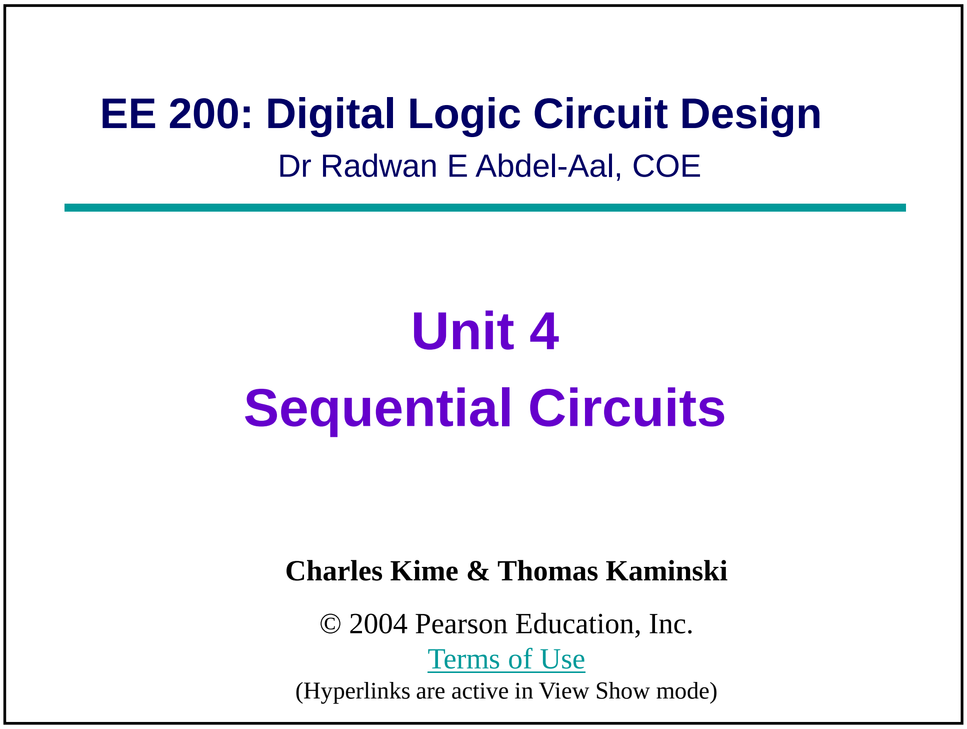


EE 200: Digital Logic Circuit Design

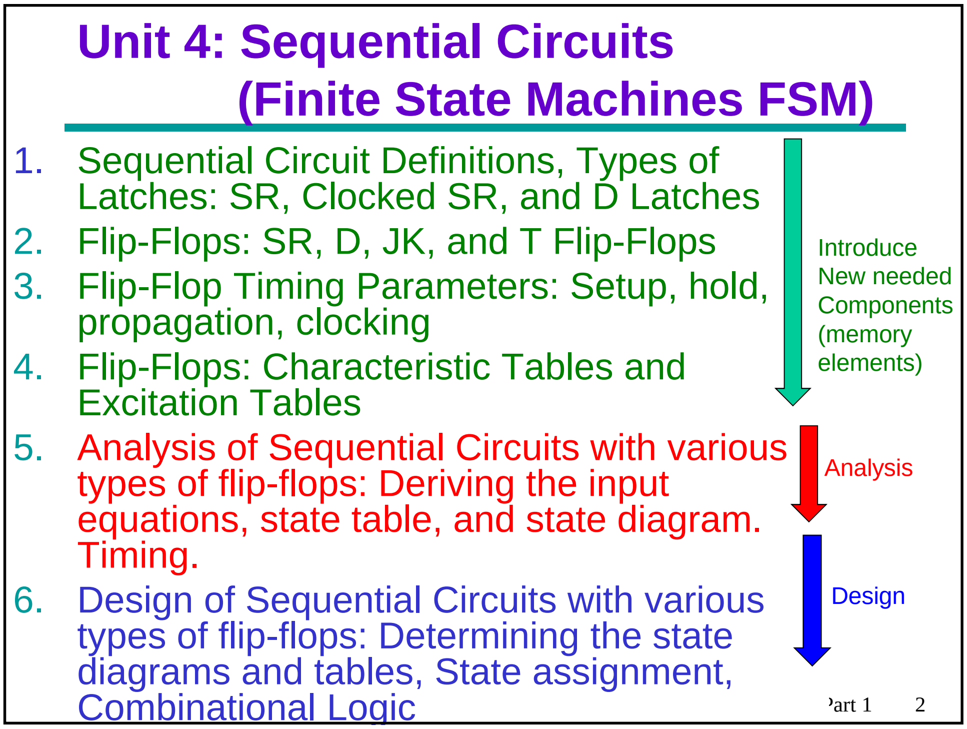
Dr Radwan E Abdel-Aal, COE

Unit 4
Sequential Circuits

Charles Kime & Thomas Kaminski

Terms of Use

(Hyperlinks are active in View Show mode)

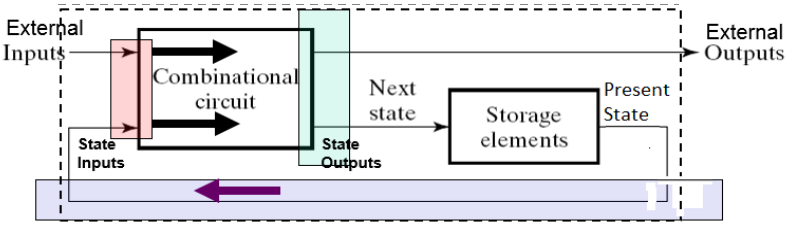


Unit 4: Sequential Circuits (Finite State Machines FSM)

1. Sequential Circuit Definitions, Types of Latches: SR, Clocked SR, and D Latches
2. Flip-Flops: SR, D, JK, and T Flip-Flops
3. Flip-Flop Timing Parameters: Setup, hold, propagation, clocking
4. Flip-Flops: Characteristic Tables and Excitation Tables
5. Analysis of Sequential Circuits with various types of flip-flops: Deriving the input equations, state table, and state diagram. Timing.
6. Design of Sequential Circuits with various types of flip-flops: Determining the state diagrams and tables, State assignment, Combinational Logic

Introduction to Sequential Circuits

Outputs are functions of inputs and some previous (state) outputs

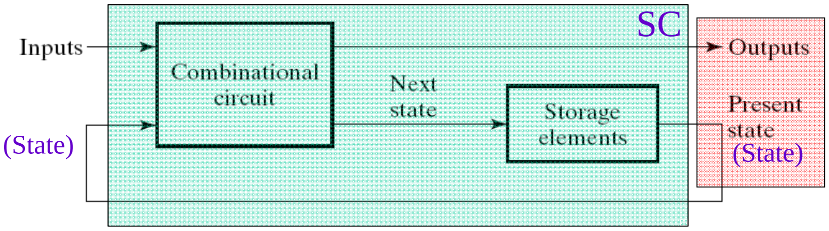


- **A Sequential Circuit (SC) consists of:**
  - **Data Storage (memory) elements: (Latches / Flip-Flops)**
  - **+ Combinatorial Logic:**

    Implements a multiple-output function
    - **External Inputs are signals from outside**
    - **External Outputs are signals to outside**
    - **State inputs (Internal to SC) = Present State at o/p of storage elements**
    - **State outputs, Next State at i/p of storage elements**

The storage (memory) elements isolate the next state from the present state, So changes occur only when desired

Introduction to Sequential Circuits (SC)



- **Combinational Circuit outputs:**
  - ***Next state:***

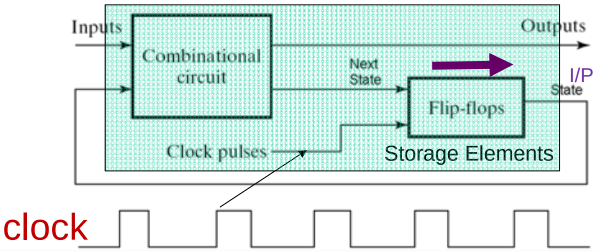
    Next State = f(Inputs, State)
  - ***External Output:- Two Possibilities:***
    - **Mealy Circuits**

      Outputs = g(Inputs, State)
    - **Moore Circuits**

      Outputs = h(State only)

Timing of Sequential Circuits
Two Approaches

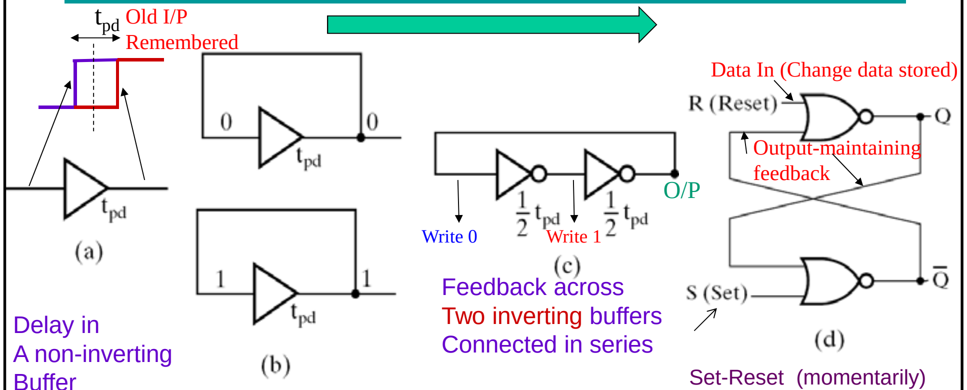
- Behavior depends on the times at which the storage elements ‘sense’ their inputs and ‘change’ their outputs (“next state” becomes “present state”)
- Asynchronous (No clock)
  - Behavior defined from knowledge of inputs **at any instant of time** and the order in which inputs change in continuous time
- Synchronous (More common)
  - Behavior is determined by the signals at **discrete** times **(clock pulses)**
  - Storage elements sense their inputs / change state only according to a timing and synchronizing signal (a **clock**)
- Will use mainly the synchronous approach here



I/P → O/P **at the arrival of a clock pulse**

When a clock pulse arrives, the SC moves from one state to the next

Data Storage Logic Structures for SCs



Delay in
A non-inverting
Buffer

Problem:
Data stored only
for short time =
Propagation delay t_{pd}

Non-inverting buffer
With feedback- indefinite

Problem: No separate input for data.
Difficult to change data stored

Feedback across
Two inverting buffers
Connected in series

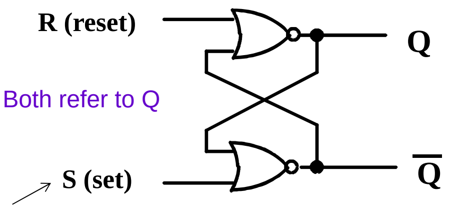
Set-Reset (momentarily)
NOR **Latch**

Now 2 separate inputs:
- **Data-in** (momentarily make the change to be stored)
- **Feedback** to maintain that change

Basic NOR Set–Reset (SR) Latch

Set: Make Q = 1,   Reset: Make Q = 0

- **Cross-coupling two NOR gates gives the Set – Reset Latch:**
- **Which has the time sequence behavior:**



