

EE2000 Logic Circuit Design

Sequential Logic Circuit Design

Outline

6.1 Concepts of States

- Mealy Machines
- Moore Machines

6.2 State Reduction

- Equivalent State
- Direct Inspection
- Partitioning
- Implication Table

6.3 FF Excitation Table

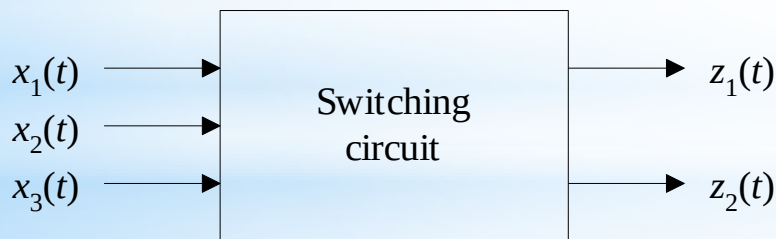
6.4 Circuit Analysis

6.5 Design Example

Sequential and combinational circuits

Switching system: a combinational circuit which transforms the input excitation to its corresponding output, the input and output are all function of time. The output is dependent only on the present input conditions.

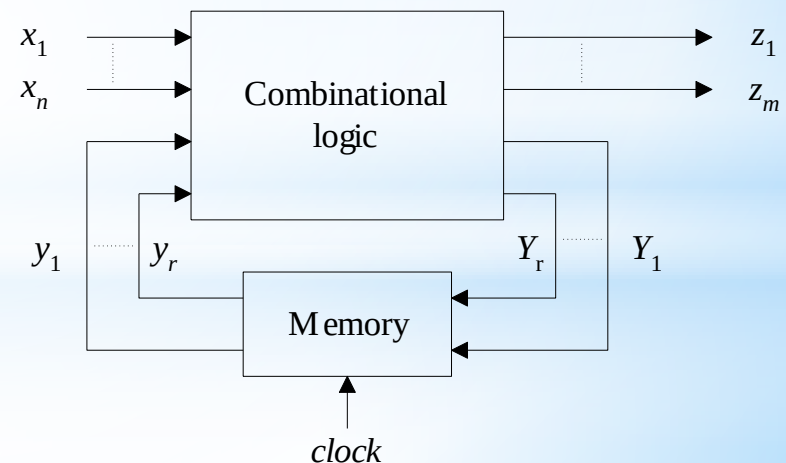
combinational circuit



$$z_i(t) = f(x_i(t))$$

Sequential circuit: the system outputs are dependent not only on the present input conditions but also on the past history of the system. The past history is usually regarded **states** which obviously require memory devices (e.g. FFs) to remember their states.

sequential circuit



$$z_i(t) = f(x_i(-\infty, t))$$

Synchronous and asynchronous systems

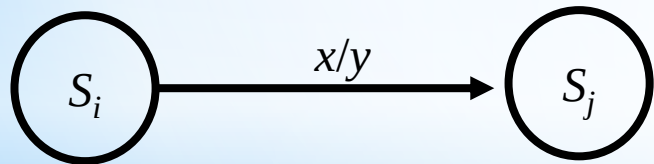
- ***Synchronous*** sequential systems in which the change of state takes place at discrete instants of time defined by a synchronizing input called the *clock* (*common clock signal for all FFs*).
- ***Asynchronous*** sequential systems, the states can change at any time as a function of input changes, there is no external synchronization (no common clock signal).

6.1 Concept of States

- Due to the repeated nature of inputs, the input time functions can be group into classes in such a way that all time functions have the same effect on the output at time t are in the same class.
- Practically, the number of classes is finite.
- These classes are represented by the auxiliary variables s_i called states.
- States summarize the effect of past inputs.

Mealy machine

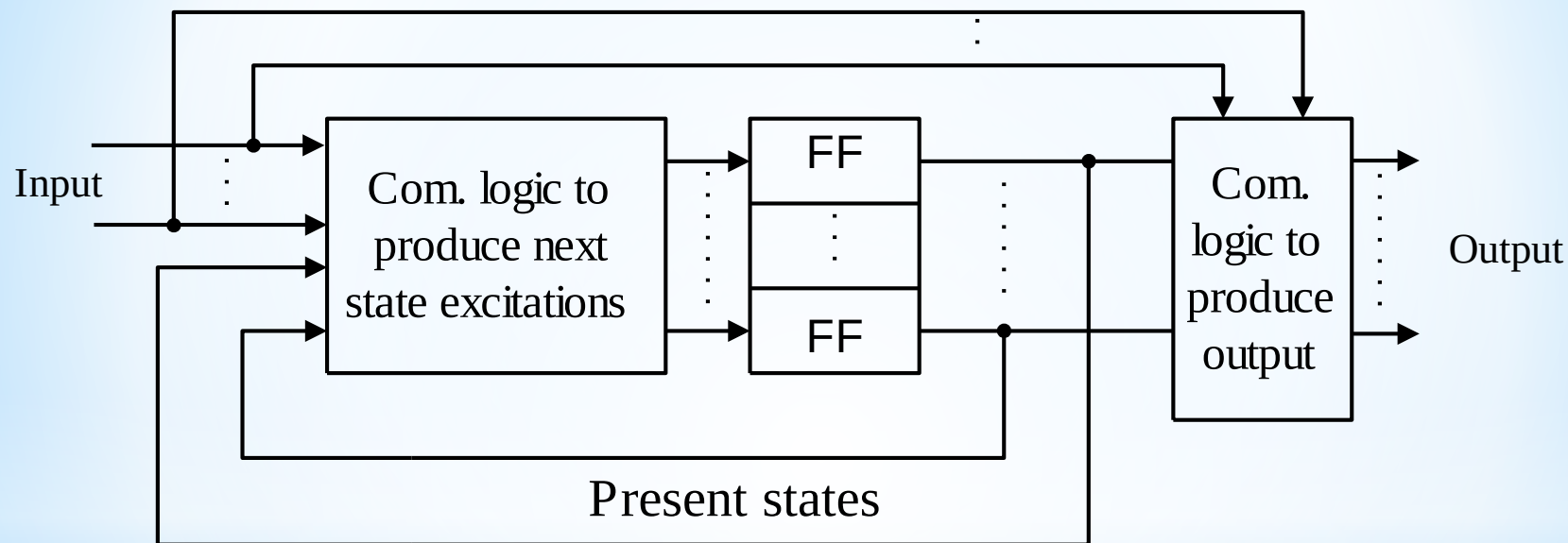
- State machine is used to describe the behavior of a sequential circuit.
- The state machine is illustrated using state diagram.
- A state diagram consists of circles which present the states; and lines connecting the circles that represent the transition of states.



S_i : present state S_j : next state
 x : input y : output

- The above state transition means that for a present state S_i with an input condition of x , the state will change to S_j and produce a output of y .

Mealy machine

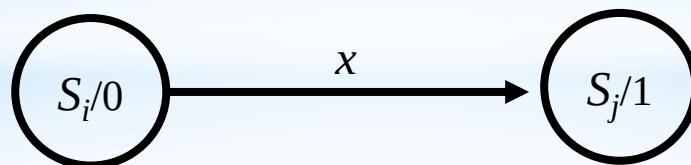


Structure of a synchronous sequential circuit using Mealy machine

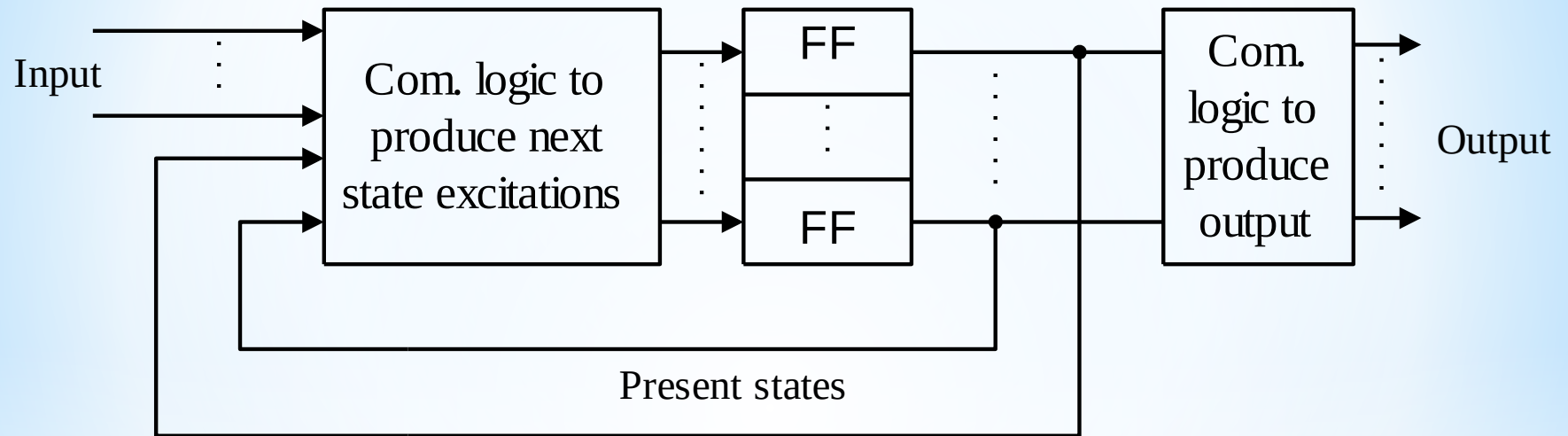
For Mealy machine, the output(s) depends on both present state(s) and input(s).

