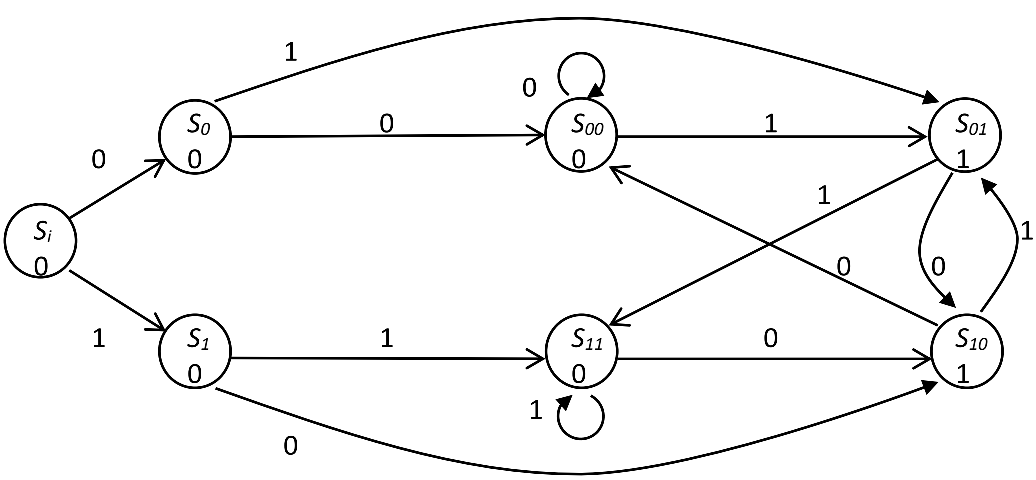
1. Recap of FSM design steps
    a. Turn a word description into a state diagram
    b. Turn the state diagram into a transition table
    c. Use transition table to make K-maps to simplify circuit
    d. Implement simplified circuit

2. Our word problem
    a. Want to create an edge detecting circuit
    b. Edge – position in a string of 0s and 1s where a 0 is adjacent to a 1
    c. Essentially, XOR of adjacent bits
    d. 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



        iv. Mealy model differences
            1. Mealy model state diagrams look slightly different
                a. Outputs are on the edges/transitions between states, rather than the states themselves
                b. Usually, the number before the slash indicates the input
                c. Number after the slash indicates the output
                d. 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
    b. Output z is specified with respect to the present state
    c. 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

    d. Choice of flip-flops
        i. Here, will keep it simple and use DFFs
        ii. Could use more complicated FFs, like J-K
            1. 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
    b. Need to assign binary codes to each state
        i. Since we have 7 states, will need 3 DFFs to represent all possible states
        ii. A, B, and C will represent the present state of the corresponding flip-flops
        iii. 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