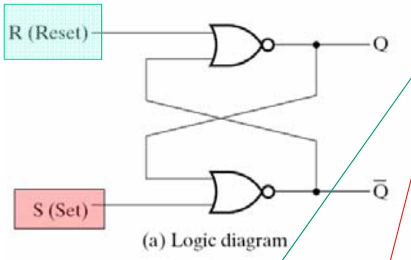
Time	R	S	Q	$\overline{Q}$	Comment
	0	0	Q	$\overline{Q}$	Show the last stored data
	0	1	1	0	“Set” Q to 1 (Write 1)
	0	0	1	0	Q “remembers” written 1 after I/P disappeared
	1	0	0	1	“Reset” Q to 0 (Write 0)
	0	0	0	1	Q “remembers” written 0 after I/P disappeared
	1	1	0	0	Both Q and $\overline{Q}$ = 0 (Avoid)
	0	0	?	?	Undefined! (Q = 1 or 0)

00 = Normal input condition
No input change
(show last stored I/P)

Input R S stored (written)
in $\overline{Q}$ Q
(remains at O/P after input is removed)

S = 1, R = 1 simultaneously is a forbidden input pattern

Basic NOR Set–Reset (SR) Latch



(a) Logic diagram

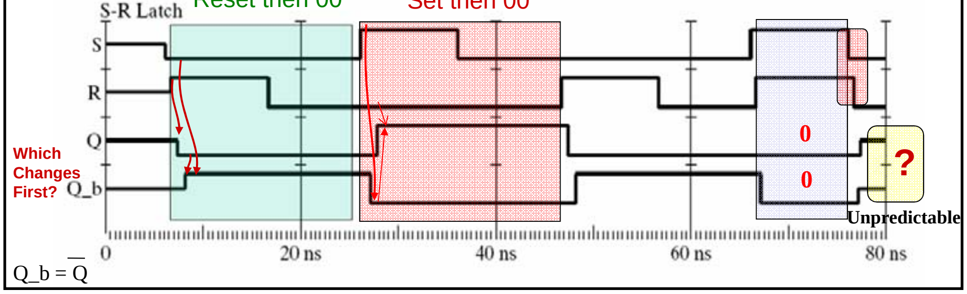
S	R	Q	$\overline{Q}$	
0	1	0	1	Reset state
0	0	0	1	
1	0	1	0	Set state
0	0	1	0	
1	1	0	0	Undefined (Avoid)

(b) Function table

Should not try to Set and Reset at the same time!
Forbidden I/P Comb.

Reset then 00

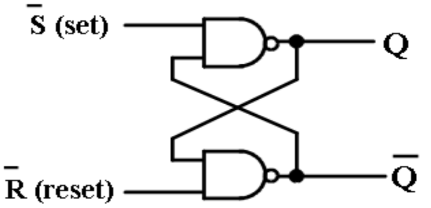
Set then 00



Q_b = $\overline{Q}$

Basic NAND $\overline{\text{Set}}$–$\overline{\text{Reset}}$ ($\overline{\text{S}}\overline{\text{R}}$) Latch

- Cross-coupling two NAND gates gives the $\overline{S}$ – $\overline{R}$ Latch:



- Which has the time sequence behavior:

Time	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Comment
	1	1	Q	$\overline{Q}$	Show the last stored data
	0	1	1	0	“Set” Q to 1 (Write 1)
	1	1	1	0	Q “remembers” written 1 after I/P disappeared
	1	0	0	1	“Reset” Q to 0 (Write 0)
	1	1	0	1	Q “remembers” written 0 after I/P disappeared
	0	0	1	1	Both Q and $\overline{Q}$ go high (Avoid)
	1	1	?	?	Undefined! (Q could be 0 or 1

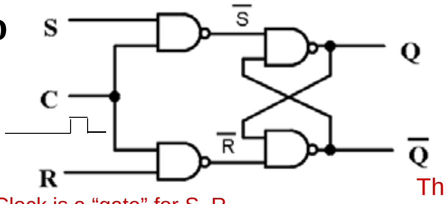
11 = Normal input condition
No input change

Input $\overline{S}$ $\overline{R}$ stored in $\overline{Q}$ Q (remains at O/P after input is removed)

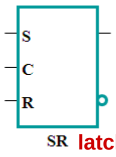
$\overline{S}$ = 0, $\overline{R}$ = 0 simultaneously is a forbidden input pattern

Clocked (or Controlled) S-R NAND Latch

- Adding two NAND gates to the basic $\overline{S}$ - $\overline{R}$ NAND latch gives the clocked S – R latch:



Clock is a “gate” for S, R



SR latch

This latch is Transparent: i.e. O/P follows the I/P directly when C = 1

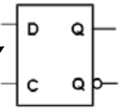
- C = normally 0 → $\overline{S}$ $\overline{R}$ inputs to the latch = normally 1 1 (No output change)
  So this prevents the forbidden conditions $\overline{S}$ $\overline{R}$ = 0 0 with C = 0
- C = 1 Opens the two input NANDs for the S R, inverting them.
  This gives normal S R (not $\overline{S}$ $\overline{R}$) latch operation → Allow changes in latch state
  But here both S R = 1 1 with C = 1 is still a problem
- C means “control” or “clock”. Changes in SR affect the latch only during the clock pulse

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

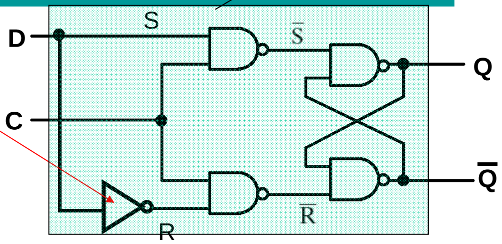
Freeze (Read)

The Clocked D Latch-
Totally avoids the SR = 11 Problem!

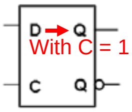
D type latch



- Adding this inverter to the S-R Latch, makes it a **D Latch**
- We got rid of the unwanted condition (SR =11 with C = 1)
- But **also lost** SR = 00 **(no change)!**



When C = 1, this latch is **transparent**: The D I/P is **effectively 'connected'** to output Q, i.e. Q follows D



With C = 0:
Freeze Output at last value written when C was 1,
(can change only when C becomes 1 again)

To get 'no change':
Must block the clock pulses!

C	D	Next state of Q
0	X	No change (Q = last Q before C went from 1 to 0)
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

i.e. with C = 1: D → Q

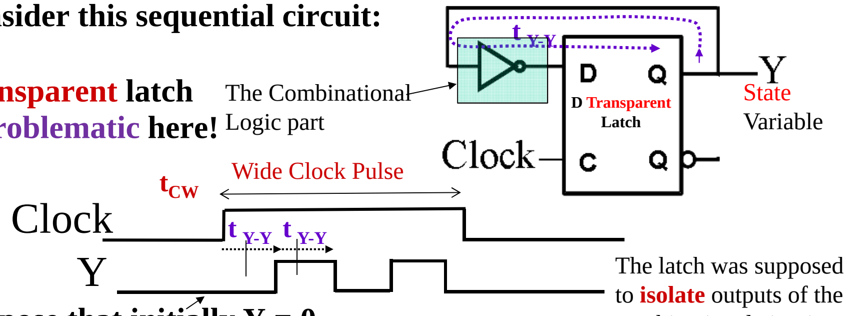
Function Table

From Latches to Flip-Flops

- The transparent latch timing problem
- Solution: Flip-Flop
  - Master-slave flip-flop
  - Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops
- Flip-flop timing
- Types of flip-flops: D, JK, T

The Transparent Latch as a Storage Element: Timing Problem

- **Consider this sequential circuit:**
- **Transparent latch is problematic here!**



The latch was supposed to isolate outputs of the combinational circuit from its inputs. Is it???

- **Suppose that initially Y = 0.**
- **As long as C = 1, the value of Y keeps changing!**
- **Changes occur based on the delay in the Y-to-Y loop**
- **If $t_{Y\text{-}Y}$ << t_{CW} this causes several unwanted state changes to occur during the same clock pulse- unacceptable!**
- **Desired behavior: Y should change only once per clock pulse, in order to get only one state transition per clock pulse!**

Solving the Latch Timing Problem Flip flops instead of latches

- Two approaches:
  - Break the path within the storage element into **two** successive (mutually exclusive) steps in time:

    Step - 1. Register the change in input D (then stop)

    Step - 2. Apply that change to the output Y (then stop)

    This uses a **master-slave** (Pulse Triggered) flip-flop

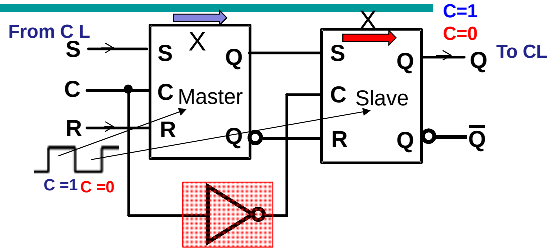
OR

  - Use an **edge-triggered** flip-flop:

    Change in D is sensed **and** applied to the Q output in **one go at the edge** of the clock pulse (+ ive or – ive edge)

    i.e. Effectively as if we have a Zero-width clock pulse, which obviously solves the problem (see previous slide)

S-R Master-Slave (Pulse-Triggered) Flip-Flop



- Consists of two clocked S-R latches in series, with the clock to the second latch inverted
- C = 1: - Master is open

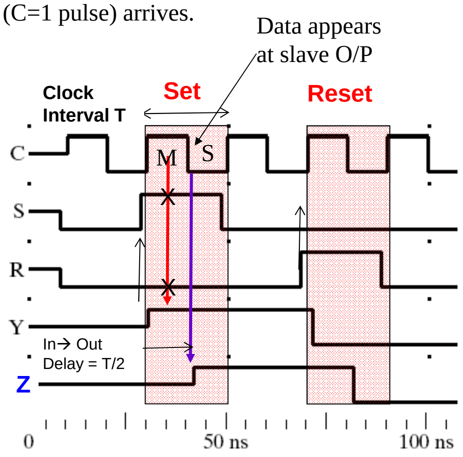
  1st half Of clock Pulse - Slave is blocked

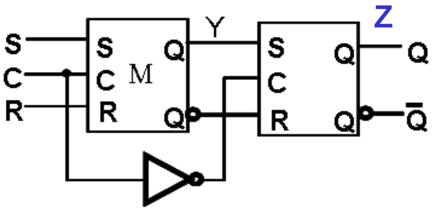
  Only “input is sensed” by master for this pulse duration (→ **pulse-triggered)** while output is unchanged by slave
- C = 0: - Master is Blocked

  2nd half Of clock Pulse - Slave is open → “output is changed”
- The path from input to output is thus broken by the different 2 clock levels for the two latches (C = 1 and C = 0)
- Sensing I/P then changing O/P are now **two separate steps** - not one transparent step as the case with the transparent latch

S-R Master-Slave Flip-Flop: Simulation

Ideally, changes in S, R inputs from combinational circuit should **arrive to master** **before** the next clock interval (C=1 pulse) arrives.

