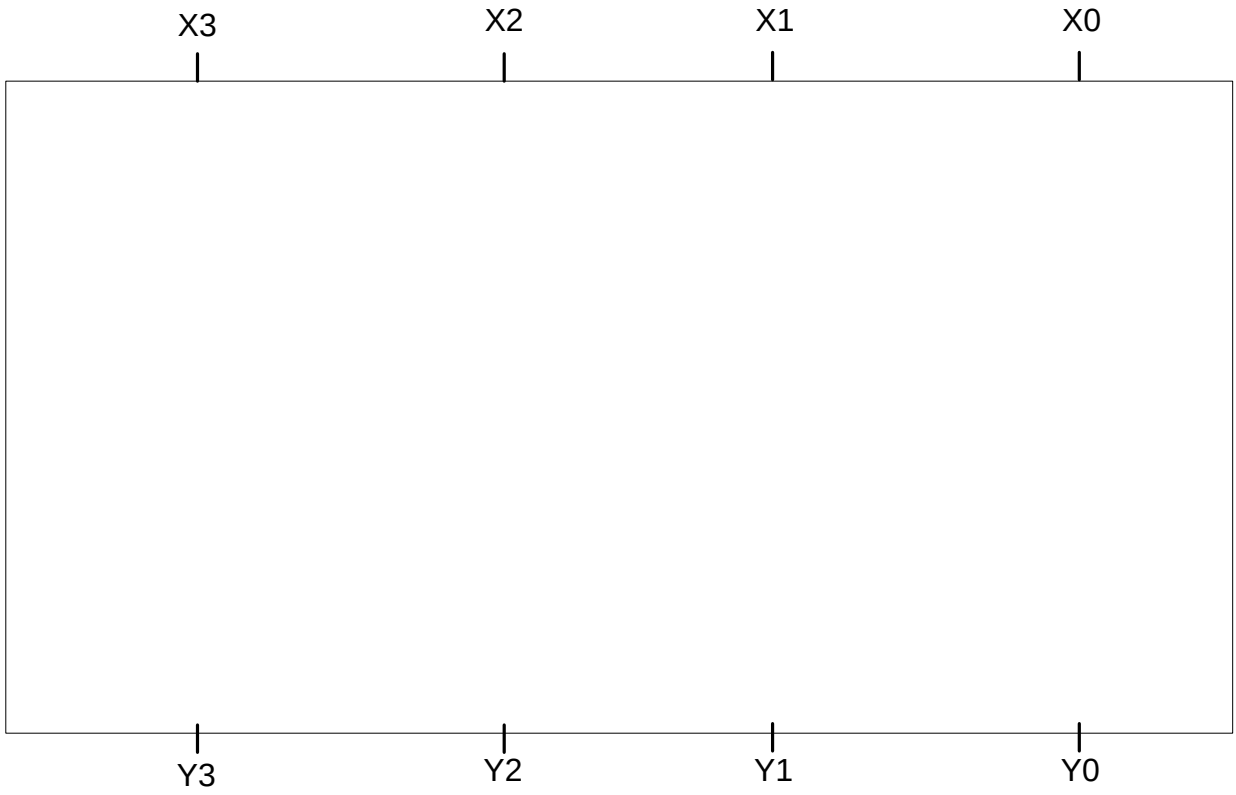
Q2.2 Waveform. Complete this.



