Introduction to Digital Logic

Assist. Prof. Hamza Osman İLHAN

hoilhan@yildiz.edu.tr

Course Outline

1. Digital Computers, Number Systems, Arithmetic Operations, Decimal, Alphanumeric, and Gray Codes
2. Binary Logic, Gates, Boolean Algebra, Standard Forms
3. Circuit Optimization, Two-Level Optimization, Map Manipulation, Multi-Level Circuit Optimization
4. Additional Gates and Circuits, Other Gate Types, Exclusive-OR Operator and Gates, High-Impedance Outputs
5. Implementation Technology and Logic Design, Design Concepts and Automation, The Design Space, Design Procedure, The major design steps
6. Programmable Implementation Technologies: Read-Only Memories, Programmable Logic Arrays, Programmable Array Logic,Technology mapping to programmable logic devices
7. Combinational Functions and Circuits
8. Arithmetic Functions and Circuits
9. Sequential Circuits Storage Elements and Sequential Circuit Analysis
10. Sequential Circuits, Sequential Circuit Design State Diagrams, State Tables
11. Counters, register cells, buses, & serial operations
12. Sequencing and Control, Datapath and Control, Algorithmic State Machines (ASM)
13. Memory Basics

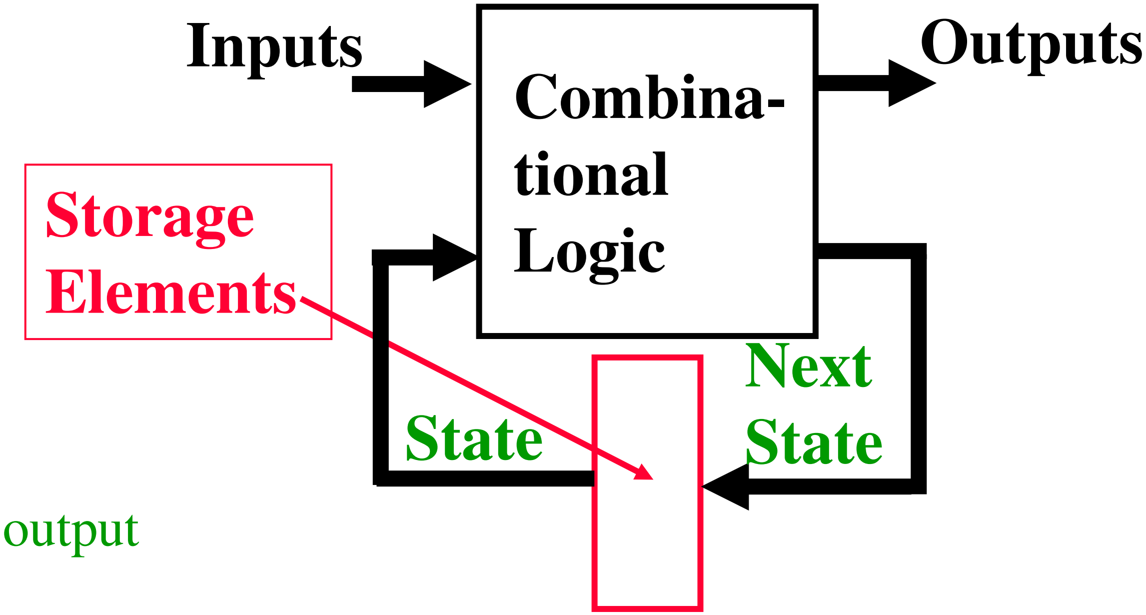
Introduction to Digital Logic

Lecture 9

Sequential Circuits

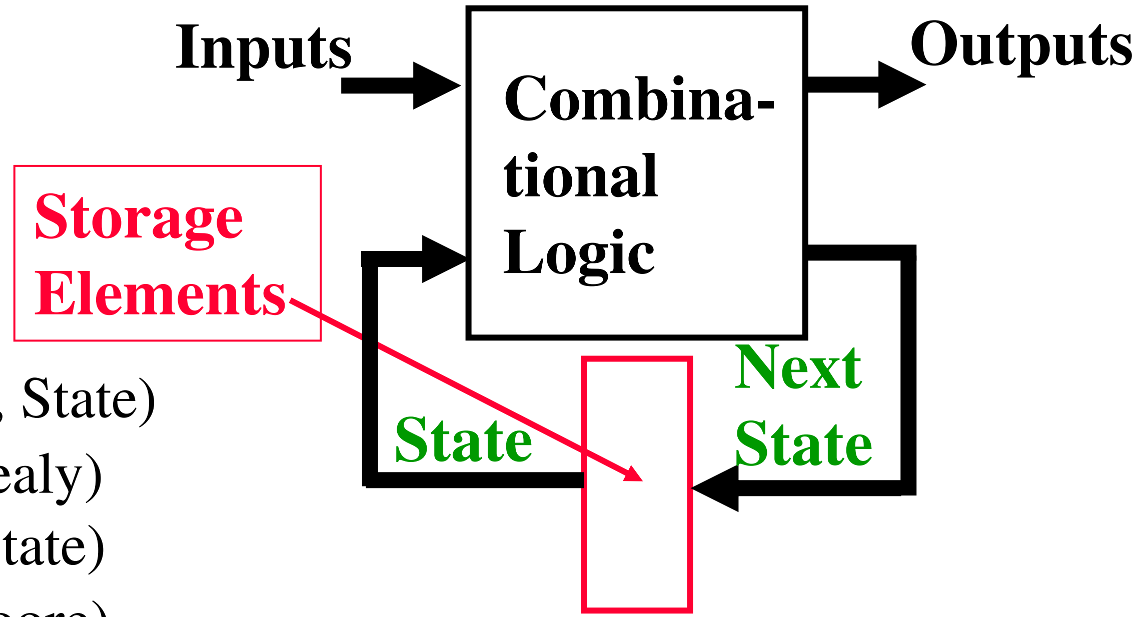
Storage Elements and Sequential Circuit Analysis

Introduction to Sequential Circuits



- A Sequential circuit contains:
  - Storage elements: Latches or Flip-Flops
  - Combinatorial Logic:
    - Implements a multiple-output switching function
    - Inputs are signals from the outside.
    - Outputs are signals to the outside.
    - Other inputs, State or Present State, are signals from storage elements.
    - The remaining outputs, Next State 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 times at which:
  - storage elements observe their inputs, and
  - storage elements change their state

1 Synchronous
  - Behavior defined from knowledge of its signals at discrete instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)

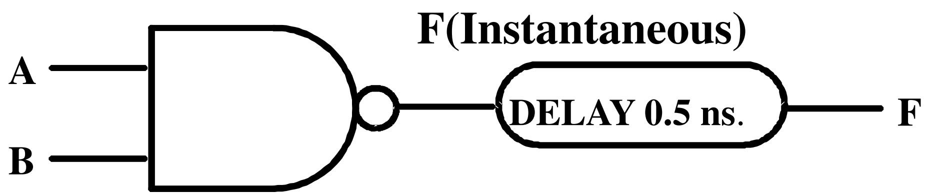
2 Asynchronous
  - Behavior defined from knowledge of inputs an any instant of time and the order in continuous time in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!

Discrete Event Simulation

- In order to understand the time behavior of a sequential circuit we use discrete event simulation.
- Rules:
  - Gates modeled by an ideal (instantaneous) function and a fixed gate delay
  - Any change in input values is evaluated to see if it causes a change in output value
  - 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:

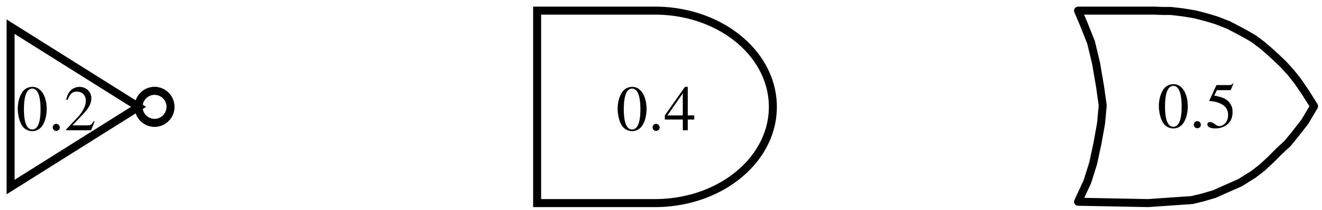


- Assume A and B have been 1 for a long time
- At time t=0, A changes to a 0 at t= 0.8 ns, back to 1.