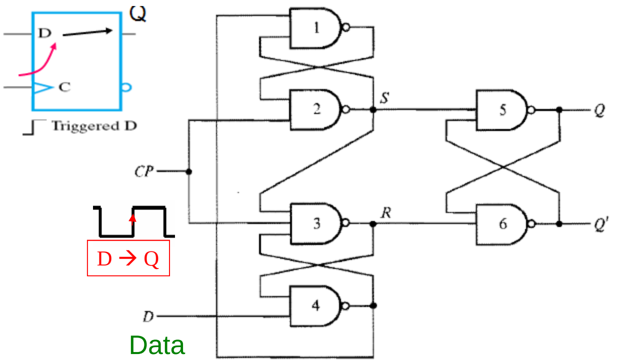




Edge-Triggered D-type Flip-Flop

- This is a **Positive Edge-triggered D-type flip-flop**

- Is currently the most preferred FF for building sequential circuits



- The D data input is transferred to the Q output only at the rising edge of the clock, subject to timing constraints on the D input relative to effective clock edge: Setup time before edge and Hold time after edge
- Negative edge triggered D FF is also available

Flip-Flop Timing Parameters:
- ive Edge Triggered FF (Section 6.3)

Requirements:

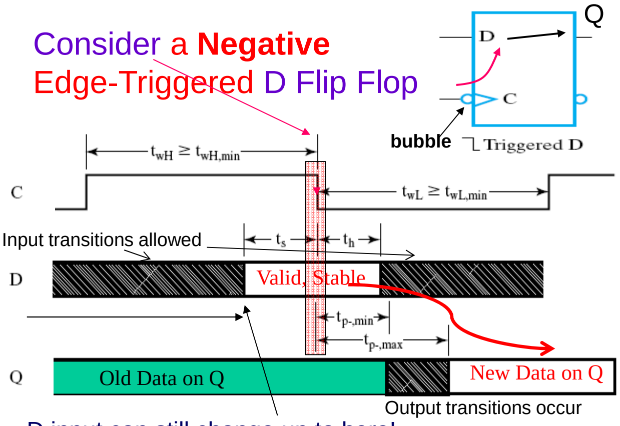
- t_w - clock pulse width (for both low & high)

- t_s : setup time
- t_h : hold time (usually 0 ns)

Outcomes:

- t_p: propagation delay

Consider a **Negative** Edge-Triggered D Flip Flop



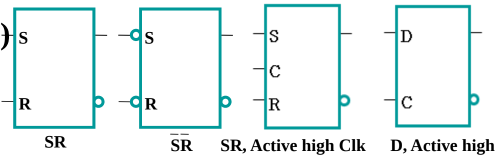
D input can still change up to here!
Better utilization of time
→faster sequential cct designs
compared to Master-Slave FF

Standard Symbols for Storage Elements

Unclocked

clocked

1. Latches (Transparent)

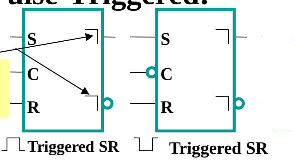


Transparent Latches, Provide No I-O isolation

2. Flip Flops

a. Master-Slave (M-S) Pulse-Triggered:

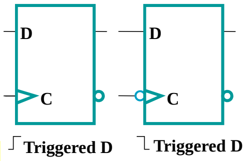
O/P **determined during clock pulse width and changed** at its end



(b) Master-Slave Flip-Flops

In a sequential that uses different Types of FFs, Ensure that all FFs circuit change their outputs at the **same clock edge**. Invert clock signal to some FFs if needed

b. Edge-Triggered:



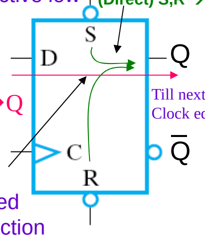
(c) Edge-Triggered Flip-Flops

O/P **determined & changed** on the indicated clock **edge**

One problem with D type FF is that no D inputs produce "no change" at the output
Solution:
- Block the clock pulses
- Feed back the Q to the D input when no change is required
(See Unit 5)

FF Direct Inputs

- When power is turned ON, the state of a sequential circuit FFs could be anything!
- We usually need to initialize the circuit to a known state before operation starts
- This initialization is often done directly outside the clocked behavior of the circuit, i.e., asynchronously
- Direct S and/or R inputs that control the state of the latches within the flip-flops are added to FFs for this purpose
- For the example the flip-flop shown
  - 0 applied to $\overline{R}$ directly ($\overline{S}$=1): resets the flip-flop to the 0 state regardless of the clock
  - 0 applied to $\overline{S}$ directly ($\overline{R}$=1): sets the flip-flop to the 1 state regardless of the clock



Bubble → I/P is active low

Asynchronous (Direct) S,R → Q

Synchronous (clocked) D→Q

Till next Clock edge

Edge-Triggered Clock action

Active Low

S	R	C	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	Undefined	
1	1	↑	0	0	1
1	1	↑	1	1	0

Asynchronous Action- **Regardless** of the clock

Synchronous (clocked) action

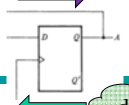
(b) Function table

Other Types of Flip-Flops

- We know about the master-slave S-R and D flip-flops
- We will briefly introduce the J-K and T flip-flops
  - Implementation
  - Behavior
    - Characteristic Table/Equation: For use in SC Analysis
    - Excitation Table/Equation: For use in SC Design

Analysis *Characteristic*

Basic Flip-Flop Descriptors



Excitation design

- In **analysis**: Given a Circuit:

  Present state, I/Ps → CL → FF Inputs → ? FF O/P (Next state) ?

  FF: FF Inputs & Present output → Next FF output?

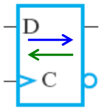
  - *Characteristic table* - defines the next output of the flip-flop given its present output and its inputs
  - *Or Characteristic equation* - defines the next output of the flip-flop as a Boolean function of its present output and its inputs
- In **design**: Given a **state transition** behavior → ? CL ?

  FF: Present output & Next output → ? FF inputs? (that give such behavior)

  - *Excitation table* - defines the flip-flop inputs that give a required present-to-next output behavior

D Flip-Flop

Analysis



Design

Analysis

- **Characteristic Table**

Analysis

Input-Driven

D(t)	Q(t+1)	Operation
0	0	Reset
1	1	Set

Does not depend on the present O/P

Next O/P = Present I/P

- **Characteristic Equation**

$$Q(t+1) = D(t)$$

Design

- **Excitation Table**

Design

Output-Driven

Q(t +1)	D(t)	Operation
0	0	Reset
1	1	Set

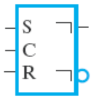
Limitation of D FF:
No built-in "No Change" Condition
while clock is connected

Present I/P = Desired Next O/P

Hence simplicity of using D FF

S-R Flip-Flop

Pulse or Edge -Triggered?



Analysis

- Characteristic Table

Input-Driven

S	R	Q(t+1)	Operation
0	0	*Q*(*t*)	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

Given the present FF I/Ps, O/P
→ Next FF O/P = ?

← Improvement on D !

Limitation:
SR=11: "Undesirable" Condition

- Characteristic Equation

$$Q(t+1) = S + \overline{R}\,Q(t)$$

Design

- Excitation Table

Given Present, next O/P s → FF Inputs = ?

Get from above

Output behavior-Driven

Q(t)	Q(t+1)	S	R	Operation
0	0	0	X	No change , Or Reset
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change , Or Set

Change

J-K Flip-Flop- Improvements on SR and D types

→ on SR: Avoids the "SR = 11", JK = 11 → Toggle, i.e. Q(t+1) = $\overline{Q(t)}$

→ on D type: Allows a 'No Change' condition

- **Characteristic Table**

S R	Q(t+1)	Operation
0 0	$Q(t)$	No change
0 1	0	Reset
1 0	1	Set
1 1	?	Undefined

Now OK! ✓

J K	Q(t+1)	Operation
0 0	$Q(t)$	No change
0 1	0	Reset
1 0	1	Set
1 1	$\overline{Q}(t)$	Complement (Toggle)

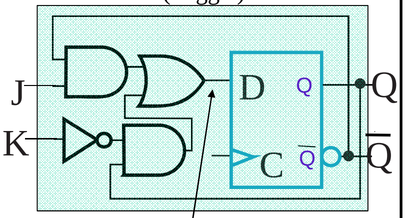
- **Characteristic Equation**

$$Q(t+1) = J\,\overline{Q}(t) + \overline{K}\,Q(t)$$

- **Excitation Table**

Q(t)	Q(t+1)	J K	Operation
0	0	0 X	No change , Or Reset
0	1	1 X	Set , Or Toggle
1	0	X 1	Reset , Or Toggle
1	1	X 0	No Change , Or Set

Change

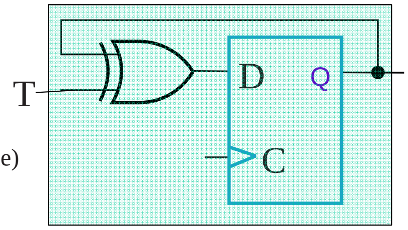


D(t) → Q(t+1)

T (Toggle) Flip-Flop
D FF with only "No change" & "toggle" capabilities

- **Characteristic Table**

T	Q(t+1)	Operation
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement (Toggle)



- **Characteristic Equation**

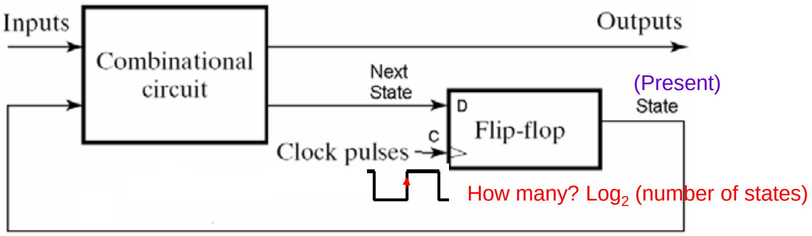
$$Q(t+1) = D(t) = T \oplus Q(t)$$

- **Excitation Table**

Q(t+1) →	T	Operation
$Q(t)$	0	No change
$\overline{Q}(t)$	1	Complement (Toggle)

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Sequential Circuit
Analysis & Design



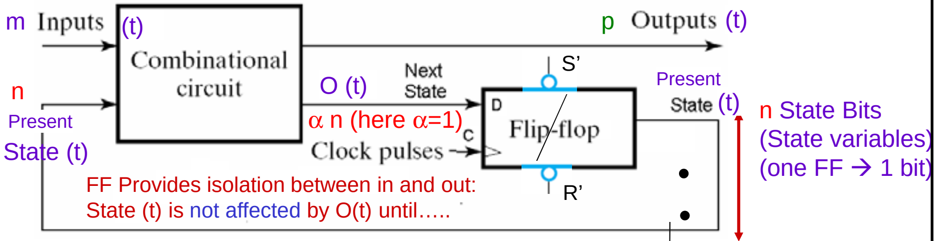
How many? Log_2 (number of states)

Analysis:
Given a circuit. → Describe how it behaves, in terms of: Present State, Inputs → Next State? Outputs?,

Design:
Given how a circuit behaves, in terms of: Present State, Inputs → Next State, Outputs, → **Determine the circuit**

Sequential Circuit Analysis: With D-type FFs

- General Model
  - Present State (state) at time (t) are the O/Ps of an array of flip-flops
  - Next State at (t+1) are O(t) combinational fns of {State & Inputs}
  - (External) Outputs at time (t) are a combinational function of State (t) only (Moore) and also Inputs (t) (Mealy)



n State Bits (State variables) (one FF → 1 bit)

FF Provides isolation between in and out: State (t) is not affected by O(t) until…..

Max # of states for a circuit with 4 FFs?

How many FFs needed for a circuit with 6 states?

