Design of Finite State Machines

Steps involved in designing FSMs

1. Understand the description of the system. Usually, a timing diagram is helpful to consider different scenarios.

2. Construct the state/output table or the state diagram corresponding to the description.

3. Minimize number of identified states (optional, not required). The idea of minimization is to identify *equivalent states*. A pair of equivalent states can be replaced by a single state.

   Two states S1 and S2 are equivalent if and only if two conditions are true:
   1. S1 and S2 must produce the same values at the state-machine output(s).
   2. For each *input combination*, S1 and S2 must have either the same next state or equivalent

4. State Assignment: Choose state variables to represent the states in the state table. For n states, the minimum number of variables you need is equal to $\lceil \log_2 n \rceil$. You may end up with *"unused states"*.

   defines the # of FFs

   - *Minimal risk*: Assumes the machine may get into unused states, and for any input combination, the unused states go to the "initial" state.
   - *Minimal cost*: Assumes the machine will never enter an unused state. The next-state entries of the unused states can be marked as "don't-cares."

5. Assign binary codes to state variable. Examples: counting order, one-hot, almost one-hot, decomposed, …

   - *Counting order:* Simplest coding, however, it does not always lead to the simplest excitation equations, output equations, and or simplest logic circuits.
   - *One-hot:* uses one bit per state. Therefore, it may require more number of state variables than the minimum ($\lceil \log_2 n \rceil$). This approach usually leads to short excitation equations, since each FF must be set to 1 for transitions into only one state. But it also requires more number of FF than the minimum number.
   - *Almost One-hot:* uses the "none-hot" combination for the initial state.
   - *Gray Code:* one state variable changes on each state transition.
   - *Decomposed:* see Pages 463-464 in the textbook.

	one-hot Coding	almost one-hot Coding
Q_0	00001	0000
Q_1	00010	0001
Q_2	00100	0010
Q_3	01000	0100

Q_4 1 0 0 0 0 1 0 0 0

6. Substitute states in the state/output table with their assigned binary codes, to build the transition table.

7. Find FF characteristics equations, excitation and output equations. K-Maps could become useful here. If you use D FFs for state memory (the case in this course), you can easily obtain the excitation equations from the transition equations.

8. Draw the logic diagram.

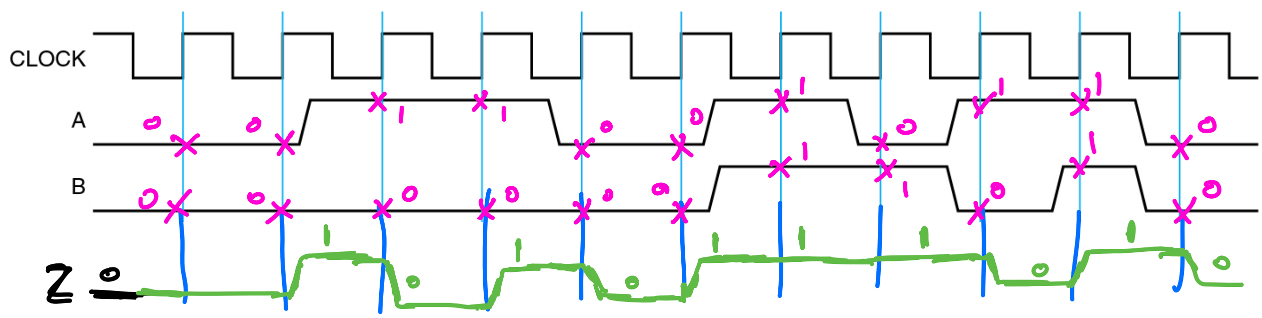
Example: Design a state machine with two inputs, A and B, and a single output Z. Z is 1 if:

– A had the same value at each of the two previous clock ticks,

or

– B has been 1 since the last time that the first condition was true

Step 1: Understand the description of the system.



Step 2: Construct the state/output table or the state diagram corresponding to the description.

INIT: When the system powers up. The output Z will be 0 in this state.

A0: Got A = 0 on the previous tick, A ≠ 0 on the tick before that, and B ≠ 1 at some time since the previous pair of equal A inputs. The output Z will be 0 in this state.

A1: Got A = 1 on the previous tick, A ≠ 1 on the tick before that, and B ≠ 1 at some time since the previous pair of equal A inputs. The output Z will be 0 in this state.