Moore machine

- For Moore machine, the output at time t depends only on the present state.
- The influence of the input on the output is only through the state.
- The state diagram is then different from the one used by the Mealy machine.



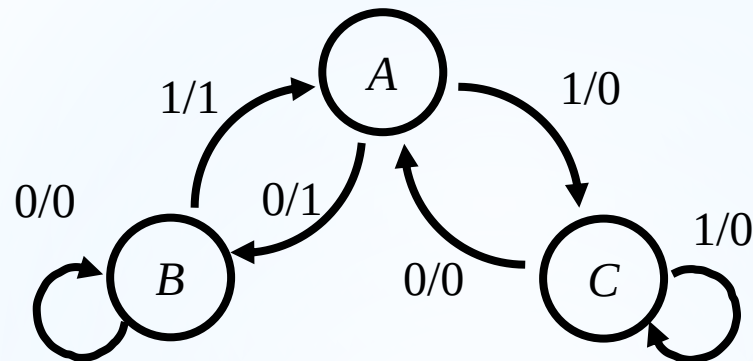
Moore machine



Structure of a synchronous sequential circuit using Moore machine

For Moore machine, the output(s) depends on present state(s) only.

Mealy Machine (Example)



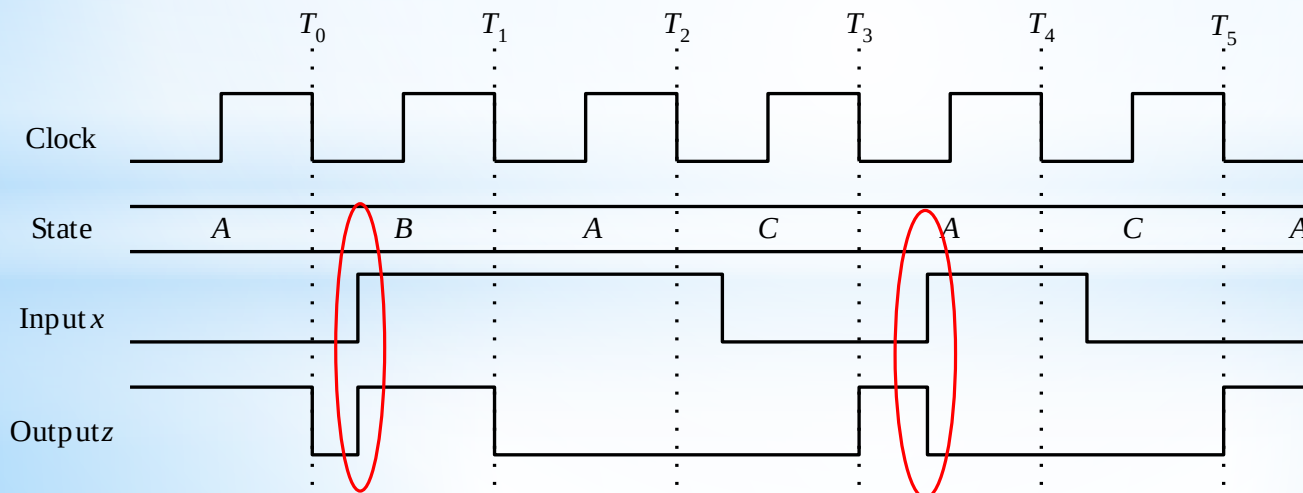
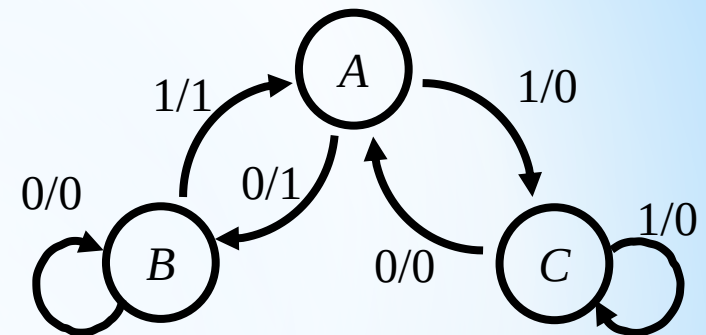
PS	NS/output	
	Input = 0	Input = 1
A	B / 1	C / 0
B	B / 0	A / 1
C	A / 0	C / 0

State table

Mealy Machine - Example (Cont'd)

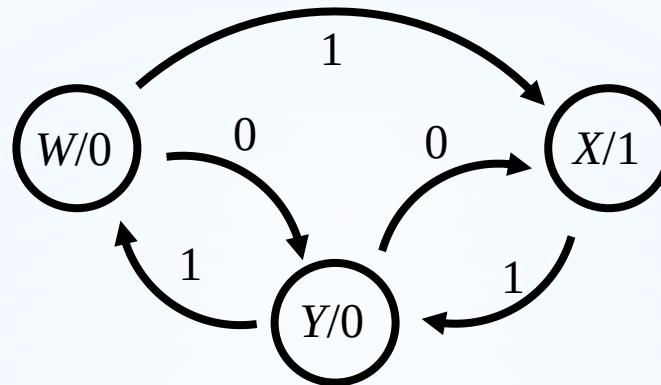
Circuit behavior with present state A:

Time:	0	1	2	3	4	5	
Present state:	A	B	A	C	A	C	A
Input:	0	1	1	0	1	0	
Output:	1	1	0	0	0	0	
Next state:	B	A	C	A	C	A	



Output changes when input changes (other than triggered by clock edges) !!

Moore Machine (Example)



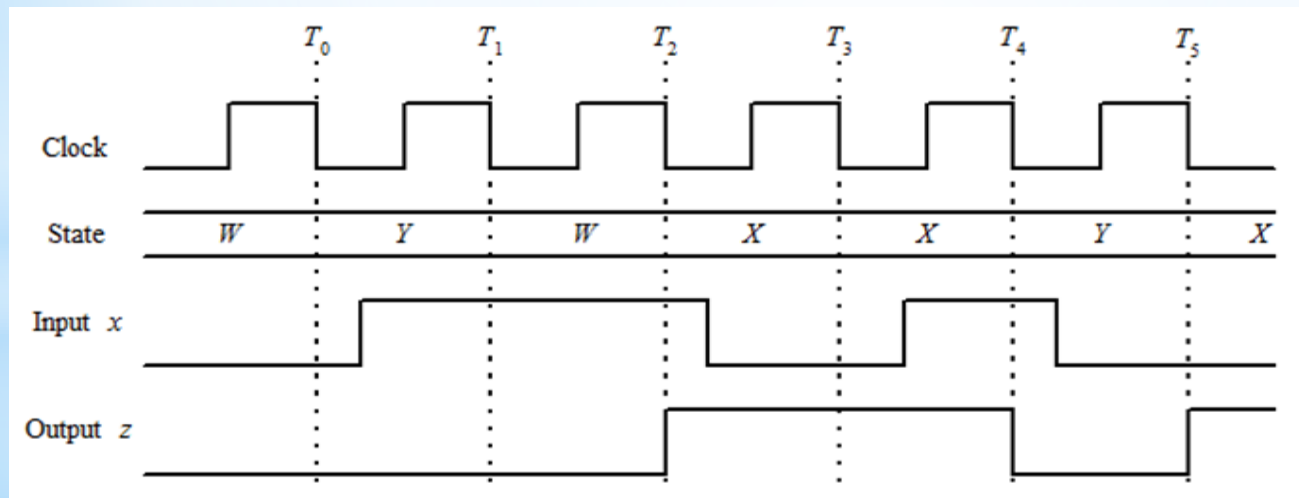
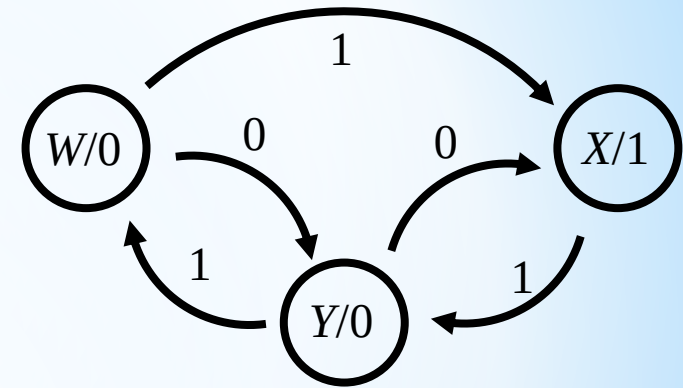
PS	NS		Output
	Input = 0	Input = 1	
W	Y	X	0
X	X	Y	1
Y	X	W	0

State table

Moore Machine - Example (Cont'd)

Circuit behavior with present state A:

Time:	0	1	2	3	4	5	
Present state:	W	Y	W	X	X	Y	X
Input:	0	1	1	0	1	0	
Output:	0	0	0	1	1	0	
Next state:	Y	W	X	X	Y	X	



Output only changes at the corresponding triggering edges !!