3. (13 pts.) Combinational Logic Design

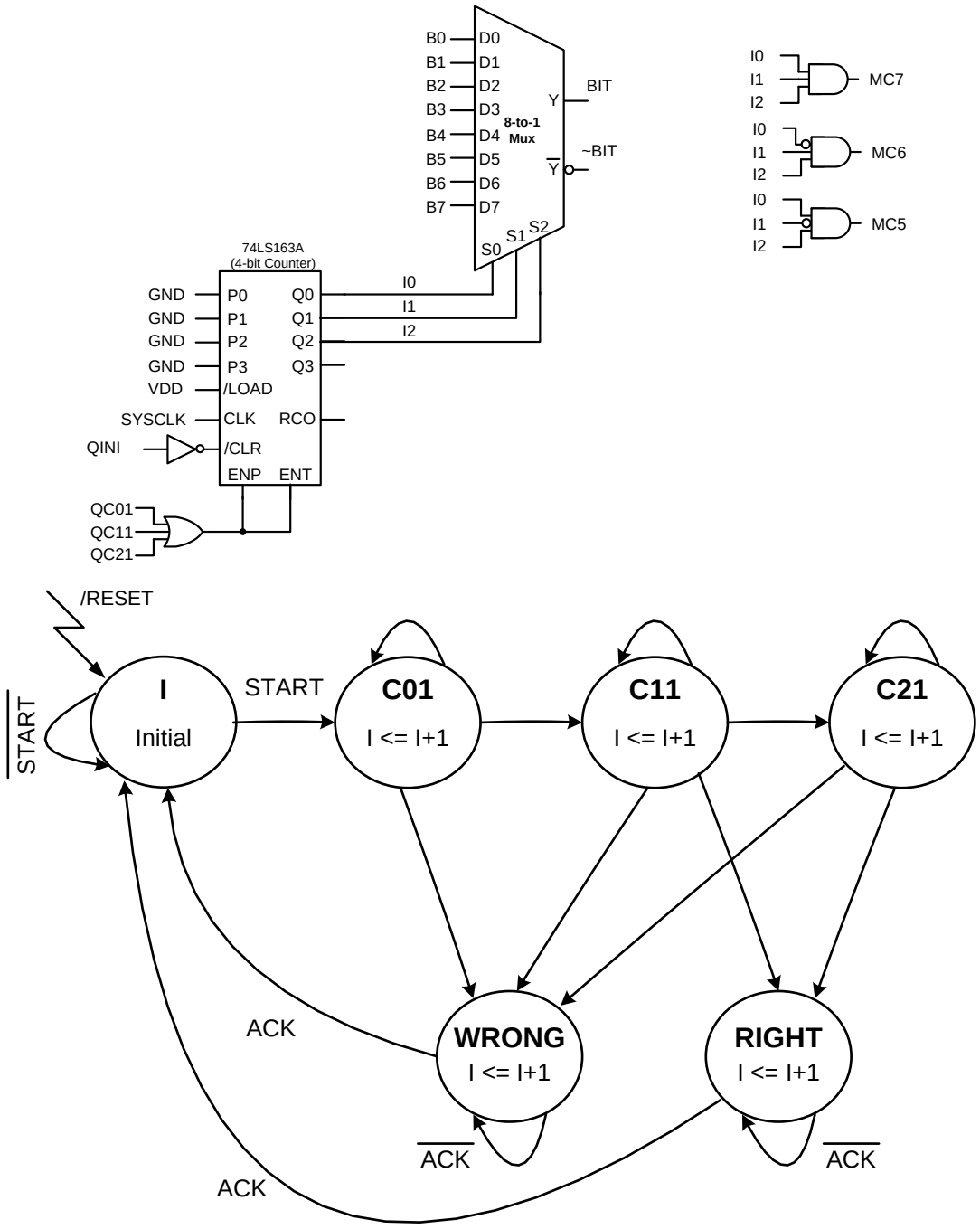
We need an adder to perform Y = (X + 11 dec.) MOD 16. X and Y are 4-bit unsigned numbers represented as X[3:0] and Y[3:0]. Note: 11 dec. = 1011 bin. Build this special adder from SSI gates (Small Scale Integrated circuits such as INVERTER, AND, OR, XOR, NAND, NOR, and XNOR). You may refer to full adder and half-adder designs in your book if you wish. However you should use ***as few gates as possible***.

The constant to be added is always 11 dec. and not a variable! This should make optimization possible. Note: MOD 16 means modulo 16 so if X=12 dec. then 12 + 11 = 23; 23 MOD 16 = 7. Hence Y = 7 dec. Because of the MOD 16 operation you need not produce a carry out (a.k.a. Y4) output.



