Sequential Logic

CO 2206 Computer Organization

Topics

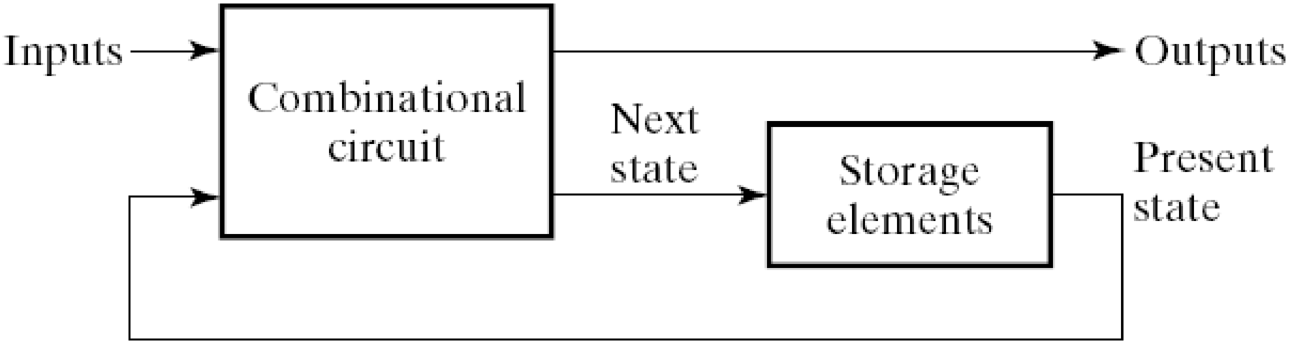
- Basics
- Building Blocks: Latches and Flip-Flops
- Sequential Circuit Analysis
- Sequential Circuit Design

Sequential Logic

- Outputs of ***sequential logic circuits*** depend not only on present inputs, but also on stored values (states), which are a function of previously applied inputs
- Output determined by
  - inputs
  - present state of the storage elements
    - ‘previous’ outputs
- Timing is a factor

Sequential Circuit

- A *sequential circuit* consists of a combinational circuit to which storage elements are connected to form a feedback path



- Next states and output(s) are interrelated. In some cases, the feedback consists of a simple interconnection of some outputs of the combinational circuit to its inputs.

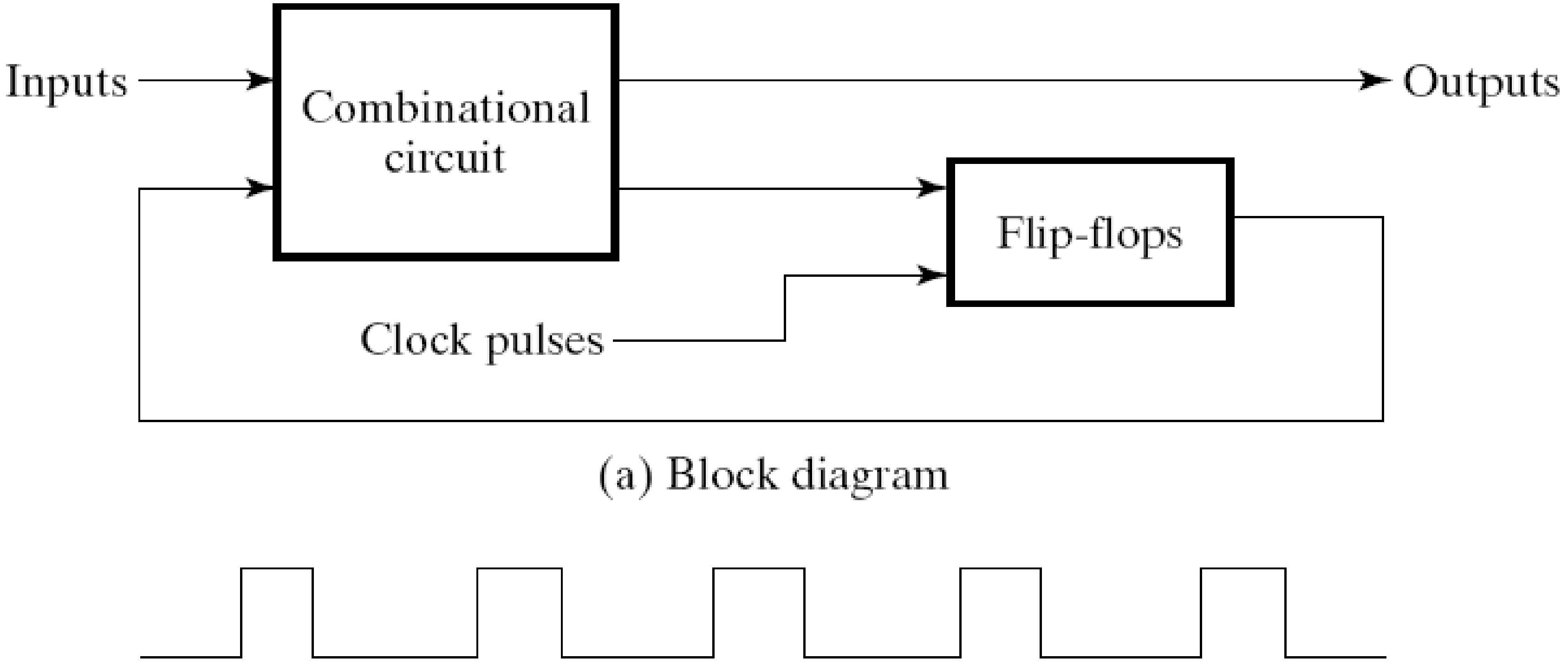
Two Types

- Main types are:
  - *Synchronous* sequential circuit
    - whose behaviour can be defined from the knowledge of its signals at discrete instants of time
    - All circuits in the system change their state at some precisely defined instant – driven by a system clock
  - *Asynchronous* sequential circuit
    - whose behaviour depends upon the order in which its input signals change and can be affected at any instant of time
    - implies that all required inputs may not be valid at the same time

Synchronous – the preferred

- Synchronization is achieved by a timing device – a *clock generator*
  - Which produces a periodic train of *clock pulses*
- The outputs of storage elements can change their value in the presence of clock pulses

Synchronous

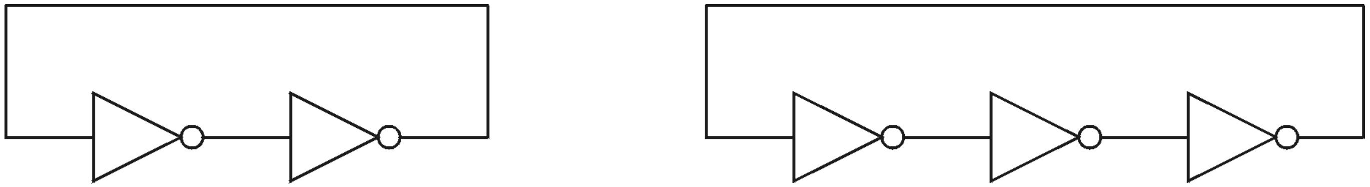


(a) Block diagram

(b) Timing diagram of clock pulses

Storage – using feedback

- Storage circuit can be
  - a simple interconnection from outputs to input
  - i.e. feedback
- Feedback can potentially introduce instability



(a) Stable circuit

(b) Unstable circuit

- Can’t change information stored → need another input → as in latch (using NOR or NAND)

Building Blocks for Sequential Logic Circuits

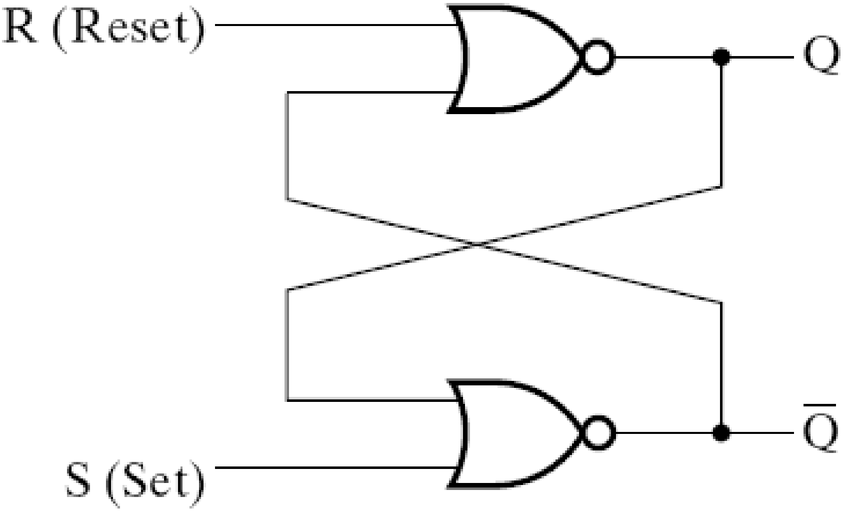
- In the same way that gates are the building blocks of combinatorial circuits, latches and flip-flops are the building blocks for sequential circuits
- While gates had to be built directly from transistors, latches can be built from gates, and flip-flops can be built from latches
- Major differences among various types of latches (and flip-flops) are in
  - number of inputs they possess
  - manner in which inputs affect the binary state

Latches

- A bistable multivibrator electronic device which has two stable states
- Latches can store one bit of information; it is the most basic storage element
- Today the word is mainly used for simple transparent storage elements

SR Latches (NOR) - 1

- Simplest of latches are constructed from 2 cross-coupled NOR gates or NAND gates



(a) Logic diagram

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

(b) Function table

SR Latches (NOR) - 2

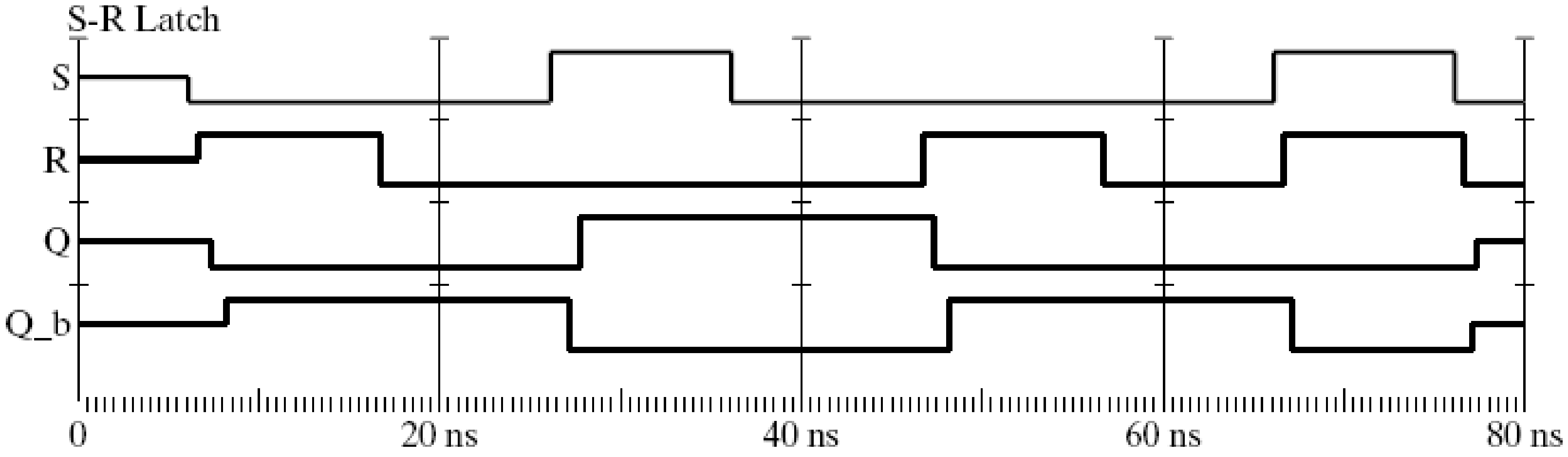
- Cross-coupled connection constitutes a feedback path
- When both S and R = 0, next output depends on current output
  - Under normal conditions, both inputs of the latch remain 0 unless the state is to be changed
  - the latch can be in either the set or reset state when both inputs = 0

