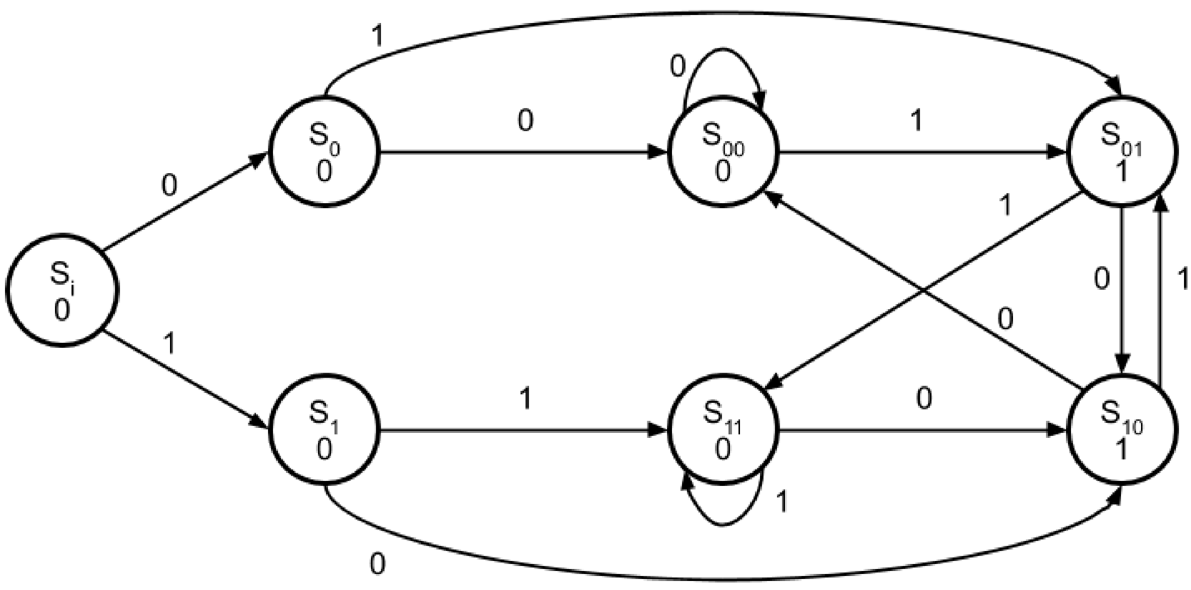
1. Recap of FSM design steps
    a. Turn a word description into a state diagram
    b. Turn the state diagram into a transition table
    c. Use transition table to make K-maps to simplify circuit
    d. Implement simplified circuit

2. Our word problem
    a. Want to create an edge detecting circuit
        i. Edge – position in a string of 0s and 1s where a 0 is adjacent to a 1
        ii. Essentially, XOR of adjacent bits
    b. Since this involves remembering the previous value, machine will have different states

3. State transition diagram to minimize the number of states
    a. Convert word description
        i. States based on patterns of inputs
        ii. Transitions between states based on individual input values
    b. For this problem
        i. S_i is our start state
        ii. Every other state is labeled S_{xy} where x is the previous bit, y is the current bit
            1. Each state will have a unique label
            2. This minimizes the number of states
        iii. Values inside each state are the output of the circuit
            1. 0 indicates no edge
            2. 1 indicates an edge was detected



    c. Mealy model differences
        i. Mealy model state diagrams look slightly different
            1. Outputs are on the edges/transitions between states, rather than the states themselves
            2. Usually, the number before the slash indicates the input
            3. Number after the slash indicates the output
            4. Will come back to this later

4. State table
    a. Table that lists all transitions from each *present state* to the *next state* for different values of inputs
        i. Output z is specified with respect to the present state
        ii. x is the next input

Present State	**Next State**		**Output**
	***x* = 0**	***x* = 1**	***z***
i	0	1	0
0	00	01	0
1	10	11	0
00	00	01	0
01	10	11	1
10	00	01	1
11	10	11	0

    b. Choice of flip-flops
        i. Here, will keep it simple and use DFFs
            1. Could use more complicated FFs, like J-K
            2. Using these means more logic in front of FFs

5. Derivation of next-state and output expressions
    a. Will use a Moore model for this example
        i. Need to assign binary codes to each state
            1. Have assigned them simply above
            2. Will talk about better way to do later
        ii. Since we have 7 states, will need 3 DFFs to represent all possible states
        iii. A, B, and C will represent the present state of the corresponding flip-flops
        iv. Initial state we start out in *i* must always be assigned to binary code of all 0s
            1. Flip flops are assumed to be 0 when we first start circuit

Present State	**Binary Code**	**Present State**			**Input**	**Next State**			**Output**
		A	***B***	***C***	***x***	***A'***	***B'***	***C'***	***z***
i	000	0	0	0	0	0	0	1	0
i	000	0	0	0	1	0	1	0	0
0	001	0	0	1	0	0	1	1	0
0	001	0	0	1	1	1	0	0	0
1	010	0	1	0	0	1	0	1	0
1	010	0	1	0	1	1	1	0	0
00	011	0	1	1	0	0	1	1	0
00	011	0	1	1	1	1	0	0	0
01	100	1	0	0	0	1	0	1	1
01	100	1	0	0	1	1	1	0	1
10	101	1	0	1	0	0	1	1	1
10	101	1	0	1	1	1	0	0	1
11	110	1	1	0	0	1	0	1	0
11	110	1	1	0	1	1	1	0	0
	111	1	1	1	0	d	d	d	d
	111	1	1	1	1	d	d	d	d

b. Use above to create K-maps for FF input combinational circuits
   i. Inputs are the concatenation of the current state variables A, B, C, and the input x
   ii. Outputs of each K-map are the next state variables A', B', C' and the output z
      1. Need to create 4 different K-maps, one for each variable
      2. Since state 111 wasn't assigned, combinations of that with x are don't cares

A'

Cx \ AB	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	1	1	d	1
10	0	0	d	0

$A' = B\bar{C} + A\bar{C} + Cx$

B'

Cx \ AB	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	d	0
10	1	1	d	1

$B' = \bar{C}x + C\bar{x}$

C'

Cx \ AB	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	d	0
10	1	1	d	1

$C' = \bar{x}$

c. Create K-map to determine the output combinational circuit
   i. Remember, Moore models don't use the input to determine the current output
      1. Therefore, K-map for z only uses A, B, and C
      2. Could place x in as well but result will be identical
   ii. Mealy model differences
      1. Mealy models use the current input to determine the current output
      2. Final K-map for output z would incorporate current flip-flop values as well as input x
      3. In this case, final K-map below would have 4 inputs

z

C \ AB	00	01	11	10
0	0	0	0	1
1	0	0	d	1

$z = A\bar{B}$

6. Finishing steps
   a. After K-maps done, implement these circuits
   b. Mealy model differences
      i. In general, Mealy model usually involves less circuitry
         1. In this case, it would
         2. Only need two DFFs to remember previous bit
         3. XOR those values with input to give final output
      ii. Not always the case, though