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

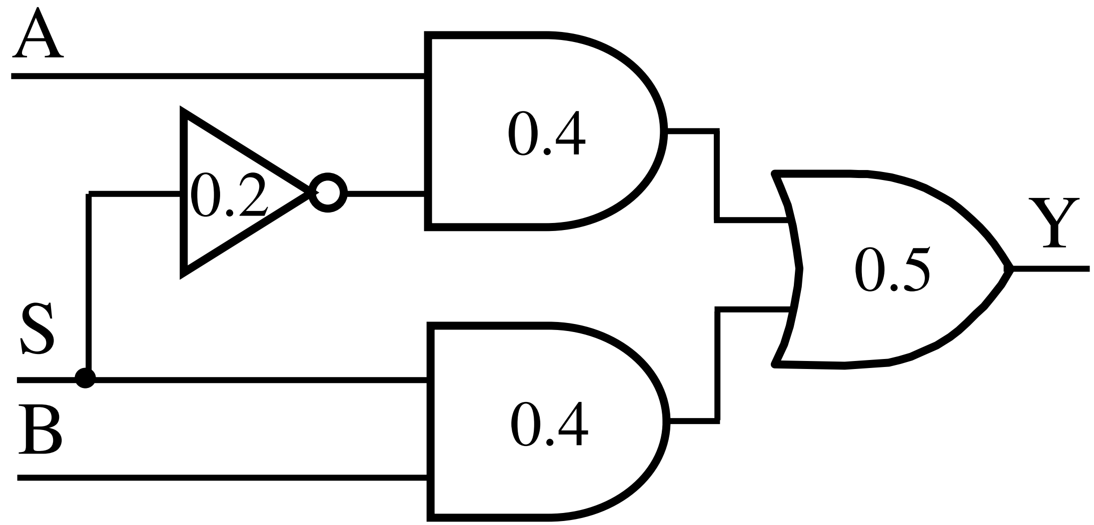


Circuit Delay Model

- Consider a simple 2-input multiplexer:
- With function:

  Y = A for S = 0

  Y = B for S = 1

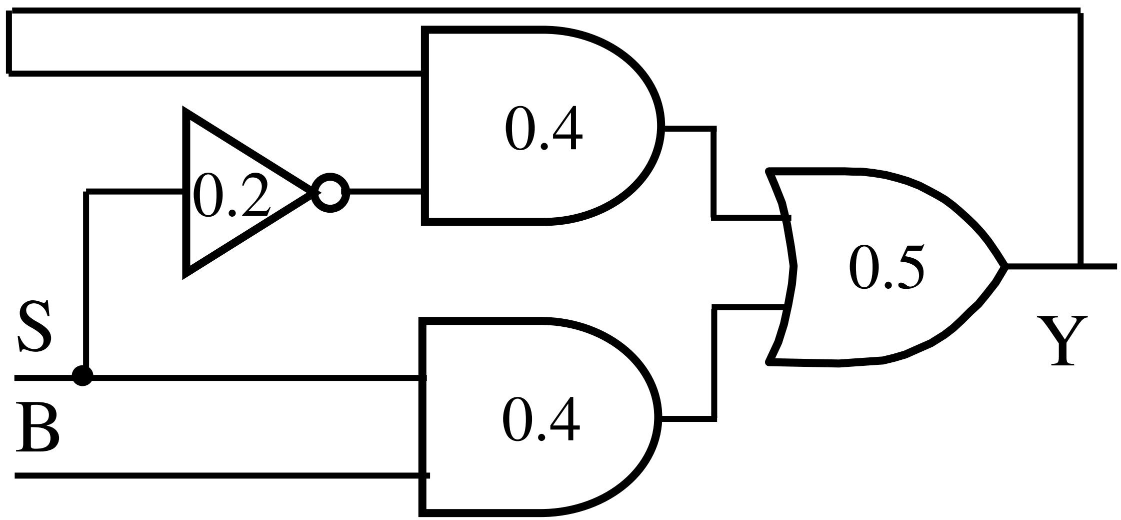


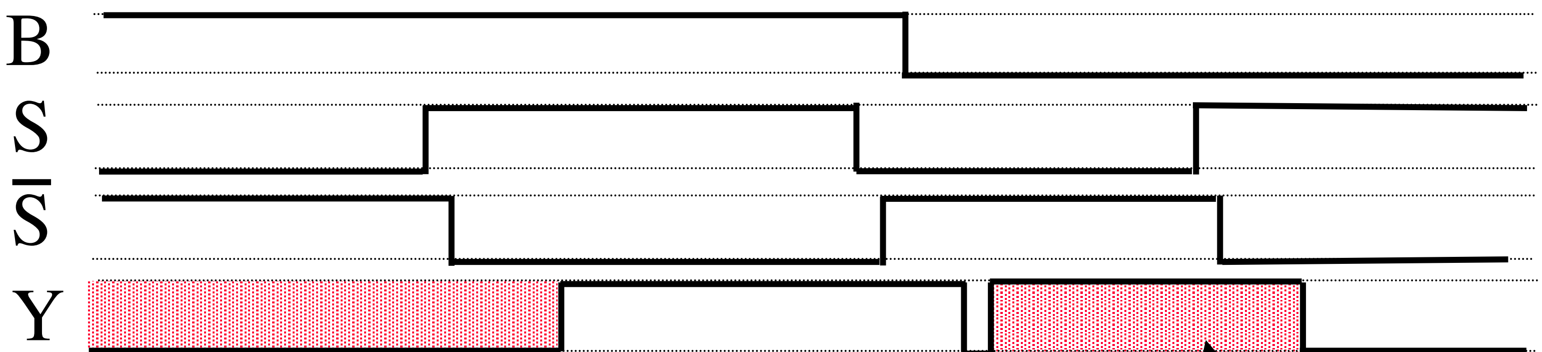
- "Glitch" is due to delay of inverter

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.

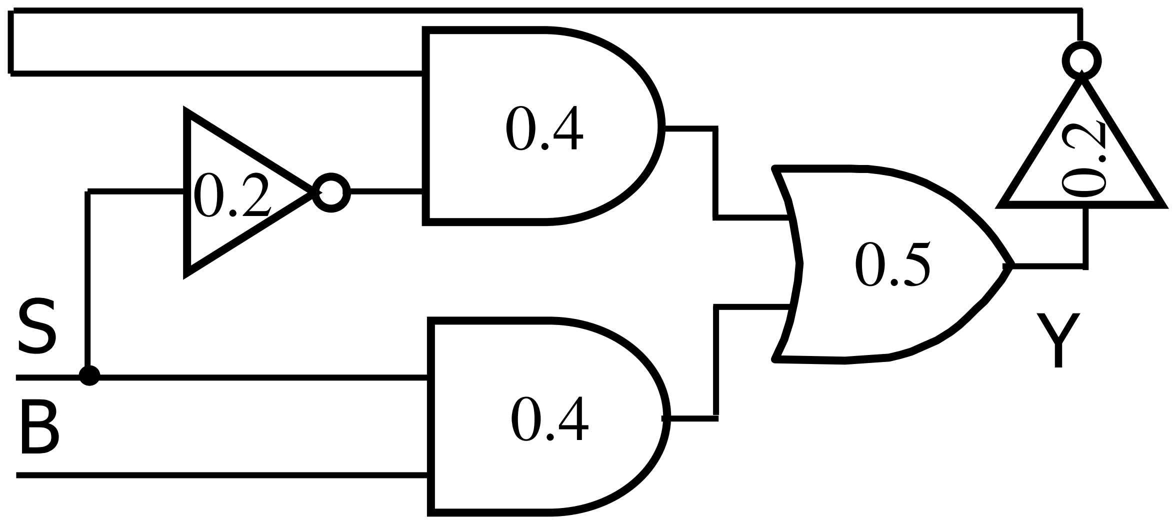
Time ↓

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)

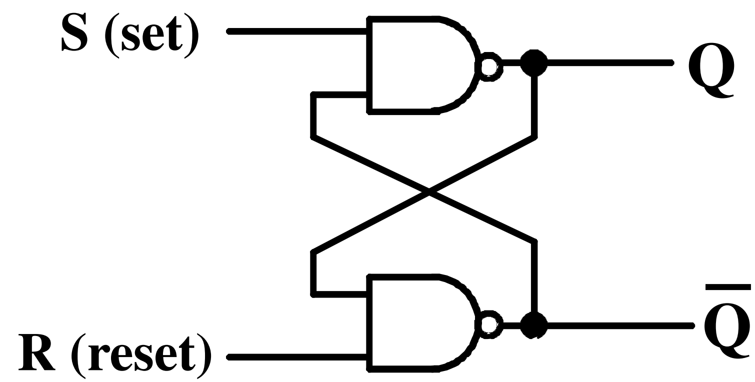
- Suppose we place an inverter in the "feedback path."