SR Latches (NOR) - 3

- When S = 1, R = 1, both Q and Q' = 0
  - Violates the fact that they are the complement of each other
  - When input changes from 11 to 00, the output is also indeterminate
  - These can be avoided by making sure that 1's are not applied to both inputs simultaneously

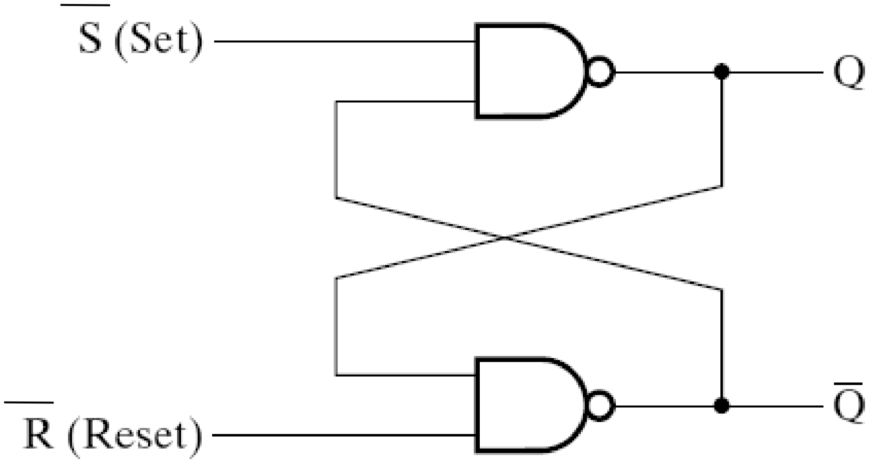
SR Latches (NOR) - 4

- To write 1 into SR latch, set SR as 10
- To write 0, use SR = 01
- To retain a stored bit, keep SR at 00

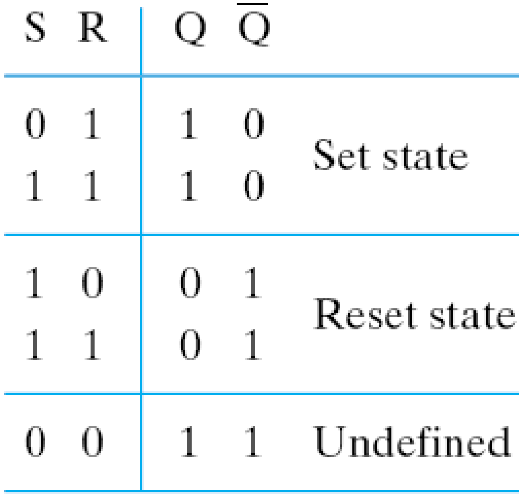


S’R’ Latches (NAND) - 1

- This is an S’R’ (active-low) latch because the NAND latch requires a 0 signal to change its state



(a) Logic diagram

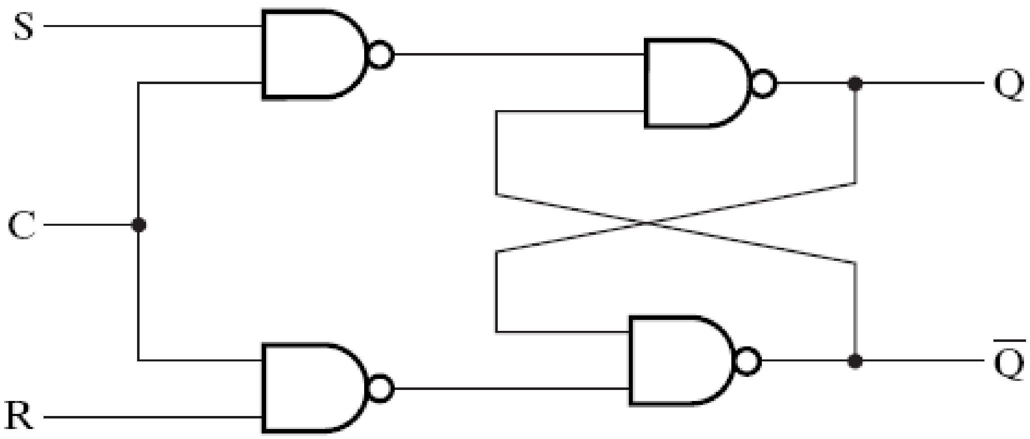


S	R	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

SR Latches with Control Input

- Control input (C) acts as an enable signal for the other 2 inputs
- When C returns to 0, circuit remains in current state



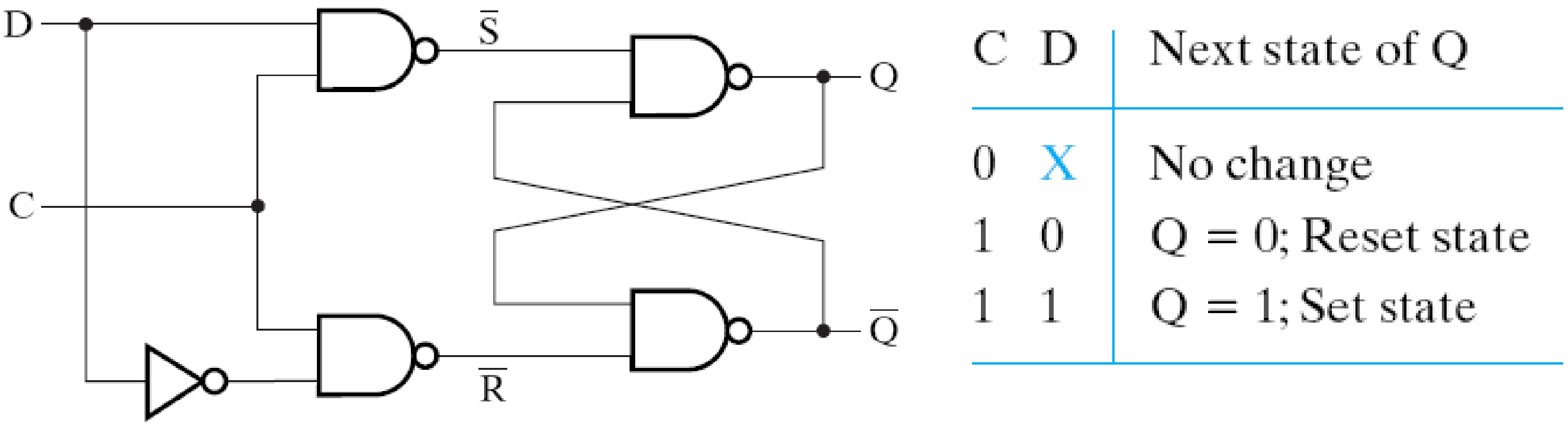
(a) Logic diagram

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

(b) Function table

D Latch - 1

- To eliminate indeterminate state, ensure S and R never equal 1 at the same time

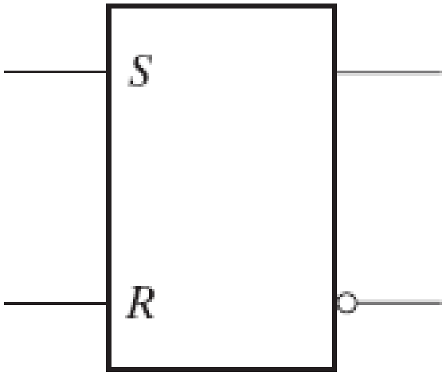


C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

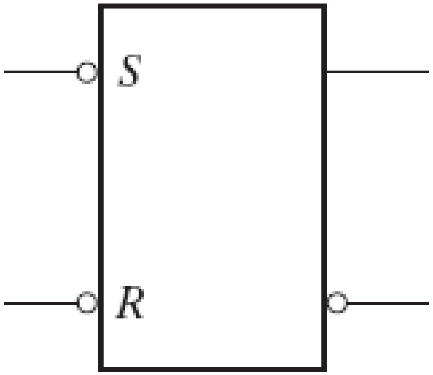
D Latch - 2

- D latch has only 2 inputs
  - D (data)
  - C (control)
- When control input is enabled (C=1)
  - data input of D latch is transferred to the Q output
- When control input is disabled (C=0)
  - Last data input value is retained until enabled again

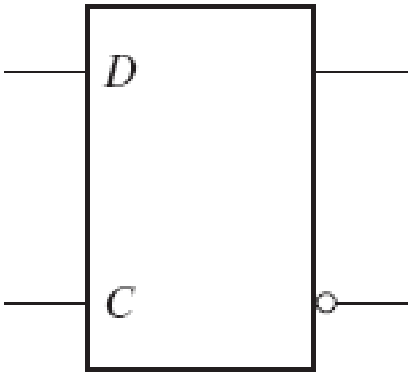
Graphic Symbols for Latches



SR



$\bar{S}\bar{R}$



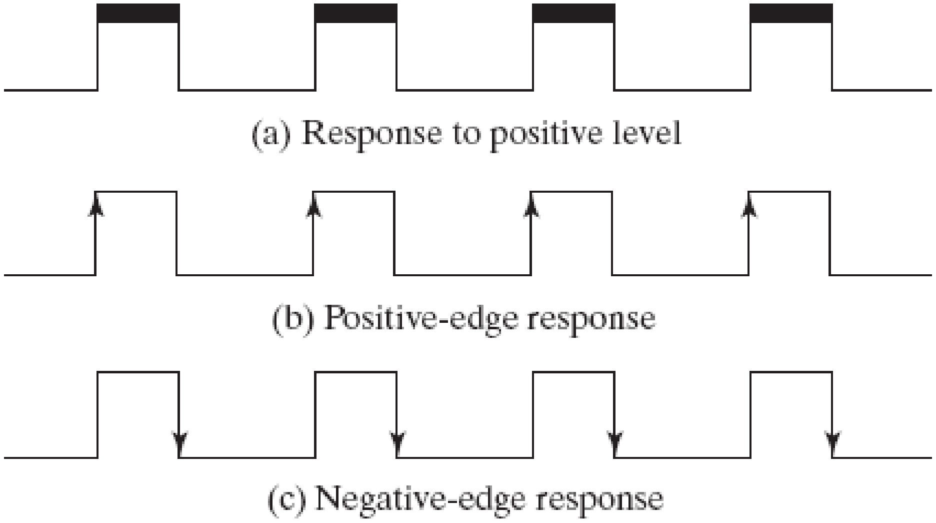
D

Problems with Latch

- D latch with clock pulses in its control input is triggered every time the pulse goes to logic 1 level
- The output and state of latches may keep changing as long as the clock pulse stays in the active level
  - Because of unpredictability, the output of the latch cannot be applied to the input of the same or another latch when the latches are triggered by a common clock source

