Logic and Computer Design Fundamentals

Chapter 5 – Sequential Circuits

Part 1 – Storage Elements and Sequential Circuit Analysis

Charles Kime & Thomas Kaminski

© 2008 Pearson Education, Inc.
(Hyperlinks are active in View Show mode)

Overview

- **Part 1 - Storage Elements and Analysis**
  - **Introduction to sequential circuits**
  - **Types of sequential circuits**
  - **Storage elements**
    - **Latches**
    - **Flip-flops**
  - **Sequential circuit analysis**
    - **State tables**
    - **State diagrams**
    - **Equivalent states**
    - **Moore and Mealy Models**
- **Part 2 - Sequential Circuit Design**
- **Part 3 – State Machine Design**

1

Introduction to Sequential Circuits

- **A Sequential circuit contains:**
  - Storage elements: Latches or Flip-Flops
  - Combinational Logic:
    - Implements a multiple-output switching function
    - <u>Inputs</u> are signals from the outside.
    - <u>Outputs</u> are signals to the outside.
    - Other inputs, <u>State</u> or <u>Present State</u>, are signals from storage elements.
    - The remaining outputs, <u>Next State</u> are inputs to storage elements.
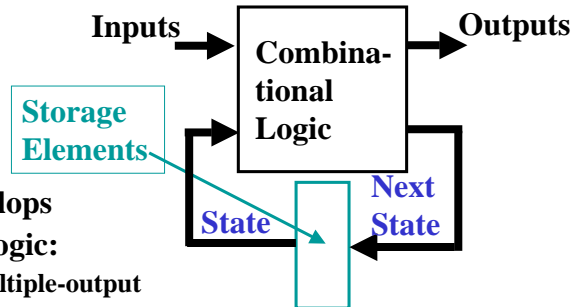
Introduction to Sequential Circuits

- **Combinatorial Logic**
  - *Next state function*
    Next State = f(Inputs, State)
  - *Output function* (Mealy)
    Outputs = g(Inputs, State)
  - *Output function* (Moore)
    Outputs = h(State)
- **Output function type depends on specification and affects the design significantly**
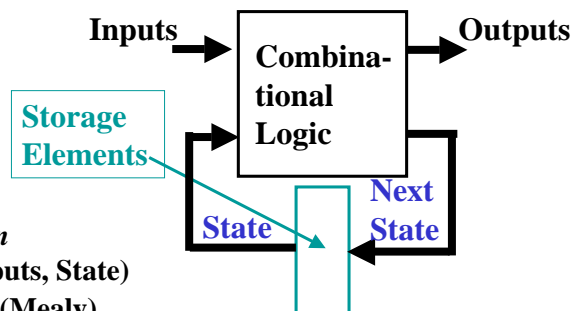
Types of Sequential Circuits

- **Depends on the <u>times</u> at which:**
  - storage elements observe their inputs, and
  - storage elements change their state
- <u>**Synchronous**</u>
  - Behavior defined from knowledge of its signals at <u>discrete</u> instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (<u>clock pulses</u> from a <u>clock</u>)
- <u>**Asynchronous**</u>
  - Behavior defined from knowledge of inputs an any instant of time and the order in continuous time in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!
  - Nevertheless, the synchronous abstraction makes complex designs tractable!

Discrete Event Simulation

- **In order to understand the time behavior of a sequential circuit we use <u>discrete event simulation</u>.**
- **Rules:**
  - Gates modeled by an <u>ideal</u> (instantaneous) function and a <u>fixed gate delay</u>
  - Any <u>change in input values</u> is evaluated to see if it causes a <u>change in output value</u>
  - Changes in output values are scheduled for the fixed gate delay after the input change
  - At the time for a scheduled output change, the output value is changed along with any inputs it drives

Simulated NAND Gate

- **Example: A 2-Input NAND gate with a 0.5 ns. delay:**

F(Instantaneous)

A ————
B ————
[NAND gate] o— **DELAY 0.5 ns.** —— F
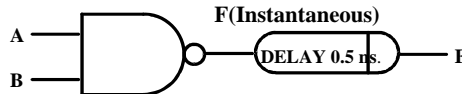
- **Assume A and B have been 1 for a long time**
- **At time t=0, A changes to a 0 at t= 0.8 ns, back to 1.**

t (ns)	A	B	F(I)	F	Comment
$-\infty$	1	1	0	0	A=B=1 for a long time
0	1⇒0	1	1⇐0	0	F(I) changes to 1
0.5	0	1	1	1⇐0	F changes to 1 after a 0.5 ns delay
0.8	1⇐0	1	1⇒0	1	F(Instantaneous) changes to 0
0.13	1	1	0	1⇒0	F changes to 0 after a 0.5 ns delay