6.2 State Reduction

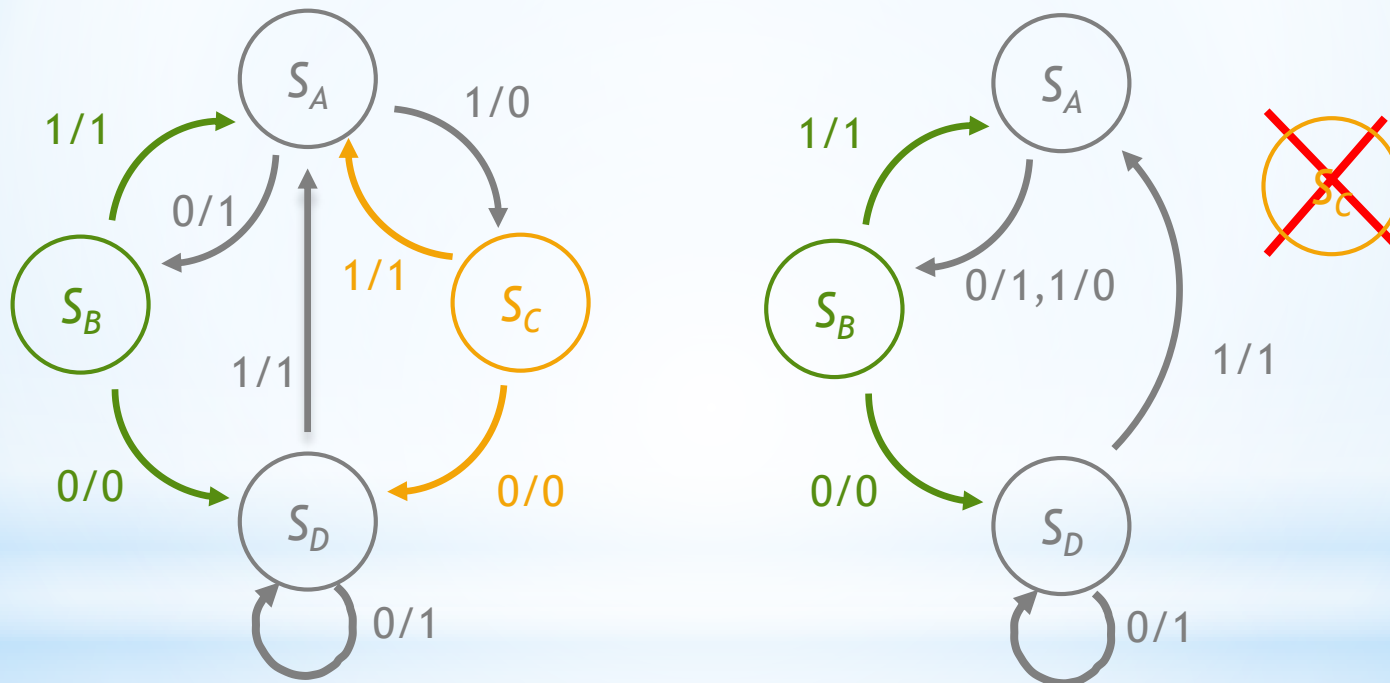
- For combination logic circuits, minimizing the output functions means:
 - reducing the number of logic gates
 - reducing the cost and space of the circuit
- For sequential logic circuits, minimizing the state table means
 - reducing the redundant states (e.g. number of Flip-Flops)
 - Reducing the cost and space

Principle of Equivalent States

- Two conditions:
 - *For the same input conditions, the output must be the same.*
 - *For the same input conditions, they go to the same next state.*
- For all present states that satisfy the above two criteria, these states are **equivalent**.
- If the a redundant state is found, then it can be **eliminated**.

Finding Equivalent States

Are these two state diagram equivalent?

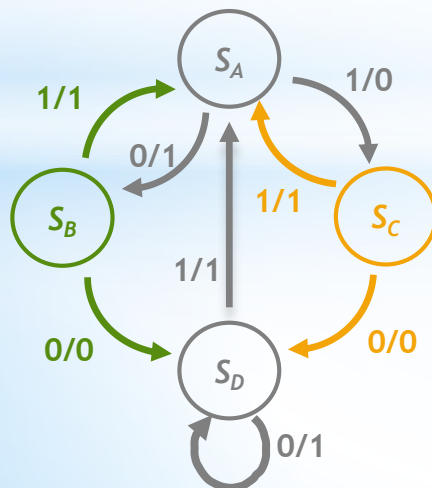


Not easy to find them from state diagram !!

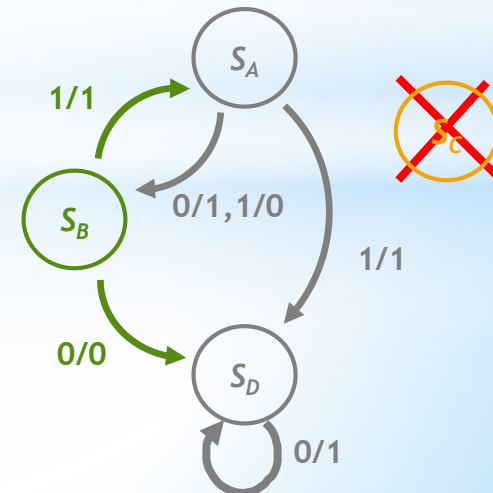
1st Method - Direct Inspection

Compare their state tables

	NS/Z	
PS	X = 0	X = 1
S_A	$S_B/1$	$S_C/0$
S_B	$S_D/0$	$S_A/1$
S_C	$S_D/0$	$S_A/1$
S_D	$S_D/1$	$S_A/1$



	NS/Z	
PS	X = 0	X = 1
S_A	$S_B/1$	$S_B/0$
S_B	$S_D/0$	$S_A/1$
~~S_C~~	~~$S_D/0$~~	~~$S_A/1$~~
S_D	$S_D/1$	$S_A/1$



More example - Direct Inspection

	NS/Z	
PS	X = 0	X = 1
A	B/0	C/1
B	C/0	A/1
C	D/1	B/0
D	C/0	A/1
E	D/0	C/1

B = D

	NS/Z	
PS	X = 0	X = 1
A	B/0	C/1
B	C/0	A/1
C	B/1	B/0
E	B/0	C/1

A = E

	NS/Z	
PS	X = 0	X = 1
A	B/0	C/1
B	C/0	A/1
C	B/1	B/0

D & E - eliminated

More example - Direct Inspection

	NS/Z	
PS	X = 0	X = 1
A	B/0	C/1
B	B/0	A/1
C	D/1	B/0
D	D/0	A/1

Case 1

	NS/Z	
PS	X = 0	X = 1
A	B/0	C/1
B	D/0	A/1
C	B/1	B/0
D	B/0	A/1

Case 2

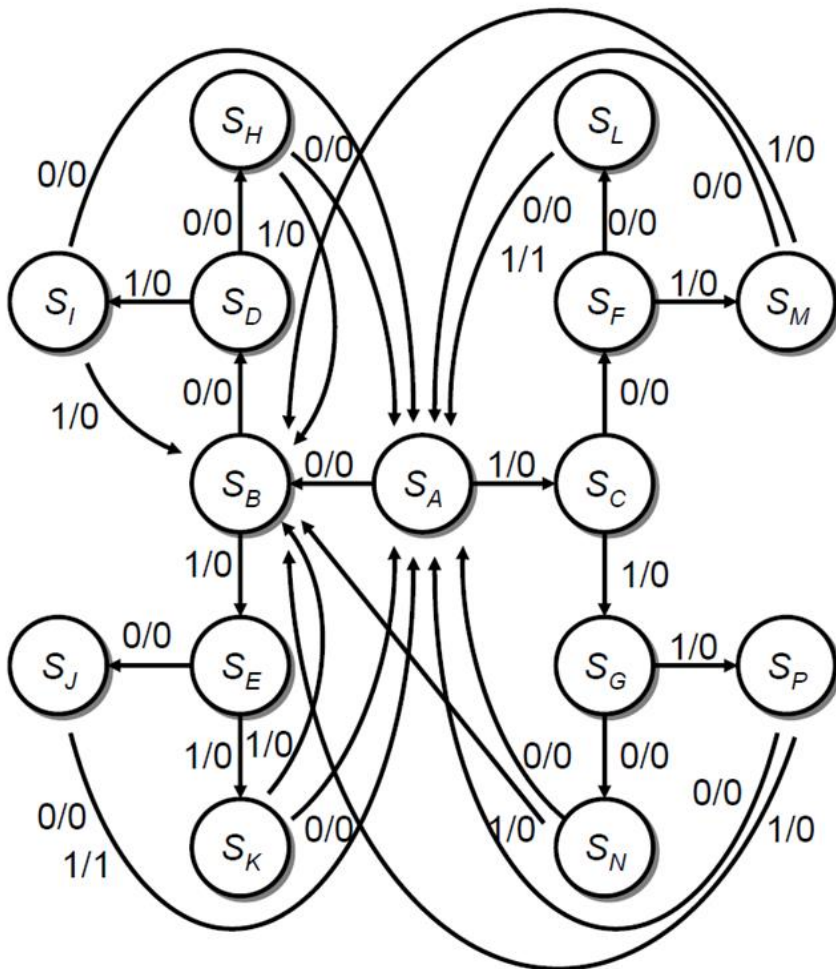
	NS/Z	
PS	X = 0	X = 1
A	B/0	C/1
B	B/0	A/1
C	B/1	B/0

For Case I: the NS of PS B remains unchange for X=0
 the NS of PS D remains unchange for X=0
THUS states B & D are equivalent

For Case II: the NS of PS B is D for X=0
 the NS of PS D is B for X=0
THUS states B & D are also equivalent 19

More example - Direct Inspection

Derive state table from state diagram



Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_I / 0$
S_E	$S_J / 0$	$S_K / 0$
S_F	$S_L / 0$	$S_M / 0$
S_G	$S_N / 0$	$S_P / 0$
S_H	$S_A / 0$	$S_B / 0$
S_I	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$
S_K	$S_A / 0$	$S_B / 0$
S_L	$S_A / 0$	$S_A / 1$
S_M	$S_A / 0$	$S_B / 0$
S_N	$S_A / 0$	$S_B / 0$
S_P	$S_A / 0$	$S_B / 0$