Flip-flops

- A bistable multivibrator that has two stable states and thereby is capable of serving as one bit of memory
- Today, the term flip-flop has come to mostly denote non-transparent (clocked or edge-triggered) devices
- Flip-flops operate properly because it is triggered only during signal *transition*
  - From 0 to 1 (positive-edge transition)
  - From 1 to 0 (negative-edge transition)

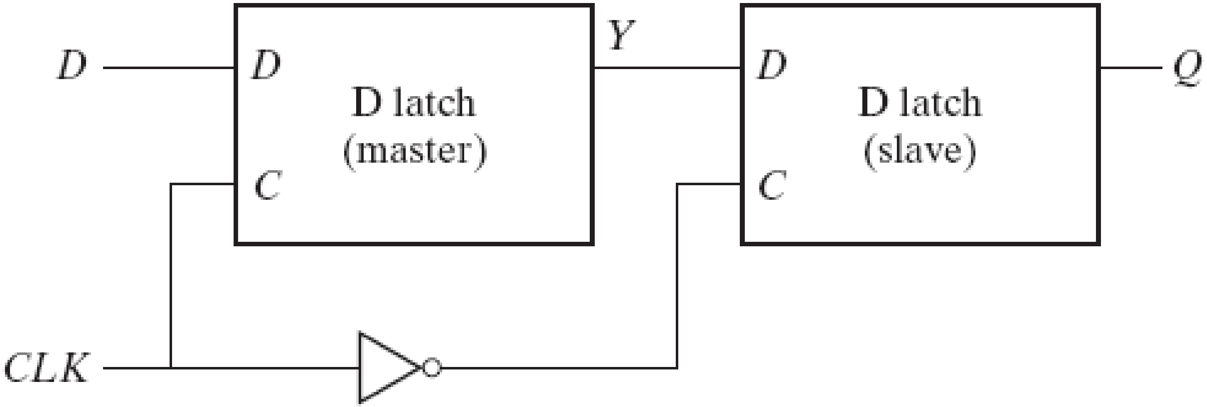

(a) Response to positive level
(b) Positive-edge response
(c) Negative-edge response

Flip-flops vs. Latches

- Both latches and flip-flops are circuit elements whose output depends not only on the current inputs, but also on previous inputs and outputs
- The difference between a latch and a flip-flop are:
  - Latches are level triggered device while flip-flops are edge triggered device
  - Latches are asynchronous and transparent
    - the output follows any transition at the input
  - Flip-flops are synchronous and not transparent
    - it holds the captured value at the output
  - Flip-flops are often built from latches

Master-slave Flip-flops: D - 1

- Master-slave D flip-flop



Master-slave Flip-flops: D - 2

- Constructed from 2 separate latches: *master* and *slave*
- When Clk=1
  - External input D determine the value stored in master and output Y
  - Slave disabled because C at slave is 0
- When Clk=0
  - Master is disabled, Slave is enabled
  - slave output Q = master output Y
- Note that the circuit changes its output value on the negative edge of the clock pulse

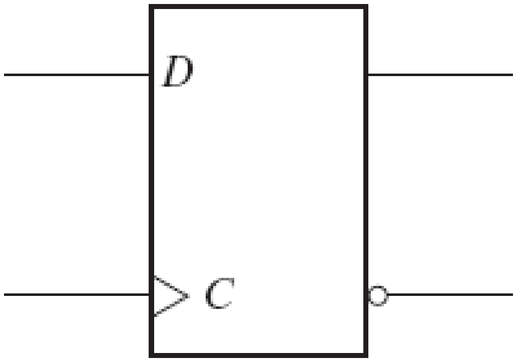
Master-slave Flip-flops: Connectivity - 1

- Consider many master-slave flip-flops, with outputs of some going to inputs of other
- Assume Clk inputs to all flip-flops are synchronised (occur at the same time)
- At the beginning of Clk, some masters change state, but flip-flop outputs remain at previous values

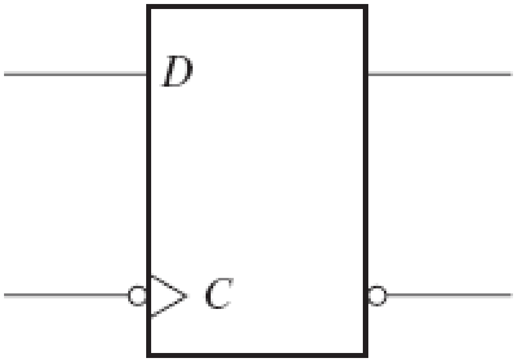
Master-slave Flip-flops: Connectivity - 2

- After Clk=0, some outputs change state, but not affecting any other masters until next Clk
- Therefore content of first can be transferred to second and second to first, and occurring during the same Clk pulse
- Flip-flop's output may change only during –ve edge transition of the clock

Edge-triggered D flip-flops



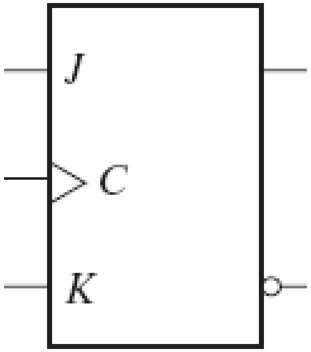
(a) Positive-edge

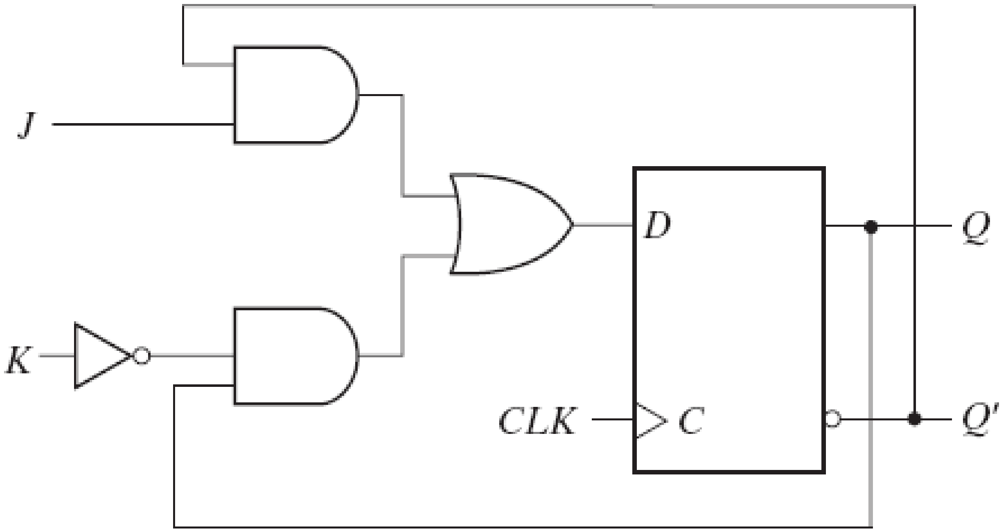


(a) Negative-edge

JK flip-flops

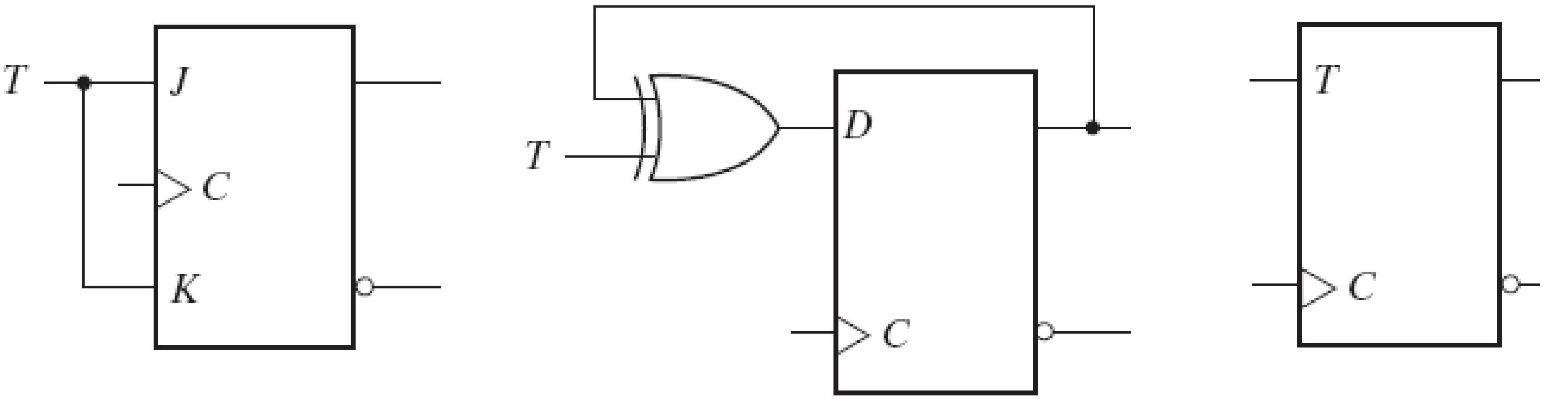
- Input J and K behave like S and R to set and clear the flip-flop
- When both J = K = 1, flip-flop switches to its complement state





T flip-flops

- T = 1, present state complemented
- T = 0, no change



(a) From *JK* flip-flop

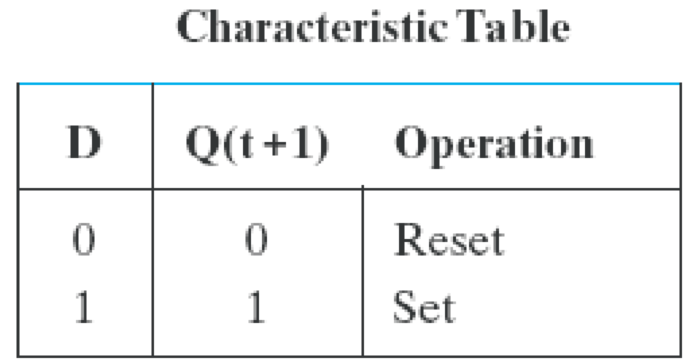
(b) From *D* flip-flop

(c) Graphic symbol

Characteristic and Excitation Tables

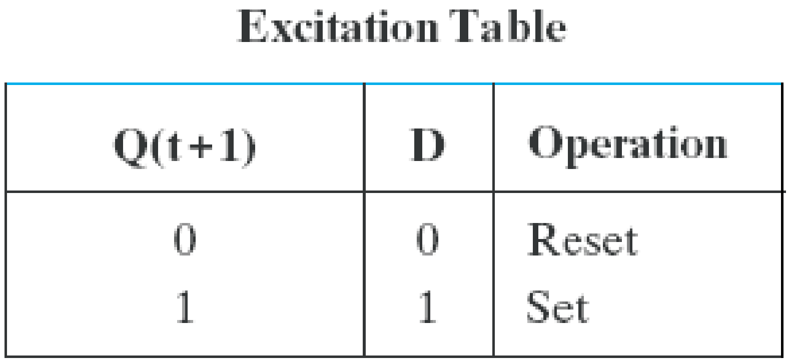
- Function of a Combinational Logic Circuit can be described using a Truth Table
  - Truth Table, however does not contain time variable
- Characteristic table is useful for analysis and for defining flip-flop's operation
  - it specifies the next state when inputs and present state are known
  - it contains 2^n rows, where **n** is the number of inputs
- Excitation table lists the required flip-flop inputs for a given change of state
  - It contains 2^{2m} rows, where **m** is the number of state variables (which may be the same as the outputs in simple circuits)