Gate Delay Models

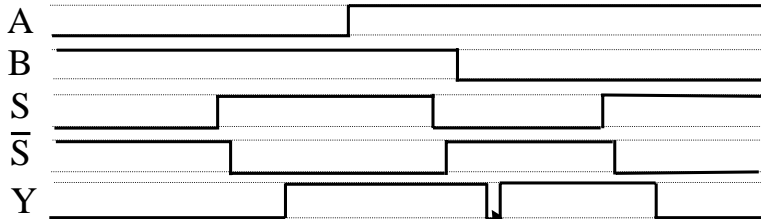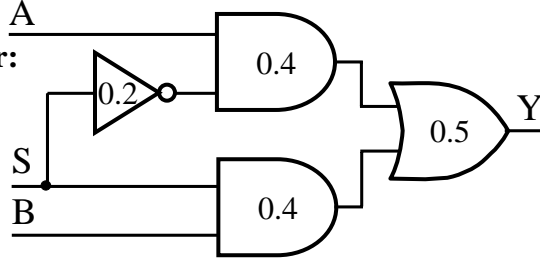- **Suppose gates with delay *n* ns are represented for *n* = 0.2 ns, *n* = 0.4 ns, *n* = 0.5 ns, respectively:**
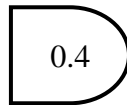
0.2 [inverter symbol] o

0.4 [AND gate symbol]

0.5 [OR gate symbol]

4

Circuit Delay Model

- **Consider a simple 2-input multiplexer:**
- **With function:**
  - Y = A for S = 1
  - Y = B for S = 0

- **"Glitch" is due to delay of inverter**

Storing State

- **What if A connected to Y?**
- **Circuit becomes:**
- **With function:**
  - Y = B for S = 1, and Y(t) dependent on Y(t – 0.9) for S = 0
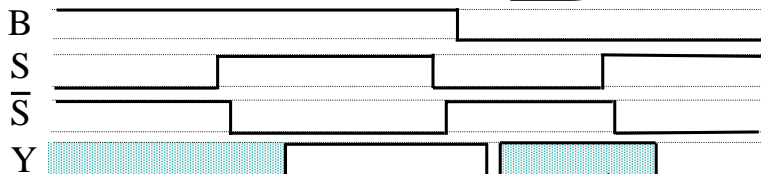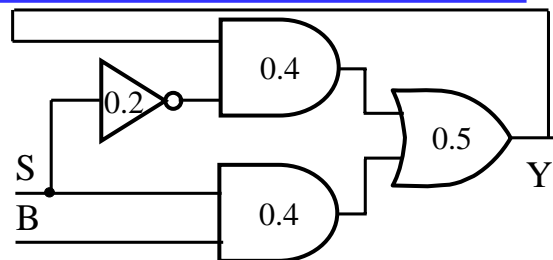
- **The simple <u>combinational circuit</u> has now become a <u>sequential circuit</u> because its output is a function of a time sequence of input signals!**

**Y is stored value in shaded area**

Storing State (Continued)

- **Simulation example as input signals change with time. Changes occur every 100 ns, so that the tenths of ns delays are negligible.**

Time →

B	S	Y	Comment
1	0	0	Y "remembers" 0
1	1	1	Y = B when S = 1
1	0	1	Now Y "remembers" B = 1 for S = 0
0	0	1	No change in Y when B changes
0	1	0	Y = B when S = 1
0	0	0	Y "remembers" B = 0 for S = 0
1	0	0	No change in Y when B changes

- **Y represent the <u>state</u> of the circuit, not just an output.**

Storing State (Continued)

- **Suppose we place an inverter in the "feedback path."**
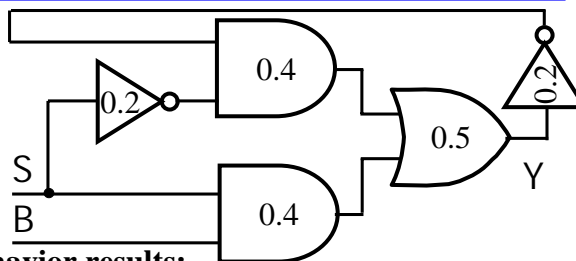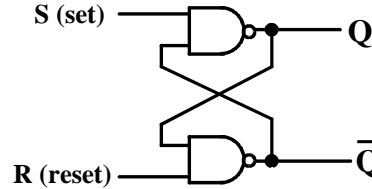

