Spring 2015 Week 6 Module 32

Digital Circuits and Systems

State Machines 3: State Assignment

*Shankar Balachandran**
Associate Professor, CSE Department
Indian Institute of Technology Madras

*Currently a Visiting Professor at IIT Bombay

State Assignment

- *State Assignment* is a binary encoding used to represent states of a sequential machine in its digital circuit implementation.
- In our designs so far we have assumed some state assignment without considering any alternatives.
- Two different assignments may result in vast differences in hardware.
- Appropriate choice of state assignment may result in lower cost and improved performance.

State Assignment Example

- Consider the following machine with 7 states.

Present State	Next State / Output	
	$x = 0$	$x = 1$
A	B/0	E/0
B	C/0	G/0
C	D/0	F/0
D	A/1	A/0
E	G/0	C/0
F	A/0	A/1
G	F/0	D/0

Let us use the following assignments to encode the states A through G

Present State	State Assignments	
	Assignment I	Assignment II
	$Q_1Q_2Q_3$	$Q_1Q_2Q_3$
A	000	001
B	001	000
C	011	010
D	010	011
E	101	100
F	110	101
G	111	110

Comparing Assignments I and II

- If the state machine was implemented using *JK* flip-flops …

Using Assignment I

$$J_1 = \overline{q_2}x + q_3x \qquad K_1 = \overline{q_3} + x$$
$$J_2 = q_3 \qquad K_2 = \overline{q_3}$$
$$J_3 = \overline{q_2} \qquad K_3 = q_2$$
$$z = \overline{q_1}q_2\overline{q_3}\,\overline{x} + q_1\overline{q_3}x$$

Using Assignment II

$$J_1 = \overline{q_3}\,x + \overline{q_2}\,x \qquad K_1 = q_3 + x$$
$$J_2 = q_1\overline{q_3} + \overline{q_1}q_3 \qquad K_2 = q_3 + q_1\overline{x} + \overline{q_1}\,x$$
$$J_3 = q_2 + \overline{q_1}\,\overline{x} \qquad K_3 = 1$$
$$z = q_1q_3x + q_2q_3\overline{x}$$

2-input Gate Count

NOT	4
AND	7
OR	3
Total	14

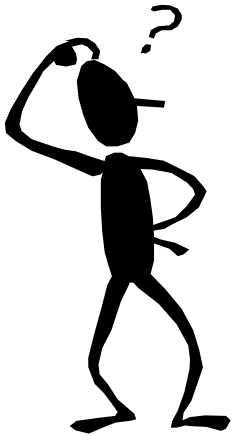
2-input Gate Count

NOT	4
AND	11
OR	7
Total	22

Assignment II requires almost twice as many gates as Assignment I !

Let's try all possible assignments ....

How many



s	*n*	*# Assignments*	
2	1	2	☺
3	2	24	😐
6	3	20,160	☹
8	3	40,320	
12	4	871,782,912,0000	
16	4	209,227,898,880,00	

State Assignment Problem

- It is not practical to go through all possible assignments to determine the optimal assignment for machines with more than two or four states.
- Unfortunately, there is no simple technique to determine optimal state assignment.
- Reasonably good assignments can be obtained by using a few guidelines.

State Assignment Guidelines

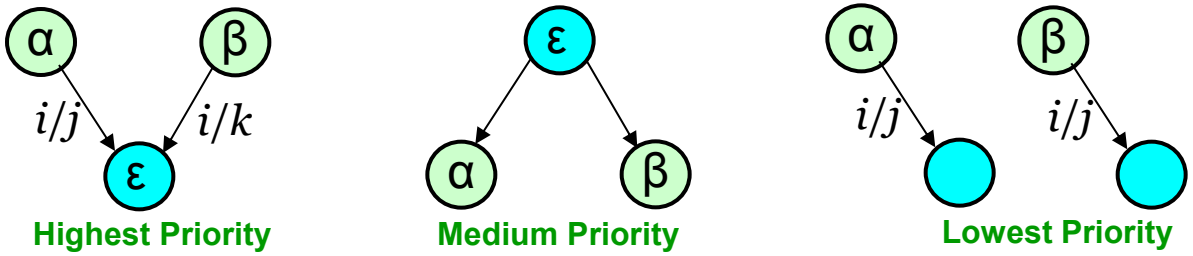
- In order to reduce the complexity of logic equations, the state assignment should be such that it forces large groupings of the logic 1s in the binary transition table, or transition K-map.
  - *The larger the group of 1s, ⇨ simpler excitation and output equations ⇨ reduced complexity of combinational circuit.*

- Minimum-Bit-Change Heuristic:
  - Minimize the number of bit changes for all state transitions.

- Guidelines based on Next State and Input/Outputs:
  - ***Highest priority***: States with the same next state for a given input should be given adjacent assignments in the state map.
  - ***Medium priority***: Next states of the same state should be given adjacent assignments in the state map.
  - ***Lowest priority***: States with the same output for a given input should be given adjacent assignments in the state map.

