



EE 2001

DIGITAL SYSTEMS

COURSE SUMMARY



OVERVIEW

1. Number Representation
2. Boolean Algebra
3. Gate level Minimization
4. Combinational Logic
5. Synchronous Sequential Logic
6. Registers and Counters
7. State Reduction and Hazards



BIBLIOGRAPHY

1. Prof. Ananth Krishnan, “EE2001 Lectures”
2. Prof. TG Venkatesh, “EE2001 Lectures”
3. Morris Mano, “Digital Design”
4. Soham Roy, “EE2001 Lecture Notes”
5. <https://electricalvoice.com/>



NUMBER REPRESENTATION

“There are 10 types of people in the world. Those who understand binary, those who don’t and those who didn’t expect this joke to be in base 3.”



Common Bases :

1. Binary (2) : 0 , 1
2. Octal (8) : 0,1,2,3,4,5,6,7
3. Decimal (10) : 0,1,2,3,4,5,6,7,8,9
4. HexaDecimal : 0-9 , A,B,C,D,E,F

$(123.45)_{10}$, where 10 is the Base/Radix and is to be mentioned
 $(123.45)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$




Base X to Decimal conversion

$(1010.10)_2 \rightarrow \text{Decimal}$

$$\Rightarrow 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2}$$

$$\Rightarrow 8 + 0 + 2 + 0 + 0.5 + 0$$

$$\Rightarrow (10.5)_{10}$$



$(123.4)_{16} \rightarrow \text{Decimal}$

$$\Rightarrow 1 \times 16^2 + 2 \times 16^1 + 3 \times 16^0 + 4 \times 16^{-1}$$

$$\Rightarrow 256 + 32 + 3 + 0.25$$

$$\Rightarrow (291.25)_{10}$$



Decimal to Base X conversion

$(10.5)_{10} \rightarrow \text{Binary}$


i. Take the integer part & do long division

$$\Rightarrow (10)_{10} = (1010)_2$$

ii. Take decimal part & keep multiplying by 2

$$\Rightarrow (0.5)_{10} = (0.1)_2$$

$$\Rightarrow (10.5)_{10} = (1010.1)_2$$


$$(291.25)_{10} \rightarrow (?)_{16}$$


i. Take integer part & do long division

$$\Rightarrow (291)_{10} = (123)_{16}$$

ii. Deal with fractional part

$$\Rightarrow (0.25)_{10} = (0.4)_{16}$$

$$\Rightarrow (291.25)_{10} = (123.4)_{16}$$

- 
1. For converting from base (X) to base (Y) : Go through DECIMAL base.
 2. Conversion can cause Representation Errors. Hence use CODING.

Eg:

Decimal	BCD				Excess-3			
	8	4	2	1	BCD + 0011			
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0



Representing Negative Numbers : 2S Complement

Step One : Take a number $X \rightarrow 0(11)_2$

Step Two : Flip all the bits individually. $X \rightarrow 011 \rightarrow 100$

Step Three : Add 1 at the LSB : 101

Check for yourselves , the sum is zero.

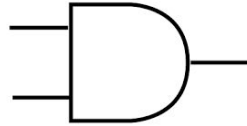


BOOLEAN ALGEBRA

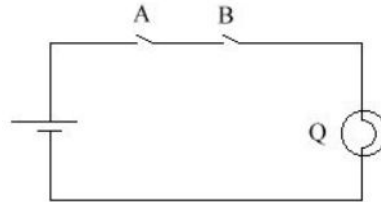
AND operation

Inputs needed : Minimum Two

Circuit Symbol :



Circuit Realization :



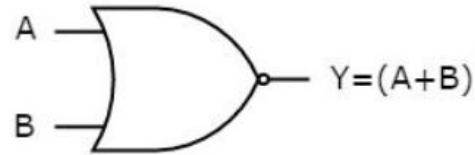


A	B	$Y = A.B$
0	0	0
0	1	0
1	0	0
1	1	1

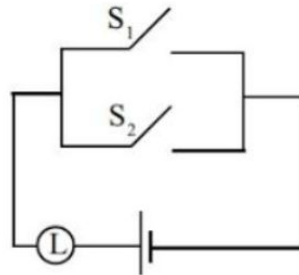
OR operation

Inputs needed : Minimum Two

Circuit Symbol :



Circuit Realization :