…. the next clock pulse comes:
→ t becomes t+1,
→ O(t) is moved to FF output, thus becoming State (t+1), i.e. next state

Sequential Circuit Analysis

- Given a sequential Circuit
- Objective: Obtain outputs & state behavior (External outputs and next state) from (External inputs and present state) for all combinations of I/Ps & present state
- Two equivalent approaches to represent the results of the analysis:
  - State table: A truth table-like approach
  - State diagram: A graphical, <u>more intuitive </u>way to represent the state table and express sequential circuit operation

Analysis Example using D FF:
Given a Sequential Circuit → Determine how it behaves

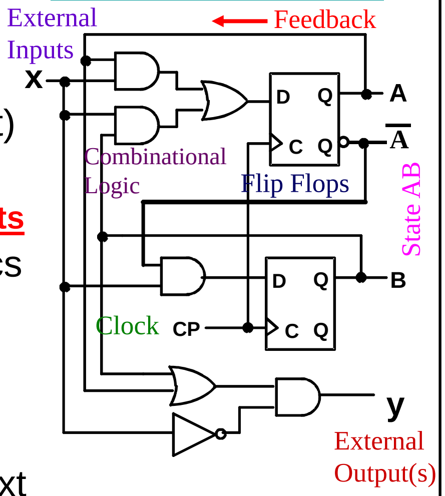
- <u>External Inputs</u>: x(t)
- <u>External Outputs:</u> y(t)
- <u>State Outputs:</u> A(t), B(t)

In Analysis we determine:

- Comb. Equations for the **<u>FF inputs</u>**

  → then use FF characteristics to get the <u>next state</u> given a present state and inputs
- Comb. Equations for the **<u>External Outputs</u>**

  → and use them to get the ext <u>outputs</u> given a present state and inputs



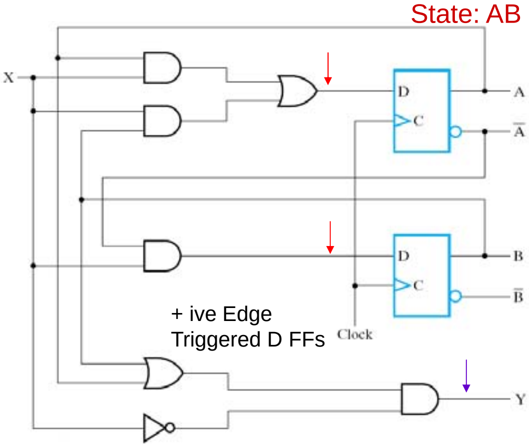
→ Synchronous or asynchronous?
→ Mealy or Moore?

Analysis Example 1: A Mealy Circuit, Ext. Output = F(state, inputs)

Deriving flip flop input equations

MSB

State: AB



+ ive Edge
Triggered D FFs

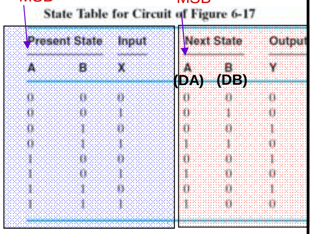
- Right at the outset, there are things we can do:
  We can derive Boolean equations for *all outputs* of the combinational logic (CL) circuits
- Two types of CL outputs:
  → Flip flop inputs
  (Will determine the next state **based on FF characteristics**)
  $D_A = AX + BX$
  $D_B = \overline{A}X$
  → External Outputs:
  $Y = (A+B)\,\overline{X}$

Note: Flip flop inputs needed to determine next state depend on the type of flip flop used, e.g. D, SR, etc.

State Table Characteristics

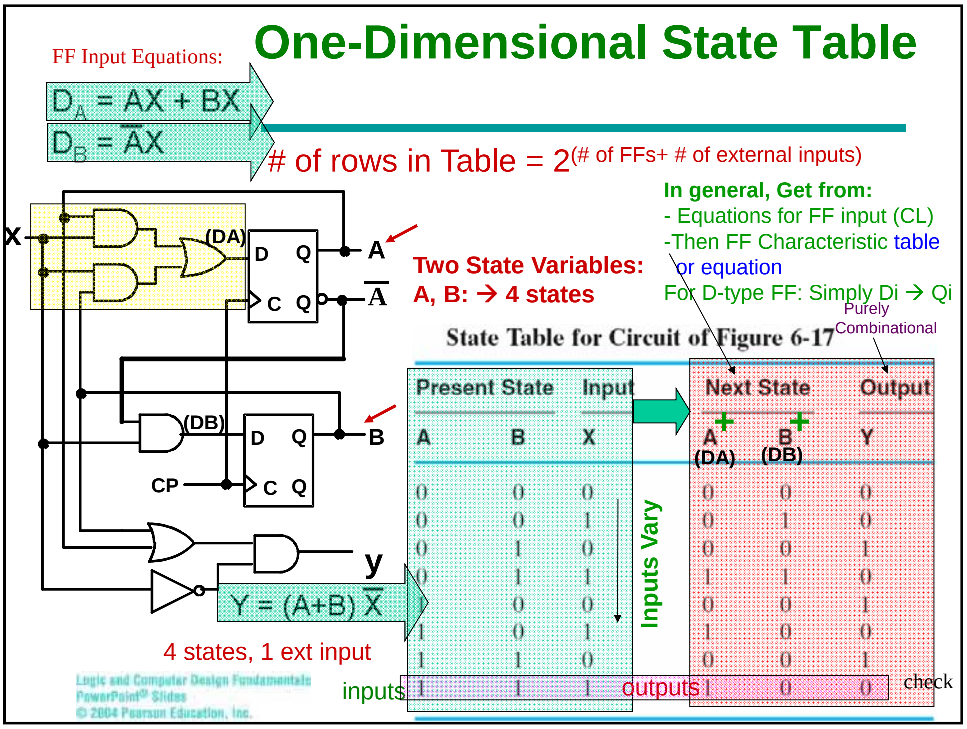
- *State table* – a multi variable table with the following four “vertical” components: CL = Combinational Logic
  → **CL Inputs:** FF = Flip Flop
  - *Present State (MSBs)* – Values of the state variables (FF outputs) for each allowed state
  - *External Inputs*
  → **Outputs:**
  - *Next-state* – Value of the state
    (FF outputs) at time (t+1). Determined by:
    - FF inputs (outputs of CL)
    - the FF characteristics: Simplest for D-type FF: D → Q
  - *CL External Outputs* – the value of the outputs as a function of the present state only (Moore) or present state & external inputs (Mealy)

MSB MSB



State Table for Circuit of Figure 6-17

Present State		Input	Next State		Output
A	B	X	A (DA)	B (DB)	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



FF Input Equations:

One-Dimensional State Table

$D_A = AX + BX$

$D_B = \overline{A}X$

# of rows in Table = $2^{(\text{\# of FFs+ \# of external inputs})}$

In general, Get from:
- Equations for FF input (CL)
-Then FF Characteristic table or equation
For D-type FF: Simply Di → Qi

Two State Variables:
A, B: → 4 states

$Y = (A+B)\overline{X}$

4 states, 1 ext input

State Table for Circuit of Figure 6-17

Purely Combinational

Present State		Input	Next State		Output
A	B	X	A^+ (DA)	B^+ (DB)	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

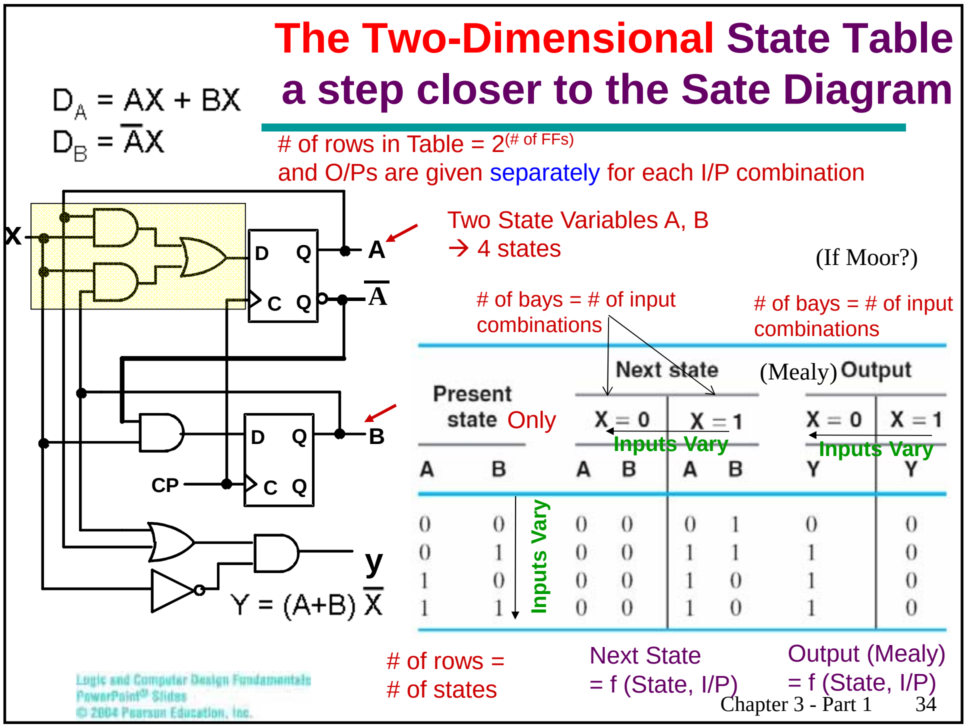
Inputs Vary

inputs

outputs

check

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.



The Two-Dimensional State Table
a step closer to the Sate Diagram

$D_A = AX + BX$

$D_B = \overline{A}X$

# of rows in Table = $2^{(\text{\# of FFs})}$
and O/Ps are given separately for each I/P combination

Two State Variables A, B
→ 4 states

(If Moor?)

# of bays = # of input combinations

# of bays = # of input combinations

$Y = (A+B)\overline{X}$

Present state Only		Next state				(Mealy) Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Inputs Vary

Inputs Vary

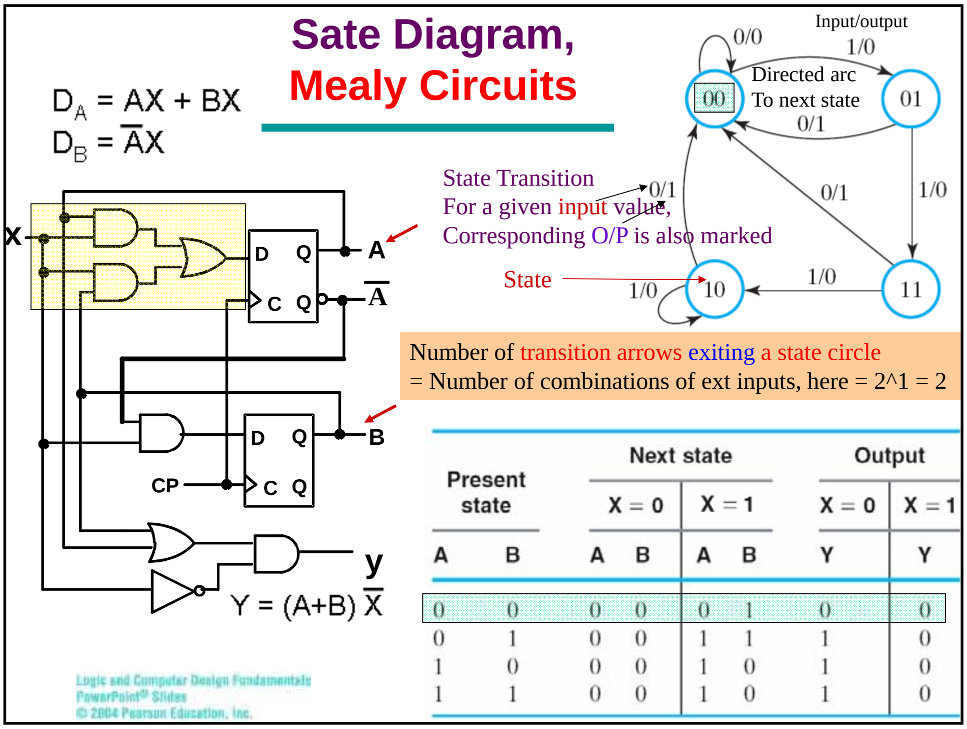
Inputs Vary

# of rows =
# of states

Next State
= f (State, I/P)

Output (Mealy)
= f (State, I/P)

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.