- **The following behavior results:**
- **The circuit is said to be <u>unstable</u>.**
- **For S = 0, the circuit has become what is called an *oscillator*. Can be used as crude <u>clock</u>.**

B	S	Y	Comment
0	1	0	Y = B when S = 1
1	1	1	
1	0	1	Now Y "remembers" A
1	0	0	Y, 1.1 ns later
1	0	1	Y, 1.1 ns later
1	0	0	Y, 1.1 ns later

6

Basic (NAND) $\bar{S}$ – $\bar{R}$ Latch

- **"Cross-Coupling" two NAND gates gives the $\bar{S}$ -$\bar{R}$ Latch:**
- **Which has the time sequence behavior:**

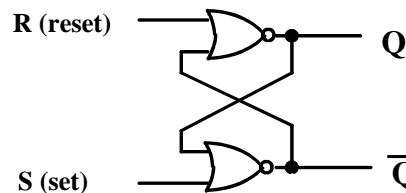- **$S = 0, R = 0$ is <u>forbidden</u> as input pattern**

S (set) ○ Q

R (reset) ○ $\bar{Q}$

Time

R	S	Q	$\bar{Q}$	Comment
1	1	?	?	Stored state unknown
1	0	1	0	"Set" Q to 1
1	1	1	0	Now Q "remembers" 1
0	1	0	1	"Reset" Q to 0
1	1	0	1	Now Q "remembers" 0
0	0	1	1	Both go high
1	1	?	?	Unstable!

Basic (NOR) S – R Latch

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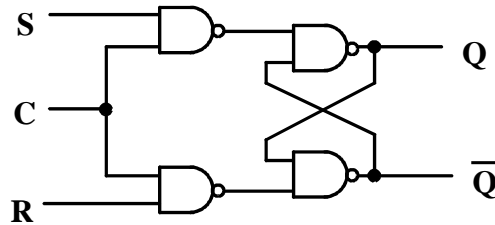- **Which has the time sequence behavior:**

R (reset) ○ Q

S (set) ○ $\bar{Q}$

Time

| R	S	Q	$\bar{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	"Set" Q to 1
0	0	1	0	Now Q "remembers" 1
1	0	0	1	"Reset" Q to 0
0	0	0	1	Now Q "remembers" 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

Clocked S - R Latch

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

- **Has a time sequence behavior similar to the basic S-R latch <u>except that</u> the S and R inputs are only observed when the line C is high.**
- **C means "control" or "clock".**

Clocked S - R Latch (continued)

- **The Clocked S-R Latch can be described by a table:**
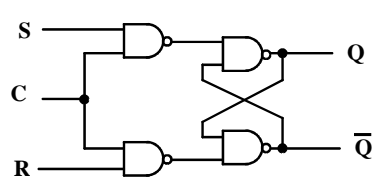
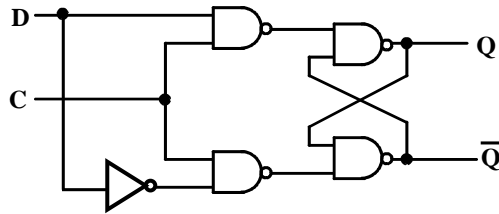| Q(t)	S	R	Q(t+1)	Comment
0	0	0	0	No change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	Indeterminate
1	0	0	1	No change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	Indeterminate

- **The table describes what happens after the clock [at time (t+1)] based on:**
  - **current inputs (S,R) and**
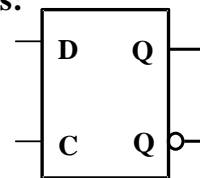 - **current state Q(t).**

8

D Latch

- **Adding an inverter to the S-R Latch, gives the D Latch:**
- **Note that there are no "indeterminate" states!**

The graphic symbol for a D Latch is:

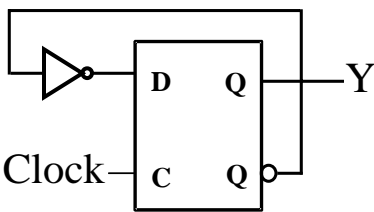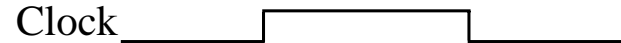| Q	D	Q(t+1)	Comment
0	0	0	No change
0	1	1	Set Q
1	0	0	Clear Q
1	1	1	No Change