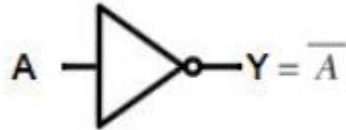


A	B	$Y = A+B$
0	0	0
0	1	1
1	0	1
1	1	1

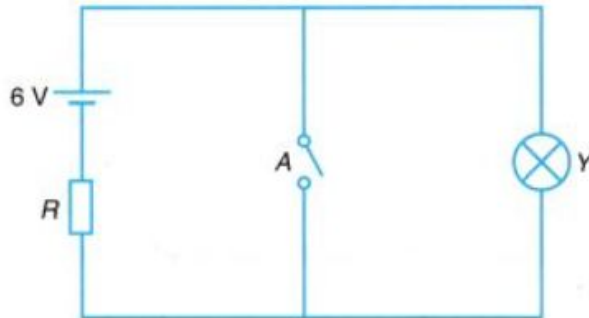
NOT operator

Inputs needed : One Only

Circuit Symbol :



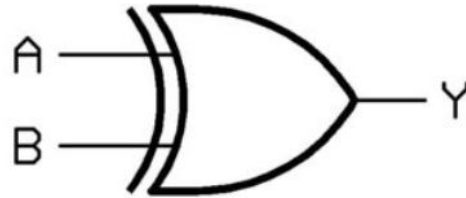
Circuit Realization :



XOR operator

Inputs needed : Minimum Two

Circuit Symbol :



$$Y = A \oplus B.$$

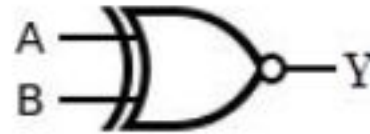


A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

XNOR operator

Inputs needed : Minimum Two

Circuit Symbol :



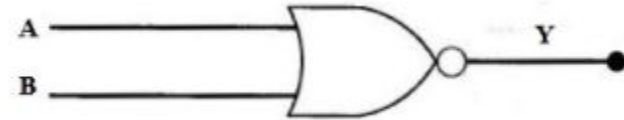
$$Y = \overline{A \oplus B}$$



A	B	$Y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1



NOR GATE



NAND GATE



Postulates and Theorems of Boolean Algebra

Postulate	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Simplification Theorems:—

$$(a) \quad a \cdot b + a \cdot b' = a$$

$$(b) \quad (a + b) \cdot (a + b') = a$$

$$(c) \quad a + a \cdot b = a$$

$$(d) \quad a + a' \cdot b = a + b$$

$$(e) \quad (a + b') \cdot b = a \cdot b$$

$$(f) \quad a \cdot b' + b = a \cdot b' + b + a \cdot b = (a + b) \cdot (b' + b) = a + b$$

—

Some other useful ones...



GATE LEVEL MINIMIZATION

**OBJECTIVE : Develop a
standard way of minimizing
logic in order to reach the
desired global minima**

—



Sum of Products (SOP)

An expression is said to be in SOP form when all products are products of single literals and added up together.

Examples :

(a) $ab + c'de + ad'e$ (Note that \cdot is omitted)

(b) $a + b + c + d' + e$



Product of Sums (POS)

An expression is in POS form when all sums are sums of single literals and then ANDed together (multiplied).

Example :

$$(a) \quad (a + b) \cdot (c + d')$$

$$(b) \quad (a + b' + c) \cdot (c + a' + d)$$



Minterm

A minterm of n variables is a product of n literals in which each variable (or literal) appears exactly once in either original or complement form, but not both.



Maxterm

A maxterm of n variables is a sum of n literals in which each variable appears exactly once in either original or complement form but not both.



a	b	c	f	f'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Where $f = 1$, write the function as SOP.

$$f = a'bc + ab'c' + ab'c + abc' + abc$$



To Write Maxterm expansion :

Write f' as SOP and then complement the result.

$$f' = a'b'c' + a'b'c + a'bc'$$

$$f = (f')'$$

$$= (a'b'c' + a'b'c + a'bc')'$$

$$= (a + b + c) (a + b + c') (a + b' + c)$$

Row No.	a	b	c	Minterm	Maxterm
0	0	0	0	$a'b'c' = m_0$	$a + b + c = M_0$
1	0	0	1	$a'b'c = m_1$	$a + b + c' = M_1$
2	0	1	0	$a'bc' = m_2$	$a + b' + c = M_2$
3	0	1	1	$a'bc = m_3$	$a + b' + c' = M_3$
4	1	0	0	$ab'c' = m_4$	$a' + b + c = M_4$
5	1	0	1	$ab'c = m_5$	$a' + b + c' = M_5$
6	1	1	0	$abc' = m_6$	$a' + b' + c = M_6$
7	1	1	1	$abc = m_7$	$a' + b' + c' = M_7$

Minterm and Maxterm corresponding to all possible combinations of 3 variables.