Sate Diagram, Mealy Circuits

$D_A = AX + BX$

$D_B = \overline{A}X$

Directed arc
To next state

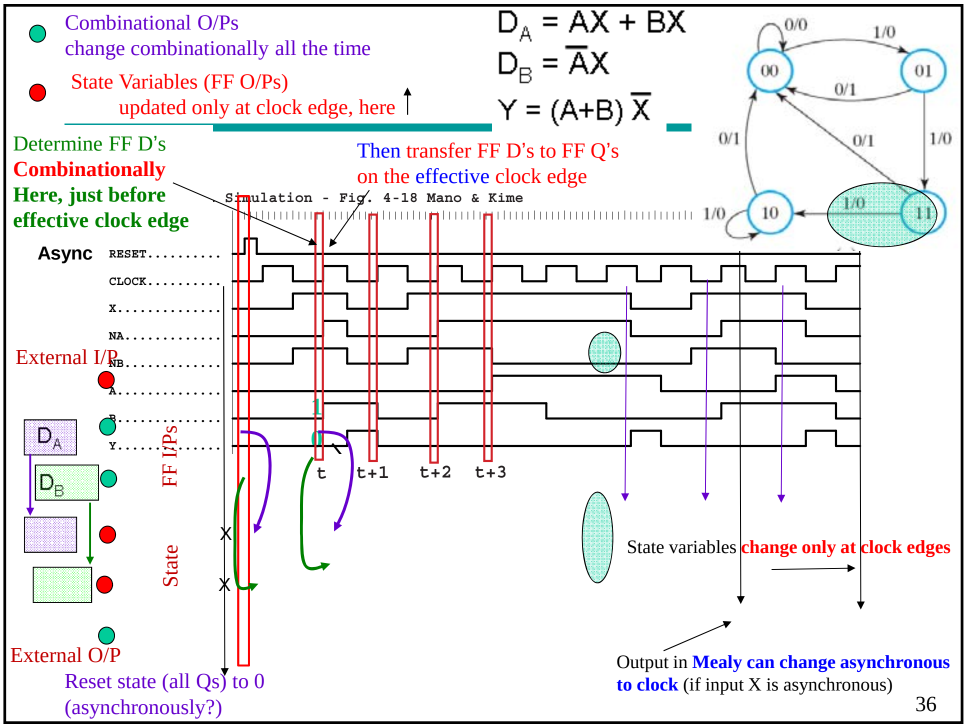
State Transition
For a given input value,
Corresponding O/P is also marked

State

Number of transition arrows exiting a state circle
= Number of combinations of ext inputs, here = 2^1 = 2

Present state		Next state				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.



● Combinational O/Ps
change combinationally all the time

● State Variables (FF O/Ps)
updated only at clock edge, here ↑

$D_A = AX + BX$

$D_B = \overline{A}X$

$Y = (A+B)\ \overline{X}$

Determine FF D's
Combinationally
Here, just before
effective clock edge

Then transfer FF D's to FF Q's
on the effective clock edge

Reset state (all Qs) to 0
(asynchronously?)

State variables **change only at clock edges**

Output in **Mealy can change asynchronous to clock** (if input X is asynchronous)

Moore and Mealy Models

- Sequential Circuits are also called ***Finite State Machines*** **(FSMs). Two formal models exist- according to dependencies of External Outputs**

- **Moore Model**
  - Named after E.F. Moore.
  - External Outputs (O/Ps) are functions of the state ONLY
- **Mealy Model**
  - Named after G. Mealy
  - External Outputs are functions of the state AND external inputs

	Moore	Mealy
In State Diagram	O/Ps are shown next to the state value (inside the state node)	O/Ps are shown on the state transition arcs- next to the I/P
In Timing Waveforms	O/Ps change only with the state → with the clock → Sync. to clk	O/Ps can change with I/Ps → possibly asynchronous to clock

- **In contemporary designs, FSM are sometimes mixed: Some O/Ps are Moore and some are Mealy**

Analysis Example 2: A Moore Circuit Output = F (States only)

- Right at the outset, there are things we can do:

  Derive Boolean expressions for all outputs of the combinational logic (CL) circuits
- These CL outputs are:

  → Inputs to the flip flops

  $D_A = X \oplus Y \oplus A$ (the odd fn)

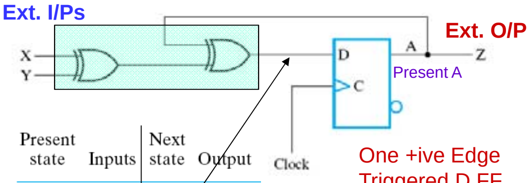
  → Output to the outside world

  $Z = A$

  Depends only on state – not on inputs, → Moore

Ext. I/Ps

Ext. O/P



One +ive Edge Triggered D FF, → $2^1 = 2$ states

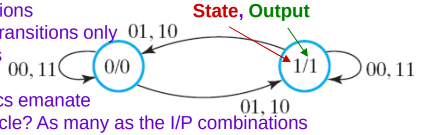
Present state	Inputs		Next state	Output
A	X	Y	Next A = D_A	Z = Present A
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Ext. O/P Determined only by the present state (not the I/Ps) → Moore

Output is determined only by the State (→ so in the circle)

State, Output

I/P combinations Affect state transitions only Not the O/Ps



How many arcs emanate From each circle? As many as the I/P combinations

8

Sequential Circuit Analysis: Using other types of Flip Flops: JK, T

- D type FF was easy to use, as the Dinput from CL is the same as the Next state bit (e.g. DA(t) = A(t+1))
- For other types of FFs, this is not so, and you will have to go through either:
  - The Characteristic Table
  - Or the Characteristic Equation

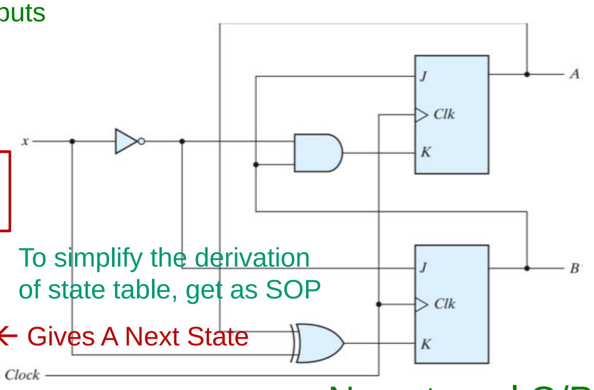
Sequential Circuit Analysis Example: Using JK Flip Flops

CL Equations of FF Inputs

$J_A = B,\ K_A = Bx'$

$J_B = x',\ K_B = A \oplus x$

JK Characteristic Eqn.
$\mathbf{Q(t+1) = J\,\overline{Q} + \overline{K}\,Q}$



To simplify the derivation of state table, get as SOP

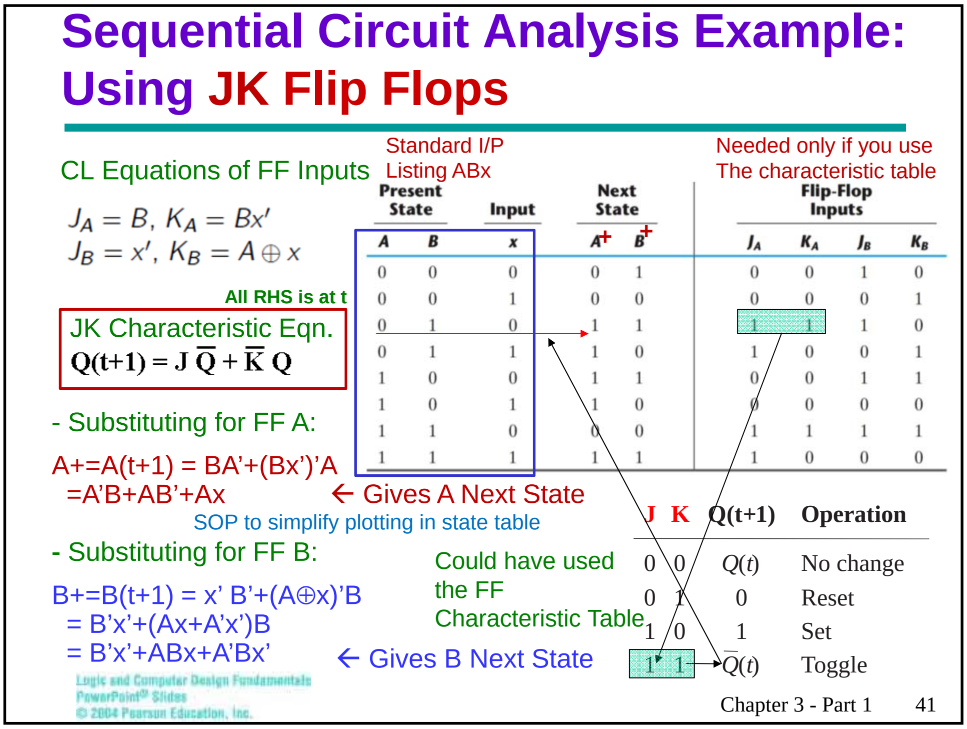
- Substituting for FF A:

A(t+1) = BA'+(Bx')'A
=A'B+AB'+Ax ← Gives A Next State

- Substituting for FF B:

No external O/Ps!