- 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**

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:

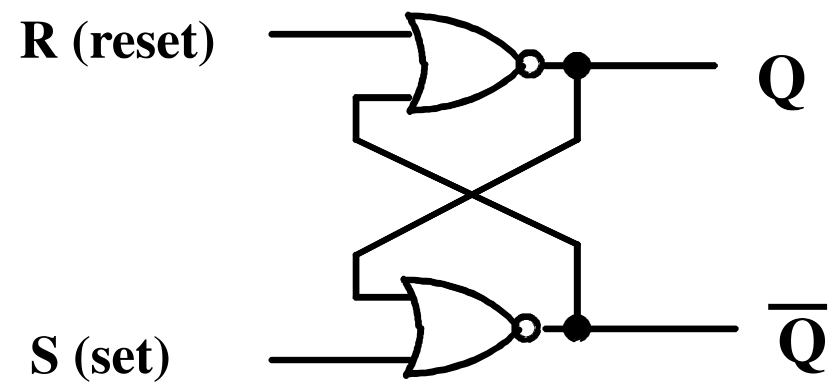
Time

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!

- S = 0, R = 0 is forbidden as input pattern

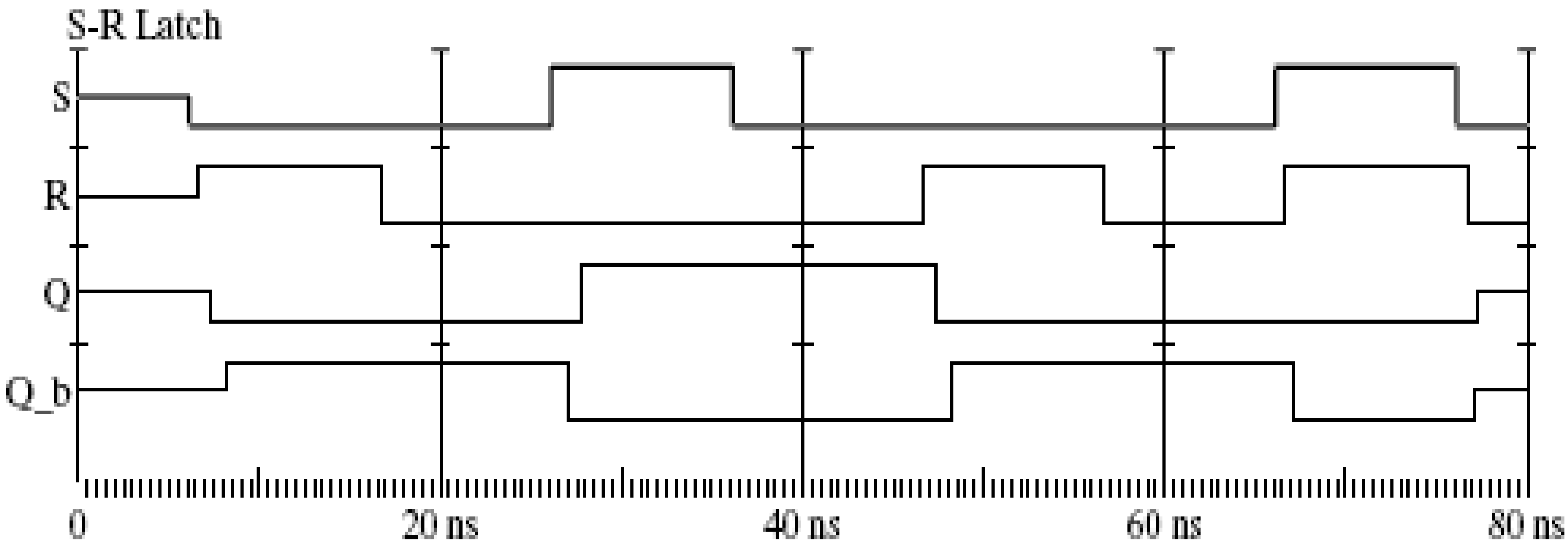
Basic (NOR) S – R Latch

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



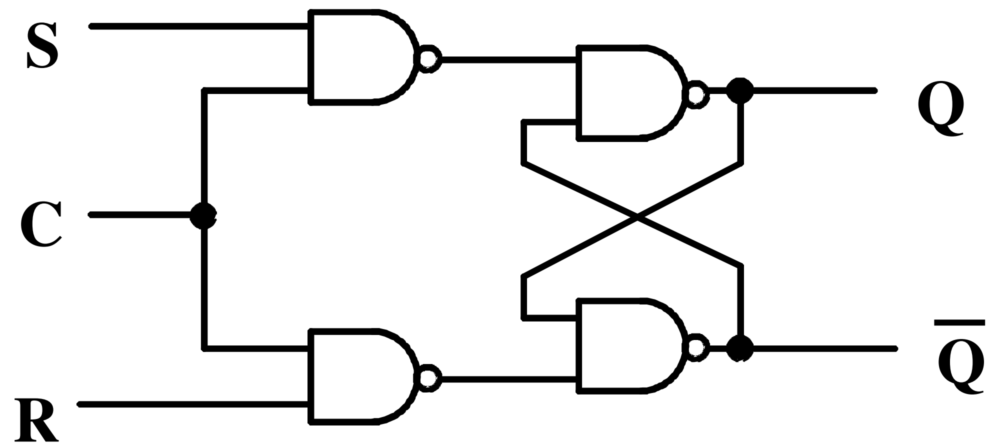
Time

R	S	Q	$\overline{Q}$	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!


S-R Latch
S
R
Q
Q_b
0
20 ns
40 ns
60 ns
80 ns

Clocked S - R Latch

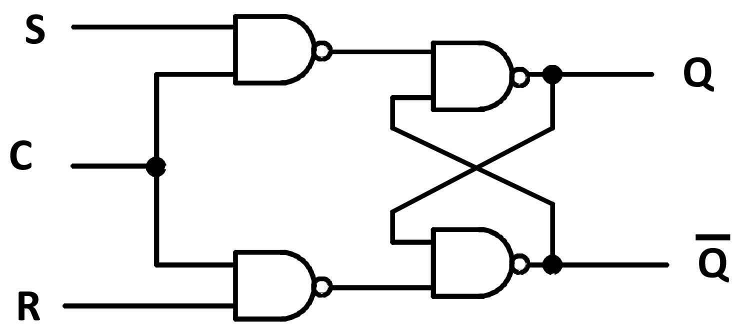
- Adding two NAND gates to the basic $\overline{S}$ - $\overline{R}$ NAND latch gives the clocked S – R latch:



- 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:

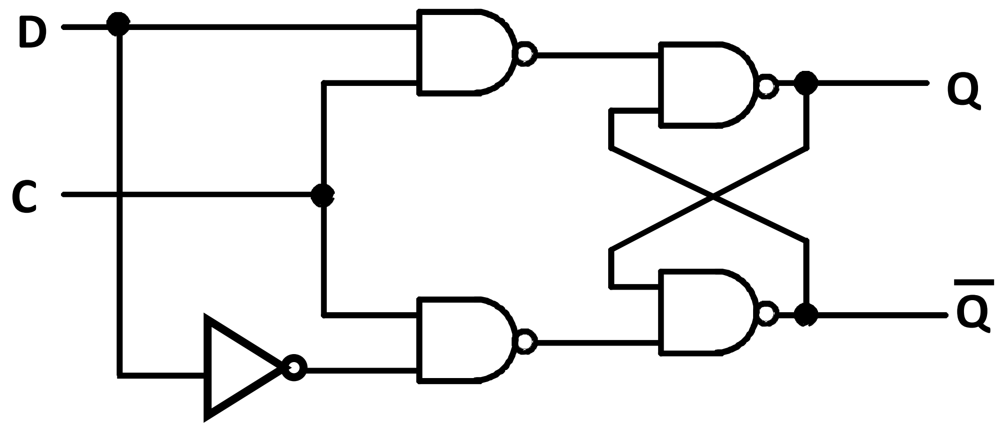


Q(t)	S	R	Q(t+1)	Comment
0	0	0	0	No change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	Indeterminate
1	0	0	1	No change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	Indeterminate

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

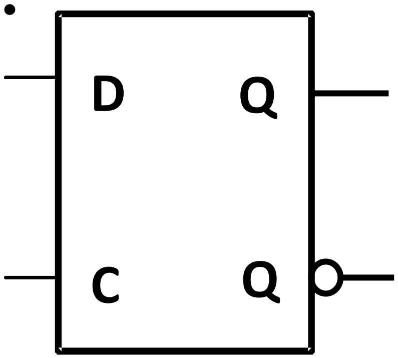
D Latch

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



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

The graphic symbol for a D Latch is:



Flip-Flops

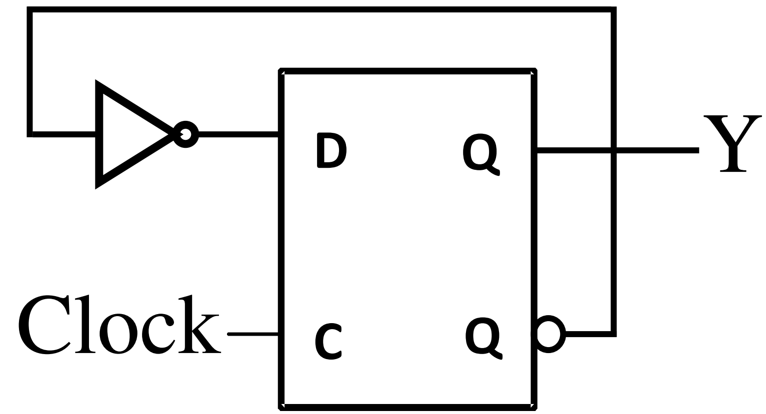
- The latch timing problem
- Master-slave flip-flop
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops
- Flip-flop timing

The Latch Timing Problem

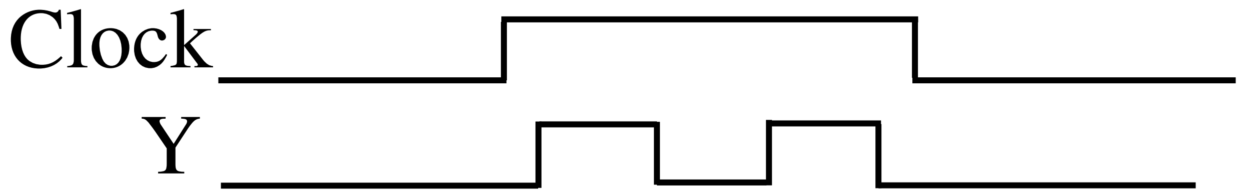
- In a sequential circuit, paths may exist through combinational logic:
  - From one storage element to another
  - From a storage element back to the same storage element
- The combinational logic between a latch output and a latch input may be as simple as an interconnect
- For a clocked D-latch, the 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.

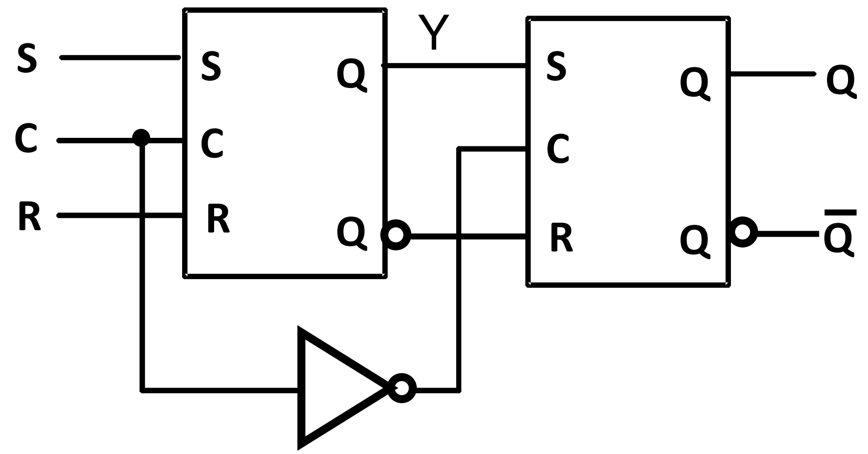


- As long as C = 1, the value of Y continues to change!
- The changes are based on the delay present on the loop through the connection from Y back to Y.
- This behavior is clearly unacceptable.
- <u>Desired behavior</u>: Y changes <u>only once</u> per clock pulse

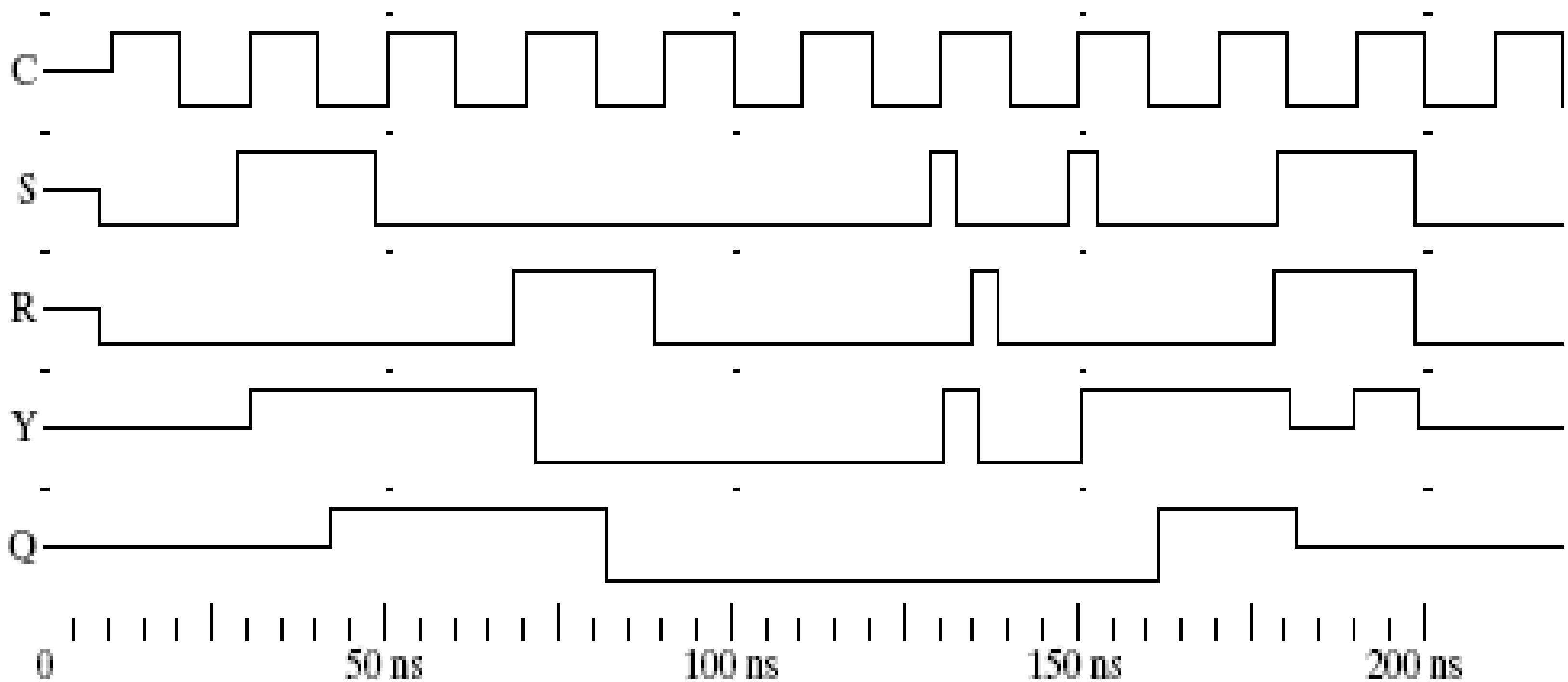
The Latch Timing Problem (continued)

- A solution to the latch timing problem is to break 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.