Characteristic and Excitation Tables: D and SR



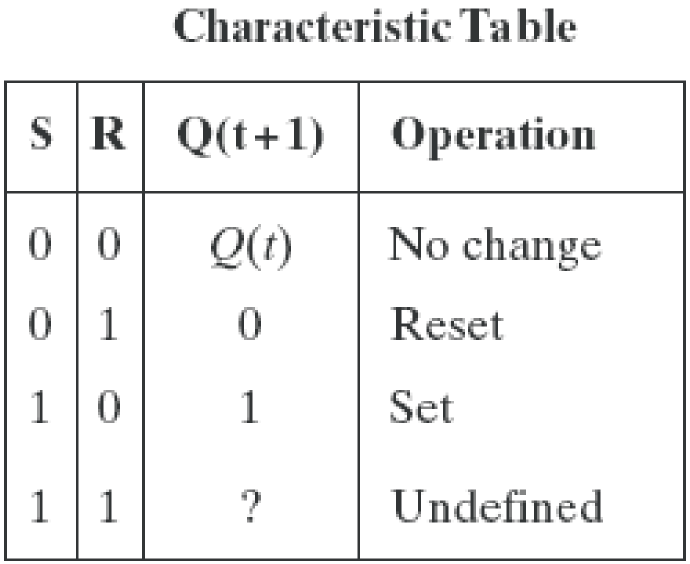
Characteristic Table

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



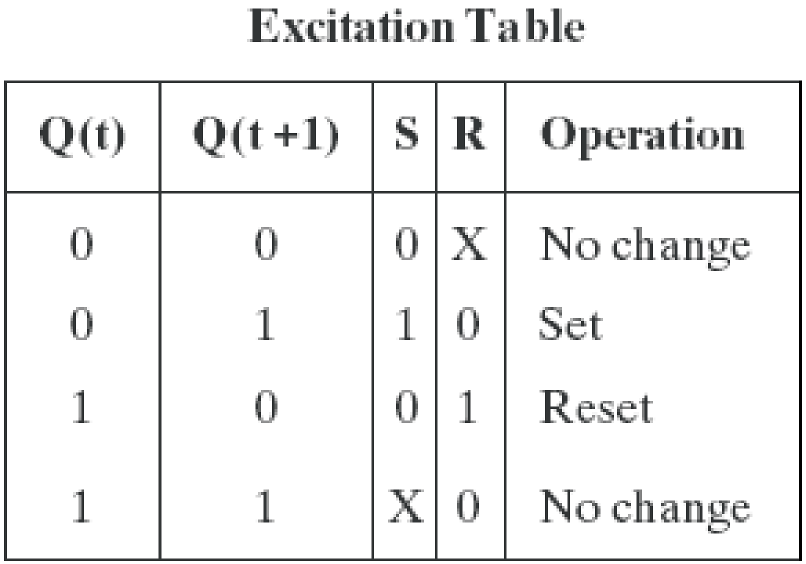
Excitation Table

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



Characteristic Table

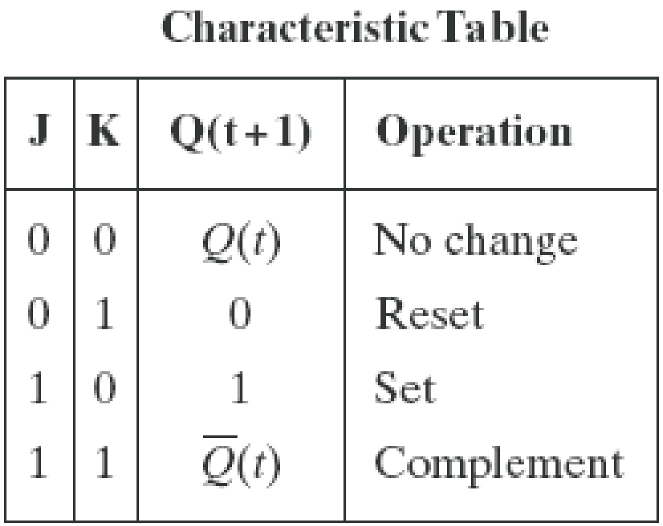
S	R	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined



Excitation Table

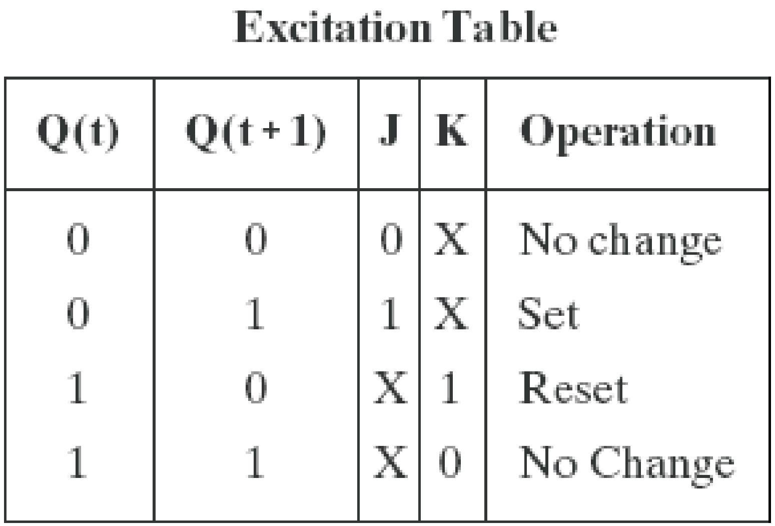
Q(t)	Q(t+1)	S	R	Operation
0	0	0	X	No change
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change

Characteristic and Excitation Tables: JK and T



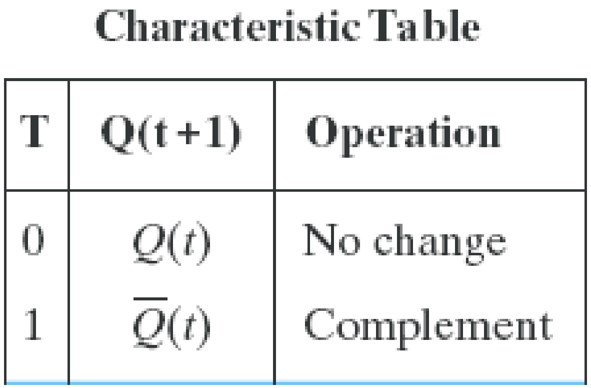
Characteristic Table

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



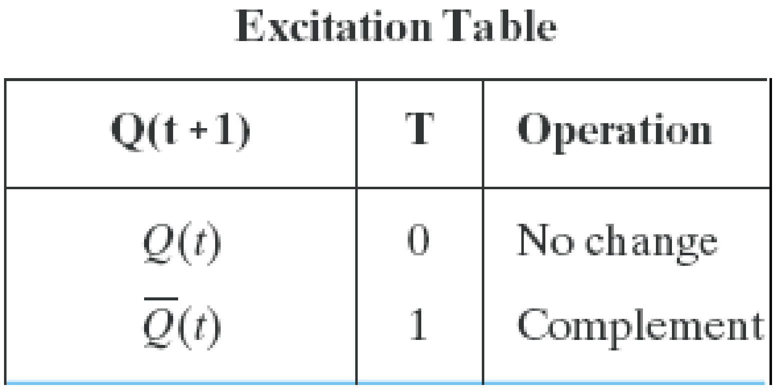
Excitation Table

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



Characteristic Table

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

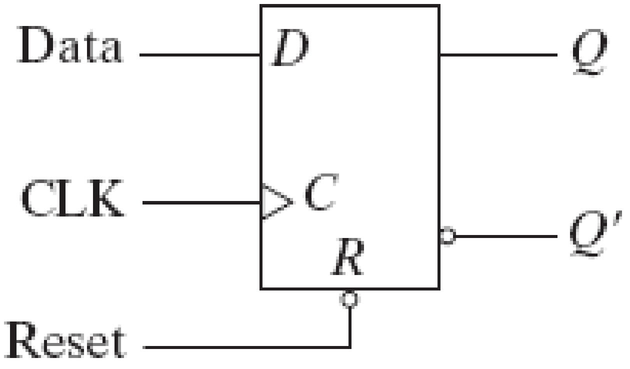


Excitation Table

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

Direct (or Asynchronous) Inputs

- Flip-flops often provide special inputs (set or reset) for setting and resetting them asynchronously
- Bubble in the output line indicates they are active at 0



(a) Graphic symbol

R	C	D	Q	Q'
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

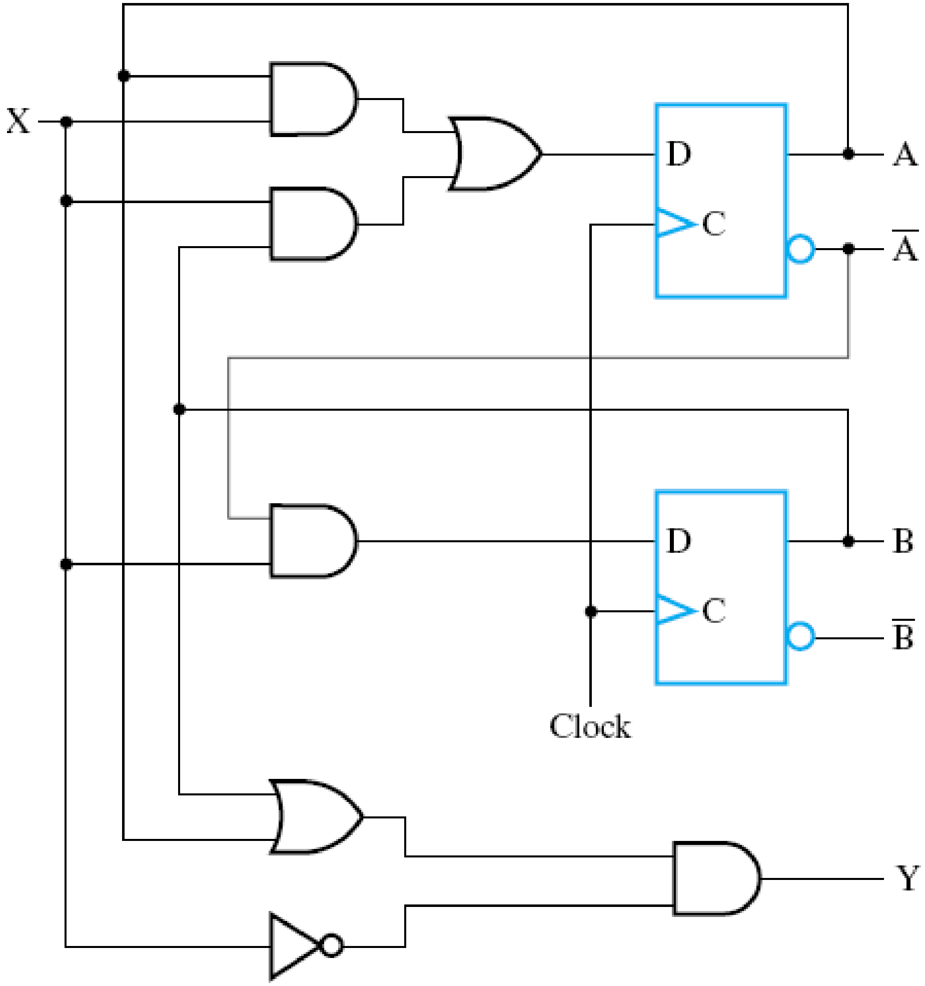
(b) Function table

Sequential Circuit Analysis

- Analysis consists of obtaining a table or diagram for the time sequence of inputs, outputs and states
- Analysis starts from a circuit diagram and culminates in a *state table* or *state diagram*
- A circuit with *n* binary state variables has 2^n possible states
  - 2^n is always finite, so sequential circuits are sometimes called *finite-state machines*