B(t+1) = x' B'+(A⊕x)'B
= B'x'+(Ax+A'x')B
= B'x'+ABx+A'Bx' ← Gives B Next State



Sequential Circuit Analysis Example: Using JK Flip Flops

CL Equations of FF Inputs

$J_A = B,\ K_A = Bx'$

$J_B = x',\ K_B = A \oplus x$

All RHS is at t

JK Characteristic Eqn.
$Q(t+1) = J\,\overline{Q} + \overline{K}\,Q$

- Substituting for FF A:

A+=A(t+1) = BA'+(Bx')'A
=A'B+AB'+Ax ← Gives A Next State

SOP to simplify plotting in state table

- Substituting for FF B:

B+=B(t+1) = x' B'+(A⊕x)'B
= B'x'+(Ax+A'x')B
= B'x'+ABx+A'Bx' ← Gives B Next State

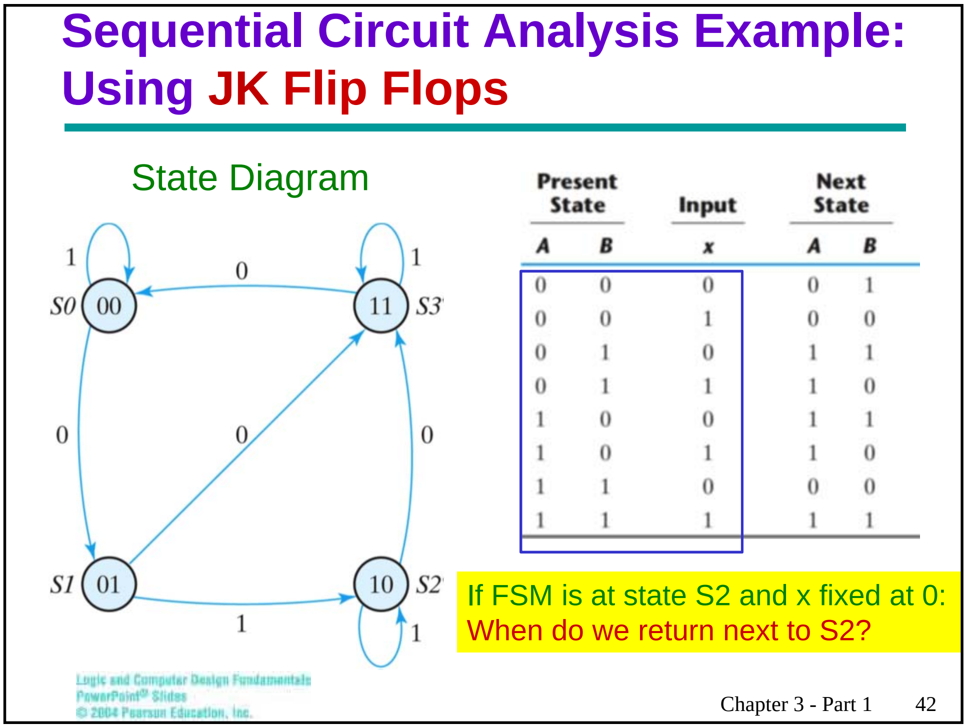
Standard I/P Listing ABx

Needed only if you use The characteristic table

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A^+	B^+	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Could have used the FF Characteristic Table

J	K	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Toggle



Sequential Circuit Analysis Example: Using JK Flip Flops

State Diagram

Present State		Input	Next State	
A	B	x	A	B
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

If FSM is at state S2 and x fixed at 0: When do we return next to S2?

Sequential Circuit Analysis Example: Using T Flip Flop

CL Equations for FF Inputs & Ext O/P

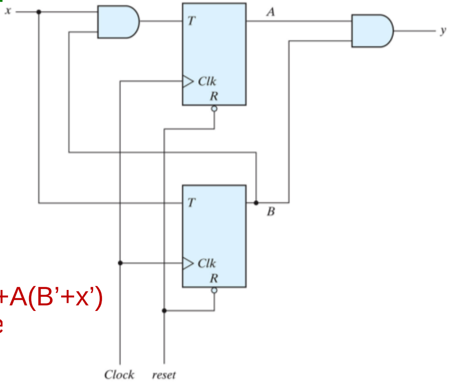
$T_A = Bx$

$T_B = x$

$y = AB$

T-FF Characteristic Eqn.

Q(t+1) = T ⊕ Q(t)



- Substituting for FF A:

A(t+1) = Bx ⊕ A = BxA'+A(BX)'=A'Bx+A(B'+x')
=A'Bx+AB'+Ax' ← A Next State

- Substituting for FF B:

B(t+1) = x⊕B ← B Next State

One external O/P
Mealy or Moore?

Sequential Circuit Analysis Example: Using T Flip Flop

CL Equations for FF Inputs & Ext O/P

$T_A = Bx$

$T_B = x$

$y = AB$

T FF Characteristic Eqn.

Q(t+1) = T ⊕ Q(t)

Present State		Input	Next State		Output
A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

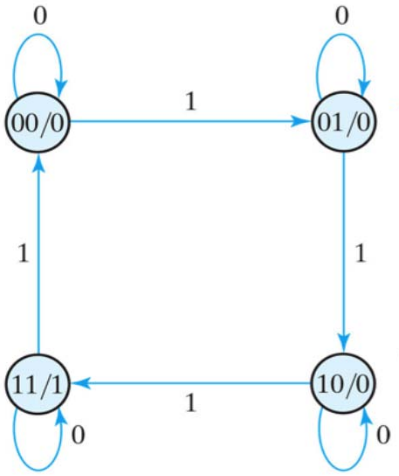
- Substituting for FF A:

A(t+1) = Bx ⊕ A
= BxA'+A(BX)'
= A'Bx+A(B'+x')
= A'Bx+AB'+Ax' ← Gives A Next State

- Substituting for FF B:

B(t+1) = x⊕B ← Gives B Next State

Sequential Circuit Analysis Example: Using T Flip Flop



Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

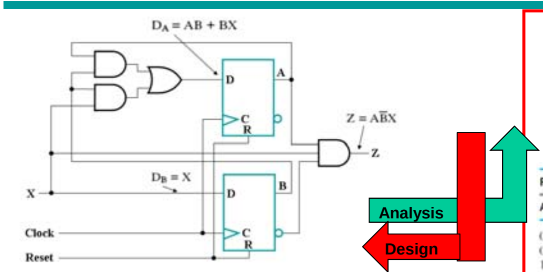
Note how O/P y is represented in the state diagram

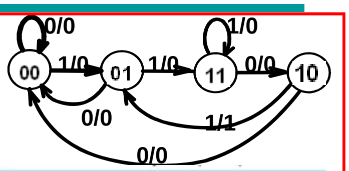
Sequential Circuit Design: The Design Procedure

Steps in green are similar to CL Desing

1. Specification (Word Description) → State Diagram (Can be **symbolic** at this “thinking” stage)
2. State **Reduction**: Try to reduce the number of states (Will cover later)
   - This may reduce the number of FFs required 10 → 8?
3. State **Assignment** - Assign binary codes to the symbolic states
4. Obtain a **<u>Binary</u>** <u>state table.</u>
5. For the selected type of Flip Flops: Use the excitation table to obtain the binary **FF inputs**
6. Derive Optimized logic expressions for **each of** the:
   - FF Inputs
   - External Outputs
7. Generate the logic diagram for the **complete sequential cct**.

Sequential Circuits
Analysis Versus Design





Present State	Next State		Output Z	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

↑ 2-D
1-D ↓

- **Analysis of a given circuit:**
  Given a circuit → behavior [state table (state diagram)]:
  {Circuit, Present state, inputs} → Next state?, Outputs?

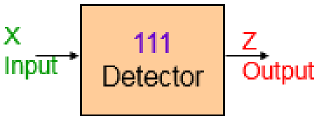
  Flip Flop Consideration: **(inputs→ outputs?)**
  → FF: Use input-driven Characteristic tables/equations

For D flip flops:
D_A (t) A (t+1)
D_B (t) B (t+1)
➡ Analysis
⬅ Design

- **Design to achieve a specified circuit performance**
  Given desired behavior [(state diagram (State table)] → get circuit (behavior: Present to next changes → Needed FF Inputs? → CL circuit?)
  Flip Flop Considerations: **(behavior at O/P → inputs that give behavior?)**
  → FF: Use output-driven Excitation tables/Equations

Complete Design Procedure
Example: “3 or more successive 1s” detector

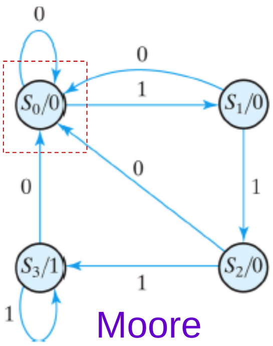
Specification (word description) → Symbolic State Diagram



I/P: 001101011110
O/P: 000000000110

Detect three or more consecutive 1's at the circuit inpu

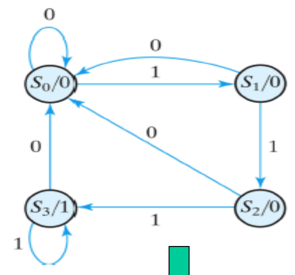
- If input = 0, stay/go at/to a state_0/reset state.
- If input = 1 after 0, go to state_1.
- If input = 1 after 01, go to state_2.
- If input = 1 after 11, go to state_3.
- We have 4 states (reset, state_1, state_2, state_3) ‹ (S_0, S_1, S_2, S_3) respectively.



Square 1

Moore

3 1s Detector Example: with D-type FFs



Circuit Details:

- Two D flip-flops (A, B) to represent the four states.
- One input (x).
- One output (y).
- $Q_{(t+1)} = D_Q$ (D- FF)

Present State	Input	Next State	Tied to State Output
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_0	0
S_1	1	S_2	0
S_2	0	S_0	0
S_2	1	S_3	0
S_3	0	S_0	1
S_3	1	S_3	1

State Assignment

	Present State		Input	Next State		Output
	A	*B*	*x*	*A* =DA	*B* =DB	*y*
S_0	0	0 ●	0	0	0 ●	0
	0	0 ●	1	0	1	0
	0	1	0	0	0 ●	0
	0	1	1	1	0	0
	1	0	0	0	0 ●	0
	1	0	1	1	1	0
	1	1	0	0	0 ●	1
	1	1	1	1	1	1

Li
PowerPoint® Slides
© 2004 Pearson Education, Inc.

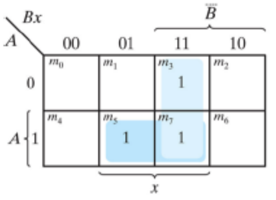
3 1s Detector Example

Present State		Input	Next State		Output
A	*B*	*x*	*A* D_A	*B* D_B	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

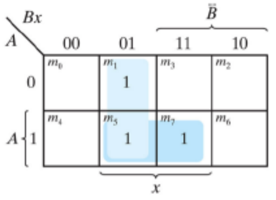
Combinational Logic Design

- $D_A(A, B, x) = \sum(3, 5, 7)$.
- $D_B(A, B, x) = \sum(1, 5, 7)$.
- $y(A, B, x) = \sum(6, 7)$.