C
S
R
Y
Q
0
50 ns
100 ns
150 ns
200 ns

Flip-Flop Problem

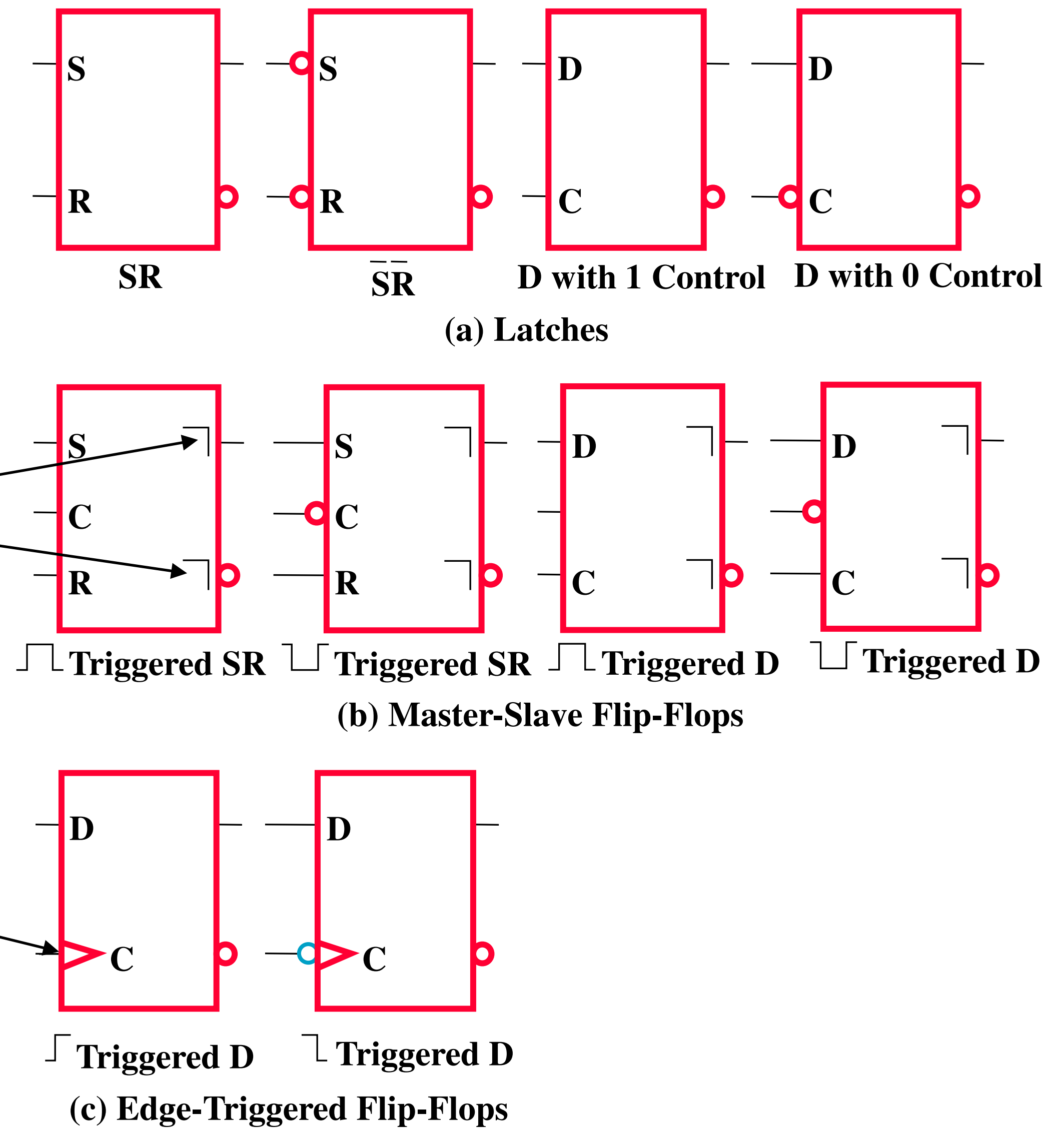
- The change in the flip-flop output is delayed by the pulse width which makes the circuit slower or
- S and/or R are permitted to change while C = 1
  - Suppose Q = 0 and S goes to 1 and then back to 0 with R remaining at 0
    - The master latch sets to 1
    - A 1 is transferred to the slave
  - Suppose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

Flip-Flop Solution

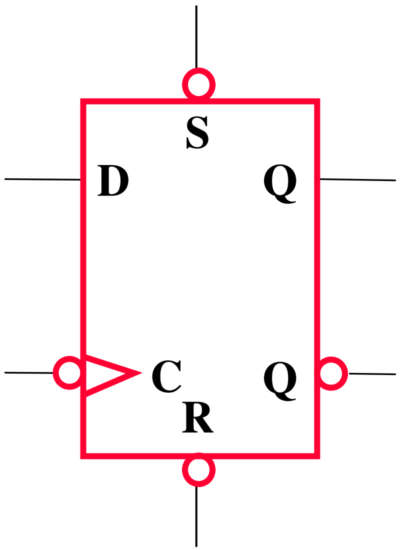
- Use edge-triggering instead of master-slave
- An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a <u>transition</u> of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level, or
- A <u>master-slave</u>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

Standard Symbols for Storage Elements

- Master-Slave: Postponed output indicators
- Edge-Triggered: Dynamic indicator



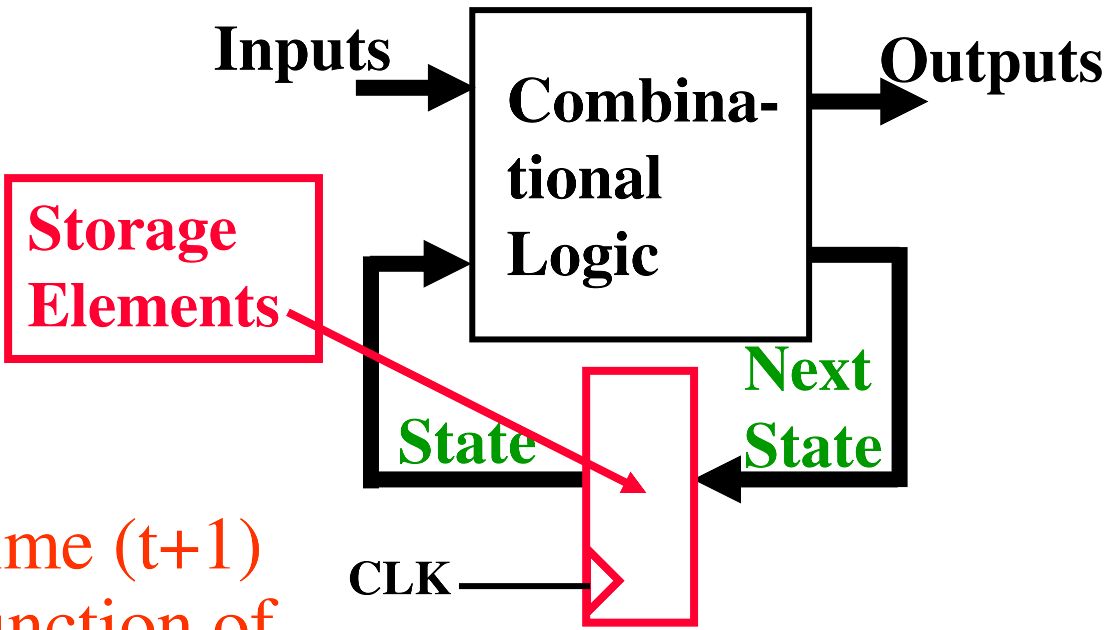
Direct Inputs



- 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 $\overline{R}$ resets the flip-flop to the 0 state
  - 0 applied to $\overline{S}$ sets the flip-flop to the 1 state

