ECE M16 Homework 4

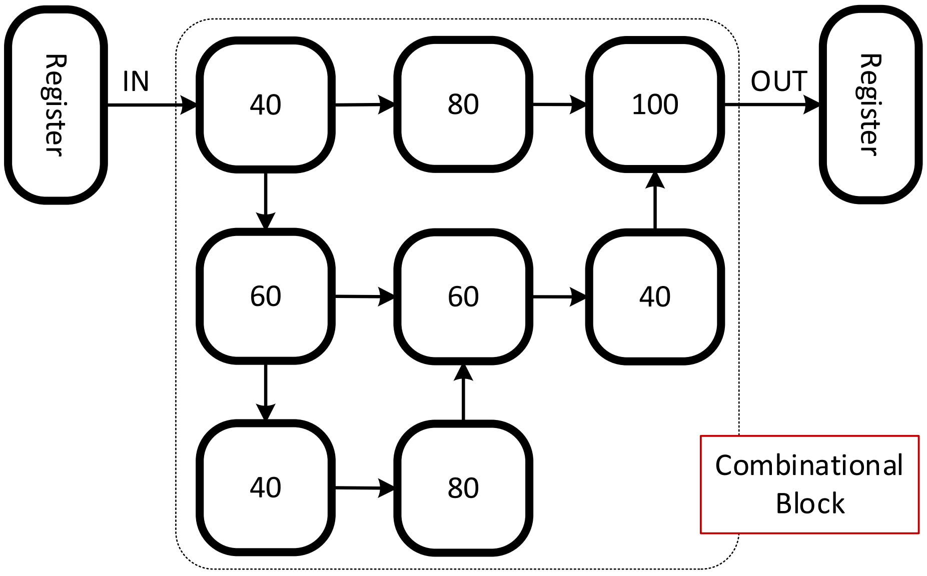
Instructor: Hooman Darabi

Sections covered: Pipelining and timing in logic design, data/control path sequential logic design, asynchronous sequential circuits.

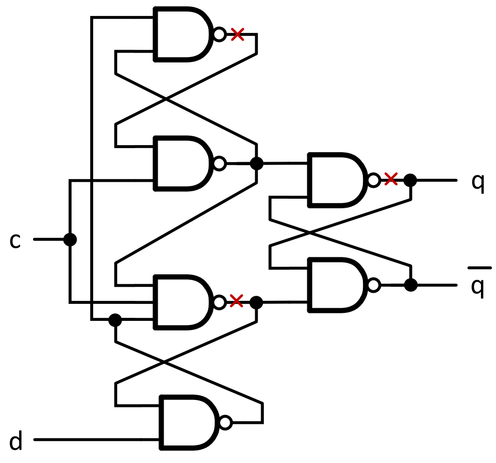
Total of 4 questions, 20 points for 1-3, 40 points for the design assignment, and 10 bonus points for the optional one. Due: 11:59PM Monday of week 8.

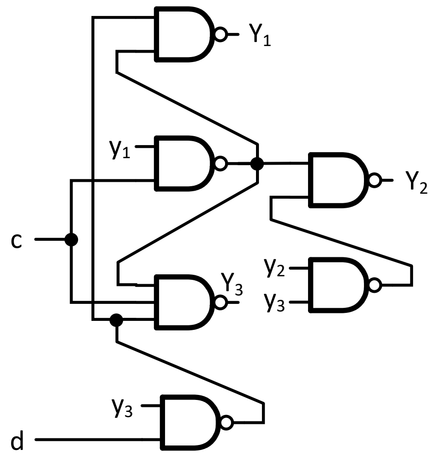
For all Logisim question, you must use version logisim-evolution-2.14.8.4-cornell.jar which you can download from:
http://www.cs.cornell.edu/courses/cs3410/2019sp/logisim/logisim-evolution.jar.
Submit a copy of your Logisim schematic, your design process, and any results. Use the chronogram capability of Logisim to visualize waveforms as appropriate.

1. Pipelining: The following combinational logic block can be broken down into modules as shown, with each delay specified. For each module, the propagation and contamination delay are the same, and given in pS. The registers comprise of DFF with contamination delay $t_{CCQ} = 10pS$, propagation delay $t_{dCQ} = 10pS$, setup time $t_s = 10pS$, and hold time $t_h = 10pS$.
   a. Determine the contamination delay of the combinational block.
   b. Determine the propagation delay of the combinational block.
   c. What is the minimum clock cycle for the combinational block?
   d. How much clock skew between the input and output registers, and in what direction will violate the hold time condition?
   e. Assume you are given an unlimited supply of registers so as to make this circuit faster via pipelining or retiming. What is the fastest clock cycle that you can obtain, and how many registers will you need (assuming that you also are minimizing the number of registers used)?
   f. What is the latency of the new pipelined system?