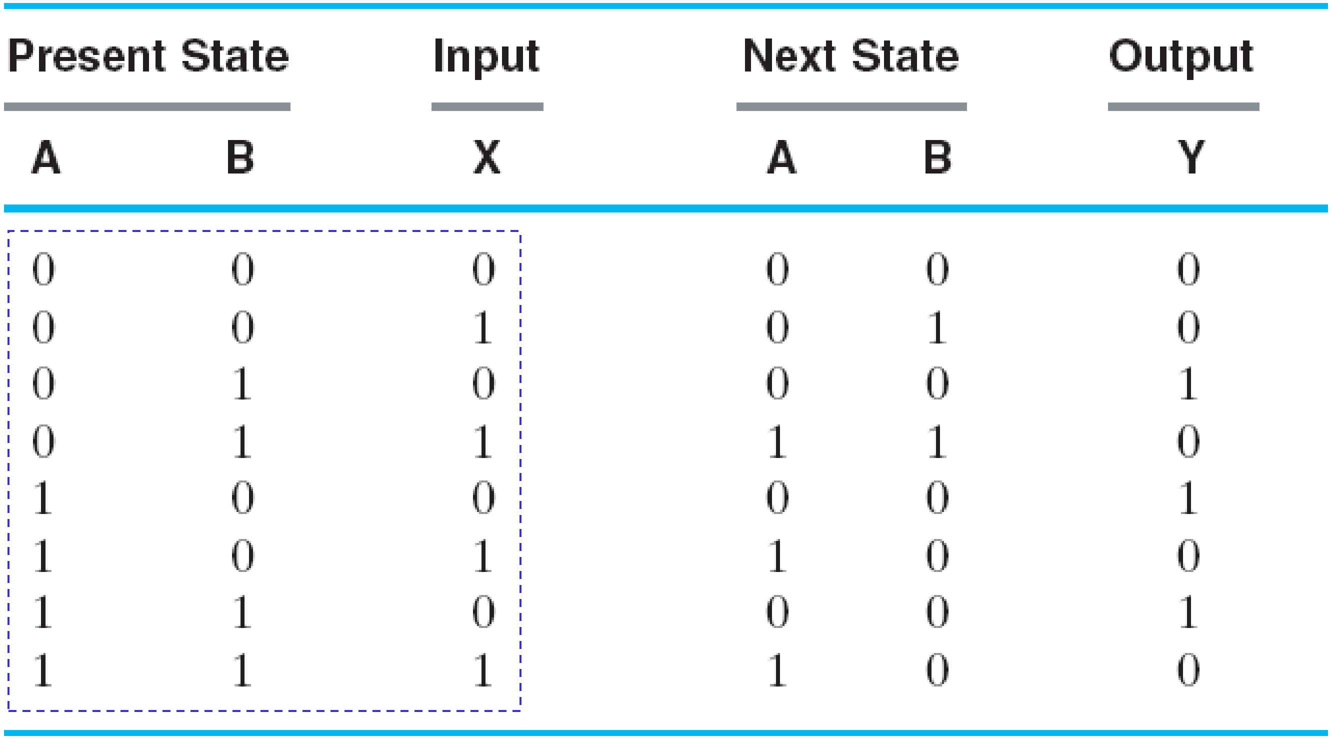
Sequential Circuit Analysis: FF Input Equations

- Flip-flop *input equations*
  - input generated by combinational circuit part
- Input equations:
  - $D_A = Ax + Bx$
  - $D_B = A'x$
- Output $y = (A + B)x'$
- A and B are state variables



Sequential Circuit Analysis: State Table

- State table
  - Next state derived from input equation



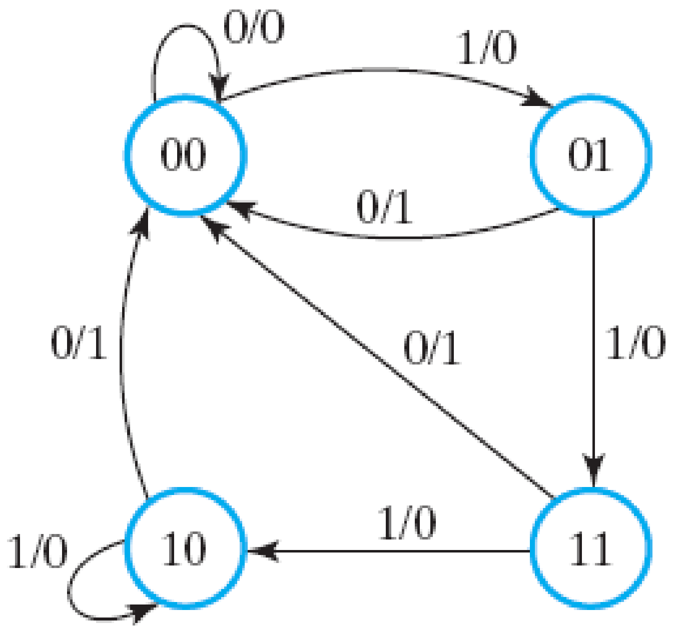
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

all combinations

Next state is D_A and D_B

Sequential Circuit Analysis: State Diagram

- No difference between state table and state diagram
- No. of inside circles = state of flip-flops
- Directed lines are labelled
  - – input during present state + "/" + output during present state

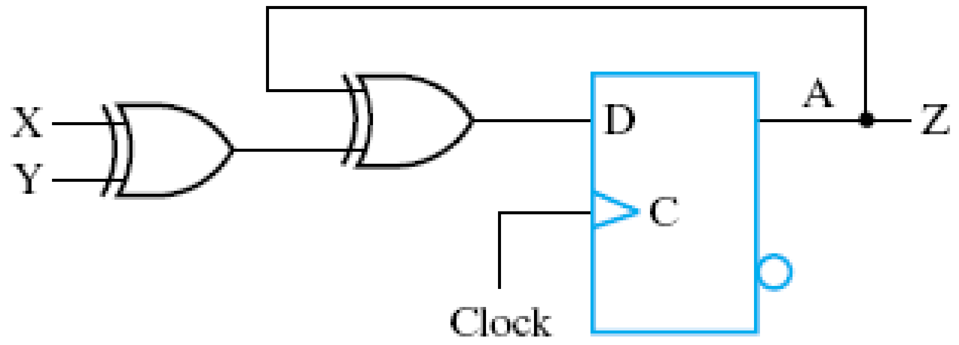


Line label = input/output

Mealy and Moore Model

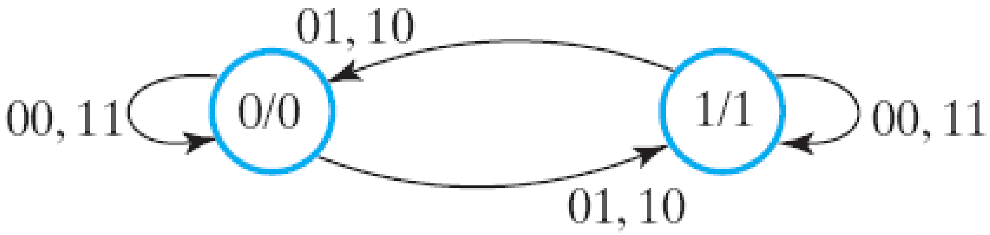
- *Mealy model* circuit
  - Sequential circuit in which outputs depend on inputs and states
  - Arcs contain inputs and outputs
- *Moore model*
  - Sequential circuit in which outputs depend only on states
  - Outputs included with the states in the circles

Moore Model Circuit Analysis: Example



$D_A = A \oplus X \oplus Y$

$Z = A$



Inside circle = state/output *Line label = input*

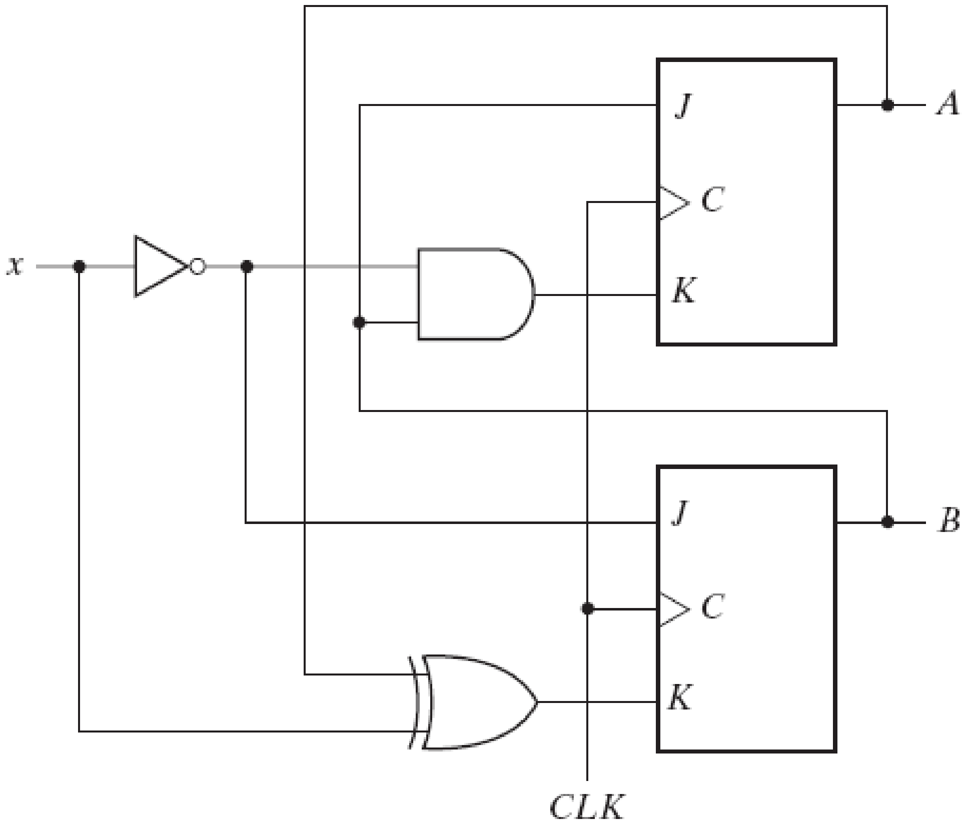
Next state is D_A

Present state	Inputs		Next state	Output
A	X	Y	A	Z
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

