ECE M16 Homework 3

Instructor: Hooman Darabi

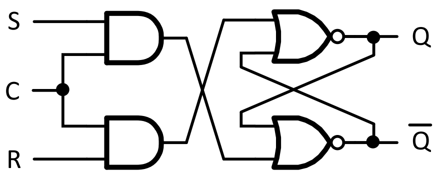
Sections covered: Flip flops, sequential logic design and verification

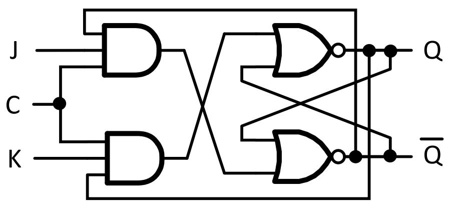
Total of 5 questions, 24 points for design assignments, 14 points for 1 and 3.

The optional question has 10 bonus points. Due: 11:59PM Friday of week 6.

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. JK latch: Shown below on left is a clocked RS (reset-set) latch. The latch cannot accept an entry $S = R = 1$. This restriction which may be inconvenient is avoided in a JK latch, as shown on the right. The circuit is the same as the RS latch, except that the outputs are fed back to the input AND gates. The naming apparently comes from Eldred Nelson who in designing a logical system assigned letters to flip flop inputs as follows: #1: A & B, #2: C & D, #3: E & F, #4: G & H, #5: J & K.
   a. Assume the latch is reset, and $J = K = 1$ when a clock pulse arrives. Analyze the circuit considering the performance of an RS latch. Note that a *toggle* or T latch is a special case of the JK latch with $J = K = 1$.
   b. Calculate the transition table of the JK latch, and compare it against the SR latch.
   c. Prove that the JK latch output may be expressed as: $Q^{n+1} = \overline{K^n}.Q^n + J^n.\overline{Q^n}$.
   d. Show that the D latch is a special case of JK latch with $J = \overline{K} = D$.





2. (Optional) Simplification by implication: For completely specified state tables, it is sometimes possible to formally reduce the number of the states. We will show the procedure in the context of an exercise here. Consider the Mealy state machine with an input x, and an output z, whose transition table is shown below on the left. We can see that there are no obvious simplification that could be made at this point.

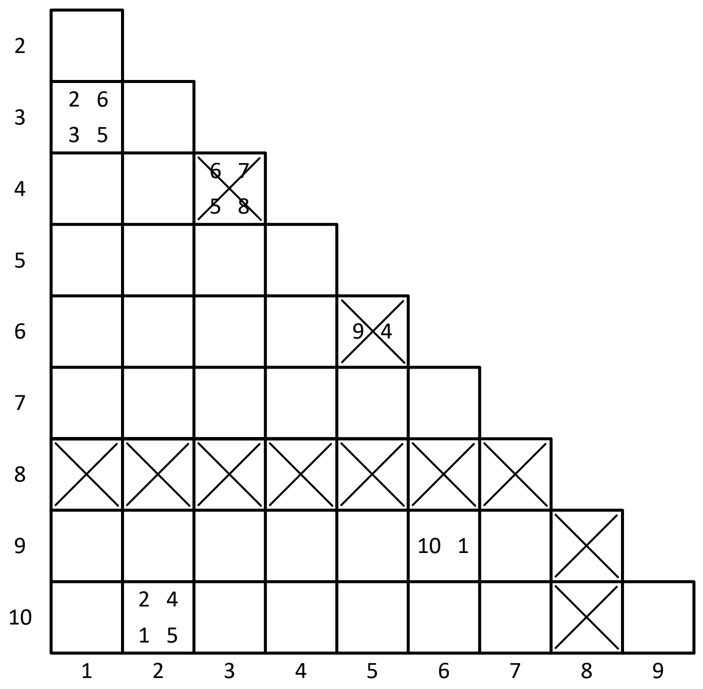
a. A set of states P are said to be implied by a set of states R if for some specific input X, P is the set of all present states of R. For example, for the state table shown, if $R_1 = (1\ \ 3)$, then the sets implied by R are $R_2 = (2\ \ 6)$ and $R_3 = (3\ \ 5)$. Now, fill out the squares of the table on the right based on the pair(s) of the states implied by the pair of states corresponding to the square. Some are shown as examples.