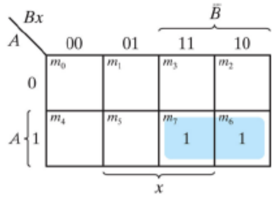
Optimized implementation
Obtained from 3 K-maps



$D_A = Ax + Bx$

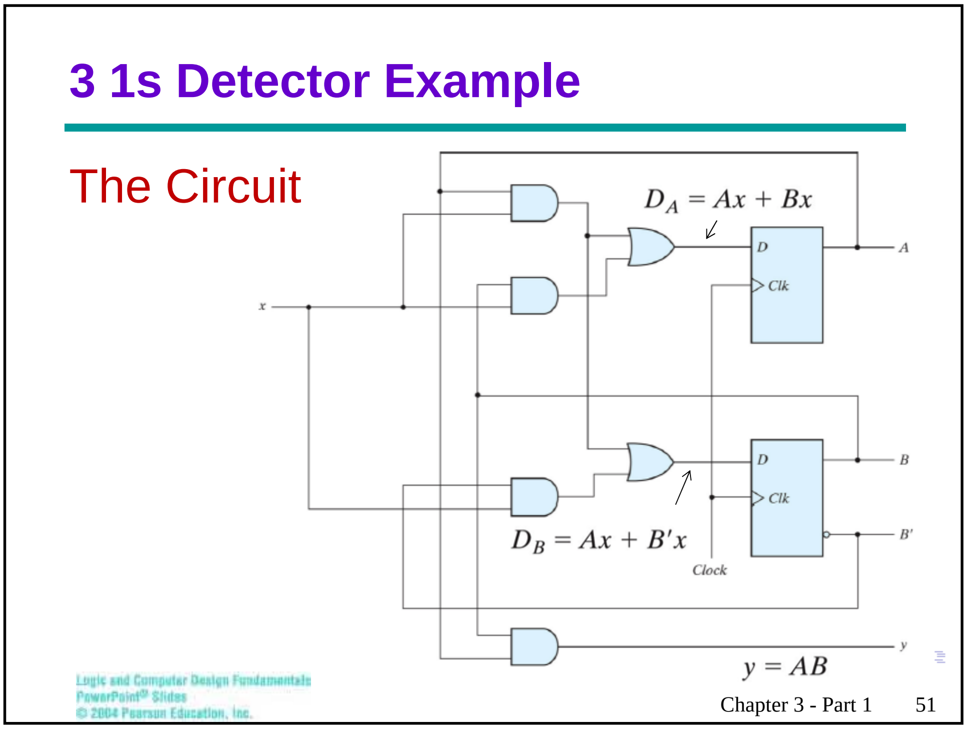


$D_B = Ax + B'x$



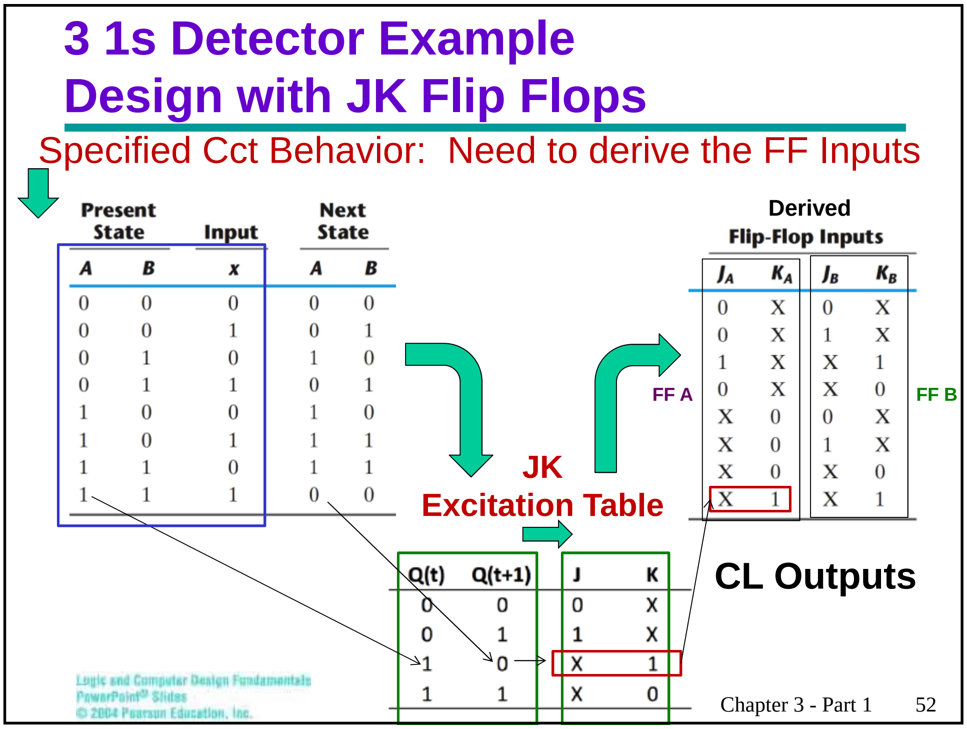
$y = AB$

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.



3 1s Detector Example

The Circuit



3 1s Detector Example
Design with JK Flip Flops

Specified Cct Behavior: Need to derive the FF Inputs

Present State		Input	Next State		Derived Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

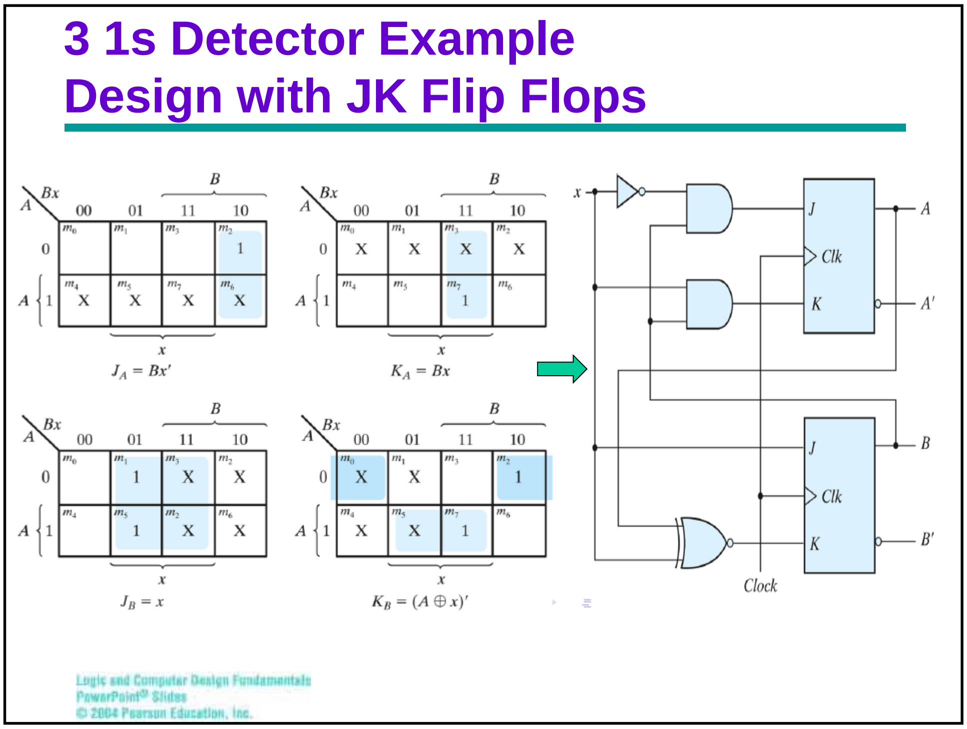
FF A

FF B

JK
Excitation Table

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

CL Outputs



3 1s Detector Example
Design with JK Flip Flops

J_A (A \ Bx):

A \ Bx	00	01	11	10
0				1
1	X	X	X	X

$J_A = Bx'$

K_A (A \ Bx):

A \ Bx	00	01	11	10
0	X	X	X	X
1			1	

$K_A = Bx$

J_B (A \ Bx):

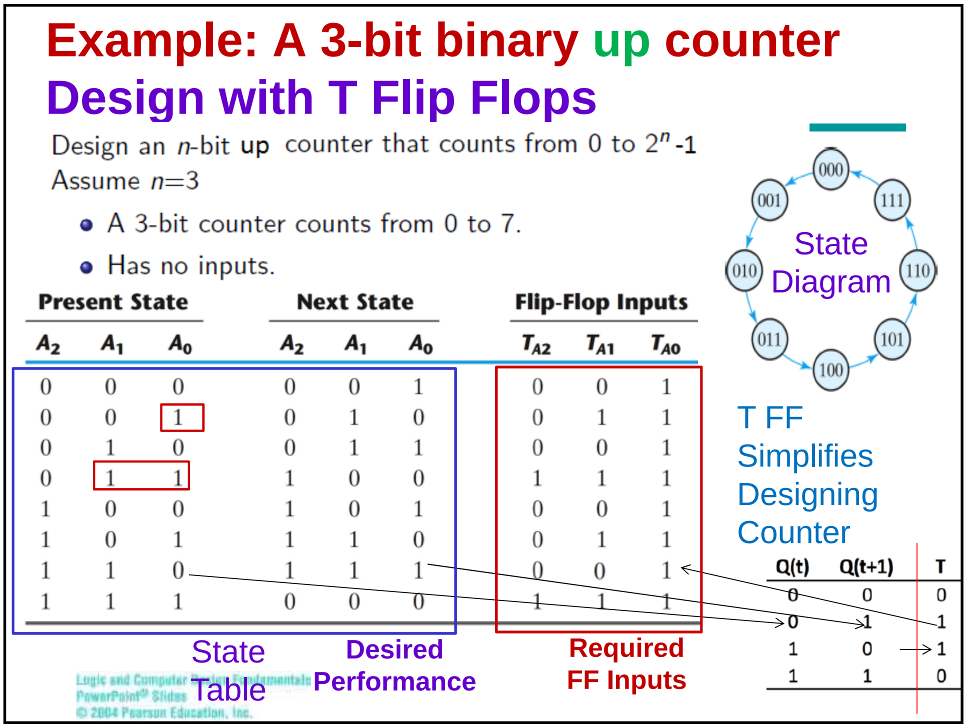
A \ Bx	00	01	11	10
0		1	X	X
1		1	X	X

$J_B = x$

K_B (A \ Bx):

A \ Bx	00	01	11	10
0	X	X		1
1	X	X	1	

$K_B = (A \oplus x)'$



Example: A 3-bit binary up counter
Design with T Flip Flops

Design an *n*-bit up counter that counts from 0 to 2^n-1

Assume *n*=3

- A 3-bit counter counts from 0 to 7.
- Has no inputs.

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

State Table

Desired Performance

Required FF Inputs

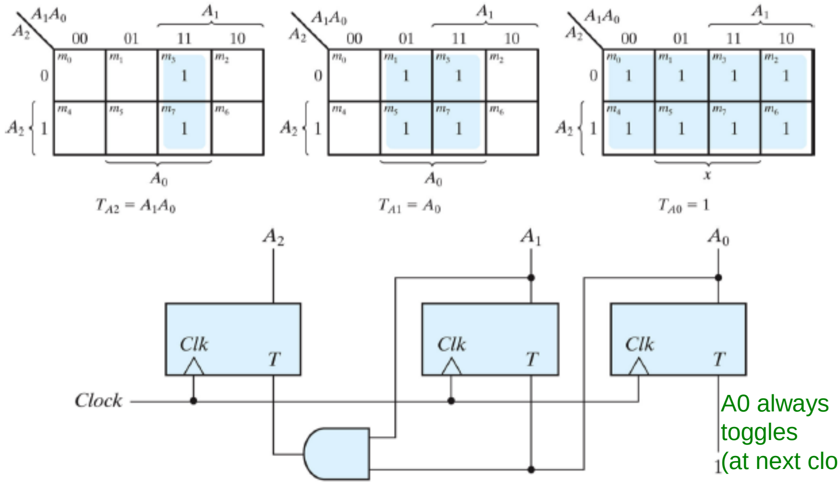
State Diagram

T FF Simplifies Designing Counter

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Design with T Flip Flops

Example: A 3-bit binary counter



A_2 \ A_1A_0	00	01	11	10
0			1	
1			1	

$T_{A2} = A_1A_0$

A_2 \ A_1A_0	00	01	11	10
0		1	1	
1		1	1	

$T_{A1} = A_0$

A_2 \ A_1A_0	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$T_{A0} = 1$

A0 always toggles (at next clock)

A2 toggles (at next clock) If present A0, A1 = 1

A1 toggles (at next clock) if present A0 = 1

State Reduction

When we are interested only in the input-output sequence and not in the state values themselves (in counters)

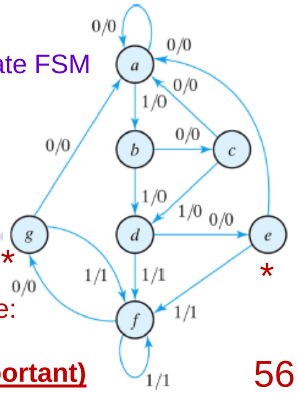
- Reducing the number of states may result in reduction in the number of flip-flops needed. (without affecting the in-out sequence)
- Sometimes, reduction in flip-flops result in a bigger combinational circuit to realize the next state and the outputs.