Flip-Flops

- **The latch timing problem**
- **Master-slave flip-flop**
- **Edge-triggered flip-flop**
- **Standard symbols for storage elements**
- **Direct inputs to flip-flops**

9

The Latch Timing Problem

- **In a sequential circuit, paths may exist through combinational logic:**
  - **From one storage element to another**
  - **From a storage element back to the same storage element**
- **The combinational logic between a latch output and a latch input may be as simple as an interconnect**
- **For a clocked D-latch, the output Q depends on the input D whenever the clock input C has value 1**

The Latch Timing Problem (continued)

- **Consider the following circuit:**

- **Suppose that initially Y = 0.**

- **As long as C = 1, the value of Y continues to change!**
- **The changes are based on the delay present on the loop through the connection from Y back to Y.**
- **This behavior is clearly unacceptable.**
- **Desired behavior: Y changes only once per clock pulse**

10

The Latch Timing Problem (continued)

- **A solution to the latch timing problem is to <u>break</u> the closed path from Y to Y within the storage element**
- **The commonly-used, path-breaking solutions replace the clocked D-latch with:**
    - **a master-slave flip-flop**
    - **an edge-triggered flip-flop**

S-R Master-Slave Flip-Flop

- **Consists of two clocked S-R latches in series with the clock on the second latch inverted**
- **The input is observed by the first latch with C = 1**
- **The output is changed by the second latch with C = 0**
- **The path from input to output is broken by the difference in clocking values (C = 1 and C = 0).**
- **The behavior demonstrated by the example with D driven by Y given previously is prevented since the clock must change from 1 to 0 before a change in Y based on D can occur.**

11
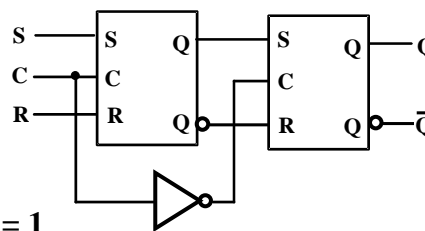
Flip-Flop Problem

- **The change in the flip-flop output is delayed by the pulse width which makes the circuit slower or**
- **S and/or R are permitted to change while C = 1**
  - **Suppose Q = 0 and S goes to 1 and then back to 0 with R remaining at 0**
    - **The master latch sets to 1**
    - **A 1 is transferred to the slave**
  - **Suppose Q = 0 and S goes to 1 and back to 0 and R goes to 1 and back to 0**
    - **The master latch sets and then resets**
    - **A 0 is transferred to the slave**
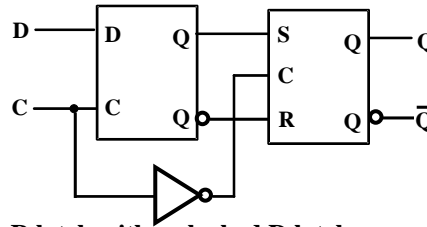 - **This behavior is called *1s catching***

Logic and Computer Design Fundamentals, 4e
PowerPoint® Slides
© 2008 Pearson Education, Inc.

Chapter 5 - Part 1   23

Flip-Flop Solution

- **Use edge-triggering instead of master-slave**
- **An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a <u>transition</u> of the clock signal**
- **Edge-triggered flip-flops can be built directly at the electronic circuit level, or**
- **A <u>master-slave</u> D flip-flop which also exhibits <u>edge-triggered behavior</u> can be used.**

Logic and Computer Design Fundamentals, 4e
PowerPoint® Slides
© 2008 Pearson Education, Inc.