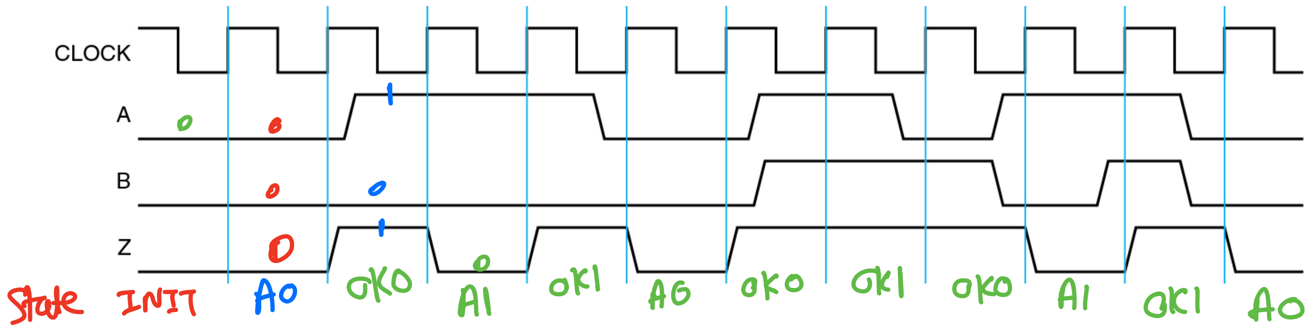
OK: Got a pair of equal A inputs (0,0 or 1,1) on the previous two ticks. The output Z will be 1 in this state.

Note: When the system goes to OK state, it remains in the OK state if 1) *A remains constant*, or, 2) *B=1*. For the system to know if A has remained constant (when B ≠ 1) , it needs to know its previous value of A. Therefore, we need to have two situations for the OK state:

OK0: the system got to this state with A0.

OK1: the system got to this state with A1.