More example (Cont'd)

Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_I / 0$
S_E	$S_J / 0$	$S_K / 0$
S_F	$S_L / 0$	$S_M / 0$
S_G	$S_N / 0$	$S_P / 0$
S_H	$S_A / 0$	$S_B / 0$
S_I	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$
S_K	$S_A / 0$	$S_B / 0$
S_L	$S_A / 0$	$S_A / 1$
S_M	$S_A / 0$	$S_B / 0$
S_N	$S_A / 0$	$S_B / 0$
S_P	$S_A / 0$	$S_B / 0$

Key: Find states that have same next state and outputs

$S_A / 0 \quad S_B / 0$

$S_A / 0 \quad S_A / 1$

Result: $S_H = S_I = S_K = S_M = S_N = S_P$

Result: $S_J = S_L$

More example (Cont'd)

Elimination of redundant states

Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_I / 0$
S_E	$S_J / 0$	$S_K / 0$
S_F	$S_L / 0$	$S_M / 0$
S_G	$S_N / 0$	$S_O / 0$
S_H	$S_A / 0$	$S_B / 0$
~~S_I~~	~~$S_A / 0$~~	~~$S_B / 0$~~
S_J	$S_A / 0$	$S_A / 1$
~~S_K~~	~~$S_A / 0$~~	~~$S_B / 0$~~
~~S_L~~	~~$S_A / 0$~~	~~$S_A / 1$~~
~~S_M~~	~~$S_A / 0$~~	~~$S_B / 0$~~
~~S_N~~	~~$S_A / 0$~~	~~$S_B / 0$~~
~~S_O~~	~~$S_A / 0$~~	~~$S_B / 0$~~



Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_H / 0$
S_E	$S_J / 0$	$S_H / 0$
S_F	$S_J / 0$	$S_H / 0$
S_G	$S_H / 0$	$S_H / 0$
S_H	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$

More example (Cont'd)

Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_H / 0$
S_E	$S_J / 0$	$S_H / 0$
S_F	$S_J / 0$	$S_H / 0$
S_G	$S_H / 0$	$S_H / 0$
S_H	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$

$$S_E = S_F$$

Repeat the elimination process

$S_H / 0 \quad S_H / 0$

Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_H / 0$
S_E	$S_J / 0$	$S_H / 0$
S_F	$S_J / 0$	$S_H / 0$
S_G	$S_H / 0$	$S_H / 0$
S_H	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$

More example (Cont'd)

The final state table:

Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_F / 0$	$S_G / 0$
S_D	$S_H / 0$	$S_I / 0$
S_E	$S_J / 0$	$S_K / 0$
S_F	$S_L / 0$	$S_M / 0$
S_G	$S_N / 0$	$S_P / 0$
S_H	$S_A / 0$	$S_B / 0$
S_I	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$
S_K	$S_A / 0$	$S_B / 0$
S_L	$S_A / 0$	$S_A / 1$
S_M	$S_A / 0$	$S_B / 0$
S_N	$S_A / 0$	$S_B / 0$
S_P	$S_A / 0$	$S_B / 0$



Present State (PS)	Input X	
	0	1
S_A	$S_B / 0$	$S_C / 0$
S_B	$S_D / 0$	$S_E / 0$
S_C	$S_E / 0$	$S_D / 0$
S_D	$S_H / 0$	$S_H / 0$
S_E	$S_J / 0$	$S_H / 0$
S_H	$S_A / 0$	$S_B / 0$
S_J	$S_A / 0$	$S_A / 1$

The final state diagram:



2nd Method - Partitioning

Assumption:

If states are in the same partition, they are equivalent.

If the next states are in different partitions, the corresponding present states are NOT equivalent.

The present state partition is divided until all next states are in the same partition.

2nd Method - Partitioning

Step 1:

For the same input conditions, group the states with the same output(s) into the same partition

Step 2:

Further partition the states into smaller subgroups if their next states belong to *different* groups

Step 3:

Repeat step 2 until no more states can be partitioned

Finally, the states in the same partition are equivalent.

Partitioning - example

	NS/output	
PS	$x = 0$	$x = 1$
<i>A</i>	<i>C/1</i>	<i>B/0</i>
<i>B</i>	<i>C/1</i>	<i>E/0</i>
<i>C</i>	<i>B/1</i>	<i>E/0</i>
<i>D</i>	<i>D/0</i>	<i>B/1</i>
<i>E</i>	<i>E/0</i>	<i>A/1</i>



	NS/output	
PS	$x = 0$	$x = 1$
<i>A</i>	<i>B/1</i>	<i>B/0</i>
<i>B</i>	<i>B/1</i>	<i>E/0</i>
<i>D</i>	<i>D/0</i>	<i>B/1</i>
<i>E</i>	<i>E/0</i>	<i>A/1</i>

How ?

Partitioning - example (Cont'd)

	Partition blocks		Action	
Partition P_0	$(ABCDE)$			
Output for $x = 0$	11100		Separate (ABC) and (DE) according from the output	
Output for $x = 1$	00011			
Partition P_1	(ABC)	(DE)		
Next state for $x = 0$	CCB	DE	Separate (A) and (BC)	
Next state for $x = 1$	BEE	BA		
Partition P_2	(A)	(BC)	(DE)	
Next state for $x = 0$	C	CB	DE	Separate (D) and (E)
Next state for $x = 1$	B	EE	BA	
Partition P_3	(A)	(BC)	(D)	(E)
Next state for $x = 0$	C	CB	D	E
Next state for $x = 1$	B	EE	B	A
Partion $P_3 = P_4$	(A)	(BC)	(D)	(E)

⇒ State **B** and **C** are equivalent.

Implication Table

Assumption: If the 2 present states are the same, it implies that their next states must be the same and with the same outputs.

Step 1: Draw the implication table

Step 2: Output partitioning - implied pairs with different output

Step 3: Find the implied pairs

Step 4: Cross out the implied pairs if they are not equivalent

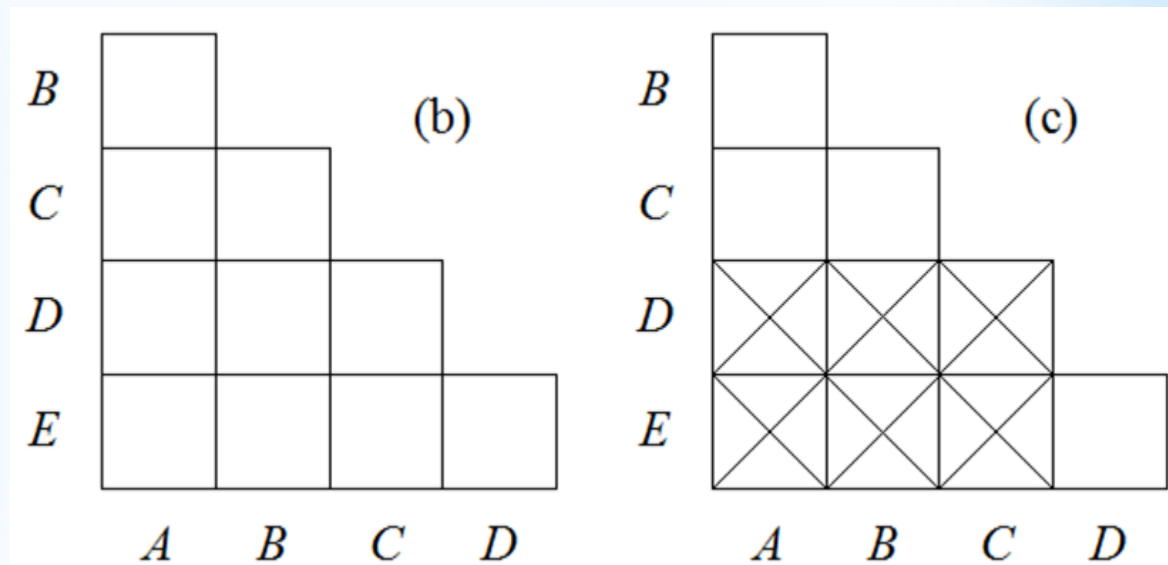
Step 5: Iteratively repeat Step 4 until no more eliminations