Sequential Circuit Analysis

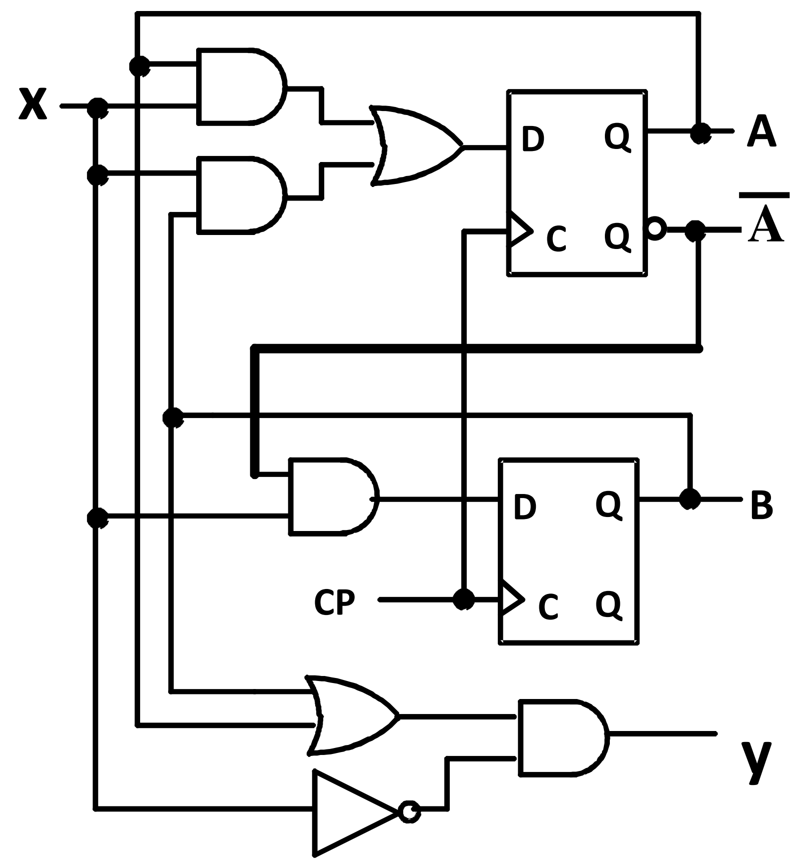
- General Model
  - Current State at time (t) is stored in an array of flip-flops.



  - 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

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



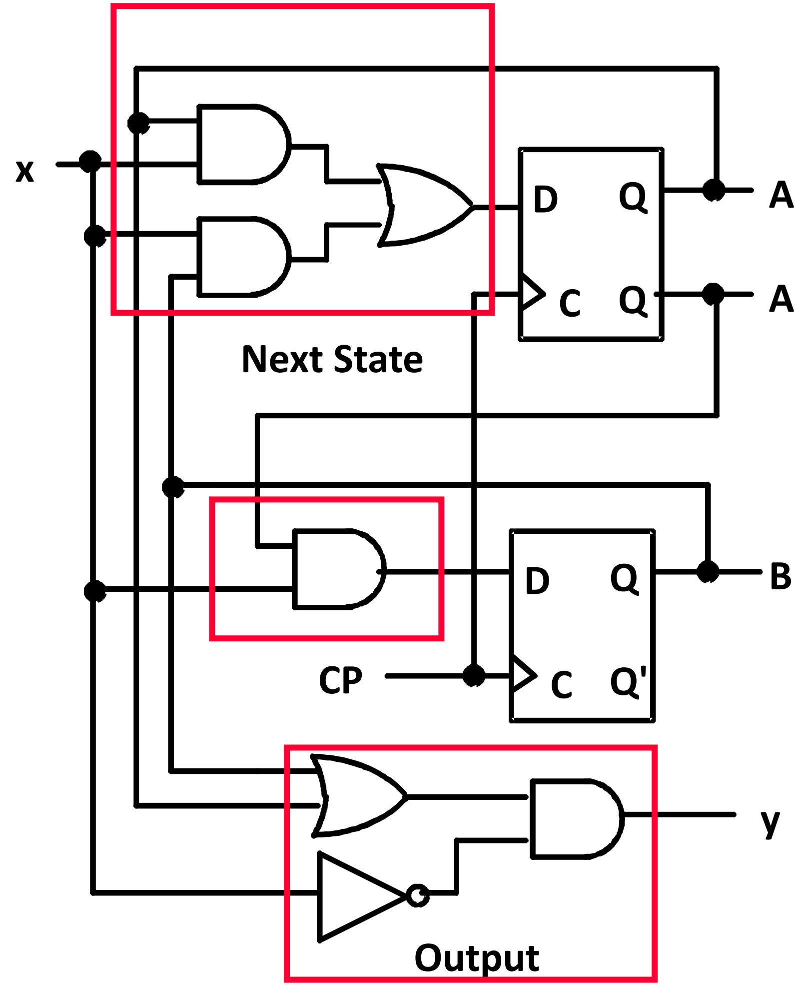
Example 1 (continued)

- Boolean equations for the functions:

$$A(t+1) = A(t)x(t) + B(t)x(t)$$

$$B(t+1) = \overline{A}(t)x(t)$$

$$y(t) = \overline{x}(t)(B(t) + A(t))$$



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 present state and the input.
  - *Output* – the value of the output as a function of the present state and (sometimes) the input.
- From the viewpoint of a truth table:
  - the inputs are Input, Present State
  - and the outputs are Output, Next State

Example 1: State Table

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

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

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

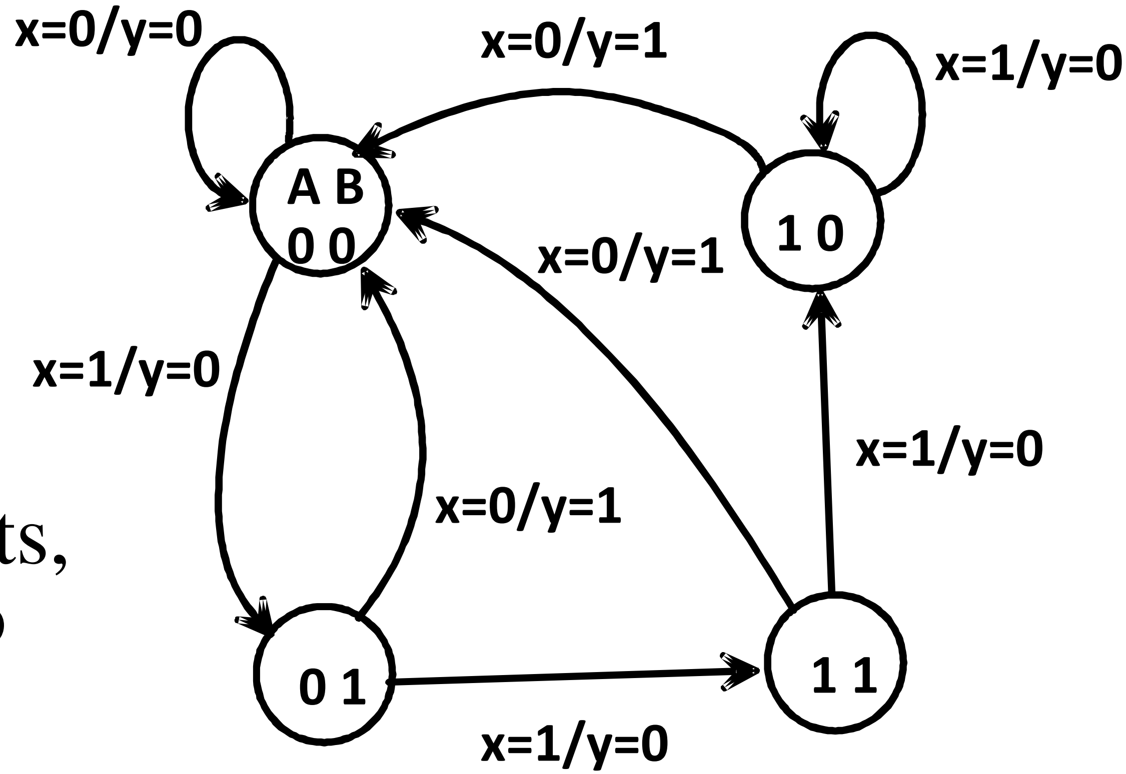
- The sequential circuit function can be represented in graphical form as a <u>state diagram</u> with the following components:
  - A <u>circle</u> with the state name in it for each state
  - A <u>directed arc</u> from the <u>Present State</u> to the <u>Next State</u> for each <u>state transition</u>
  - 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.