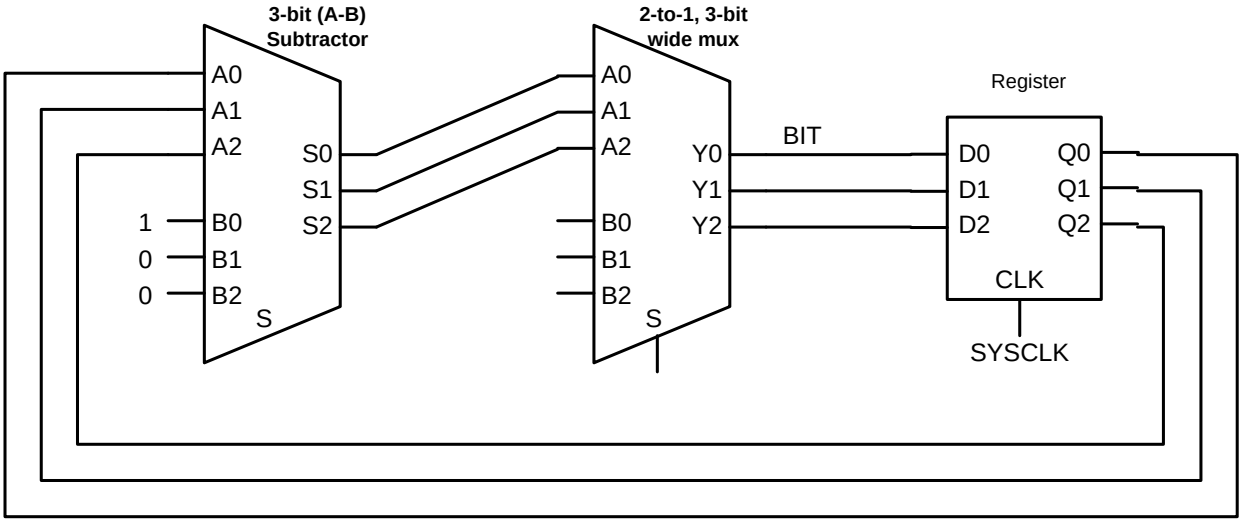
4. (12 pts.) State Diagram Design

In previous problems we have seen "one-hot" coded data where only 1-bit can be a '1'. Here the data is ***"two-hot" coded***. There shall be **TWO** ones and **SIX** zeros in any order. Example: "01001000" is correctly coded. Design a state machine to verify whether an 8-bit value: B[7:0] is correctly coded using "two-hot" coding. In the state diagram below, the states C01, C11, C21 stand for "Counted 0 '1s'", "Counted 1 '1s'", and "Counted 2 '1s'", respectively. Complete the state transition conditions. Use BIT or ~BIT together with signals such as MC7, MC6, and MC5 as necessary.



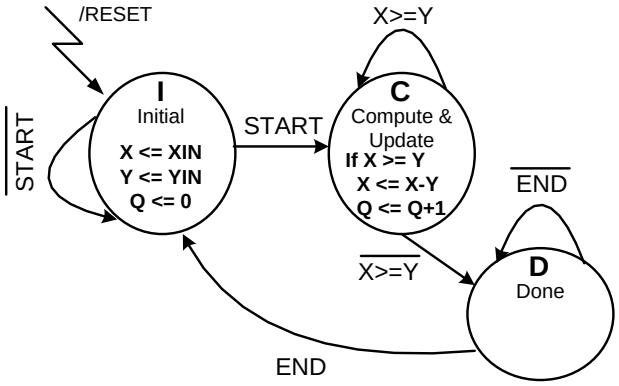
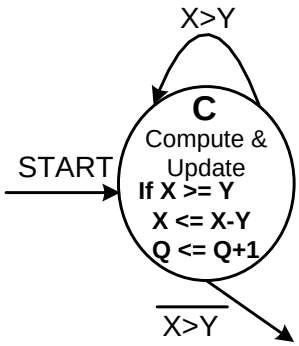
5. (5 pts.) Datapath Design

Design a special down counter which counts down (7,6,5,…). However it shall always skip 4. So the sequence shall be 7,6,5,3,2,1,0,7,6,5,3,2,1,0,…. Use the mux to skip 4. Complete all the connections and generate the select logic.



6. (20 pts.) State Machine Design

6.1. Reproduced below is the state diagram for the divider from your classnotes. A student has modified state transition conditions on the two diverging arrows on the "C" state as shown.