State table

all combinations

Analysis with JK Flip-flops: Input Equations



$J_A = B$

$K_A = Bx'$

$J_B = x'$

$K_B = A \oplus x$

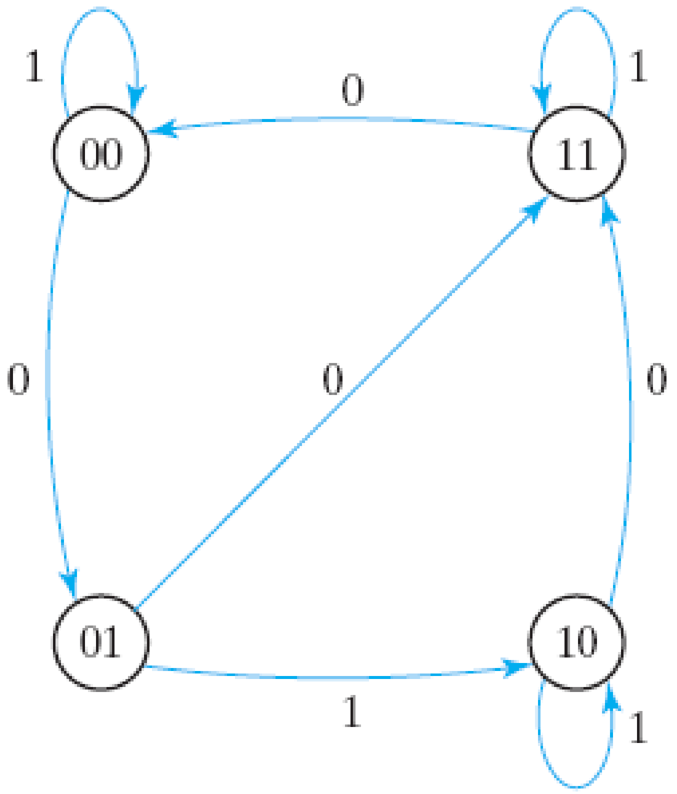
Analysis with JK Flip-flops: State Table

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

all combinations

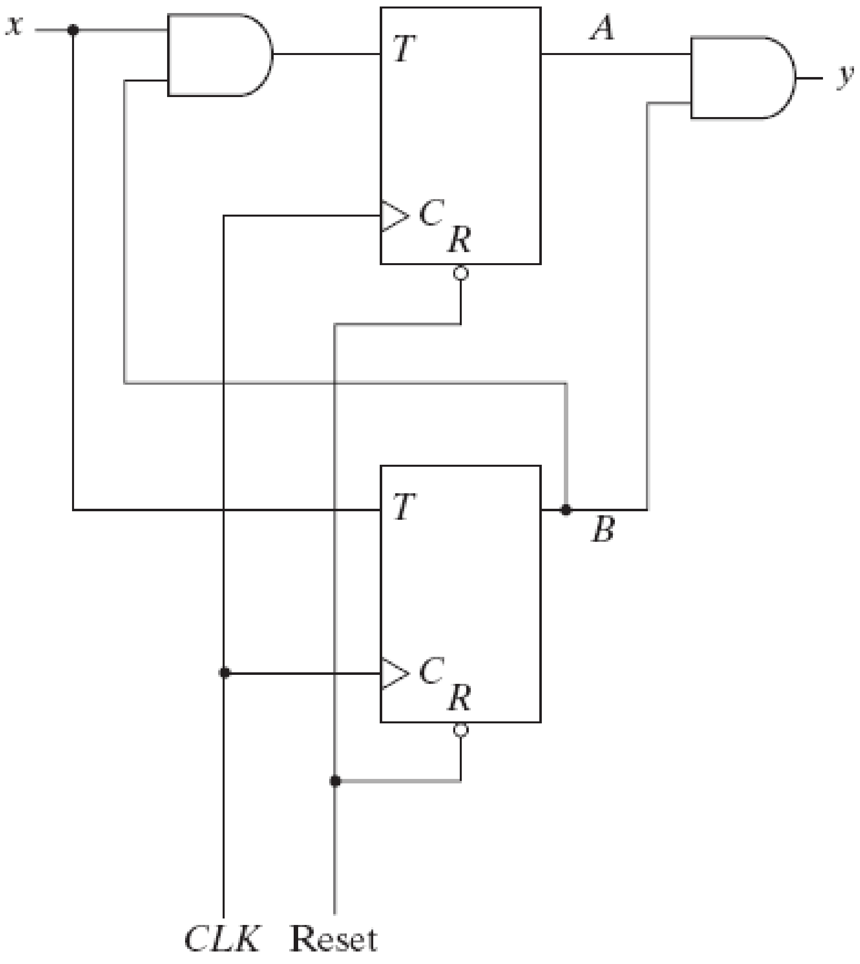
Next state is determined from flip-flop inputs

Analysis with JK Flip-flops: State Diagram



the circuit has no separate output
the state variables can be the outputs

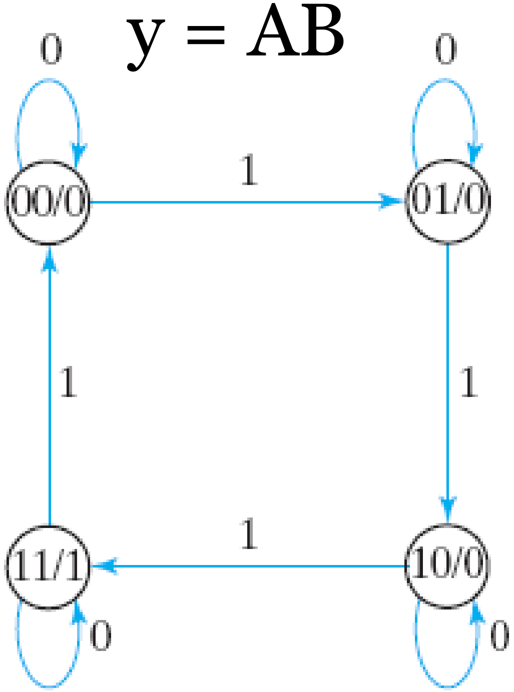
Analysis with T Flip-flops



$T_A = Bx$

$T_B = x$

$y = AB$



Analysis with T Flip-flops: State Table

Present State		Input	Next State		Output	Flip-flop Inputs	
A	B	x	A	B	y	T_A	T_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	1	0	0	0
0	1	1	1	0	0	1	1
1	0	0	1	0	0	0	0
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	0
1	1	1	0	0	0	1	1

all combinations

Next state is determined from flip-flop inputs: toggle if T=1

Sequential Circuit Design: Steps - 1

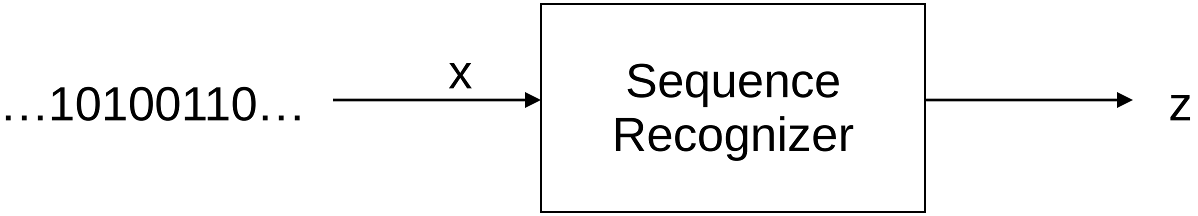
1. Specification
2. Derive state diagram
3. Obtain state table
4. State assignment
   - assign binary values to each state
   - often has major effect on excitation eqn, output eqn, and eventually circuit cost

Sequential Circuit Design: Steps - 2

5. Choose the type of flip-flop
   - JK for general applications
   - D for applications requiring transfer of data
   - T for applications involving complementation
6. Derive flip-flop input functions and the circuit output functions
7. Draw the logic diagram

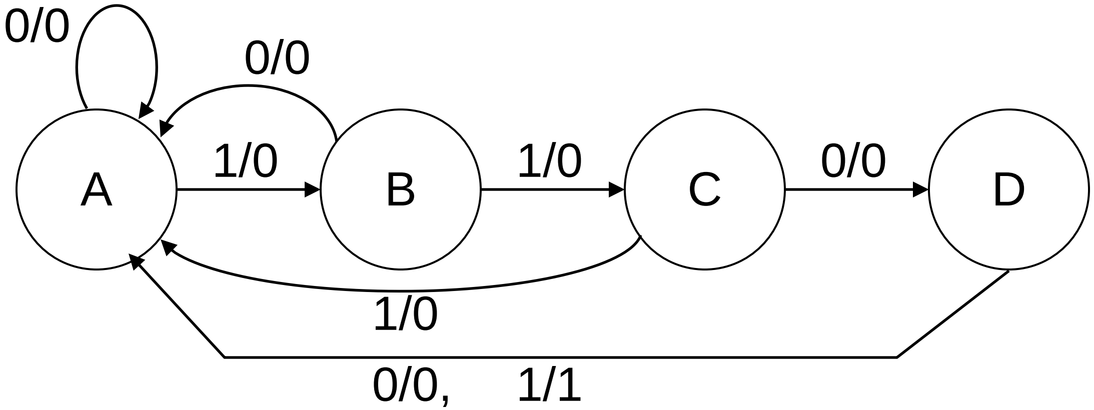
Sequence Recognizer: Specification and State Diagram

- Specification:
  - has one input X and one output Z
  - recognize the occurrence of 1101 bit sequence on X
    - Z = 1, when sequence detected, otherwise 0



Sequence Recognizer: State Diagram Explained - 1

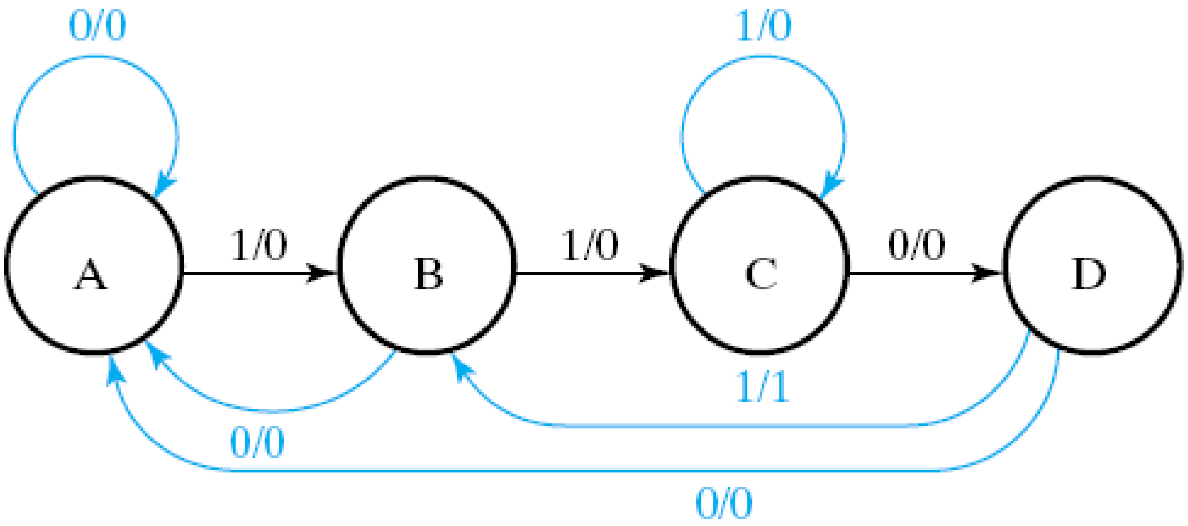
- Apparently (simply follow 1101 sequence):



- Can detect: ...0**1101**...
- Problem detecting overlaps: ...1**1101**..., ...110***1101***...