State Diagrams

- Label form:
  - On circle with output included:
    - state/output
    - Moore type output depends only on state
  - On directed arc with the output 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



Moore and Mealy Models

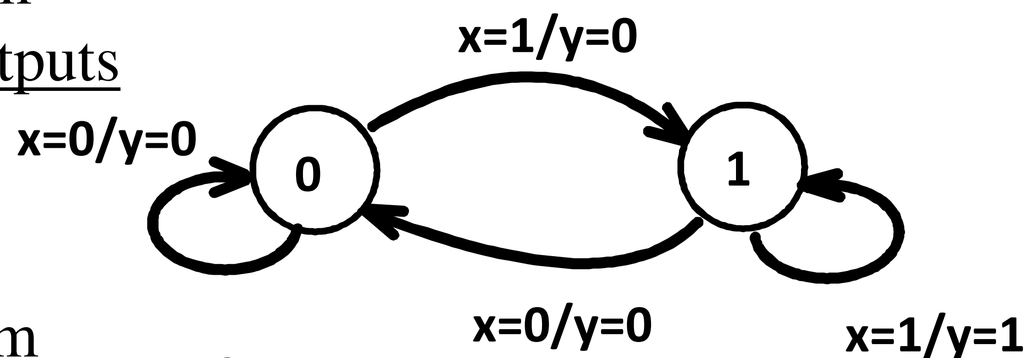
- Sequential Circuits or Sequential Machines are also called *Finite State Machines* (FSMs). Two formal models exist:

  - **Moore Model**
    - **Named after E.F. Moore.**
    - **Outputs are a function ONLY of states**
    - **Usually specified on the states.**
  - **Mealy Model**
    - **Named after G. Mealy**
    - **Outputs are a function of inputs AND states**
    - **Usually specified on the state transition arcs.**

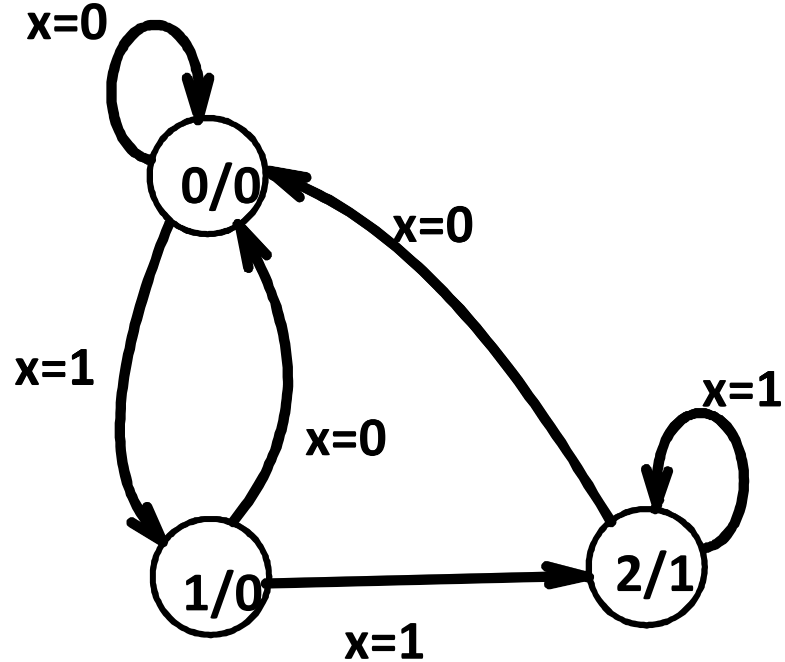
- In contemporary design, models are sometimes mixed Moore and Mealy

Moore and Mealy Example Diagrams

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



- Moore Model State Diagram maps <u>states</u> to <u>outputs</u>



Moore and Mealy Example Tables

- Mealy Model state table maps inputs and state to outputs

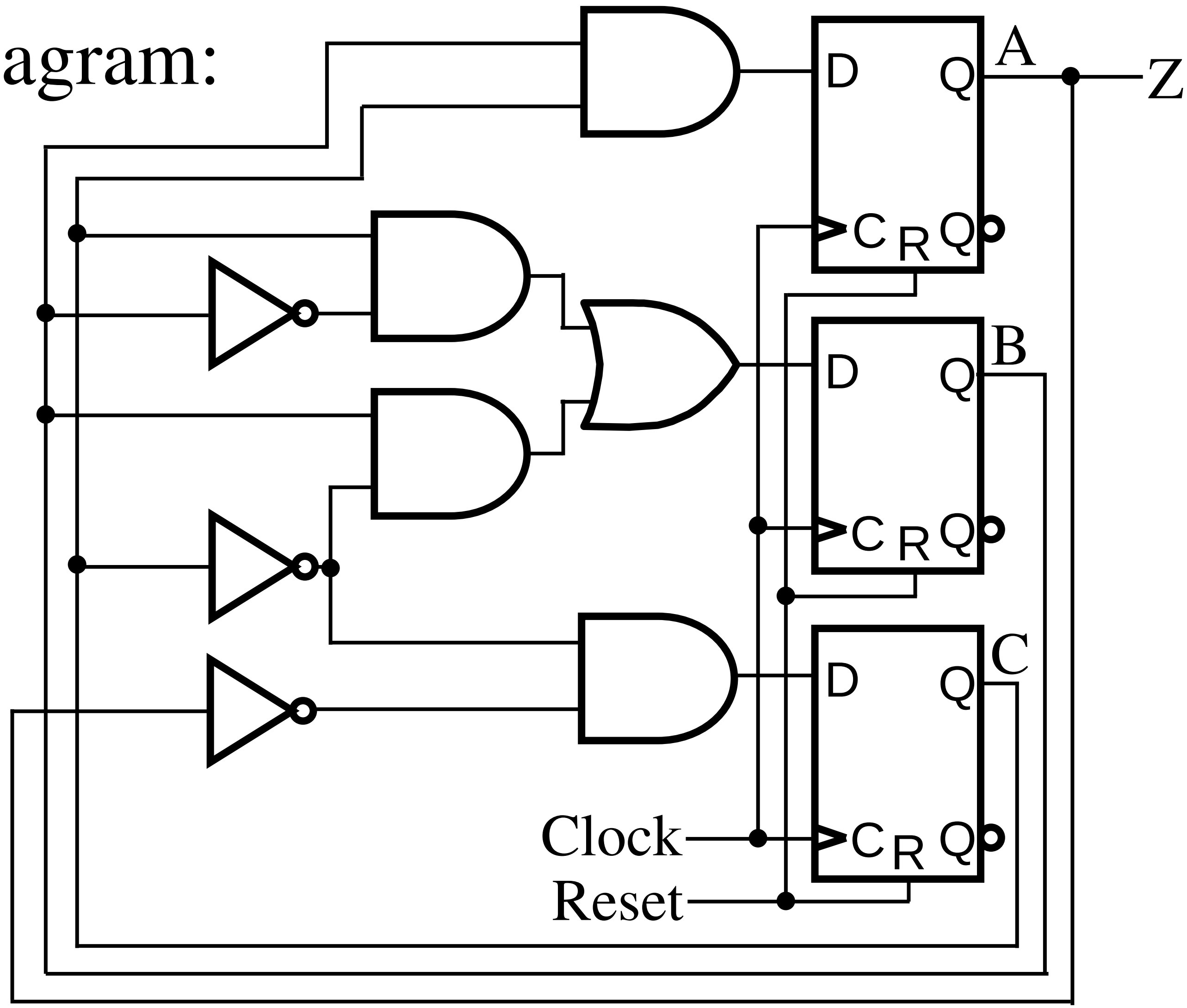
Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
0	0	1	0	0
1	0	1	0	1