Class Notes Design	Modified Design
	

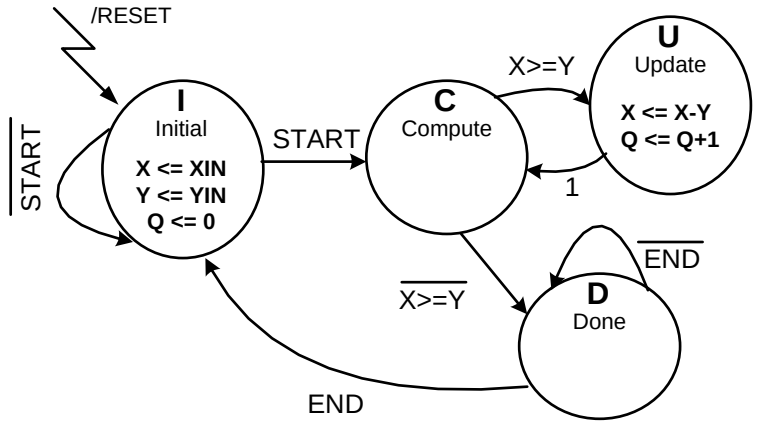
Does it work? YES / NO (Circle one)
It if works, explain how it is better or worse in performance? If it does not work, state why it does not work.

_____

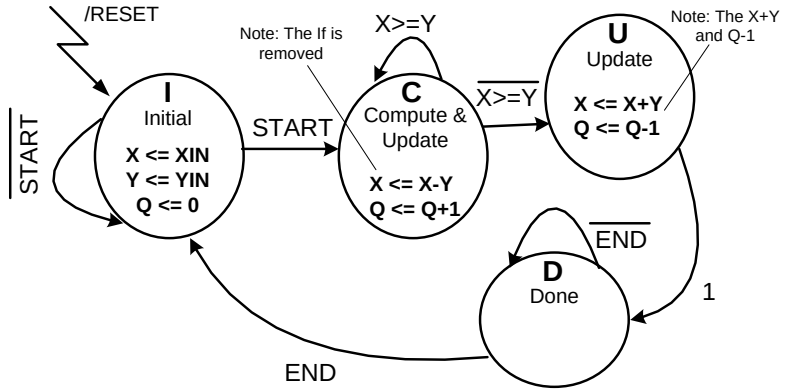
_____

_____

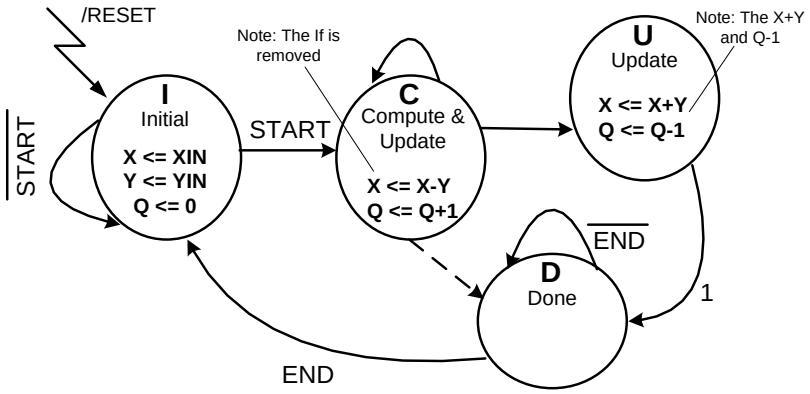
6.2. In light of the above question, can you suggest an improvement to the Moore machine for the divider (reproduced from the classnotes below) by modifying the state transition conditions only?

Class Notes Design	
	Explanation of improvement (if possible) or why no improvement is possible:

6.3. Mr. Trojan suggested a *better* Moore machine for the divider (*better* than the one in the classnotes and reproduced above) as shown below.

Class Notes Design	
	Using the example of 40 / 2 = 20, R=0, explain why this is better than our classnotes design.

6.4. Miss Trojan says that there is still a little more to improve by playing with the state transition arrows and conditions! Can you guess what is in her mind by completing the design below?

Class Notes Design	
	Using the example of 40 / 2 = 20, R=0, explain why this is a little better than Mr. Trojan's design.

6.5. Miss Trojan says that the resulting datapath of Mr. Trojan's design is more expensive. Mr. Trojan says the opposite. Explain who is right and why.