L9 – State Assignment and gate implementation

States Assignment

- Rules for State Assignment
- Application of rule
- Gate Implementation

- Ref: text Unit 15.8

Rules for State Assignment

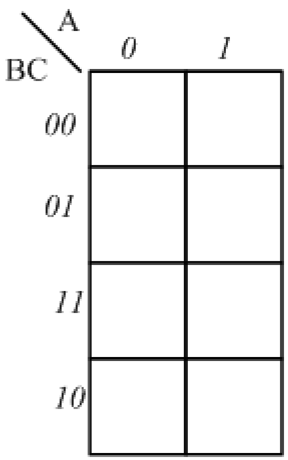
- Situation: You have arrived at the reduced state table and no further state reduction can be made.
- Does it matter how you assign the binary encoding to the states – YES!!!
- But how to do it!!!

Guidelines for State Assignment

- To try all equivalent state assignments, i.e., and exhaustive exploration of all possible state assignments. This is a *n-p complete* problem.
- Do not panic!!! (where does this come from?)
- There are guidelines that help
- 1. States which have the same next state for a given input should be given adjacent assignments.
- 2. States which are the next states of the same state should be given adjacent assignments.
- And third
- 3. States which have the same output for a given input should be given adjacent assignments.

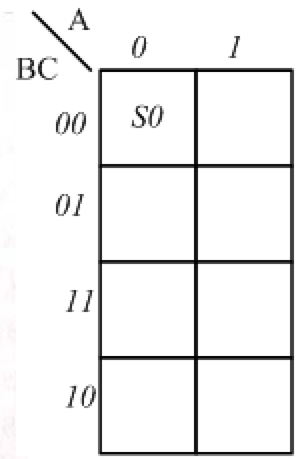
The starting state

- Assign the starting state to the “0” square on an assignment map. (An assignment map looks much like a K-map for logic minimization.)



ABC		X = 0	1	0	1
000	S_0	S_1	S_2	0	0
110	S_1	S_3	S_2	0	0
001	S_2	S_1	S_4	0	0
111	S_3	S_5	S_2	0	0
011	S_4	S_1	S_6	0	0
101	S_5	S_5	S_2	1	0
010	S_6	S_1	S_6	0	1

(a) State table



Reason for assign “0”

- Reasons for assigning “0” as the starting state:
  - The clear input on Flip Flops can be used for initialization.
  - The clear input can also be used on a reset.
  - The alternative is error prone – using a combination of preset and clears to set a specific value can lead to implementation errors.
  - A good practice even when using FPGAs.

Guidlines

- Adjacency conditions from Guideline 1 and those from Guideline 2 that are required 2 or more times should be satisfied first.
- Example – Guideline 1 for the table S0, S2, S4, and S6 should be made adjacent as they all have S1 as the next state on a 0 input.
- S3 and S5 should have adjacent assignment.
- S4 and S6 should have adjacent assignment.

ABC		$X = 0$	1	0	1
000	S_0	S_1	S_2	0	0
110	S_1	S_3	S_2	0	0
001	S_2	S_1	S_4	0	0
111	S_3	S_5	S_2	0	0
011	S_4	S_1	S_6	0	0
101	S_5	S_5	S_2	1	0
010	S_6	S_1	S_6	0	1

(a) State table

Using guidelines

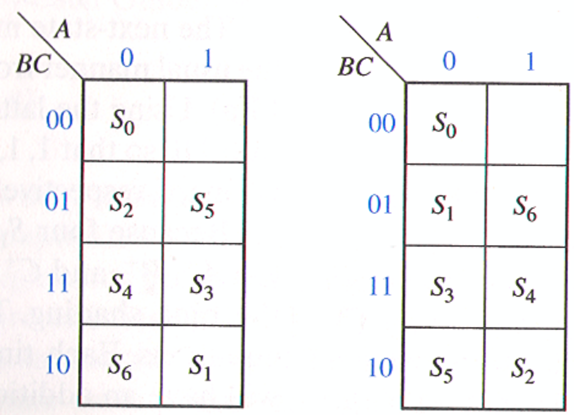
- From the state table find the following groupings:
  - 1. (S0,S1,S3,S5) (S3,S5) (S4,S6)   (S0,S2,S4,S6)
  - 2. (S1,S2) (S2,S3) (S1,S4) (S2,S5)2x (S1,S6)2x