Completely Specified

Value of function defined clearly as 0 or 1 for all rows of truth table

Incompletely Specified

Value of function is not defined as 0 or 1 for certain rows of truth table

a	b	c	f
0	0	0	1
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1


X \rightarrow Don't care whether $f = 0$ or $f = 1$.

ENTER K-MAPS



How to approach this ?

$$f = \sum_{n=4} (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$



wx^{yz}	00	01	11	10
00	$m_0 \ 1$	$m_1 \ 1$	m_3	$m_2 \ 1$
01	$m_4 \ 1$	$m_5 \ 1$	m_7	$m_6 \ 1$
11	$m_{12} \ 1$	$m_{13} \ 1$	m_{15}	$m_{14} \ 1$
10	$m_8 \ 1$	$m_9 \ 1$	m_{11}	m_{10}

$$f = y' + w'z' + xz'$$



Observations

- (a) One cell \rightarrow One Minterm with 4 literals.
- (b) 2 adjacent cells \rightarrow One Minterm with 3 literals.
- (c) 4 adjacent cells \rightarrow One Minterm with 2 literals.
- (d) 8 adjacent cells \rightarrow One Minterm with 1 literal.



Prime Implicant

Product term obtained by combining maximum number of adjacent squares to accommodate each minterm.

Essential Prime Implicant

If a minterm is covered by only one Prime Implicant, then that Prime Implicant is called Essential Prime Implicant.

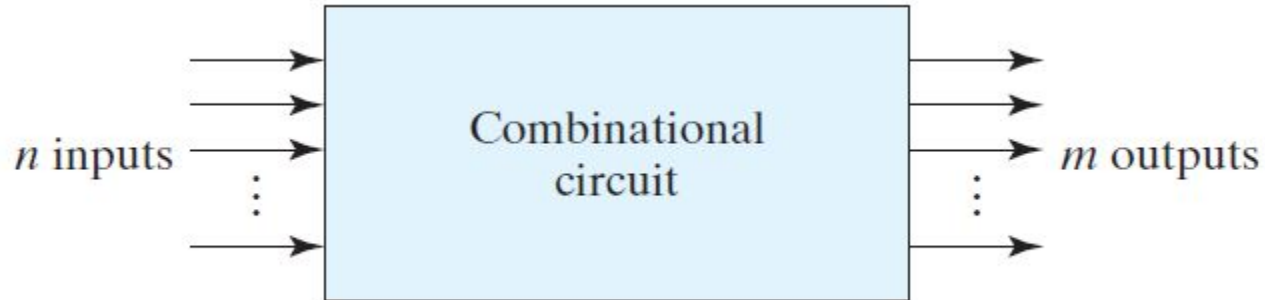


COMBINATIONAL LOGIC

ADDER - DECODER - MULTIPLEXER

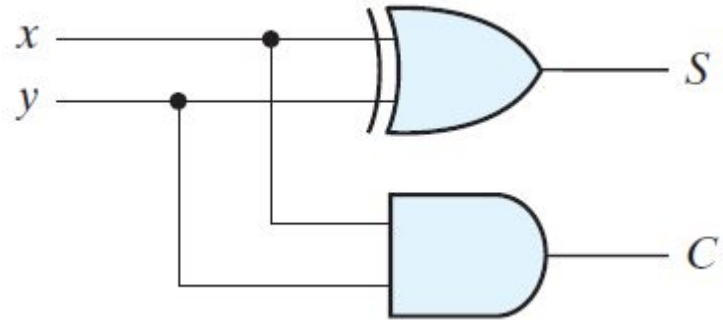


A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs.



Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



$$S = x'y + xy'$$

$$C = xy$$

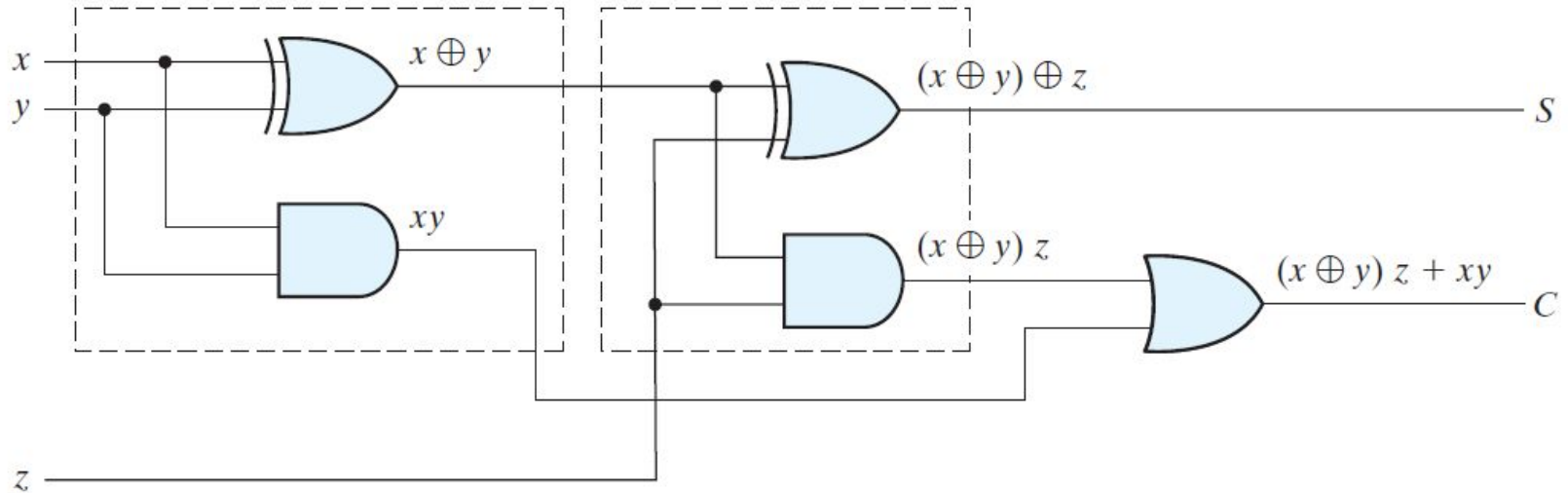


Full Adder

<i>x</i>	<i>y</i>	<i>z</i>	<i>C</i>	<i>S</i>
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

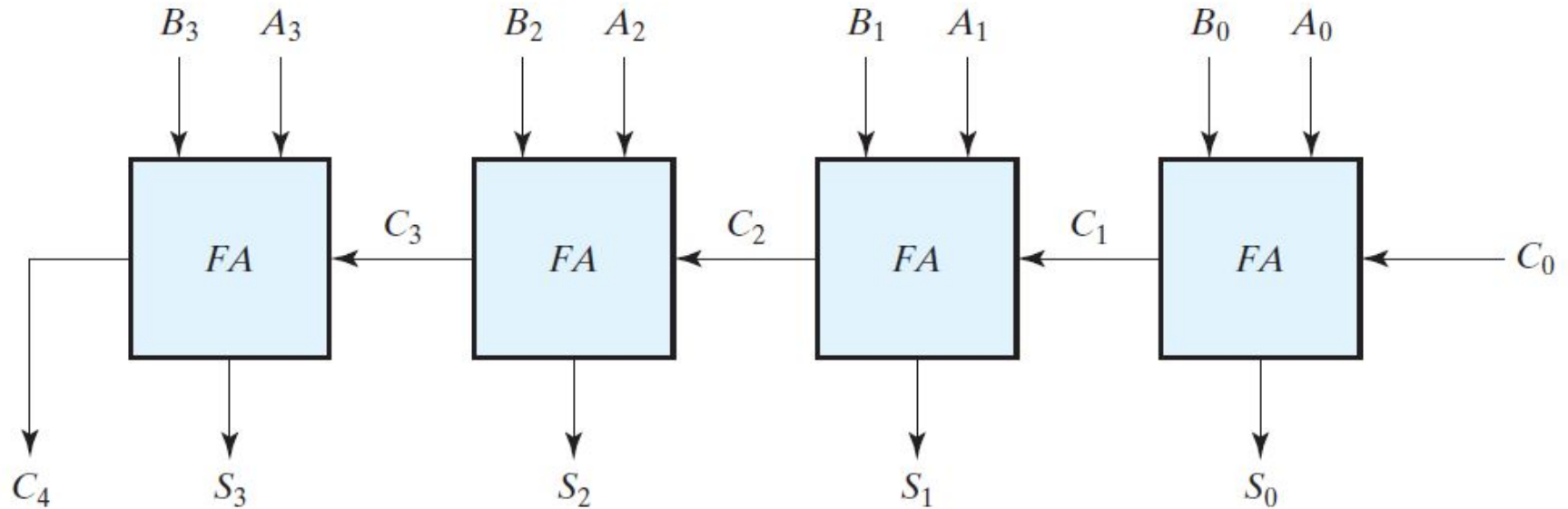
$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz$$