Step 6: The remaining squares represent equivalent states

Implication Table - Example

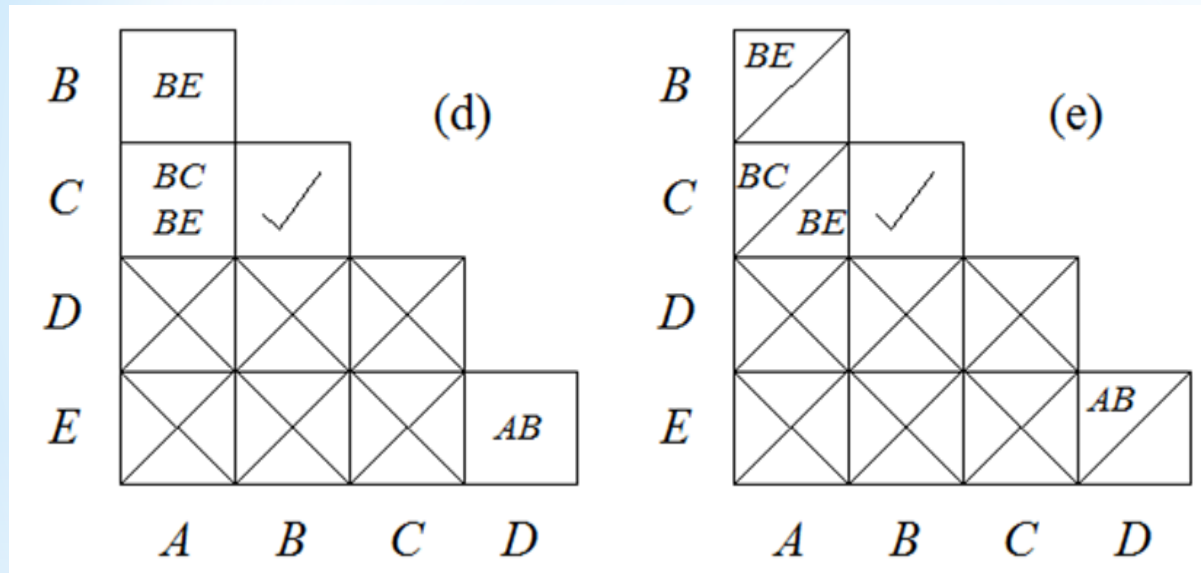
	<i>x</i>	
	0	1
<i>A</i>	<i>C</i> /1	<i>B</i> /0
<i>B</i>	<i>C</i> /1	<i>E</i> /0
<i>C</i>	<i>B</i> /1	<i>E</i> /0
<i>D</i>	<i>D</i> /0	<i>B</i> /1
<i>E</i>	<i>E</i> /0	<i>A</i> /1

(a)



The implication table for a five-state circuit.
 (a) State table. (b) Implication table. (c) Output partitioning.

Implication Table - Example

(f) $B = C$

	x	
	0	1
A	$B/1$	$B/0$
B	$B/1$	$E/0$
D	$D/0$	$B/1$
E	$E/0$	$A/1$

The implication table for a five-state circuit (Cont'd).

(d) Implied pairs. (e) Completed table.

(f) State equivalence and reduced state table.

More example - Implication Table

	<i>x</i>	
	0	1
<i>A</i>	<i>E</i> /0	<i>D</i> /0
<i>B</i>	<i>A</i> /1	<i>F</i> /0
<i>C</i>	<i>C</i> /0	<i>A</i> /1
<i>D</i>	<i>B</i> /0	<i>A</i> /0
<i>E</i>	<i>D</i> /1	<i>C</i> /0
<i>F</i>	<i>C</i> /0	<i>D</i> /1
<i>G</i>	<i>H</i> /1	<i>G</i> /1
<i>H</i>	<i>C</i> /1	<i>B</i> /1

(a)

<i>B</i>							
<i>C</i>							
<i>D</i>	<i>BE</i>						
<i>E</i>		<i>AD</i> <i>CF</i>					
<i>F</i>			<i>AD</i>				
<i>G</i>							
<i>H</i>							<i>CH</i> <i>BG</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>

(b)

The implication table for an eight-state circuit. (a) State table.
(b) Implication table

More example - Implication Table

(c)

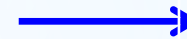
$$A=D \Rightarrow B=E$$

$$C=F \Rightarrow A=D$$

$$B=E \Rightarrow (A=D) \& (C=F)$$

$$\therefore A=D \Rightarrow B=E \Rightarrow C=F$$

	x	
	0	1
A	E/0	D/0
B	A/1	F/0
C	C/0	A/1
D	B/0	A/0
E	D/1	C/0
F	C/0	D/1
G	H/1	G/1
H	C/1	B/1



	x	
	0	1
A	B/0	A/0
B	A/1	C/0
C	C/0	A/1
G	H/1	G/1
H	C/1	B/1

(d)

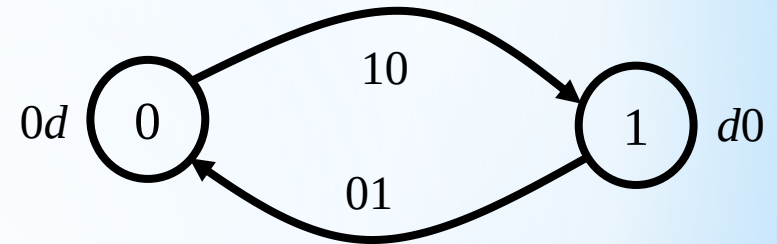
	x	
	0	1
A	B/0	A/0
B	A/1	C/0
C	C/0	A/1
G	H/1	G/1
H	C/1	B/1

The implication table for an eight-state circuit - (cont'd).
 (c) State equivalence. (d) Reduced state table.

6.3 FF Excitation Table

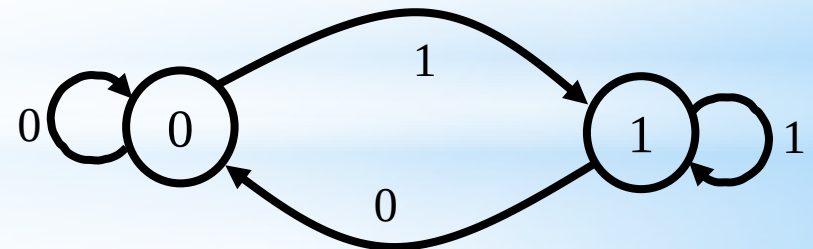
SR-FF

SR	Q_{n+1}		Q_n	Q_{n+1}	SR
	$Q_n=0$	$Q_n=1$			
00	0	1	0	→ 0	0d
01	0	0	0	→ 1	10
10	1	1	1	→ 0	01
11	-	-	1	→ 1	d0



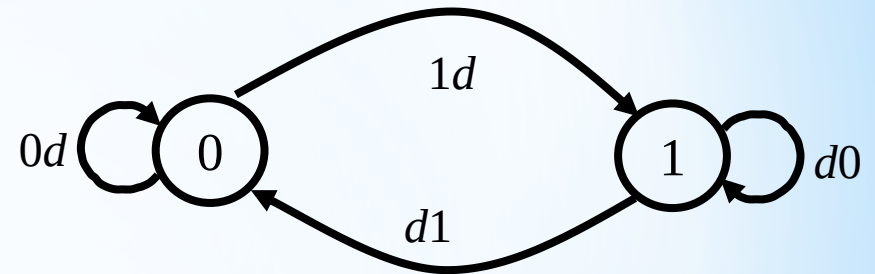
D-FF

D	Q_{n+1}		Q_n	Q_{n+1}	D
	$Q_n=0$	$Q_n=1$			
0	0	0	0	→ 0	0
1	0	1	0	→ 1	1
0	1	1	1	→ 0	0
1	1	1	1	→ 1	1



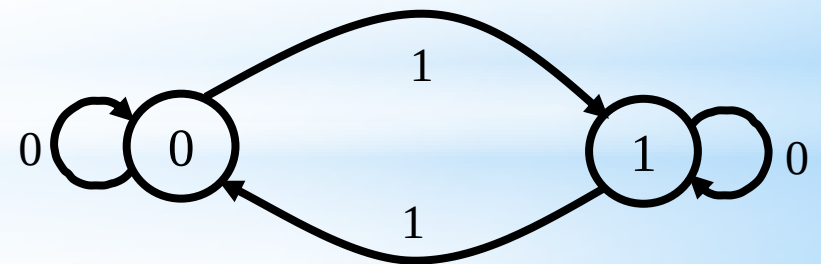
JK-FF

JK	Q_{n+1}		Q_n	Q_{n+1}	JK
	$Q_n=0$	$Q_n=1$			
00	0	1	0 →	0	0d
01	0	0	0 →	1	1d
10	1	1	1 →	0	d1
11	1	0	1 →	1	d0



T-FF

T	Q_{n+1}		Q_n	Q_{n+1}	T
	$Q_n=0$	$Q_n=1$			
0	0	1	0 →	0	0
1	1	0	0 →	1	1
			1 →	0	1
			1 →	1	0

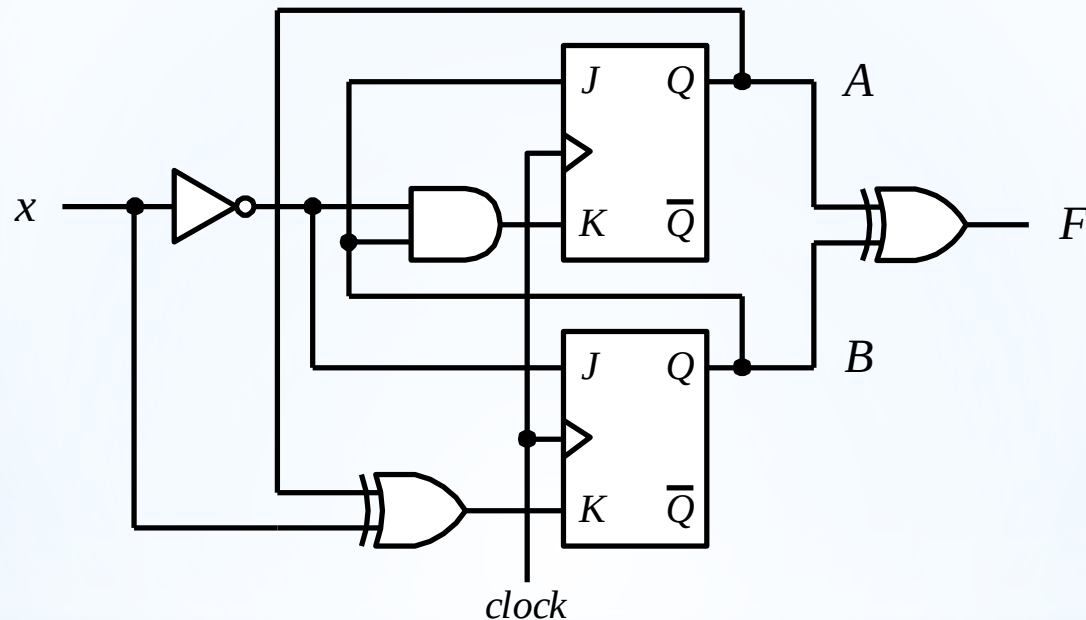


6.4 Sequential Circuit Analysis

Procedures:

- a) Determine the FFs' input logic functions.
- b) Construct the analysis table.
- c) Draw state diagram.
- d) Determine state equations.