ABC		*X* = 0	1	0	1
000	S_0	S_1	S_2	0	0
110	S_1	S_3	S_2	0	0
001	S_2	S_1	S_4	0	0
111	S_3	S_5	S_2	0	0
011	S_4	S_1	S_6	0	0
101	S_5	S_5	S_2	1	0
010	S_6	S_1	S_6	0	1

(a) State table

Two possible ways

- Two possible ways of satisfying the guidelines are:
  - 1. (S0,S1,S3,S5) (S3,S5) (S4,S6) (S0,S2,S4,S6)
  - 2. (S1,S2) (S2,S3) (S1,S4) (S2,S5)2x (S1,S6)2x



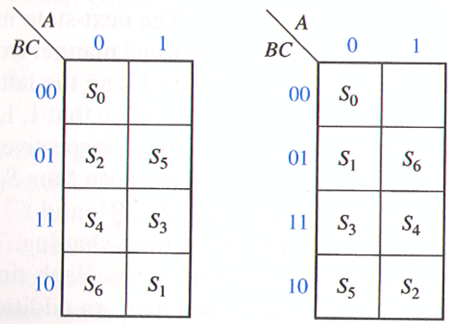
(b) Assignment maps

Next state maps

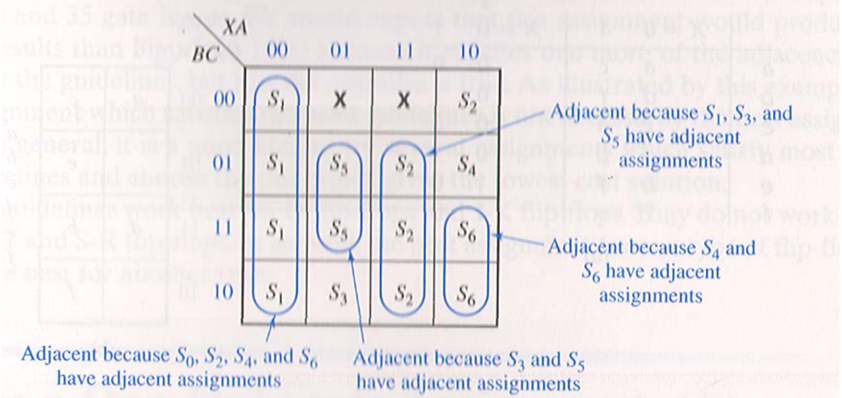
- Next state maps may help choose the better assignment.
- Look at the next state given current state and input and how this will simplify K-maps for logic.

ABC		X = 0	1	0	1
000	S_0	S_1	S_2	0	0
110	S_1	S_3	S_2	0	0
001	S_2	S_1	S_4	0	0
111	S_3	S_5	S_2	0	0
011	S_4	S_1	S_6	0	0
101	S_5	S_5	S_2	1	0
010	S_6	S_1	S_6	0	1

(a) State table



(b) Assignment maps



(a) Next-state maps for Figure 15-14

Choose an assignment

- Choose an assignment and implement in gates. Using the left assignment map get the next state map below with encoding.
- Map the encoding to K-maps

BC \ XA	00	01	11	10
00	S1 110	X	X	S2 001
01	S1 110	S5 101	S2 001	S4 011
11	S1 110	S5 101	S2 001	S6 010
10	S1 110	S3 111	S2 001	S6 010

Next States

BC \ XA	00	01	11	10
00	1	X	X	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

DA = X’