b. We start by placing an X in any square corresponding to any pair of states that have different outputs, and therefore cannot be equivalent. This is the case for state 8, and thus all the squares corresponding to 8 are crossed out. It does not matter then what states are implied by 8. The next step is to make a second pass through the table to see if any of the implied states are ruled out as possible equivalent states by the squares which have been marked X. For instance the square corresponding to states 3 and 4 are marked, as states 5 and 8 are incompatible. You may have to do a few passes to cross all the incompatible states.

c. Once the table is completed, and all the incompatibles states are crossed, you can recognize the states that are compatible. For example, you will see that states 2 and 6, 2 and 9, as well as 6 and 9 are all compatible. So we can group them into an equivalent class of (2 6 9). Show that states (1 3 5 10) are also an equivalent class. Are there any other?

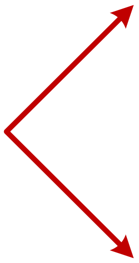
d. We can now assign new states to the equivalent classes and simplify the transition table. Show that the original FSM can be simplified into 5 states (compared to the 10). How many DFFs you will save a s a result?

	q^{n+1}		z^n	
q^n	x = 0	x = 1	x = 0	x = 1
1	2	3	0	0
2	4	5	0	0
3	6	5	0	0
4	7	8	0	0
5	9	10	0	0
6	4	10	0	0
7	7	1	0	0
8	9	1	1	0
9	4	1	0	0
10	2	1	0	0



3. Optimum state assignment: Consider the following state table, consisting of 4 states, a, b, c, and d. The output details have not been shown, To have the most economical realization one would like to group as many ones and don't cares as possible. This could potentially be different for different state assignments. Shown on the right are two different such assignments.
   a. Using gates and DFFs realize each assignment.
   b. Which assignment is more economical? Why?

	q^{n+1}	
q^n	x = 0	x = 1
a	a	c
b	a	d
c	a	b
d	d	d



Assignment 1

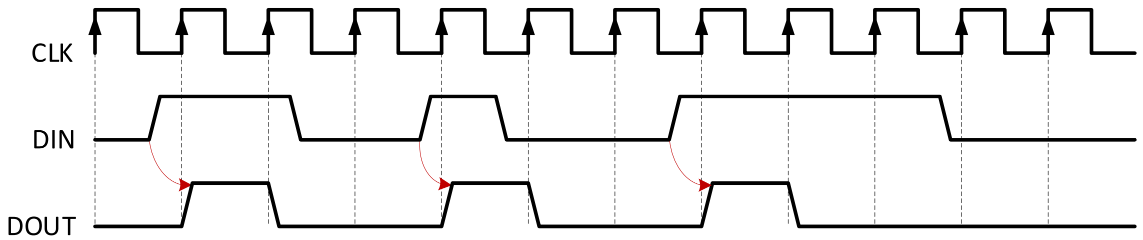
		q^{n+1}	
q^n	$(y_1y_0)^n$	x = 0	x = 1
a	00	00	01
c	01	00	11
b	11	00	10
d	10	10	10

Assignment 2

		q^{n+1}	
q^n	$(y_1y_0)^n$	x = 0	x = 1
a	00	00	01
c	01	00	10
d	11	11	11
b	10	00	11

