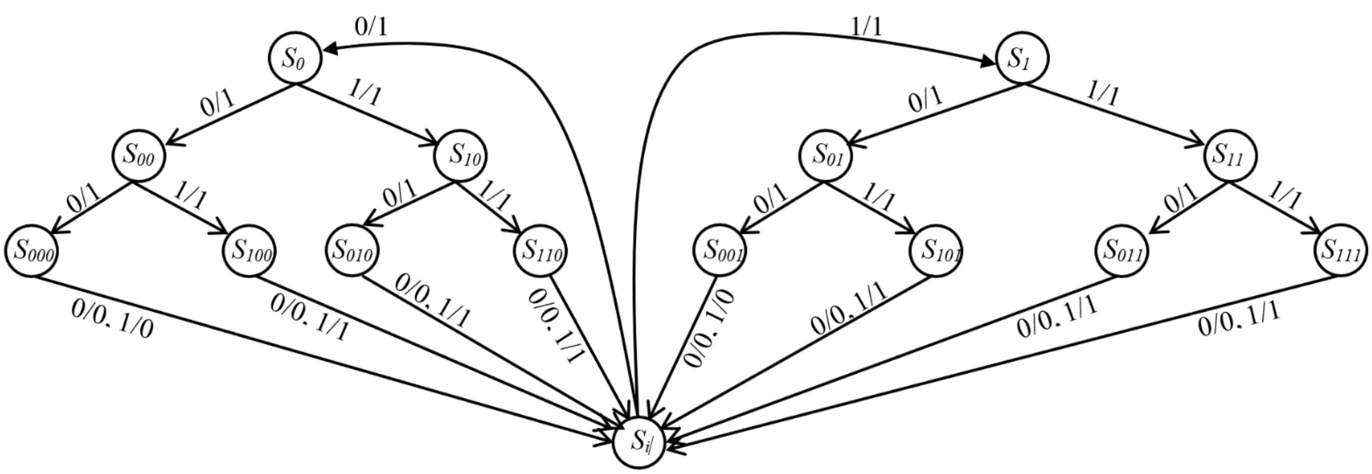
1. Mealy model example
    a. Design a circuit for checking binary coded decimal (BCD)
        i. BCD uses a 4-bit number to represent a single decimal digit
        ii. Circuit will take in 4 bits, indicate whether BCD is valid, then take in another 4 bits
        iii. First bit taken in is least significant, last bit is most significant
            1. This means newest numbers get placed on left side in our diagram below
        iv. Circuit outputs 0 if the BCD is valid (binary number formed is between 0 and 9) and 1 otherwise
    b. State transition diagram
        i. As noted previously, biggest difference for Mealy model diagrams is on the outputs
            1. States no longer have outputs on them since output can differ with input
            2. Instead, place output together with relevant input on the transition arrow
        ii. Diagram below
            1. Left branch means 0 input
            2. Right branch means 1 input



2. Minimizing a Mealy model
    a. Apply the Partitioning Minimization Procedure to reduce the number of states in the Mealy model
    b. Since this is a Mealy model, states do not have output values
        i. The k-successors for a state are the output value created by the combination of the state and the possible inputs
    c. Except for state i, for this example we will refer to the states simply by their bit patterns
    d. P_1 = (i, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111)
    e. P_2 = (i, 0, 1, 00, 01, 10, 11) (000, 001) (100, 010, 110, 101, 011, 111)
        i. The first block all have k-successors of 0/1, 1/1
        ii. The second block all have 0/0, 1/0
        iii. The last block all have 0/0, 1/1
        iv. Don't have a fourth block based on the other possible k-successor combination, 0/1, 1/0
            3. No states that have that combination

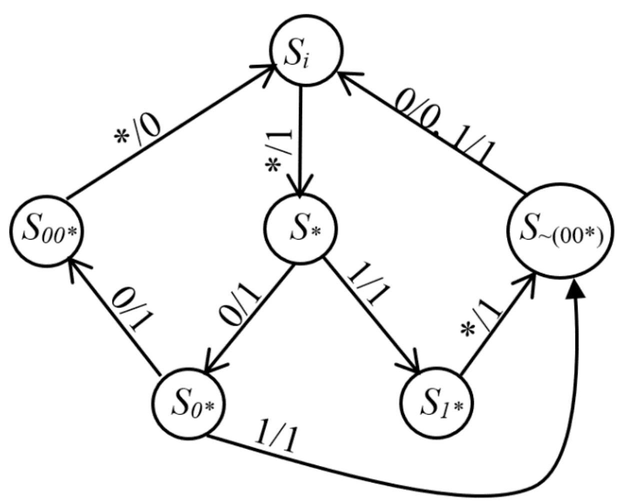
f. P_3 = (i, 0, 1) (00, 01) (10, 11) (000, 001) (100, 010, 110, 101, 011, 111)
   i. The first block all have k-successors in the first block of P_2
   ii. (00, 01) have 0-successors in (001, 001), and 1-sucessors in (100, 010, 110, 101, 011, 111)
   iii. (10, 11) have both of their k-successors in (100, 010, 110, 101, 011, 111)
   iv. (000, 001) have the single state, i, as their k-successor
      1. We can presume that that block will never split
      2. Ignore it until the end of the procedure
   v. Similarly, (100, 010, 110, 101, 011, 111) all have a single state, i, as their k-successor
      1. That block will never split either