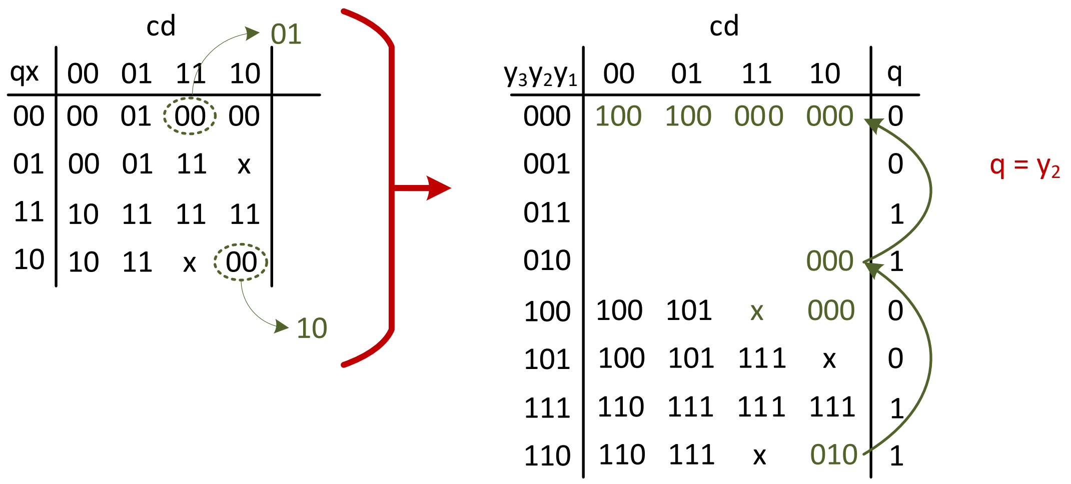


2. Design a D latch (not D flip flop) based on its flow table and synthesize one. Watch for the hazards. Design a DFF based on that, and compare to the one in the class.
3. A very common realization of the edge-triggered D flip flop is known as the Earle D flip flop, and is shown below on the left. There are three feedback loops as marked. The circuit may be analyzed by breaking the loops, and finding the expressions for Y_3, Y_2, and Y_1 based on their previous states, y_3, y_1, and y_1 and the inputs, c and d, as shown on the right.
   a. Find the expressions for Y_3, Y_2, and Y_1 based on their previous states, y_3, y_1, and y_1 and the inputs, c and d.
   b. Verify the D flip flop works as intended.





4. (Optional) It is possible in the sequential circuits that the most economical realization may not have the minimum number of states, although such cases are very much the exception. Since adding extra state variables is a try and error process, it should probably be considered only if a very large production volume is anticipated, certainly the case for the edge-triggered DFF. The flip flop flow table worked out in the class is shown below on the left. One may notice that if the circled entry in the first row changes to 01, it leads to simplification of state variable x. Likewise, if the circled entry on row four is changed to 10, it leads to simplification of state variable q (why?). Changing the flow table entries arbitrarily is of course not possible, and leads to an incorrect function. However, the simplification will occur if a third sate state variable Y_3 is added, as shown on the right.
   a. Justify that the two flow tables are equivalent, leading to the same master-slave D flip flop. Note that the row 000 shares the function of the stable state 1 with row 100. For $cd = 01$, the circuit cycles from 110 to 010 to 000 instead of directly going to 100. The entry for $cd = 11$ and $y_3y_2y_1 = 100$ can now be a don't care, since the square could be entered only if d could change simultaneously with the rising edge of the clock.
   b. Synthesize the flow table on the right using NAND only gates.
   c. Verify your design in logisim.



qx \ cd	00	01	11	10
00	00	01	00	00
01	00	01	11	x
11	10	11	11	11
10	10	11	x	00

y₃y₂y₁ \ cd	00	01	11	10	q
000	100	100	000	000	0
001					0
011					1
010				000	1
100	100	101	x	000	0
101	100	101	111	x	0
111	110	111	111	111	1
110	110	111	x	010	1

5. Design assignment: In this task you will design a system that given two integer inputs A (the dividend) and $D \neq 0$ (the divisor) outputs the quotient Q and remainder R. Here A, D, Q, and R are all 8-bit unsigned numbers, and the four numbers satisfy the following conditions: $A = Q \cdot D + R$ and $0 \leq R < D$. Your system will interact with the external user (i.e. another digital system) via two handshake signals REQ and ACK. Normally $REQ = 0$ and $ACK = 0$. To make use of the system, the user will provide the two operands on $A[7:0]$ and $D[7:0]$, and set $REQ = 1$. Your system will then compute the result according to the specified inputs, which will take multiple clock cycles, and once ready put the result on $Q[7:0]$, and $R[7:0]$, and set $ACK = 1$ to indicate that the results are ready. The user upon seeing ACK go from 0 to 1 will read the results, and then once it has read the outputs of your system it will bring REQ back to 0. Your system upon seeing REQ go to 0 should bring ACK down to 0. The user can initiate a new computation request once it sees that ACK is down to 0. The clock input to your system is clk and the reset input is RST. The circuit should be clock synchronous, operating on the rising edge of clk. The circuit should synchronously reset when $RST = 1$ with all outputs becoming 0. You may assume that after power up the circuit will be reset via a $RST = 1$ for at least one clock cycle before any operations are done. The state of the input signals is arbitrary prior to reset.

   In summary, the system you are designing has the following inputs and outputs:
   Inputs: clk, RST, REQ, $A[7:0]$ and $D[7:0]$
   Outputs: ACK, $Q[7:0]$, and $R[7:0]$

   Allowed Logisim Modules: Any except the divider module.