- Moore Model state table maps state to outputs

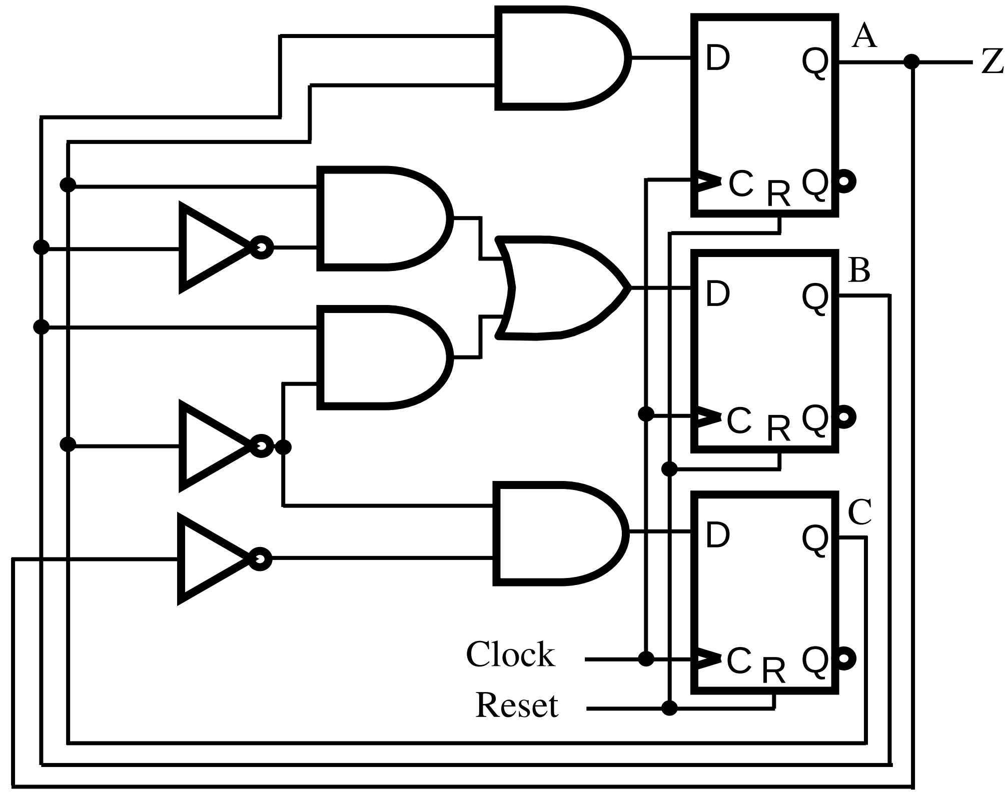
Present State	Next State x=0	Next State x=1	Output
0	0	1	0
1	0	2	0
2	0	2	1

Example 2: Sequential Circuit Analysis

- Logic Diagram:



Example 2: Flip-Flop Input Equations



- Variables
  - Inputs: None
  - Outputs: Z
  - State Variables: A, B, C
- Initialization:
  - Reset to (0,0,0)
- Equations

$$A(t+1) = B(t)C(t) \qquad Z = B(t)C(t)$$

$$B(t+1) = \overline{B}(t)C(t) + B(t)\overline{C}(t)$$

$$C(t+1) = \overline{A}(t)\overline{C}(t)$$

Example 2: State Table

X’ = X(t+1)

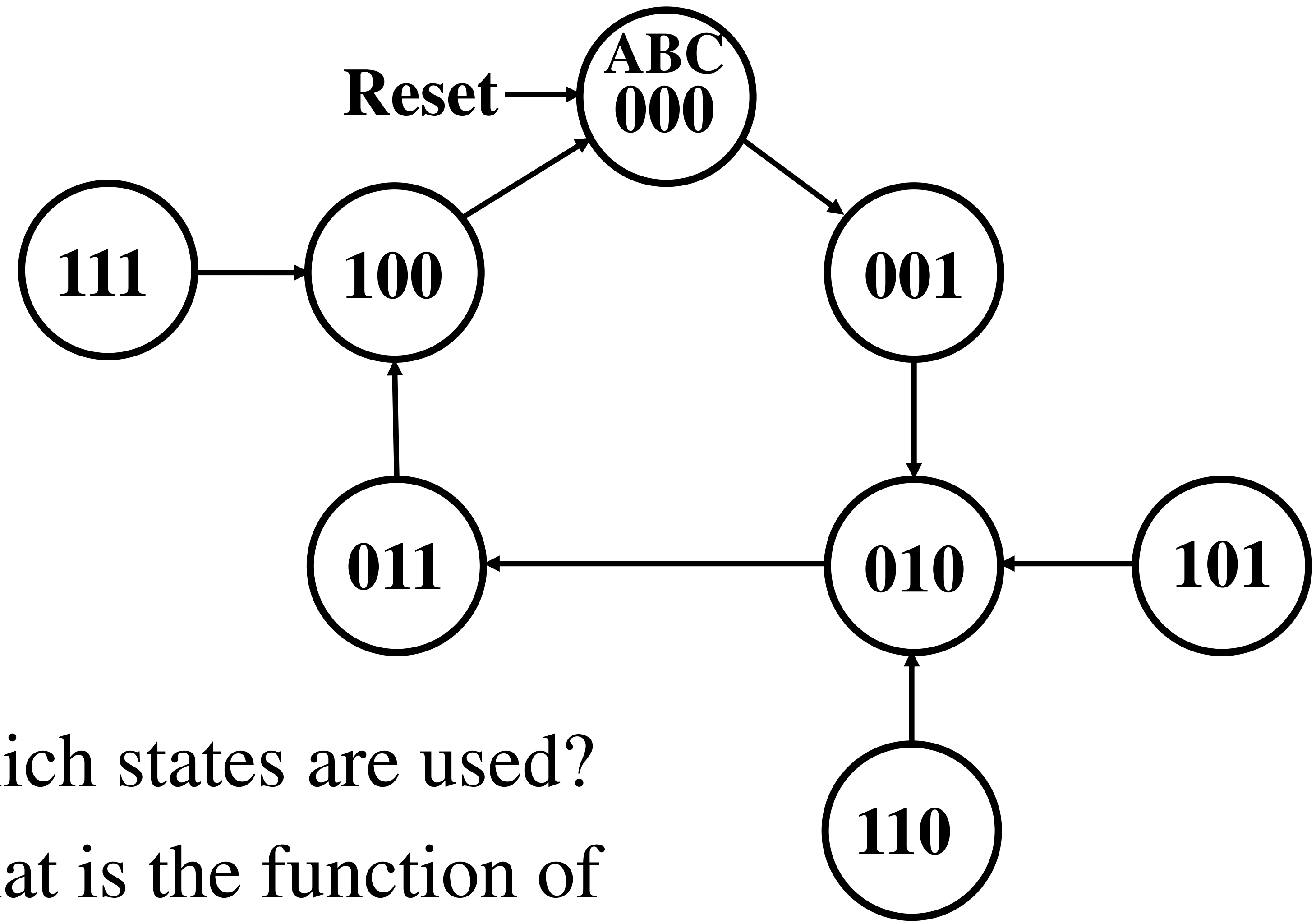
$A(t+1) = B(t)C(t)$

$B(t+1) = \overline{B}(t)C(t) + B(t)\overline{C}(t)$

$C(t+1) = \overline{A}(t)\overline{C}(t)$

A B C	A’B’C’	Z
0 0 0	0 0 1	0
0 0 1	0 1 0	0
0 1 0	0 1 1	0
0 1 1	1 0 0	1
1 0 0	0 0 0	0
1 0 1	0 1 0	0
1 1 0	0 1 0	0
1 1 1	1 0 0	1

Example 2: State Diagram



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

Circuit and System Level Timing

- Consider a system comprised of ranks of flip-flops connected by logic:
- If the clock period is too short, some data changes will not propagate through the circuit to flip-flop inputs before the setup time interval begins