Chapter 5 - Part 1   24

Edge-Triggered D Flip-Flop

- **The edge-triggered D flip-flop is the same as the master-slave D flip-flop**

- **It can be formed by:**
  - **Replacing the first clocked S-R latch with a clocked D latch or**
  - **Adding a D input and inverter to a master-slave S-R flip-flop**
- **The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs**
- **The change of the D flip-flop output is associated with the negative edge at the end of the pulse**
- **It is called a *negative-edge triggered* flip-flop**

Positive-Edge Triggered D Flip-Flop

- **Formed by adding inverter to clock input**
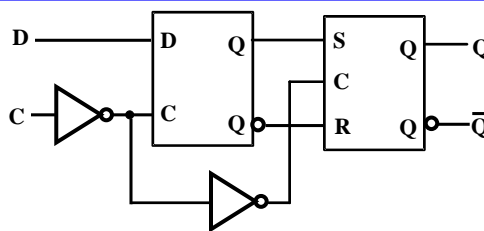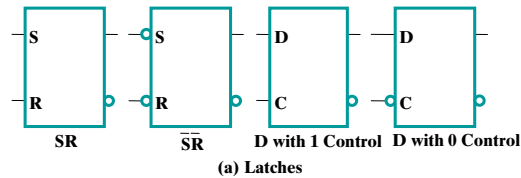
- **Q changes to the value on D applied at the positive clock edge within timing constraints to be specified**
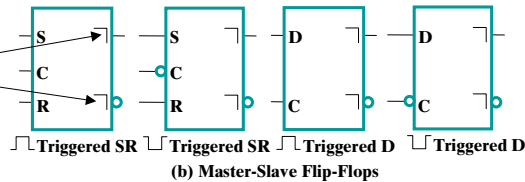- **Our choice as the <u>standard flip-flop</u> for most sequential circuits**

Standard Symbols for Storage Elements

SR $\overline{\text{SR}}$ **D with 1 Control**   **D with 0 Control**

(a) Latches

- **Master-Slave: Postponed output indicators**

Triggered SR   **Triggered SR**   **Triggered D**   **Triggered D**

(b) Master-Slave Flip-Flops

- **Edge-Triggered: Dynamic indicator**
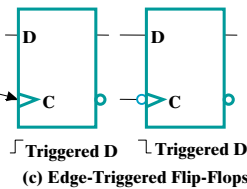
Triggered D   **Triggered D**

(c) Edge-Triggered Flip-Flops

Direct Inputs

- **At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation**
- **This initialization is often done outside of the clocked behavior of the circuit, i.e., asynchronously.**
- **Direct R and/or S inputs that control the state of the latches within the flip-flops are used for this initialization.**
- **For the example flip-flop shown**
  - **0 applied to $\overline{\text{R}}$ resets the flip-flop to the 0 state**
  - **0 applied to $\overline{\text{S}}$ sets the flip-flop to the 1 state**
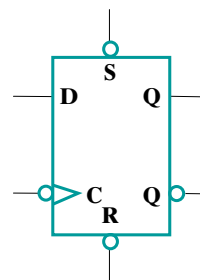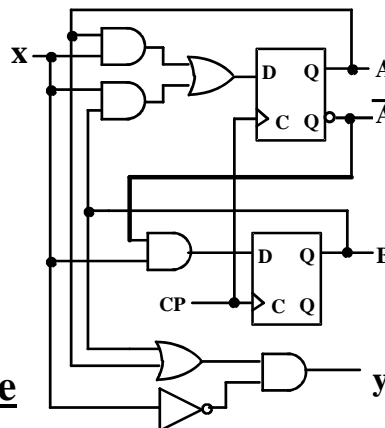
14

Sequential Circuit Analysis

- **General Model**
  - **Current State at time (t) is stored in an array of flip-flops.**

  - **Next State at time (t+1) is a Boolean function of State and Inputs.**
  - **Outputs at time (t) are a Boolean function of State (t) and (sometimes) Inputs (t).**
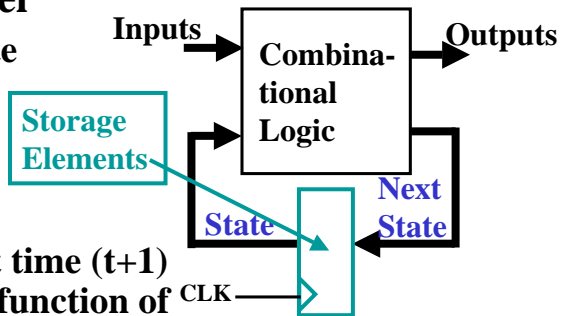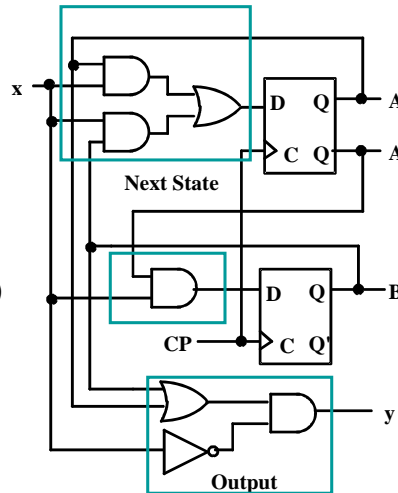