BC \ XA	00	01	11	10
00	1	X	X	0
01	1	0	0	1
11	1	0	0	1
10	1	1	0	1

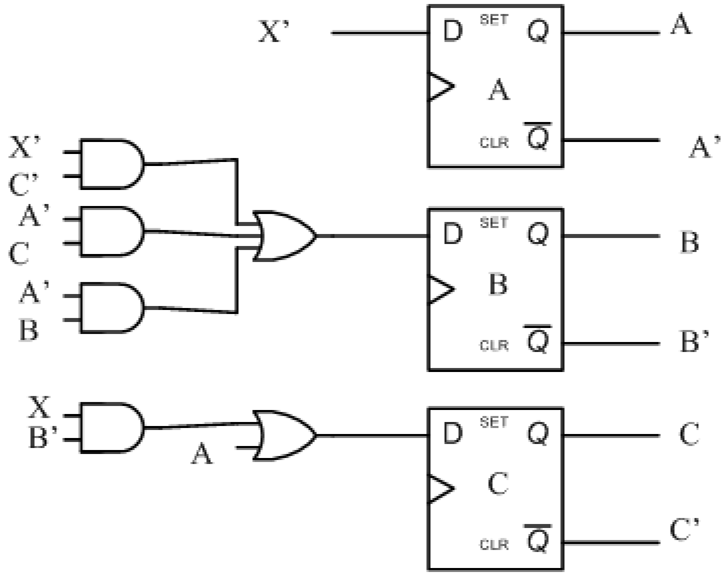
DB = X’C’ + A’C + A’B

BC \ XA	00	01	11	10
00	0	X	X	1
01	0	1	1	1
11	0	1	1	0
10	0	1	1	0

DC = A + XB’

Implement in gates

- Notes on implementation
  - All F/F outputs are used
  - 6 gates are needed for next state generation only 1 of which is 3 inputs.



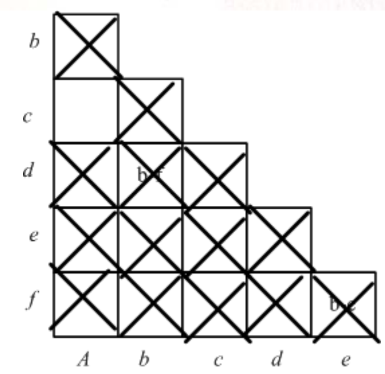
Another example

- Example 15-16 in text
  - Use guidelines
  - Next states (b,d) (c,f) (b,e) (a,c)
  - Next state of a state (a,c)2x (d,f) (d,b) (b,f) (c,e)
  - But is state table minimum?

FIGURE 15-16 State Table and Assignments

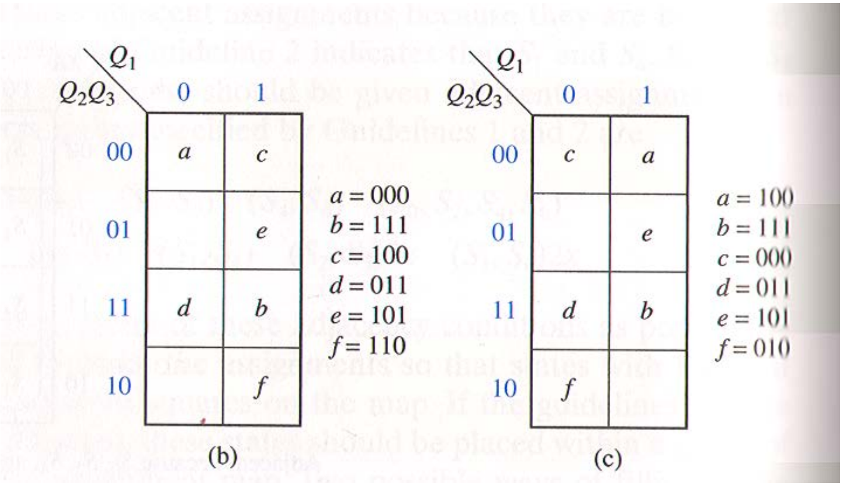
	$X = 0$	1	$X = 0$	1
a	a	c	0	0
b	d	f	0	1
c	c	a	0	0
d	d	b	0	1
e	b	f	1	0
f	c	e	1	0

(a)



Assignment map

- State table is not minimum but will continue
- The two assignment maps are



Transition table

- Resulting in a transition table of and equations of

$Q_1Q_2Q_3$	$Q_1{}^+Q_2{}^+Q_3{}^+$ $X = 0$	1	$X = 0$	1
1 0 0	100	000	0	0
1 1 1	011	010	0	1
0 0 0	000	100	0	0
0 1 1	011	111	0	1
1 0 1	111	010	1	0
0 1 0	000	101	1	0

(Figure 15-17) from the transition table. The D flip-flop input equations can be read directly from these maps:

$$D_1 = Q_1^+ = X'Q_1Q_2' + XQ_1'$$
$$D_2 = Q_2^+ = Q_3$$
$$D_3 = Q_3^+ = XQ_1'Q_2 + X'Q_3$$

and the output equation is

$$Z = XQ_2Q_3 + X'Q_2'Q_3 + X\,Q_2Q_3'$$

The cost of realizing these equations is 10 gates and 26 gate inputs.