Analysis example



Step 1: Identify the input functions of the FFs.

$$J_A = B, \quad K_A = B\bar{x}, \quad J_B = \bar{x}, \quad K_B = A \oplus x, \quad F = A \oplus B$$

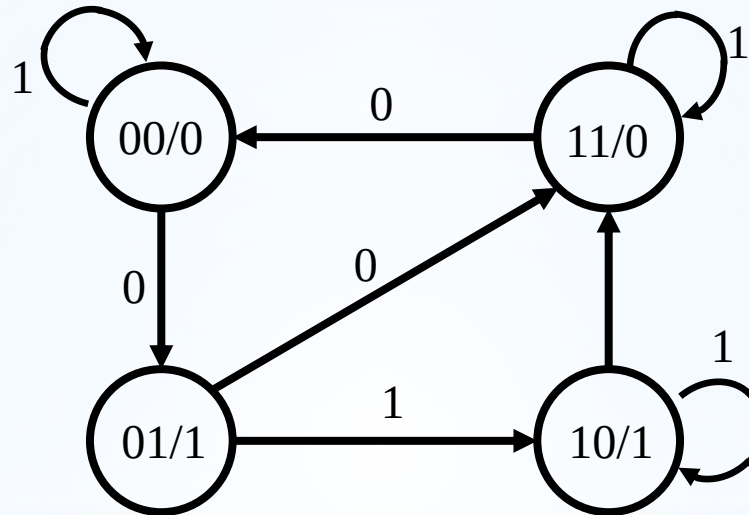
Analysis example -cont'd

Step 2: Construct an analysis table:

		PS	Input	FFs' excitation				NS	O/P
		AB	x	J_A	K_A	J_B	K_B	AB	F
<u>Steps</u>		00	0	0	0	1	0	01	0
1. Fill-in PS		00	1	0	0	0	1	00	0
2. Fill-in I/P		01	0	1	1	1	0	11	1
3. Fill-in J_iK_i		01	1	1	0	0	1	10	1
4. Find NS		10	0	0	0	1	1	11	1
5. Find O/P		10	1	0	0	0	0	10	1
		11	0	1	1	1	1	00	0
		11	1	1	0	0	0	11	0

Analysis example -cont'd

Step 3: Draw the state diagram:



State equation:

$$A_{n+1} = \overline{A_n}B_n + A_n\overline{B_n} + (B_nx) \text{ or } (A_nx)$$

$$B_{n+1} = \overline{B_n}\overline{x} + \overline{A_n}\overline{x} + A_nB_nx$$

State equation: Next stats (A_{n+1}, B_{n+1}) is a function of present state (A_n, B_n) and input (x)!!

Use K-map to determine the solution.

6.5 Sequential Circuit Design

Procedures

- a) Choose Mealy or Moore machine design from the circuit description.
- b) Design the state diagram.
- c) Tabulate the state table.
- d) Perform state reduction (will be taught in later section).
- e) Assign binary value to the states.
- f) Determine the number and which type of FFs to be used.
- g) Obtain a design table with FFs' excitation.
- h) Simplify the FFs' input functions and output functions.
- i) Draw the circuit diagram.

Synchronous sequential may NOT has input.

Clock is NOT an input, it is a timing signal for synchronizing the circuit operation.

Example- Free-running Counter

Design a free-running counter with counting sequence 2, 4, 5, 3, 7, 2, ...(and repeat) using *D* flip-flops

Operation:

Free-running $\rightarrow$ 0 inputs

$\Rightarrow$ 5 states $\Rightarrow$ 3 flip-flops are required

The largest number is 7 ($= 111_2$)

$\Rightarrow$ Require 3 outputs

The counter will count:

010 $\rightarrow$ 100 $\rightarrow$ 101 $\rightarrow$ 111 $\rightarrow$ 010 ...

Free-running Counter

Choose Moore machine design since the output should be stable for the whole clock period.

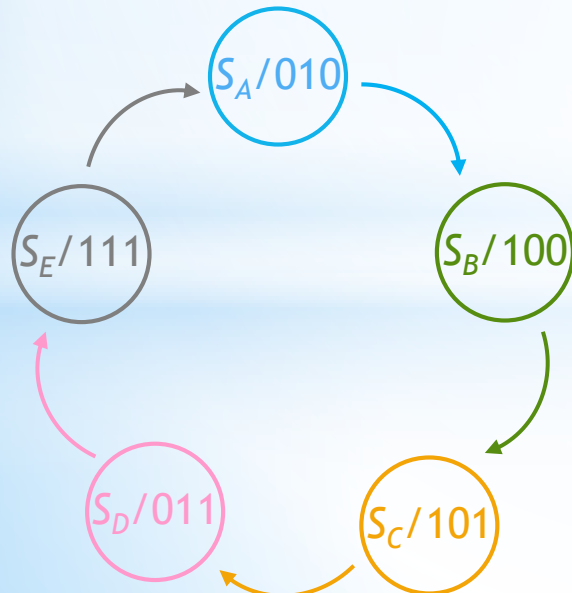
AND FFs' output can be used to represent the counter output.

No external input is required, it is a free running circuit.

The circuit state changes for every triggering edge appears.

Each count output is a “state” - therefore 5 states are required.

Derive the state diagram and state table



PS	NS	Output $Z_0 Z_1 Z_2$		
S_A	S_B	0	1	0
S_B	S_C	1	0	0
S_C	S_D	1	0	1
S_D	S_E	0	1	1
S_E	S_A	1	1	1

Free-running Counter

State assignment:

$$S_A = 010, S_B = 100, S_C = 101, S_D = 011, S_E = 111$$

PS	NS	Output $Z_0 Z_1 Z_2$		
S_A	S_B	0	1	0
S_B	S_C	1	0	0
S_C	S_D	1	0	1
S_D	S_E	0	1	1
S_E	S_A	1	1	1



PS			NS			Output		
Q_A	Q_B	Q_C	Q_A	Q_B	Q_C	Z_0	Z_1	Z_2
0	0	0	x	x	x	x	x	x
0	0	1	x	x	x	x	x	x
0	1	0	1	0	0	0	1	0
0	1	1	1	1	1	0	1	1
1	0	0	1	0	1	1	0	0
1	0	1	0	1	1	1	0	1
1	1	0	x	x	x	x	x	x
1	1	1	0	1	0	1	1	1

Use don't care state helps simplifying the design !!.

But it may have unwanted behavior !!

Free-running Counter

Obtain the design table with the D-FF excitation table:

Q_n		Q_{n+1}	D
0	→	0	0
0	→	1	1
1	→	0	0
1	→	1	1

PS			NS			FFs' input			Output		
Q_A	Q_B	Q_C	Q_A	Q_B	Q_C	D_A	D_B	D_C	Z_0	Z_1	Z_2
0	0	0	x	x	x	x	x	x	x	x	x
0	0	1	x	x	x	x	x	x	x	x	x
0	1	0	1	0	0	1	0	0	0	1	0
0	1	1	1	1	1	1	1	1	0	1	1
1	0	0	1	0	1	1	0	1	1	0	0
1	0	1	0	1	1	0	1	1	1	0	1
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	0	1	0	0	1	0	1	1	1

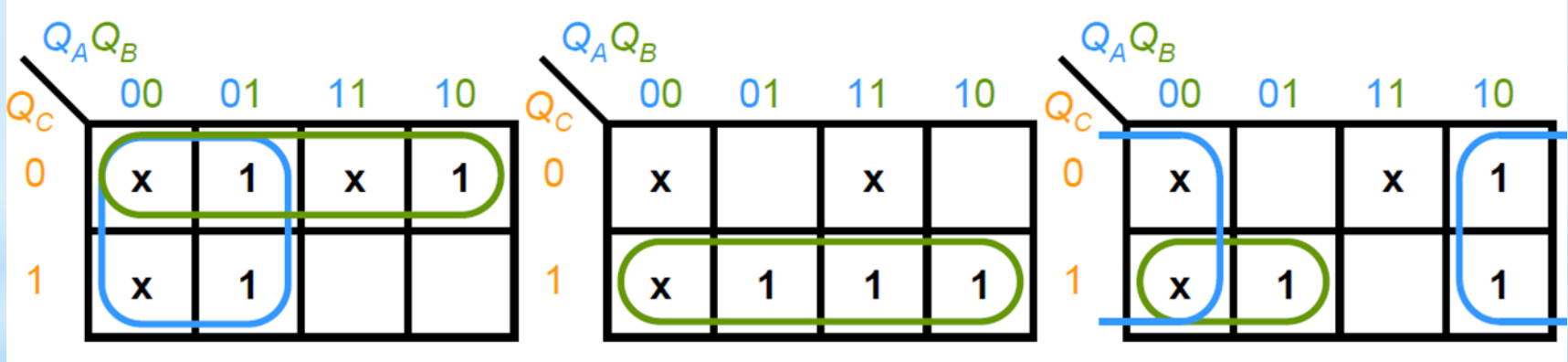
Free-running Counter

Flip-flops' input excitation:

$$D_A(Q_A, Q_B, Q_C) = \sum m(2,3,4) + \sum d(0,1,6)$$

$$D_B(Q_A, Q_B, Q_C) = \sum m(3,5,7) + \sum d(0,1,6)$$

$$D_C(Q_A, Q_B, Q_C) = \sum m(3,4,5) + \sum d(0,1,6)$$



$$D_A = Q_A' + Q_C'$$

$$D_B = Q_C$$

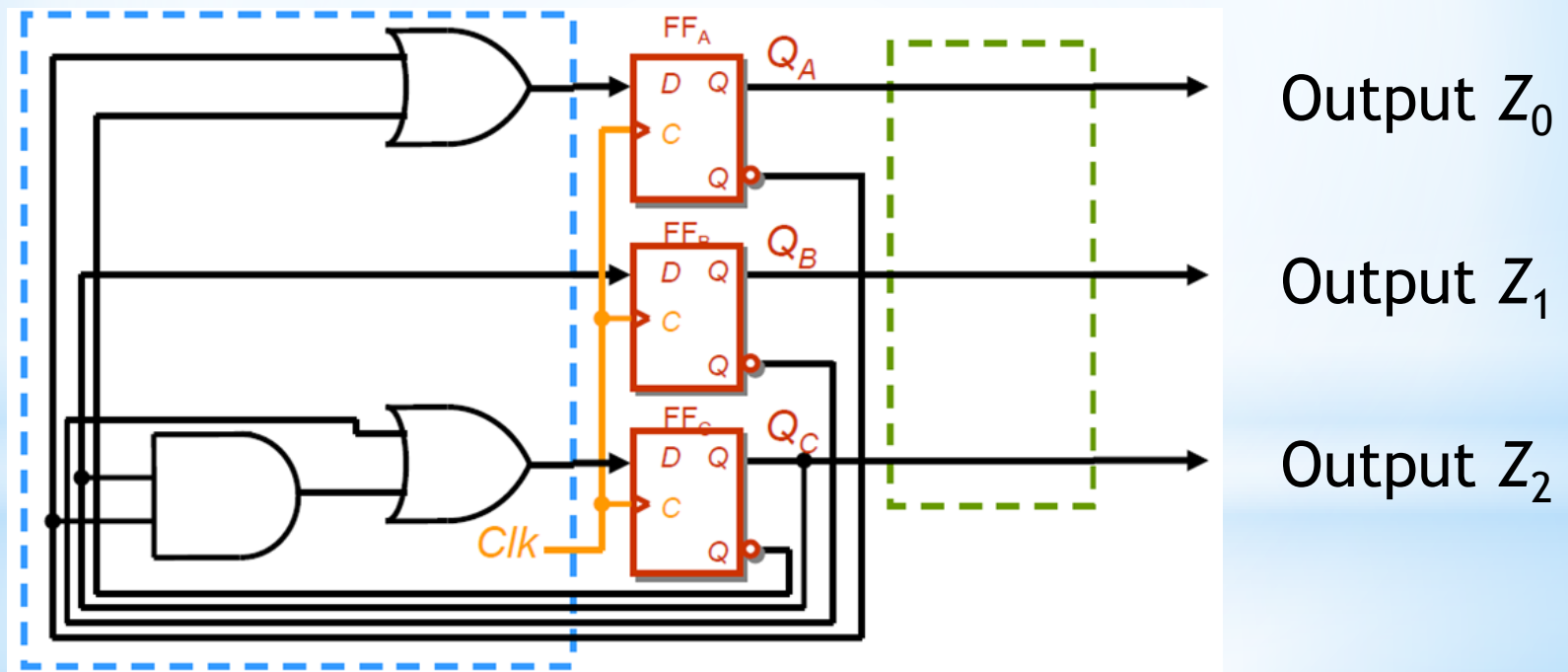
$$D_C = Q_B' + Q_A' Q_C$$

Free-running Counter

$$D_A(Q_A, Q_B, Q_C) = Q_A' + Q_C'$$

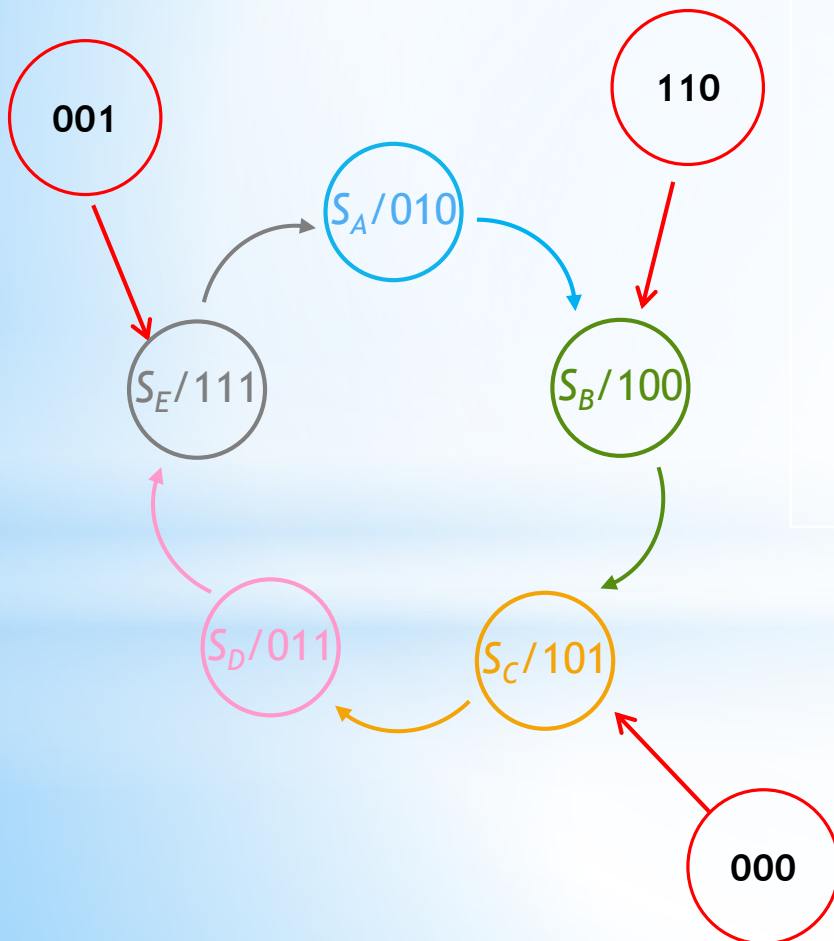
$$D_B(Q_A, Q_B, Q_C) = Q_C$$

$$D_C(Q_A, Q_B, Q_C) = Q_B' + Q_A'Q_C$$



Does this work properly ??

Analyze the circuit



<i>PS</i>			<i>FF inputs</i>			<i>NS</i>		
Q_A	Q_B	Q_C	D_A	D_B	D_C	Q_A	Q_B	Q_C
0	0	0	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1
0	1	0	1	0	0	1	0	0
0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1
1	0	1	0	1	1	0	1	1
1	1	0	1	0	0	1	0	0
1	1	1	0	1	0	0	1	0

If the initial state of the counter is not one of the valid count state, it will also produce the correct sequence after one clock pulse !!!

How to avoid this problem?

- In the design table, force the NS of the unused PS to one of the valid states.
- OR Predefine the circuit initial output by using **PRE** of **CLR** control

Another Free-running Counter

JK FF implementation ??

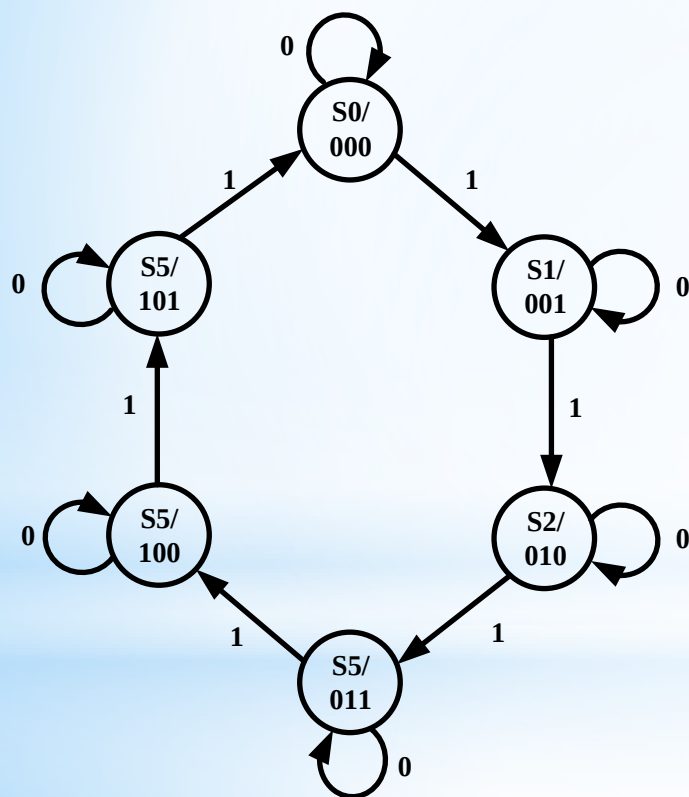
<i>PS</i>			<i>NS</i>			<i>FF inputs</i>								
Q_A	Q_B	Q_C	Q_A	Q_B	Q_C	J_A	K_A	J_B	K_B	J_C	K_C	Z_0	Z_1	Z_2
0	0	0	x	x	x							x	x	x
0	0	1	x	x	x							x	x	x
0	1	0	1	0	0							0	1	0
0	1	1	1	1	1							0	1	1
1	0	0	1	0	1							1	0	0
1	0	1	0	1	1							1	0	1
1	1	0	x	x	x							x	x	x
1	1	1	0	1	0							1	1	1

Q_n	Q_{n+1}	J	K
0	→ 0	0	X
0	→ 1	1	X
1	→ 0	X	1
1	→ 1	X	0

Complete the design !!!