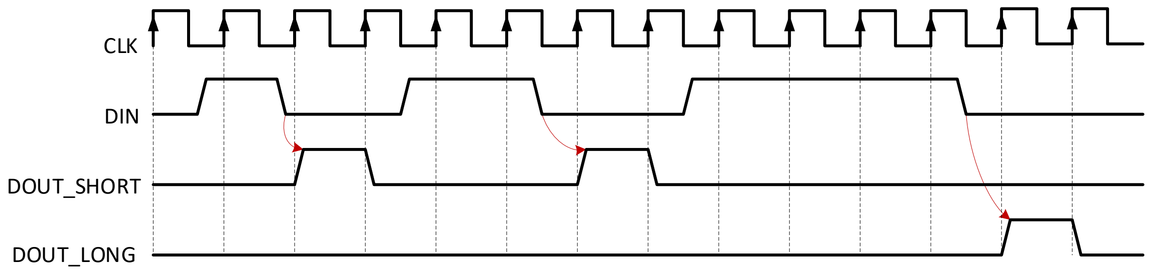
4. Design assignment: Digital systems often receive inputs from external sources which operate at much slower time scales relative to the clocks within the digital system. For example, signals coming from a button or key press change at human time scales (milliseconds or more) while clocks in MHz or higher. So the idealization that we often used in the lecture that there is a new input symbol at each clock tick is not really true: what the external world intended as a single input event spans multiple clock cycles. As an example, a signal coming from a button pressed by a human may last for multiple clock cycles. In this task you have to design a sequential system that has an input DIN, an output $DOUT$, and a CLK that will be running at a sufficiently high speed. DIN will be fed by the output of a button, which will be '0' normally but will become '1' whenever the user presses the button. Of course as noted above, DIN will typically be '1' for multiple clock cycles as we humans are slow. $DOUT$ is normally 0 and the logic of your system should arrange $DOUT$ to go high for exactly one clock cycle once for each distinct press of the button observed at a clock edge). Specifically, $DOUT$ should become 1 for exactly one clock cycle starting with the first clock edge where the button was found to be pressed. You may assume that two successive presses of the button will be separated by at least one edge during which the button was not pressed (i.e. $DIN = 0$). Assume your system is properly reset after

is starts up, though the reset is not a part of your design. The following shows an *example* run.



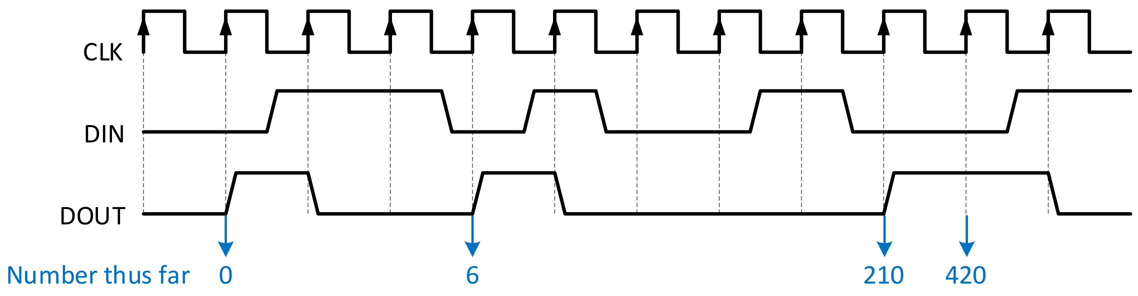
Allowed Logisim Modules: Any.

5. Design assignment: On many gadgets with limited buttons the user is asked to press and hold the button for a long duration to trigger some special action, e.g. going into a diagnostic or set up mode. In this task you have to design a Moore sequential system that has an input DIN, two outputs $DOUT\_SHORT$ and $DOUT\_LONG$, and a CLK. DIN will be fed by the output of a button, which will be '0' normally but will become '1' whenever the user presses the button. Your system will either set $DOUT\_SHORT = 1$ or set $DOUT\_LONG = 1$ for one clock cycle starting at the first clock edge after the user releases the button. Let n be the number of clock edges for which the button was pressed. Then the system should make $DOUT\_SHORT = 1$ if $1 \leq n < 4$ and $DOUT\_LONG = 1$ if $n \geq 4$. Assume your system is properly reset after is starts up, though the reset is not a part of your design. The following shows an *example* run.



Allowed Logisim Modules: Any.

6. Design assignment: Design a Moore sequential system that receives on input DIN a bit every clock tick, and outputs on $DOUT$ a bit indicating whether or not the sequence of bits received on DIN thus far represents a base-2 positive integer that is divisible by 6 or not. Specifically, $DOUT = 1$ implies that the bits received on DIN thus far represent a base-2 positive integer, whose most significant bit is the first bit that was received and whose least significant bit is the most recent bit received, is divisible by 6. Assume your system is properly reset after is starts up, though the reset is not a part of your design. The following shows an example run.



Allowed Logisim Modules: Any.