INIT , A0, A1, ok0, ok1

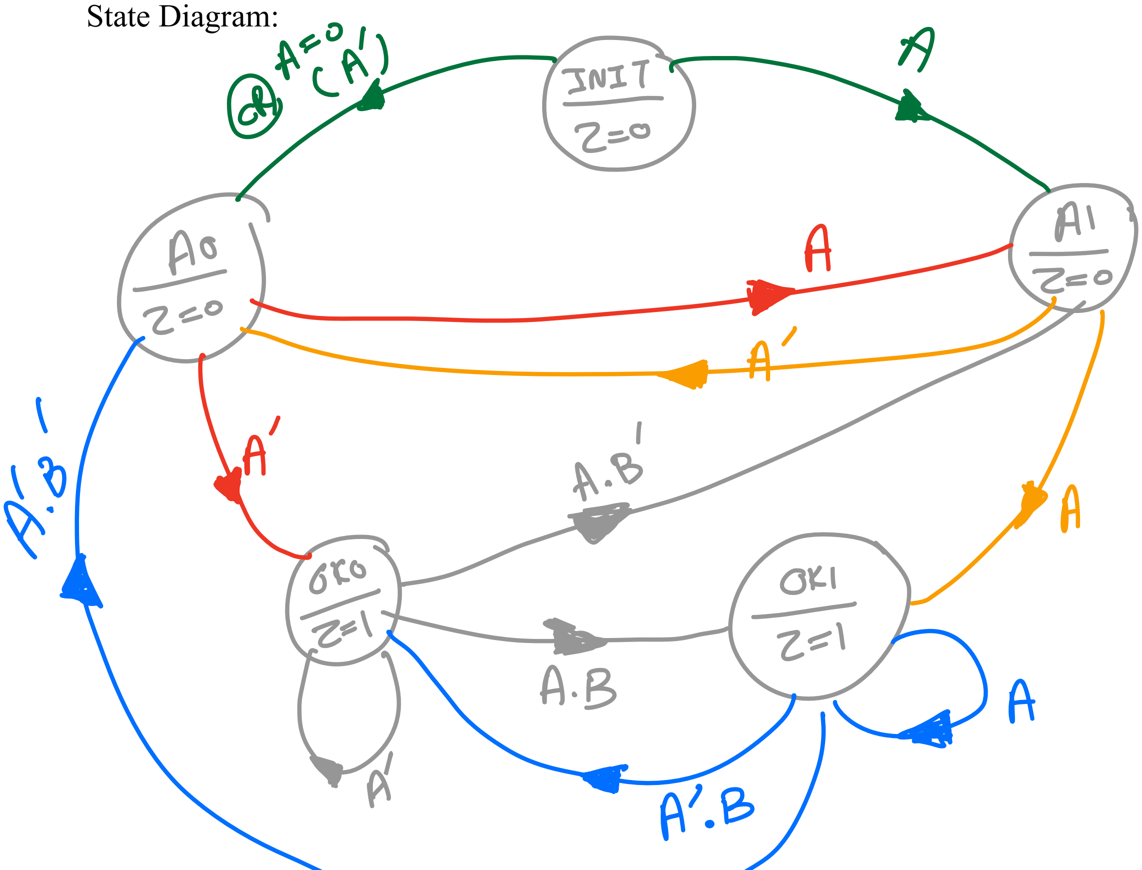


With these states, we can construct the state/output table:

S	AB = 00	01	10	11	Z
INIT	A0	A0	A1	A1	0
A0	OK0	OK0	A1	A1	0
A1	A0	A0	OK1	OK1	0
OK0	OK0	OK0	A1	OK1	1
OK1	A0	OK0	OK1	OK1	1

S*

State Diagram:



Step 3: Minimize number of identified states (optional)

Meaning	S	AB 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK00	OK00	A1	A1	0
Got a 1 on A	A1	A0	A0	OK11	OK11	0
Got 00 on A	OK00	OK00	OK00	OKA1	A1	1
Got 11 on A	OK11	A0	OKA0	OK11	OK11	1
OK, got a 0 on A	OKA0	OK00	OK00	OKA1	A1	1
OK, got a 1 on A	OKA1	A0	OKA0	OK11	OK11	1
		S*				

equivalent

equivalent States

Step 4: State Assignment: Choose state variables to represent the states in the state table.

5 States: INIT, A0, A1, OK0, OK1

To represent 5 states, we need at least 3 State variables

$2^3 = 8$ States $\rightarrow$ $\boxed{\text{3 unused States}}$

Step 5: Assign binary codes to state variable.

S	$Q_1\ Q_2\ Q_3$	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5$	$Q_1\ Q_2\ Q_3$
INIT	0 0 0	0 0 0 0 1	0 0 0
A0	0 0 1	0 0 0 1 0	1 0 0
A1	0 1 0	0 0 1 0 0	1 0 1
OK0	0 1 1	0 1 0 0 0	1 1 0
OK1	1 0 0	1 0 0 0 0	1 1 1
	Counting order	One_hot	Decomposed

Z (output)

$Q_3 = 1$ when $A = 1$

$Q_1 = 0$ when $S = INIT$

$Q_1 = 1$ otherwise

$$Z = \underbrace{Q_1 \cdot Q_2 \cdot Q_3'} + \underbrace{Q_1 Q_2 Q_3} = Q_1 \cdot Q_2 [Q_3' + Q_3]$$
$$= Q_1 \cdot Q_2$$

Step 6: Substitute states in the state/output table with their assigned binary codes, to build the transition table.

Using Decomposed Coding

S	AB 00	01	11	10	Z
INIT	A0	A0	A1	A1	0
A0	OK0	OK0	A1	A1	0
A1	A0	A0	OK1	OK1	0
OK0	OK0	OK0	OK1	A1	1
OK1	A0	OK0	OK1	OK1	1
	S*				

$Q_1 Q_2 Q_3$	AB 00	01	11	10
0 0 0	100	100	101	101
1 0 0	110	110	101	101
1 0 1	100	100	111	111
1 1 0	110	110	111	101
1 1 1	100	110	111	111
	Q_1^*	Q_2^*	Q_3^*	

20

5

A, B, Q_1, Q_2, Q_3

Q_1^*, Q_2^*, Q_3^*

Step 7: Find excitation and output equations.