Example 1 (from Fig. 5-15)

- **Input:**   **x(t)**
- **Output:**   **y(t)**
- **State:**   **(A(t), B(t))**
- **What is the Output Function?**

- **What is the Next State Function?**

15

Example 1 (from Fig. 5-15) (continued)

- **Boolean equations for the functions:**
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = \overline{A}(t)x(t)$
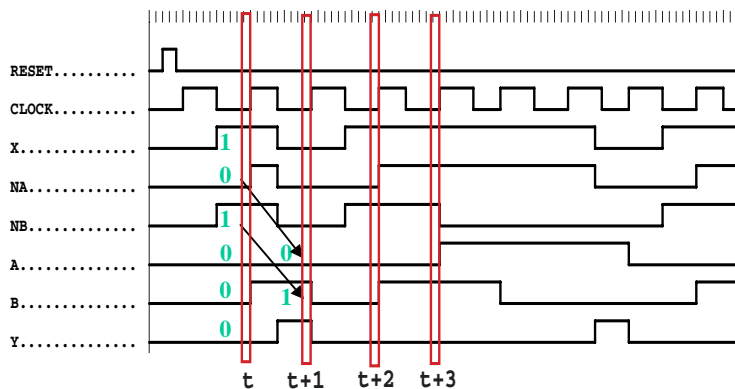 - $y(t) = \overline{x}(t)(B(t) + A(t))$

Example 1(from Fig. 5-15) (continued)

- **Where in time are inputs, outputs and states defined?**

16

State Table Characteristics

- *State table* – **a multiple variable table with the following four sections:**
  - *Present State* – **the values of the state variables for each allowed state.**
  - *Input* – **the input combinations allowed.**
  - *Next-state* – **the value of the state at time (t+1) based on the <u>present state</u> and the <u>input</u>.**
  - *Output* – **the value of the output as a function of the <u>present state</u> and (sometimes) the <u>input</u>.**
- **From the viewpoint of a truth table:**
  - **the inputs are Input, Present State**
  - **and the outputs are Output, Next State**

Example 1: State Table (from Fig. 5-15)

- **The state table can be filled in using the next state and output equations: $A(t+1) = A(t)x(t) + B(t)x(t)$ $B(t+1)$**
  $= \bar{A}(t)x(t)$ $y(t)$
  $= \bar{x}(t)(B(t) + A(t))$

Present State	Input	Next State		Output
A(t) B(t)	x(t)	A(t+1)	B(t+1)	y(t)
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

17

Example 1: Alternate State Table

- **2-dimensional table that matches well to a K-map. Present state rows and input columns in Gray code order.**
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = \overline{A}(t)x(t)$
  - $y(t) = \overline{x}(t)(B(t) + A(t))$

Present State	Next State		Output	
	x(t)=0	x(t)=1	x(t)=0	x(t)=1
A(t) B(t)	A(t+1)B(t+1)	A(t+1)B(t+1)	y(t)	y(t)
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

State Diagrams

- **The sequential circuit function can be represented in graphical form as a <u>state diagram</u> with the following components:**
  - A <u>circle</u> with the state name in it for each state
  - A <u>directed arc</u> from the <u>Present State</u> to the <u>Next State</u> for each <u>state transition</u>
  - A label on each <u>directed arc</u> with the <u>Input</u> values which causes the <u>state transition</u>, and
  - A label:
    - On each <u>circle</u> with the <u>output</u> value produced, or
   - On each <u>directed arc</u> with the <u>output</u> value produced.

18

State Diagrams

- **Label form:**
  - **On <u>circle</u> with output included:**
    - state/output
    - Moore type output depends only on state
  - **On <u>directed arc</u> with the <u>output</u> included:**
    - input/output
    - Mealy type output depends on state and input

Example 1: State Diagram

- **Which type?**
- **Diagram gets confusing for large circuits**
- **For small circuits, usually easier to understand than the state table**