g. P_4 = (i) (0, 1) (00, 01) (10, 11) (000, 001) (100, 010, 110, 101, 011, 111)
   i. (i, 0, 1) of P_3 must split
      1. i leads to (i, 0, 1) for both of its k-successors
      2. 0 and 1 both lead to (00, 01) for its 0-successor and (10, 11) for its 1-successor

h. P_5 = (i) (0, 1) (00, 01) (10, 11) (000, 001) (100, 010, 110, 101, 011, 111)
   i. All elements in each block of P_5 lead to identical blocks of P_4

i. Therefore, P_5 is our final partition, and we will use it to make the state diagram below

3. Implementing the FSM
   a. We let * stand for either a 0 or 1 at a given position
   b. ~(00*) = (100, 010, 110, 101, 011, 111) in our notation
      i. Call it so because these states are not (~) 00*
   c. State transition diagram and state table below



State transition diagram

Present State	**Next State**		**Output**	
	x = 0	x = 1	x = 0	x =1
i	*	*	1	1
*****	0*	1*	1	1
0*	00*	~(00*)	1	1
1*	~(00*)	~(00*)	1	1
00*	*i*	*i*	0	0
~(00*)	*i*	*i*	0	1

State table

4. Optimal assignment of binary codes
    a. Will need $\lceil \log_2 6 \rceil = 3$ flip flops to represent 6 states
    b. Can assign binary codes for states randomly
        i. Random assignment works
        ii. However, careful assignment reduces the combinational logic
    c. Rule of thumb for state binary code assignments
        i. Try to assign adjacent (Hamming distance of 1) code words to a state and the state that follows it
        ii. If two states have the same next state, assign those states code words adjacent to next state
        iii. Creating a K-map helps immensely with this process
        iv. Initial state *i* will always be all 0s
    d. Assign using the rules above
        i. Place i at 000, will always do this
        ii. Place * next to i at 010 since * is the successor to i
        iii. Place 0* and 1* adjacent to * at 110 and 011 respectively
        iv. 00* needs to be adjacent to i and 0*, which leaves 100 as the only place
        v. Would like ~(00*) to be adjacent to 0*, 1*, and i, but that isn't possible
            1. These are rules of thumb, not fixed laws
        vi. Can place ~(00*) at 001 to be adjacent to i and 1*, though
    e. Note that there may potentially be more than one valid code assignment that minimizes distance

Binary Code		AB			
		00	**01**	**11**	**10**
C	**0**	*i*	*	0*	00*
	1	~(00*)	1*		

    f. Make state transition table from the above with assigned binary codes
        i. Be careful when assigning values!
        ii. Table states don't line up neatly in order as they did in previous examples

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	1	0	1
i	000	0	0	0	1	0	1	0	1
~(00*)	001	0	0	1	0	0	0	0	0
~(00*)	001	0	0	1	1	0	0	0	1
*****	010	0	1	0	0	1	1	0	1
*****	010	0	1	0	1	0	1	1	1
1*	011	0	1	1	0	0	0	1	1
1*	011	0	1	1	1	0	0	1	1
00*	100	1	0	0	0	0	0	0	0
00*	100	1	0	0	1	0	0	0	0
	101	1	0	1	0	d	d	d	d
	101	1	0	1	1	d	d	d	d
0*	110	1	1	0	0	1	0	0	1
0*	110	1	1	0	1	0	0	1	1
	111	1	1	1	0	d	d	d	d
	111	1	1	1	1	d	d	d	d

g. Create K-maps for each flip flop based on input and present state in table above
   i. Be careful when entering values from the binary code table!
   ii. Some states are missing and are don't cares, like 101

A' — AB

Cx	**00**	**01**	**11**	**10**
00	0	1	1	0
01	0	0	0	0
11	0	0	d	d
10	0	0	d	d

$A' = B\overline{Cx}$

B' — AB

Cx	**00**	**01**	**11**	**10**
00	1	1	0	0
01	1	1	0	0
11	0	0	d	d
10	0	0	d	d

$B' = \overline{AC}$

C' — AB

Cx	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0	1	1	0
11	0	1	d	d
10	0	1	d	d

$C' = Bx + BC$

h. Use derivations from the K-maps to design initial combinational circuit
i. Create a K-Map based on flip-flops to determine the output combinational circuit
   i. Since this is a Mealy model, we also use the input
   ii. Don't cares are in same position as K-maps above

z — AB

Cx	**00**	**01**	**11**	**10**
00	1	1	1	0
01	1	1	1	0
11	1	1	d	d
10	0	1	d	d

$z = B + \overline{AC} + \overline{A}x$