Input Sequence

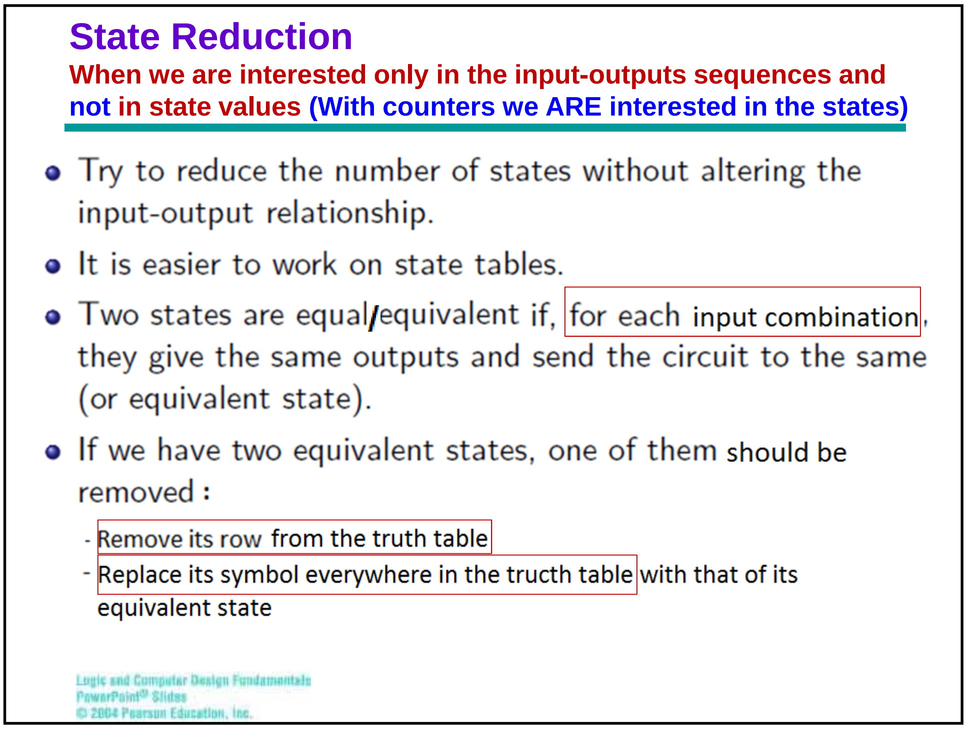
7-State FSM

- Consider the '01010110100' as your input

State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	



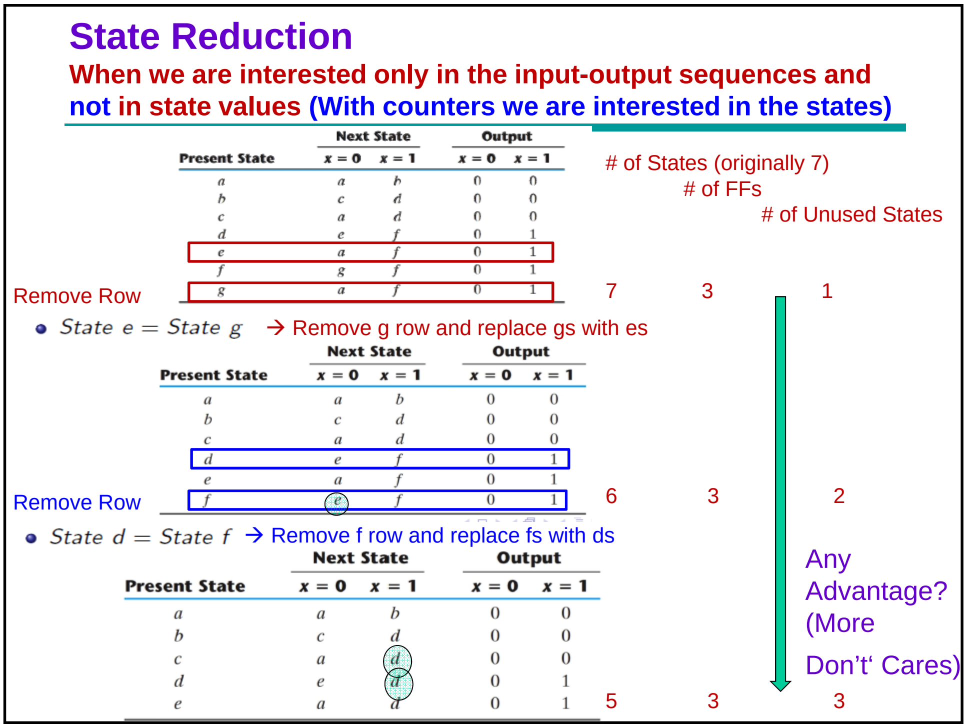
States e, g are **equivalent**, Same:
-Next state - O/P
for all input combinations (Important)



State Reduction

When we are interested only in the input-outputs sequences and not in state values (With counters we ARE interested in the states)

- Try to reduce the number of states without altering the input-output relationship.
- It is easier to work on state tables.
- Two states are equal/equivalent if, for each input combination, they give the same outputs and send the circuit to the same (or equivalent state).
- If we have two equivalent states, one of them should be removed :
  - Remove its row from the truth table
  - Replace its symbol everywhere in the tructh table with that of its equivalent state



State Reduction

When we are interested only in the input-output sequences and not in state values (With counters we are interested in the states)

Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Remove Row

# of States (originally 7)	# of FFs	# of Unused States
7	3	1
6	3	2
5	3	3

- *State* e = *State* g → Remove g row and replace gs with es

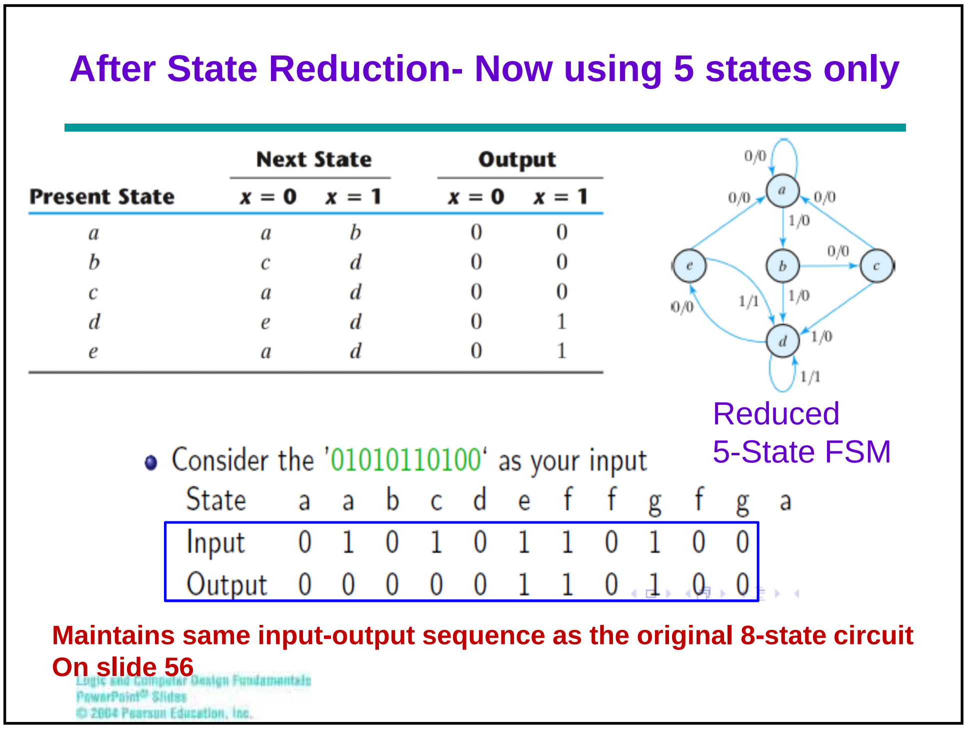
Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Remove Row

- *State* d = *State* f → Remove f row and replace fs with ds

Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Any Advantage? (More Don't' Cares)



After State Reduction- Now using 5 states only

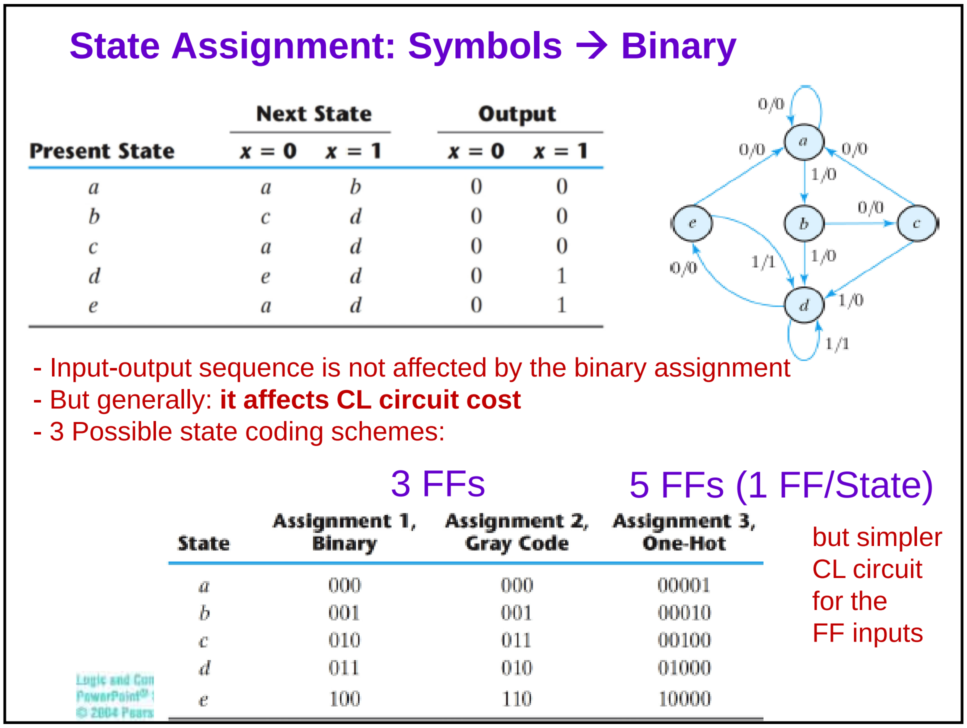
Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Reduced
5-State FSM

- Consider the '01010110100‘ as your input

State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

Maintains same input-output sequence as the original 8-state circuit On slide 56



State Assignment: Symbols → Binary

Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

- Input-output sequence is not affected by the binary assignment
- But generally: **it affects CL circuit cost**
- 3 Possible state coding schemes:

3 FFs

5 FFs (1 FF/State)

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

but simpler CL circuit for the FF inputs