Count Design with Enable input

Design a modulo-6 counter with $E=1$ counting upward and $E=0$ stop counting.



<i>PS</i>			<i>I/P</i>	<i>NS</i>			<i>FF inputs</i>		
Q_2	Q_1	Q_0		Q_2	Q_1	Q_0	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$
0	0	0	0	0	0	0			
0	0	0	1	0	0	1			
0	0	1	0	0	0	1			
0	0	1	1	0	1	0			
0	1	0	0	0	1	0			
0	1	0	1	0	1	1			
0	1	1	0	0	1	1			
0	1	1	1	1	0	0			
1	0	0	0	1	0	0			
1	0	0	1	1	0	1			
1	0	1	0	1	0	1			
1	0	1	1	0	0	0			

States are same as counter output.

When $E=0$, the counter counts downward, what will be the design?

More example

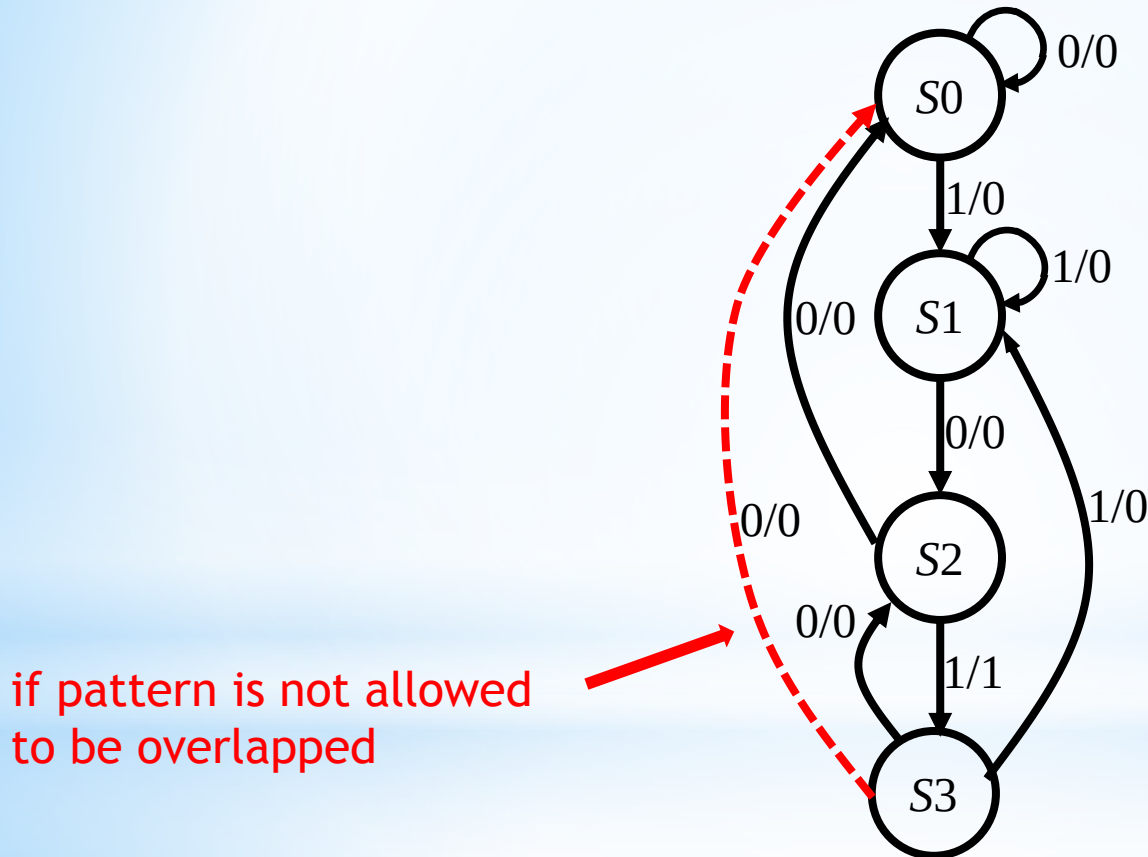
Sequence recognizer:

Design a circuit using JK FF to detect the pattern 101 occurring in a serial input data line. The circuit is synchronized with the positive-edge of the clock pulse. The operation of the circuit is illustrated as following:

Input: $x = 00110111001010100$

Output: $y = 00000100000010100$

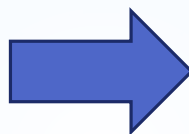
Sequence recognizer:



Sequence recognizer - cont'd

State table:

	<i>NS</i>		<i>Output</i>	
<i>PS</i>	<i>x</i> =0	<i>x</i> =1	<i>x</i> =0	<i>x</i> =1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S3	0	1
S3	S2	S1	0	0



Reduced state table:

	<i>NS</i>		<i>Output</i>	
<i>PS</i>	<i>x</i> =0	<i>x</i> =1	<i>x</i> =0	<i>x</i> =1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

$$S1 = S3$$

Sequence recognizer - cont'd

State assignment:

2 JK FFs are required.

State	Q_A	Q_B
S0	0	0
S1	0	1
S2	1	0

JK FF excitation table:

Q_n	Q_{n+1}	J	K
0	$\rightarrow$ 0	0	X
0	$\rightarrow$ 1	1	X
1	$\rightarrow$ 0	X	1
1	$\rightarrow$ 1	X	0

Sequence recognizer - cont'd

Design table:

<i>PS</i>		<i>Input</i>	<i>NS</i>		<i>FF inputs</i>		<i>Output</i>
Q_A	Q_B	x	Q_A	Q_B	J_A	K_B	y
0	0	0	0	0	0	x	0
0	0	1	0	1	0	x	0
0	1	0	1	0	1	x	0
0	1	1	0	1	0	x	0
1	0	0	0	0	x	1	0
1	0	1	0	1	x	1	1

Sequence recognizer - cont'd

K-maps:

	$Q_B x$			
Q_A				1
	x	x	x	x

$$J_A = Q_B \bar{x}$$

	$Q_B x$			
Q_A	x	x	x	x
	1	1	x	x

$$K_A = 1$$

	$Q_B x$			
Q_A		1	x	x
		1	x	x

$$J_B = x$$

	$Q_B x$			
Q_A	x	x		1
	x	x	x	x

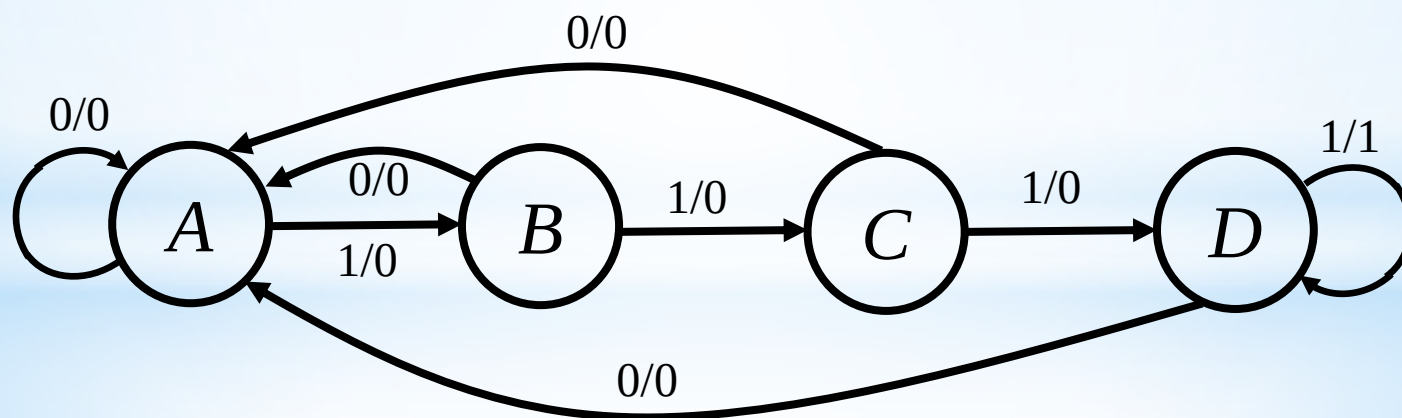
$$K_B = \bar{x}$$

$$y = Q_A \bar{Q}_B x \quad \text{or} \quad y = Q_A x$$

More Example

Implement a “1111” sequential recognizer using PAL16R6 with *D* flip-flops inside

State diagram:



Sequential circuit design using PLD - cont'd

	x	
PS	0	1
A	A/0	B/0
B	A/0	C/0
D	A/0	D/1
C	A/0	D/0

State table

		x	
$y_1^k y_2^k$		0	1
0 0		00	01
0 1		00	10
1 1		00	11
1 0		00	11
	$y_1^{k+1} y_2^{k+1}$		

Transition table

		x	
$y_1^k y_2^k$		0	1
0 0		0	0
0 1		0	0
1 1		0	1
1 0		0	0
		z	

Output map

State assignment:

A: 00

B: 01

C: 10

D: 11

$$D_1 = xy_1 + xy_2$$

$$D_2 = xy_1 + x\overline{y_2}$$

$$z = xy_1y_2$$