19
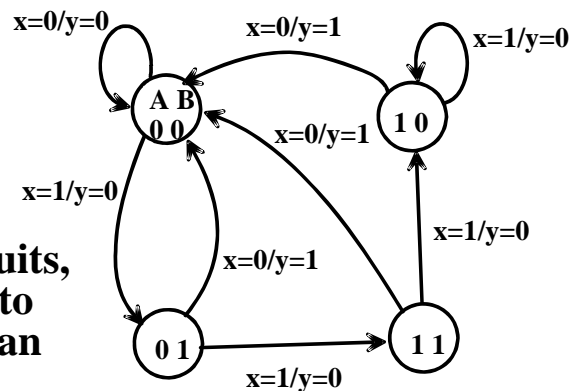
Equivalent State Definitions

- **Two states are *equivalent* if their response for each possible input sequence is an identical output sequence.**

- **Alternatively, two states are *equivalent* if their outputs produced for each input symbol is identical and their next states for each input symbol are the same or equivalent.**

Equivalent State Example

- **Text Figure 5-17(a):**
- **For states S3 and S2,**
  - **the output for input 0 is 1 and input 1 is 0, and**
  - **the next state for input 0 is S0 and for input 1 is S2.**
  - **By the alternative definition, states S3 and S2 are equivalent.**
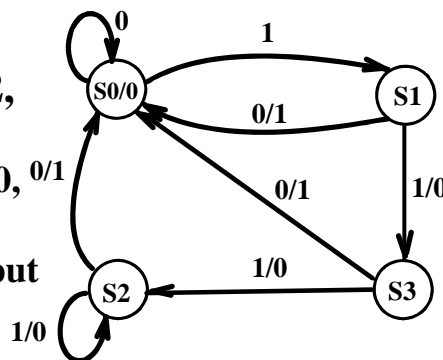
Equivalent State Example

- **Replacing S3 and S2 by a single state gives state diagram:**
- **Examining the new diagram, states S1 and S2 are equivalent since**
  - **their outputs for input 0 is 1 and input 1 is 0, and**
  - **their next state for input 0 is S0 and for input 1 is S2,**
- **Replacing S1 and S2 by a single state gives state diagram:**
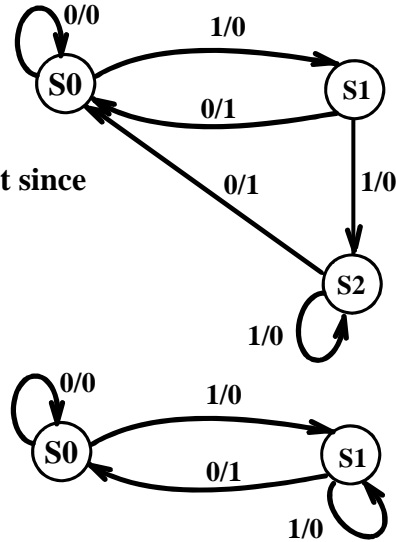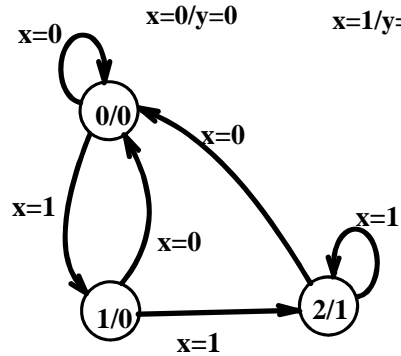
Moore and Mealy Models

- **Sequential Circuits or Sequential Machines are also called *Finite State Machines* (FSMs). Two formal models exist:**
- **Moore Model**
  - **Named after E.F. Moore**
  - **Outputs are a function ONLY of <u>states</u>**
  - **Usually specified on the states.**

- **Mealy Model**
  - **Named after G. Mealy**
  - **Outputs are a function of <u>inputs</u> AND <u>states</u>**
  - **Usually specified on the state transition arcs.**

21

Moore and Mealy Example Diagrams

- **Mealy Model State Diagram maps <u>inputs and state</u> to <u>outputs</u>**

x=1/y=0

x=0/y=0   (0)   (1)

x=0/y=0   x=1/y=1
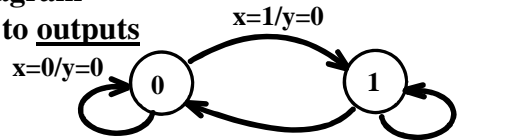
- **Moore Model State Diagram maps <u>states</u> to <u>outputs</u>**

x=0

(0/0)   x=0

x=1   x=0   x=1

(1/0)   (2/1)