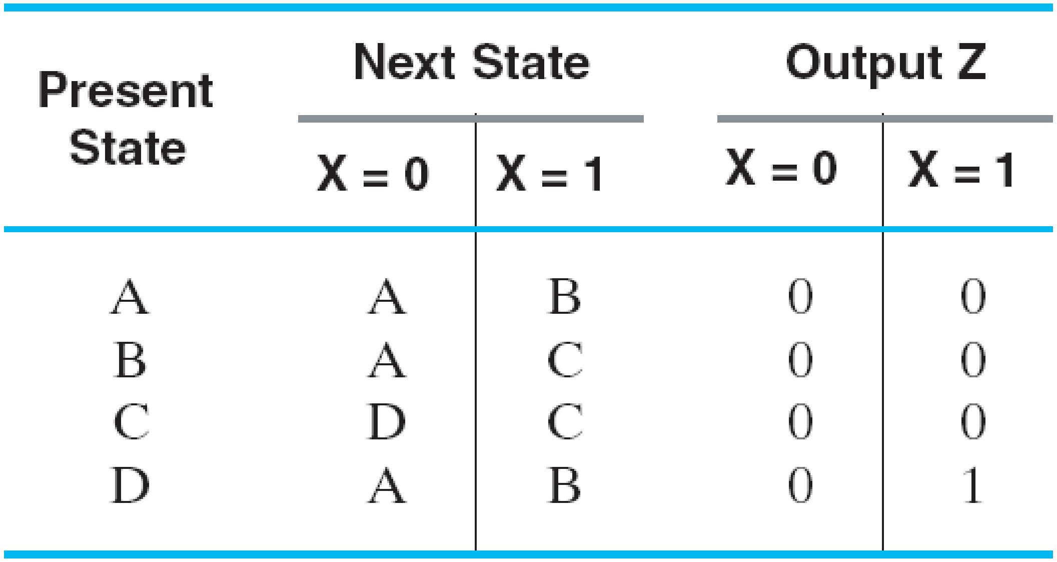
Sequence Recognizer: State Diagram Explained - 2

- Observations (e.g.):
  - To detect **11101**, whenever there was a **1**, we should count the next **1** as the “second”, i.e. going to state **C**
  - To detect tail overlap 110***1101***, the last **1** may be the first **1** in next pattern, i.e. going to state **B**



Sequence Recognizer: State Table

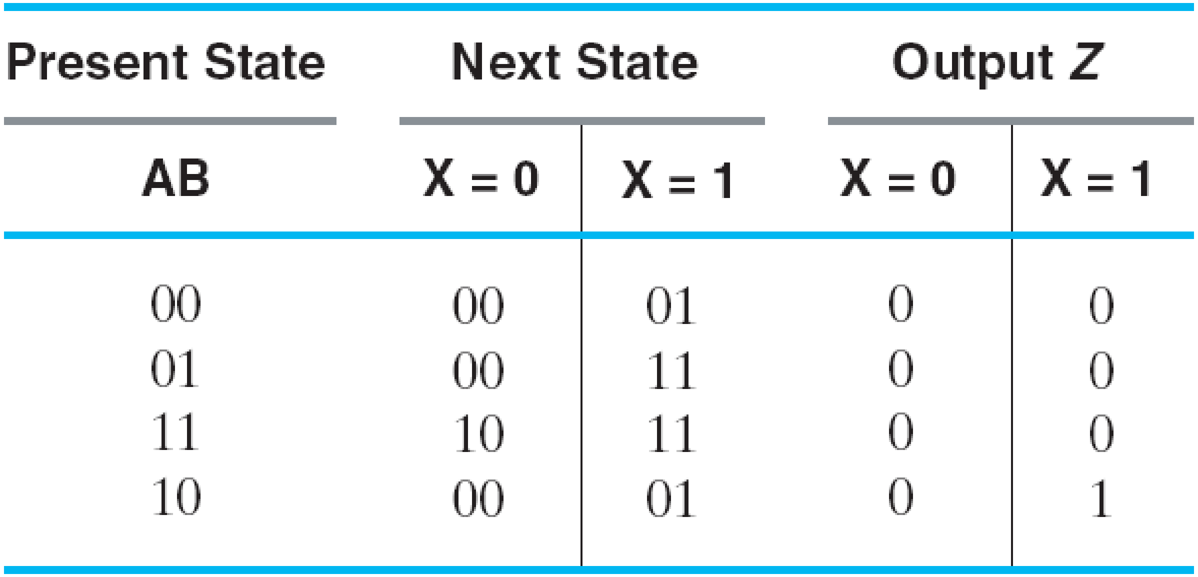
- If excess states are present, desirable to combine states into fewest needed



Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Sequence Recognizer: State Assignment

- Simplest to assign in binary counting order, or
- Gray code order
- Number of State Variables *n* gives 2^n states
  - for four (4) states, requires 2 variables (2 bits)



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

Sequence Recognizer: State Table Expanded

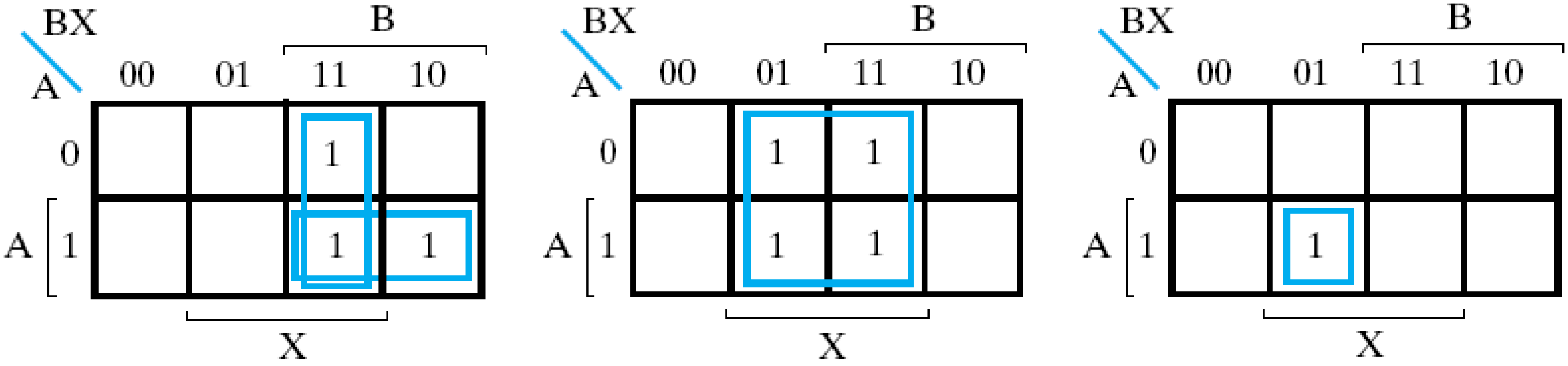
Present State		Input	Next State		Output
A	B	x	A	B	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	0

all combinations

this table allows for determination of flip-flop input equations

Sequence Recognizer: Determine Flip-Flop Input Equations

- Choose D flip-flop (simplest)
  - – if other flip-flop type chosen, the inputs (e.g. J and K) should be expanded into the State Table in previous slide



A \ BX	00	01	11	10
0			1	
1			1	1

$D_A = AB + BX$

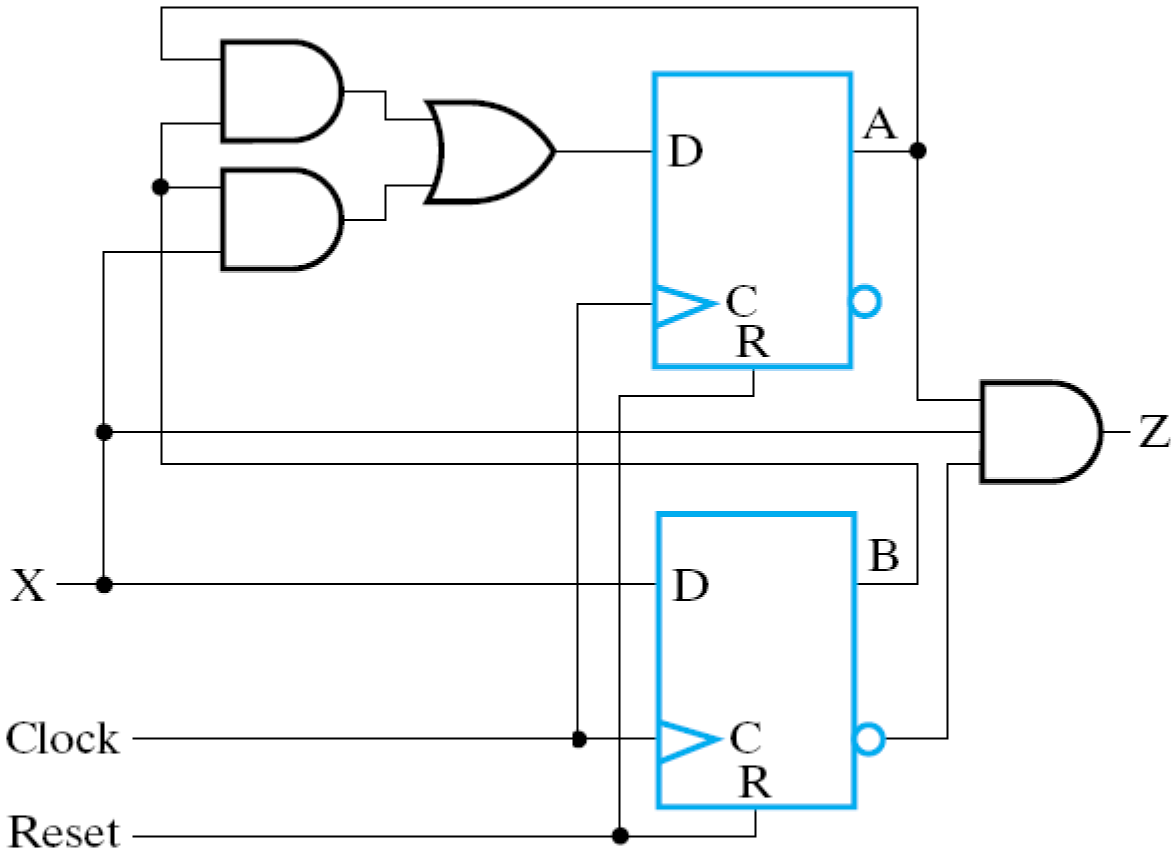
A \ BX	00	01	11	10
0		1	1	
1		1	1	

$D_B = X$

A \ BX	00	01	11	10
0				
1		1		

$Z = A\bar{B}X$

Sequence Recognizer: The Circuit



Sequence Recognizer: Design using JK Flip-flops

Present State		Input	Next State		Output	Flip-flop Inputs			
A	B	x	A	B	z	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	0	x	1	x
0	1	0	0	0	0	0	x	x	1
0	1	1	1	1	0	1	x	x	0
1	0	0	0	0	0	x	1	0	x
1	0	1	0	1	1	x	1	1	x
1	1	0	1	0	0	x	0	x	1
1	1	1	1	1	0	x	0	x	0