Use of State Assignment Guidelines:

1. Write down all sets of states which should be given adjacent assignments.



2. Compare the adjacencies of states for different assignments by plotting the states on *state assignment K-maps* with each cell representing the state variable combination assigned to one state of the circuit.
3. If all adjacencies suggested by the top two priorities are not satisfied due to some conflicting requirements, then resolve them in favor of conditions from highest priority and the adjacency conditions which are required two or more times.
4. When guidelines require that 3 or 4 states be adjacent, these states should be placed within a group of 4 adjacent cells on the assignment K-map.

Example:

Consider the following state machine having 4 states

Present State	Next State / Out	
	$x = 1$	$x = 0$
A	D/0	C/0
B	A/0	C/0
C	D/0	B/0
D	B/1	A/1

Adjacencies from highest priority guideline:

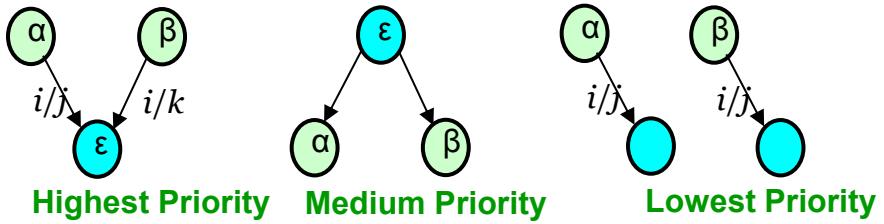
for x=0: {A, B}
for x=1: {A, C}

Adjacencies from medium priority guideline:

for state A: {C, D}
for state B: {A, C}
for state C: {B, D}
for state D: {A, B}

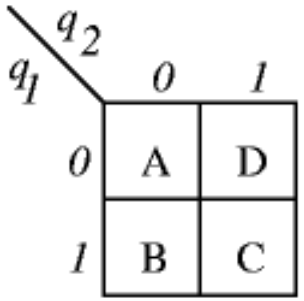
Adjacencies from lowest priority guideline:

for 0/0: {A, B, C}
for 1/0: {A, B, C}

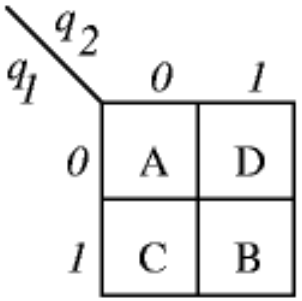


- Compare the adjacencies of states for the different assignments by plotting the states on state (assignment) maps. In general it is a good idea to assign the reset state (state A in this example) to state map cell 0).

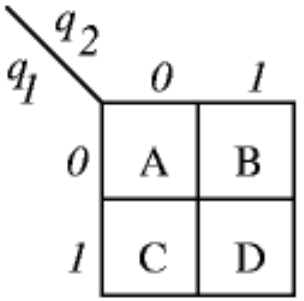
Adjacencies:
{A,B}; {A,C}
{C,D}; {A,C}; {B,D}; {A,B}
{A,B,C}; {A,B,C}



Assignment 1



Assignment 2



Assignment 3

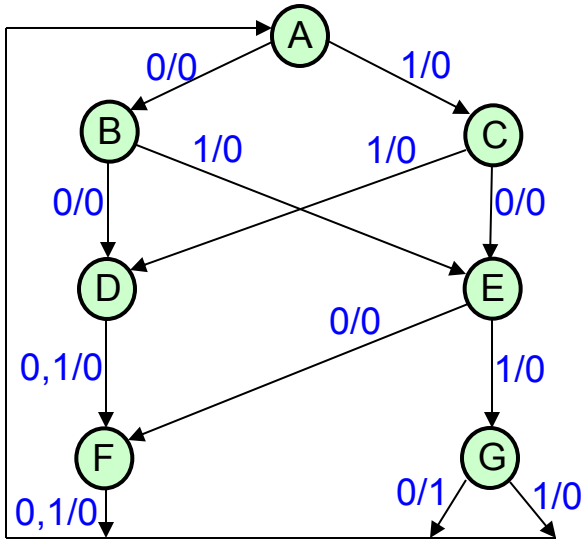
- It can be seen that assignment 3 fulfills most of the adjacencies and hence should produce the best results for this example.
- The state encoding (q_1 q_2) is

  A:00 C:10

  B:01 D:11

Example:

Encode the states using the state assignment guidelines for the sequential circuit described by the following state machine



Adjacencies from highest priority guideline:

for x=0: {D, E}; {F, G}

for x=1: {F, G}

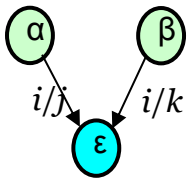
Adjacencies from medium priority guideline:

for state A: {B, C}

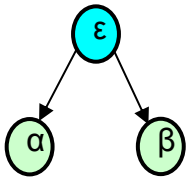
for states B and C: {D, E} 2x

for state E: {F, G}

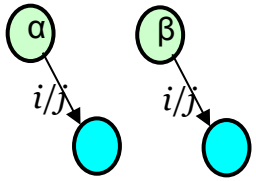
(*note: "2x" next to an adjacency => it exists twice on the list*)



