1. From last time
    a. Finished filling out the binary code table for our edge-detector circuit
    b. Table repeated below

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

    c. Use above to create K-maps
        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
        iii. 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

$C = \bar{x}$

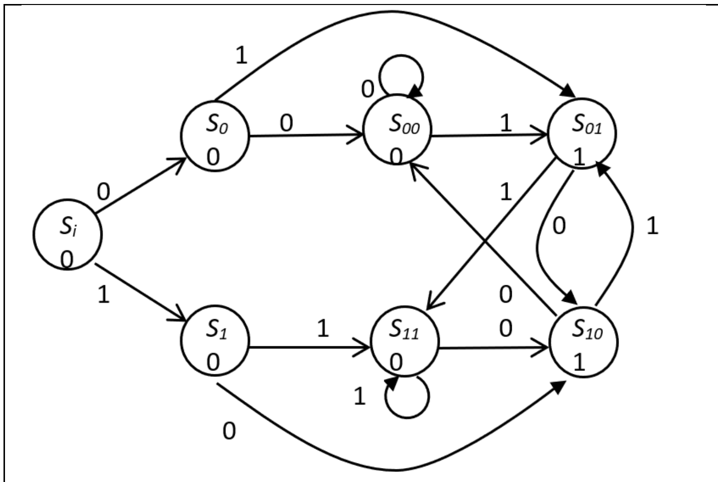
    d. Use above derivations from K-maps to design initial combinational circuit
    e. Create K-map based on value to determine the output combinational circuit
        i. Remember, Moore models don't use the input to determine the current output
            1. Therefore, K-map for z only uses A, B, and C
        ii. Mealy model difference
            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

z

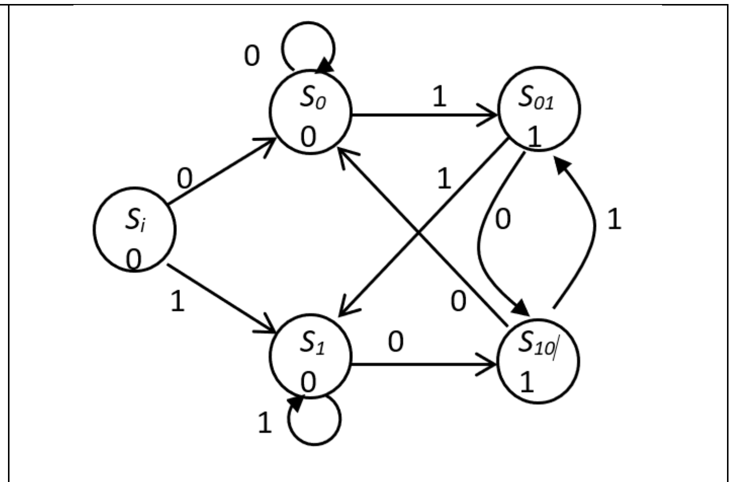
C \ *AB*	**00**	**01**	**11**	**10**
0	0	0	0	1
1	0	0	0	1

$z = A\bar{B}$

2. Mealy model versus Moore
    a. In general, Mealy model usually involves less circuitry
        i. In this case, it does
        ii. Only need two DFFs to remember previous bit
        iii. XOR those values with input to give final output
3. Minimization
    a. Did the 7-state Moore model edge detector earlier
    b. Can minimize this down to 5 states, as shown below



Original state diagram



Minimized state diagram

    c. Some definitions
        i. Two states S_i and S_j are *equivalent* if and only if for every possible input sequence, the same output sequence will be produced, regardless of whether S_i or S_j is the initial state
        ii. A *successor* to state S_i is a state that it transitions to based on its input
            1. S_0 in the unminimized FSM has S_{00} and S_{01} as its successors
            2. Differentiate successors based on input
                a. S_{00} is the *0-successor* of S_0
                b. S_{01} is the *1-successor* of S_0
                c. Collectively, all immediate successors of a state form the *k-successors* of the state
        iii. A *block* is a subset of states that may be equivalent
        iv. A *partition* is a set of blocks where the states in each block are not equivalent to the states in the other blocks
4. Partition minimization procedure
    a. Will use the unminimized edge detector FSM for the rest of this example
    b. Start with all states in one partition and in same block
        i. $P_1 = (S_i, S_0, S_1, S_{00}, S_{01}, S_{10}, S_{11})$
    c. Create P_2 by dividing states in P_1 that have same outputs
        i. From definition of *equivalent*, states that have different outputs cannot be equivalent
        ii. $P_2 = (S_i, S_0, S_1, S_{00}, S_{11})\ (S_{01}, S_{10})$
    d. Create P_3 by looking at k-successors of each state
        i. States of a block that have k-successors that in are different blocks from others in the block must be placed in new blocks, grouped by their shared k-successors
        ii. Look at first block $(S_i, S_0, S_1, S_{00}, S_{11})$
            1. 0-successors for the $(S_i, S_0, S_1, S_{00}, S_{11})$ block of P_2 are $S_0, S_{00}, S_{10}, S_{00}, S_{10}$, respectively
                a. Need to divide states into those that stay in the block, and those that move to the (S_{01}, S_{10}) of P_2
                b. Thus, we get (S_i, S_0, S_{00}) and (S_1, S_{11}).

2. 1-successors for (S_i, S_0, S_{00}) are S_1, S_{01}, S_{01}, respectively
   a. So, we divide into (S_i) and (S_0, S_{00}).
3. 1-succesors for (S_1, S_{11}) are S_{11}, S_{11}, so it will not need to be divided.
4. First block of P_2 will be divided into the three blocks (S_i), (S_1, S_{11}) and (S_0, S_{00}) in P_3.

iii. Now look at second block (S_{01}, S_{10})
   1. 0-successors for the (S_{01}, S_{10}) block of P_2 are S_{10}, S_{00}, respectively
   2. These are in different blocks from each other in P_2
   3. Will have to divide into two separate blocks (S_{01}), and (S_{10})

iv. So P_3 = (S_i)(S_1, S_{11})(S_0, S_{00})(S_{01})(S_{10})

e. Further partitions
   i. Once a block has one state in it, don't need to partition it further
      1. Will still need to use it to determine k-successors, though
   ii. P_4 and further partitions look at each multiple-element block of the previous partition to see if the k-successors of its elements lead to the same blocks of the previous partition
      1. If not, the block must be further divided
      2. If any block splits, then we must continue to another step of partitioning
      3. If no block splits, then we are done
   iii. Look at P_4 now
      1. 0-successors of (S_1, S_{11}) are both block (S_{10}), so that will not cause the block to split.
      2. 1-successors are both (S_1, S_{11}), so there is no need to separate the block further during this partitioning step
      3. 0-successors of (S_0, S_{00}) are both block (S_0, S_{00}), so that will not cause the block to split
      4. 1-successors are both (S_{01}), so there is no need to separate the block further during this partitioning step
   iv. Since no block split, the final minimized partition is P_3 = (S_i)(S_1, S_{11})(S_0, S_{00})(S_{01})(S_{10})
      1. This matches the five-state minimized state diagram