Sequence Recognizer: Design using JK Flip-flops – Input Equations

J_A	B’x’	B’x	Bx	Bx’
A’	0	0	1	0
A	x	x	x	x

$J_A = Bx$

K_A	B’x’	B’x	Bx	Bx’
A’	x	x	x	x
A	1	1	0	0

$K_A = B'$

J_B	B’x’	B’x	Bx	Bx’
A’	0	1	x	x
A	0	1	x	x

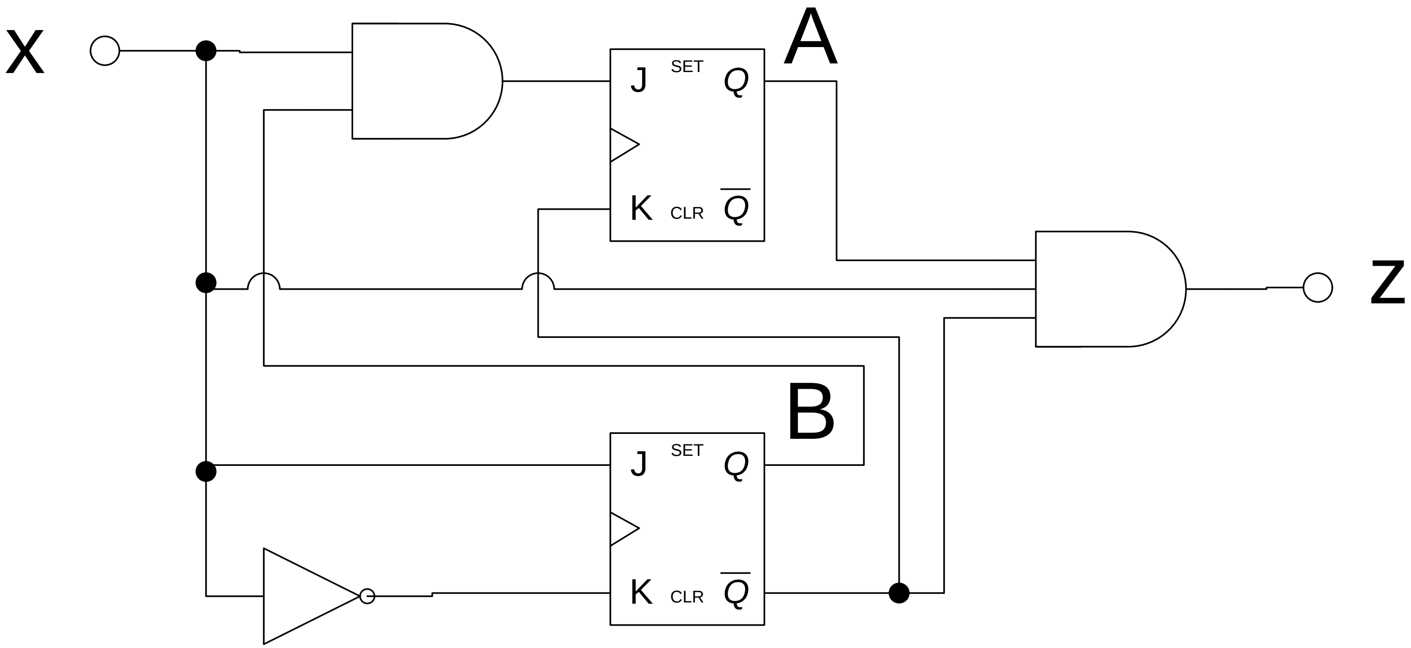
$J_B = x$

K_B	B’x’	B’x	Bx	Bx’
A’	x	x	0	1
A	x	x	0	1

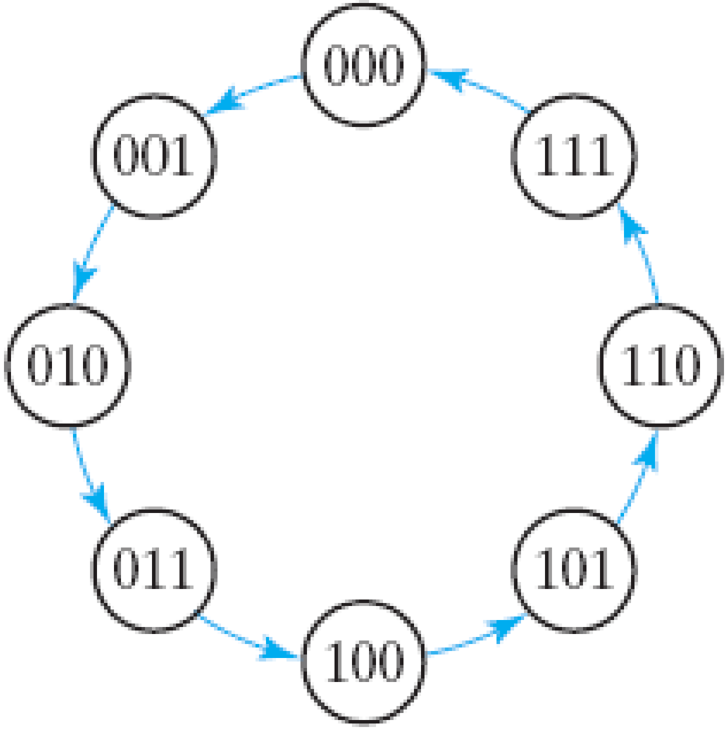
$K_B = x'$

z is not changed

Sequence Recognizer: Design using JK Flip-flops – Circuit



Counter: State Diagram

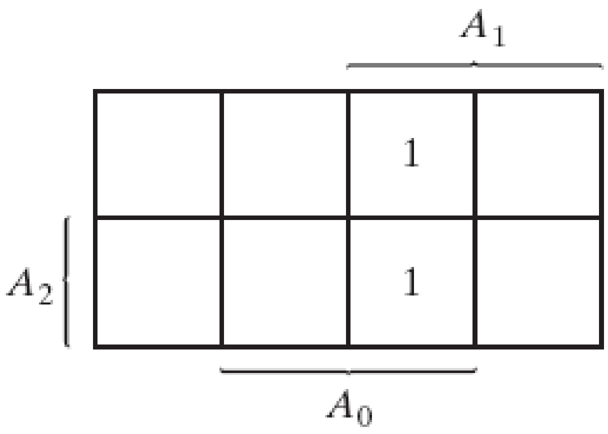


Counter: State Table Using T-FF

Present State			Next State			Flip-flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	TA_2	TA_1	TA_0
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

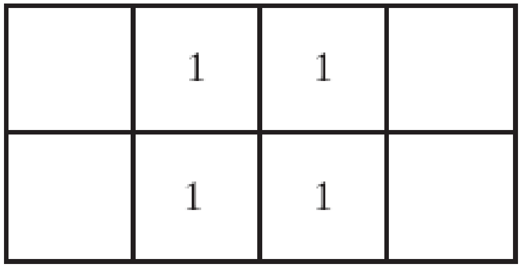
Next state is determined from flip-flop inputs: toggle if T=1

Counter: Flip-Flop Input Equations



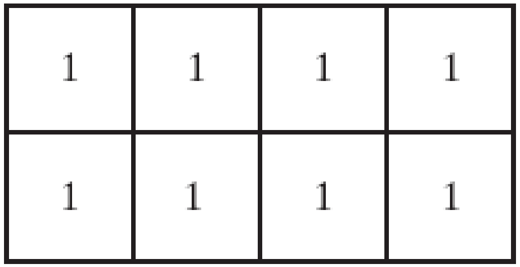
		1	
		1	

$$T_{A2} = A_1 A_0$$



	1	1	
	1	1	

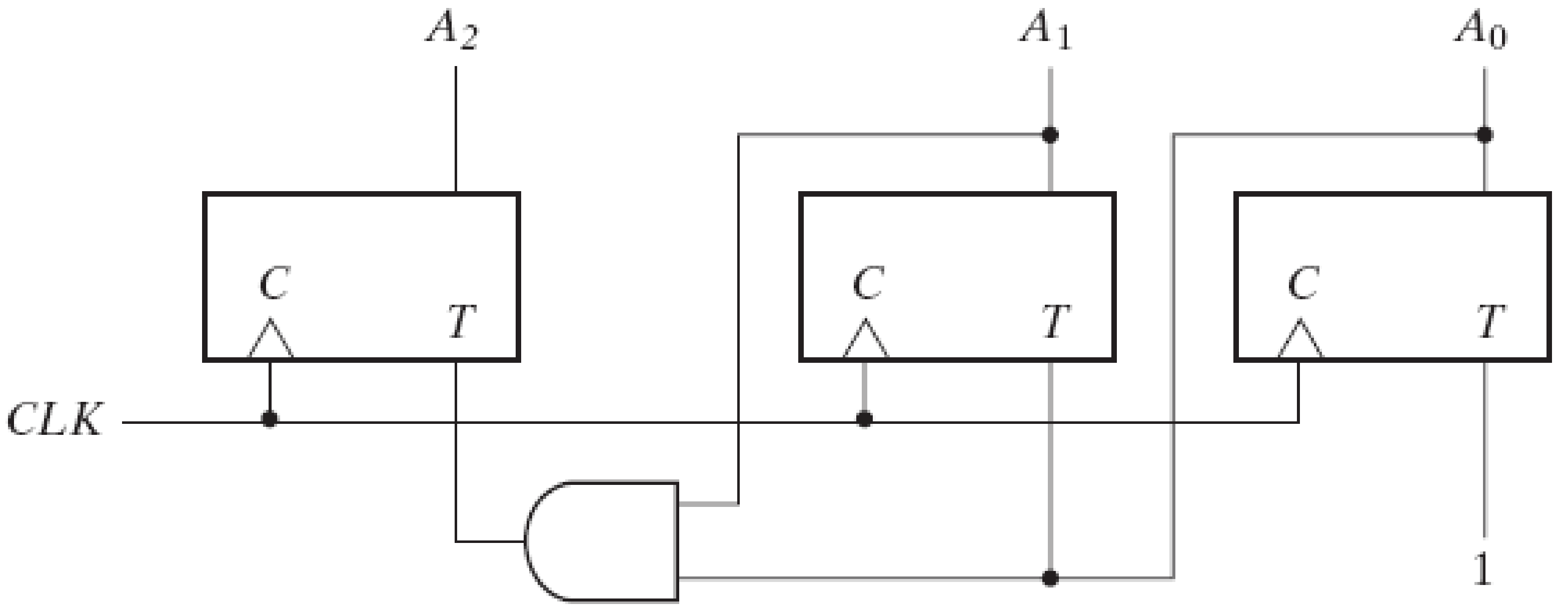
$$T_{A1} = A_0$$



1	1	1	1
1	1	1	1

$$T_{A0} = 1$$

Counter: Circuit Diagram



Summary

- Sequential logic circuit has output(s) determined not only by its input(s) but also by its previous state (inputs and outputs)
  - also called finite-state machines
- Building blocks are latches and flip-flops
  - flip-flops are edge-triggered while latches are level-triggered
- Two models of sequential logic circuit: Mealy and Moore
  - Mealy – output(s) depend on input(s) and state
  - Moore – output(s) depend on state only
- Analysis of a sequential logic circuit consists of obtaining the flip-flop input equations, state table and then state diagram
- Designing of a sequential logic circuit follows the reverse order of the analysis; it obtain the state diagram, state table, flip-flop input equations and finally the circuit
- The type of flip-flop used affects the columns in state table and the flip-flop input equations