x=1

Moore and Mealy Example Tables

- **Moore Model state table maps state to outputs**

Present State	Next State x=0 x=1	Output
0	0 1	0
1	0 2	0
2	0 2	1

- **Mealy Model state table maps inputs and state to outputs**

Present State	Next State x=0 x=1	Output x=0 x=1
0	0 1	0 0
1	0 1	0 1

Mixed Moore and Mealy Outputs

- **In real designs, some outputs may be Moore type and other outputs may be Mealy type.**
- **Example: Figure 5-17(a) can be modified to illustrate this**
  - **State 00: Moore**
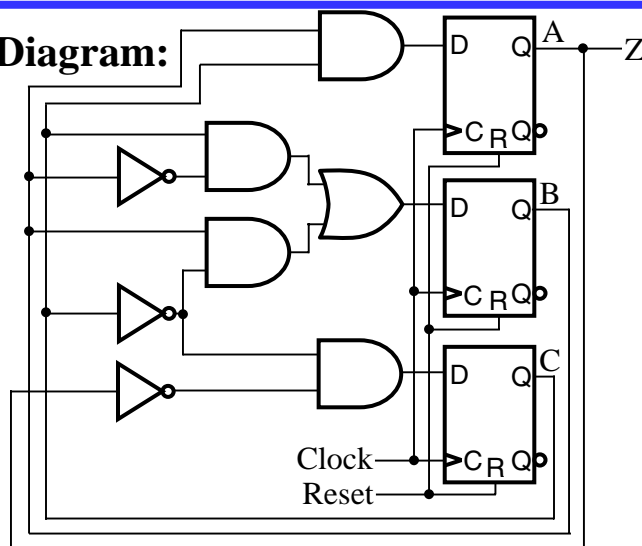 - **States 01, 10, and 11: Mealy**
- **Simplifies output specification**
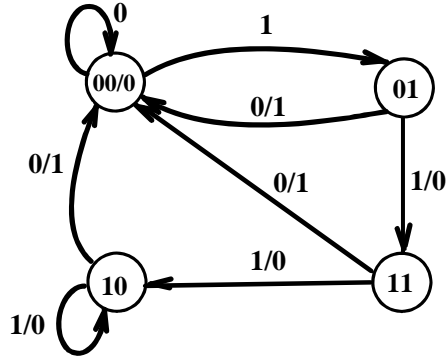
Example 2: Sequential Circuit Analysis

- **Logic Diagram:**

23

Example 2: Flip-Flop Input Equations

- **Variables**
  - **Inputs: None**
  - **Outputs: Z**
  - **State Variables: A, B, C**
- **Initialization: Reset to (0,0,0)**
- **Equations**
  - **A(t+1) =**   **Z =**
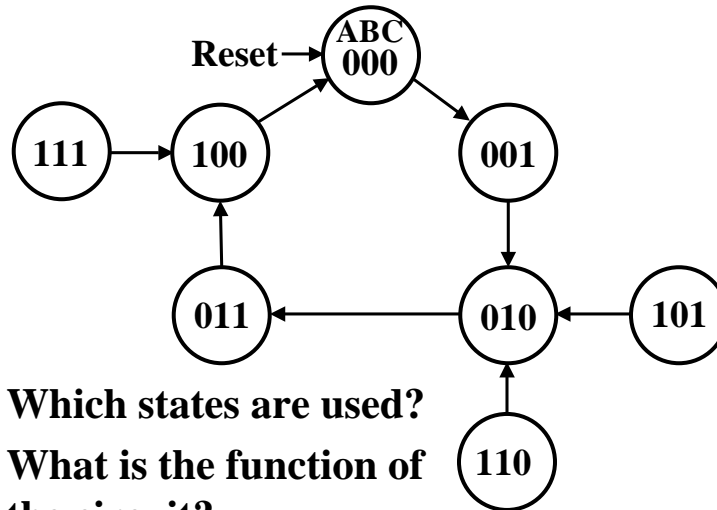 - **B(t+1) =**
  - **C(t+1) =**

Example 2: State Table

$X' = X(t+1)$

A B C	A'B'C'	Z
0 0 0		
0 0 1		
0 1 0		
0 1 1		
1 0 0		
1 0 1		
1 1 0		
1 1 1		

Example 2: State Diagram

- **Which states are used?**
- **What is the function of the circuit?**

Terms of Use

- **All (or portions) of this material © 2008 by Pearson Education, Inc.**
- **Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.**
- **These materials or adaptations thereof are not to be sold or otherwise offered for consideration.**
- **This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.**