32-cell K-Map

$32 - 20 = 12$

$$Q_1^* = Q_1 + Q_2' \cdot Q_3'$$

Unused States

$Q_1 = 0$

Q_2Q_3 \ AB	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$Q_1 = 1$

Q_2Q_3 \ AB	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

Q_1

$$Q_2^* = Q_1 Q_3 \cdot A + Q_1 \cdot Q_3' \cdot A' + Q_1 \cdot Q_2 \cdot B$$

$Q_1 = 0$

Q_2Q_3 \ AB	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$Q_1 = 1$

Q_2Q_3 \ AB	00	01	11	10
00	1	1	0	0
01	0	0	1	1
11	0	1	1	1
10	1	1	1	0

$$Q_3^* = Q_1 \cdot A + A \cdot Q_2' \cdot Q_3'$$

Q_2Q_3 \ AB	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

Q_2Q_3 \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

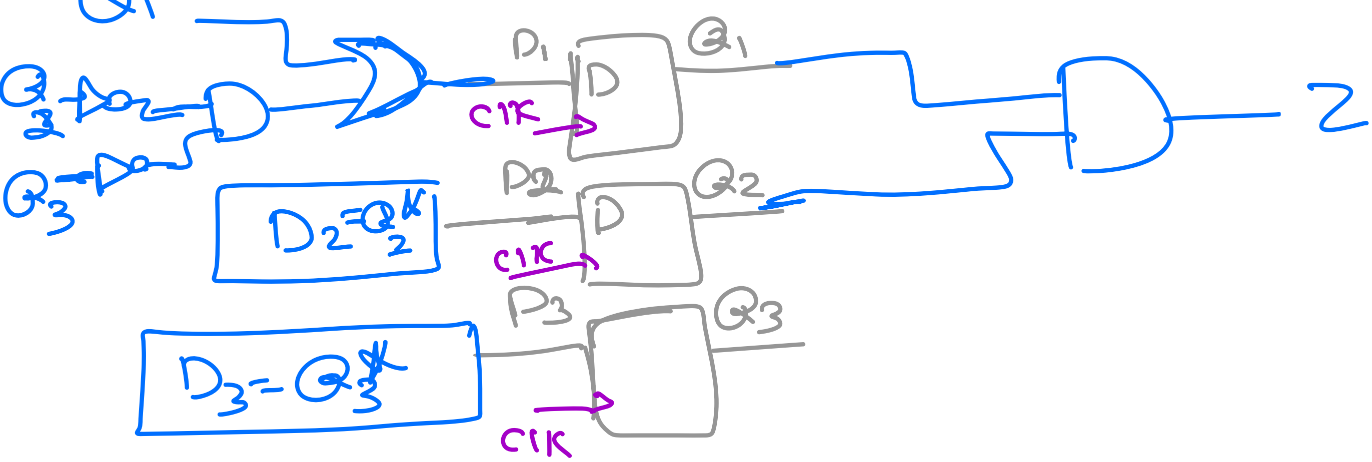
$Q_1 = 0$ $Q_1 = 1$

Taking the "minimal risk" approach, if the machine goes to unused state, the next state will be "INIT" or

$Q_1^* Q_2^* Q_3^* = 0\ 0\ 0$

Step 8: Draw the logic diagram.

$D_1 = Q_1^*$, $D_2 = Q_2^*$, $D_3 = Q_3^*$



In case we were considering *minimal cost* approach, in Step 7 we would have used the following K-Maps:

$Q_1^* = 1$

$Q_1 = 0$

Q_2Q_3 \ AB	00	01	11	10
00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$Q_1 = 1$

Q_2Q_3 \ AB	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	

$Q_2^* = Q_2 \cdot B + Q_3 \cdot A + Q_1 \cdot Q_3' \cdot A$

Q_2Q_3 \ AB	00	01	11	10
00	0	0	0	0
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

Q_2Q_3 \ AB	00	01	11	10
00	1	1	0	0
01	0	0	1	1
11	0	1	1	1
10	1	1	1	0

$Q_3^* = A$

Q_2Q_3 \ AB	00	01	11	10
00	0	0	1	1
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

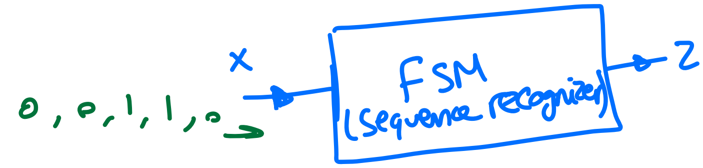
Q_2Q_3 \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Z:

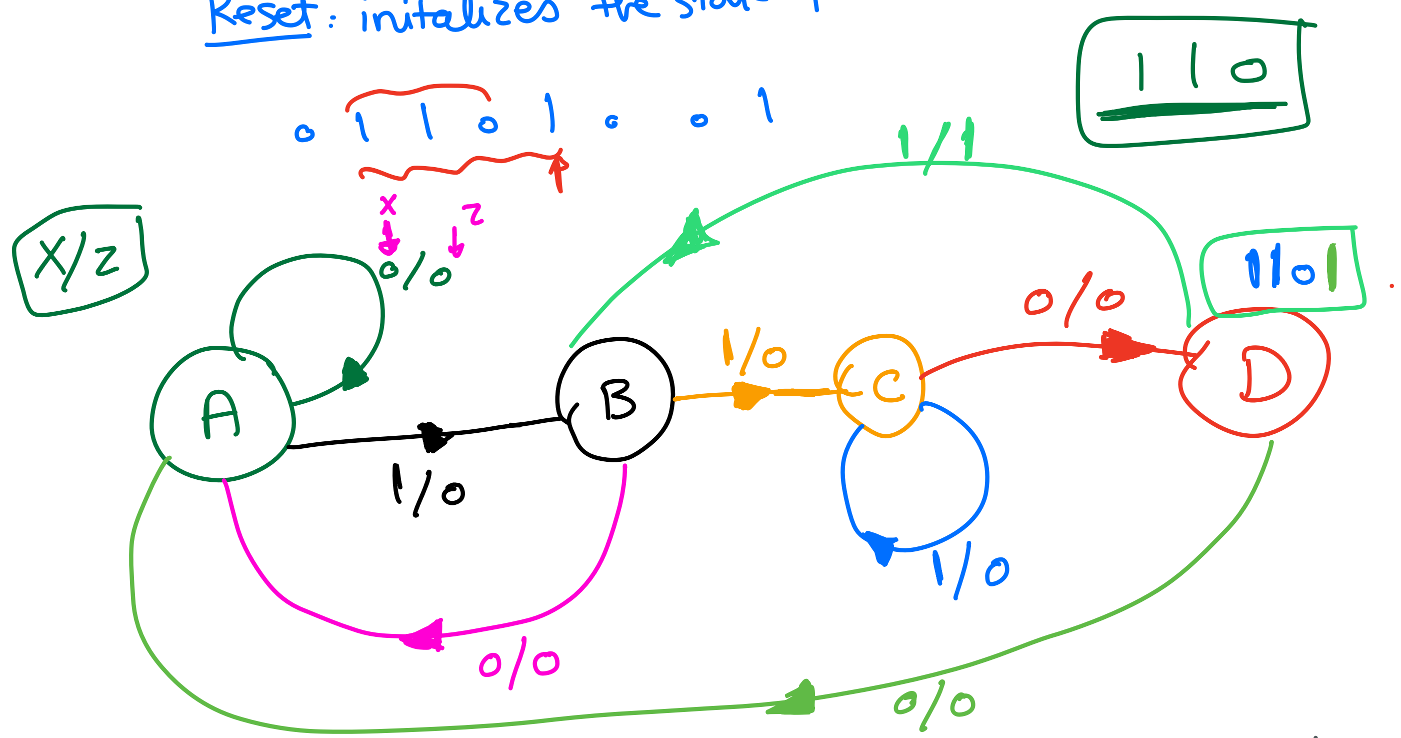
Q_1 \ Q_2Q_3	00	01	11	10
0	0	X	X	X
1	0	0	1	1

$Z = Q_2$

Example: A "sequence recognizer" has one input X and one output Z. It has Reset applied to the direct reset inputs on its flip-flops to initialize the state of the circuit to all zeros. The circuit is to recognize the occurrence of the sequence of bits 1101 on X by making Z equal to 1 when the previous three inputs to the circuit were 110 and current input is a 1. Otherwise, Z equals 0.



Reset: initalizes the state of the circuit to all zeros.



A: → 0 → Stay (A)
→ 1 → State (B) → X=0 → State (A)
→ X=1 → State (C) → X=0 → State D → X=0 → State (A)
→ X=1 → State B (Z=1)
→ X=1
(1)
11

	2			
S	X=0	X=1	X=0	X=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

S^* X

S	Q_1	Q_2
A	0	0
B	0	1
C	1	1
D	1	0

X=0		X=1	
0	0	0	1
0	0	1	1
1	0	1	1
0	0	0	1

Q_1^* Q_2^* $Q_1^* Q_2^*$

$\boxed{Q_1, Q_2, X}$

X \ Q_1Q_2	00	01	11	10
0	0	0	1	0
1	0	1	1	0

X \ Q_1Q_2	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$$Q_1^* = Q_1 . Q_2 + X . Q_2$$
$$= D_1$$

$$Q_2^* = X = D_2$$

X \ Q_1Q_2	00	01	11	10
0	0	0	0	0
1	0	0	0	1

$$Z = X \cdot Q_1 \cdot Q_2'$$