Next state generation K-maps

- The K-maps for next state generation are

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	0	1
01	X	1	0	X
11	0	0	0	1
10	0	X	X	1

$Q_1^+ = D_1$

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	0	0	0
01	X	1	1	X
11	1	1	1	1
10	0	X	X	0

$Q_2^+ = D_2$

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	0	0	0
01	X	1	0	X
11	1	1	0	1
10	0	X	X	1

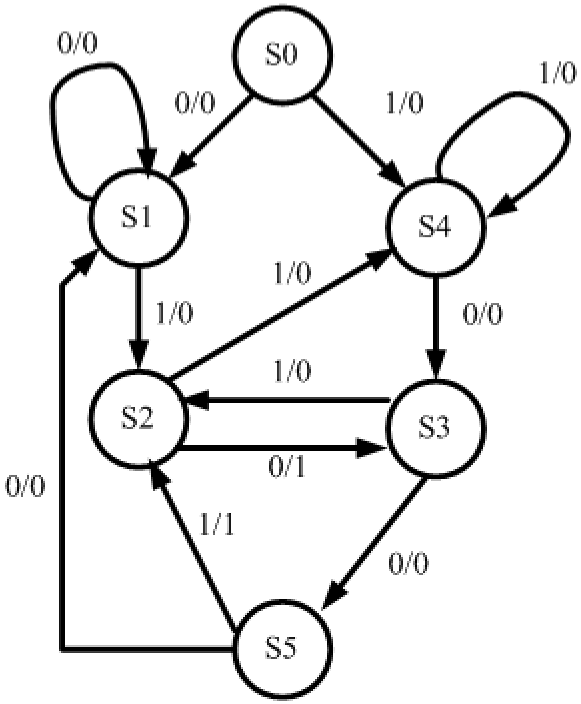
$Q_3^+ = D_3$

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	0	0	0
01	X	1	0	X
11	0	0	1	1
10	1	X	X	0

Z

Another example

- From our previous work.



	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

Use guidlines

- Same next state
  - (S0,S1,S5) (S2,S4) (S0,S2,S4) (S1,S3,S5)
- Next state pairs
  - (S1,S4) (S1,S2)2x (S3,S4)2x (S2,S5)

	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

The assignment map

- Choose S0 as the “0” state and then use guidelines
- A possible solution

BC \ A	0	1
00	S0	S2
01	S1	S4
11	S5	S3
10		

Next State Table

- Enter the state assignment onto the table
- Then generate K-maps and generate logic

Present State	Next State		Output	
ABC	X =0	X=1	X=0	X=1
S0 000	S1 001	S4 101	0	0
S1 001	S1 001	S2 100	0	0
S2 100	S3 111	S4 101	1	0
S3 111	S5 011	S2 100	0	0
S4 101	S3 111	S4 101	0	0
S5 011	S1 001	S2 100	0	1

The K maps

- Generate the K maps
  - Next State logic A (2 gates) B (1 gate) C (2 gates)

BC \ XA	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	0	0	1	1
10	x	x	x	x

A = X + AB'

BC \ XA	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	0	0
10	x	x	x	x

B = X'A

BC \ XA	00	01	11	10
00	1	1	1	1
01	1	1	1	0
11	1	1	0	0
10	x	x	x	x

C = X' + C' +AB'

K map for the output Z

- 3 gates for the output (2-3 input AND)
- (1 OR)
- Total logic count
  - 3 D F/Fs
  - 2 – 3 input AND gates
  - 3 – 2 input AND gates
  - 2 – 2 input OR gates
  - 1 – 3 input OR gate

BC \ XA	00	01	11	10
00	0	1	0	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

Z = X'AC' + XA'B

Lecture summary

- Have seen several examples of implementation from the statement of the problem (specification) to implementation.