Highest Priority



Medium Priority



Lowest Priority

Adjacencies from lowest priority guideline:

for 0/0: {A, B, C, D, E, F}

for 1/0: {A, B, C, D, E, F, G}

- Create state maps to satisfy the adjacencies

Adjacencies:
{D,E}; {F,G}
{A,B}; {D,E}2x; {F,G}
{A,B,C,D,E,F}; {A,B,C,D,E,F,G}

Assignment I

q_1 \ q_2q_3	00	01	11	10
0	A	B	D	F
1		C	E	G

A = 000
B = 001
C = 101
D = 011
E = 111
F = 010
G = 110

Assignment II

q_1 \ q_2q_3	00	01	11	10
0	A	B	D	
1	F	C	E	G

A = 000
B = 001
C = 101
D = 011
E = 111
F = 100
G = 110

Both assignments satisfy all of the high- and medium-priority guidelines, as well as most of the lowest-priority ones.

Do it Yourself

Find a state assignment for the sequential machine described below:

Present State	Next State / Output	
	$x = 1$	$x = 0$
A	C/0	B/0
B	C/0	D/0
C	E/0	B/0
D	C/0	F/0
E	G/0	B/0
F	C/0	F/1
G	G/1	B/0

Unused States

- A state machine has *unused states* when the number of states with *n* flip-flops, 2^n, is greater than the number of states required, *s*. There are two approaches to deal with unused states, depending on the application requirements:

1. Minimal Cost: This approach assumes that the machine will never enter an unused state. Therefore, next state entries of unused states can be marked as "don't-cares". In most cases, this simplifies the excitation logic.

2. Minimal Risk: This approach assumes that it is possible for the machine somehow to get into one of the unused ("illegal") states. Therefore, all unused states should have explicit next state entries for any input combination so that they always reach some other "safe" state.

State Assignment using One Hot Encoding

- So far, our goal has been *dense* encodings: state encodings in as few bits as possible.

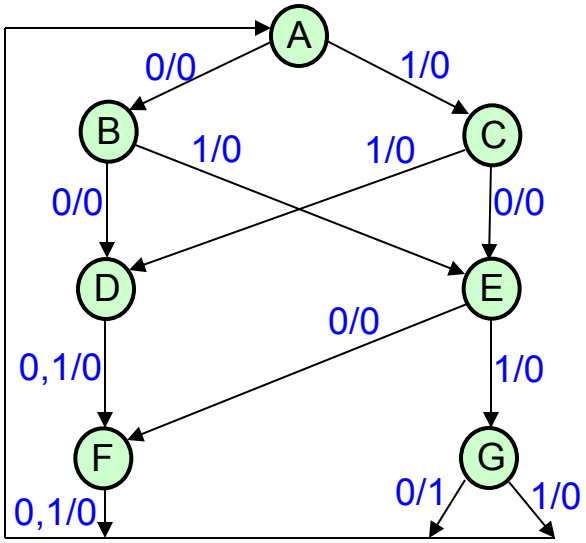
- *One hot encoding* is an alternative approach in which additional flip-flops are introduced in the hope of reducing the next-state and output logic complexity.

  ⇨ improved performance

- For a machine with *n* states, *one hot* encoding uses exactly *n* flip-flops.

- One hot ⇨ each state is represented by an *n*-bit binary code in which *exactly 1 bit is asserted*.

Example: One hot encoding



Present State	Next State / Output	
	$x = 0$	$x = 1$
A	B/0	C/0
B	D/0	E/0
C	E/0	D/0
D	F/0	F/0
E	F/0	G/0
F	A/0	A/0
G	A/1	A/0

State Assignments	
Dense	One hot
$q_1q_2q_3$	$q_1q_2q_3q_4q_5q_6q_7$
000	1000000
001	0100000
101	0000100
011	0001000
111	0000001
010	0010000
110	0000010

Implementation using *D* flip-flops:

with Dense Encoding

$$D_1 = \overline{q_2}(x \oplus q_1) + q_1 q_2 q_3 x$$
$$D_2 = q_3$$
$$D_3 = \overline{q_2}$$
$$z = q_1 \overline{q_3}\,\overline{x}$$

with One Hot Encoding

$$D_1 = q_3 + q_6 \qquad D_5 = q_1 x$$
$$D_2 = q_1 \overline{x} \qquad D_6 = q_7 x$$
$$D_3 = q_4 + q_7 \overline{x} \qquad D_7 = q_5 \overline{x} + q_2 x$$
$$D_4 = q_2 \overline{x} + q_5 x \qquad z = q_6 \overline{x}$$

End of Week 6: Module 32

Thank You