Implementation of a full adder using 2 Half adder and an OR gate

FOUR BIT BINARY ADDER





DECODER

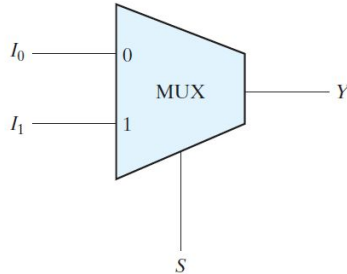
- A binary code of n bits is capable of representing up to 2^n distinct elements of coded information.
- A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.

Inputs			Outputs							
<i>x</i>	<i>y</i>	<i>z</i>	<i>D</i> ₀	<i>D</i> ₁	<i>D</i> ₂	<i>D</i> ₃	<i>D</i> ₄	<i>D</i> ₅	<i>D</i> ₆	<i>D</i> ₇
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Truth table of a Three-to-Eight-Line Decoder

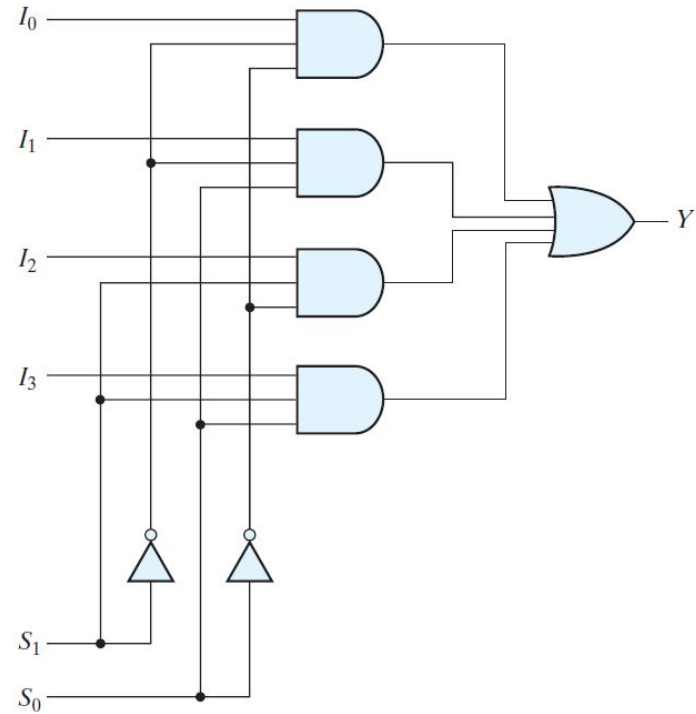
MULTIPLEXERS (MUX)

- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Four-to-one-line Mux





SYNCHRONOUS SEQUENTIAL LOGIC

LATCHES AND FLIP FLOPS

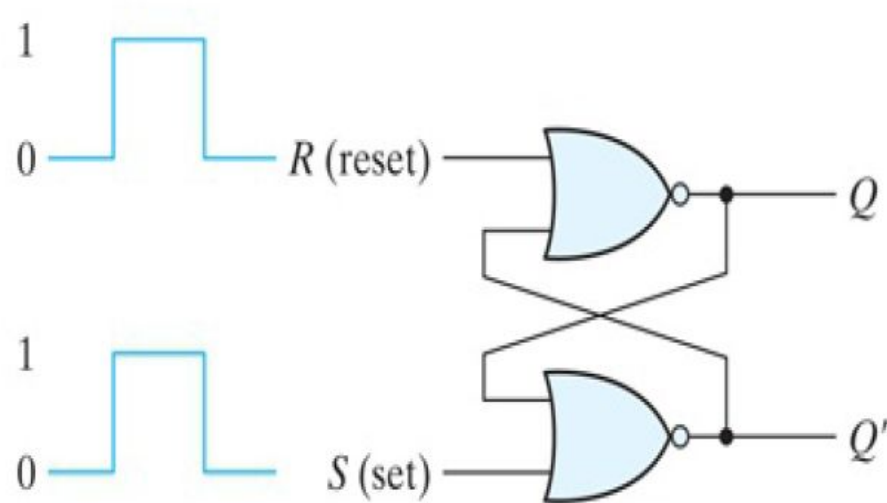
Combinatorial circuit → Where output depends on the present inputs 'only'

Sequential circuit → Where output depends on the internal states 'as well'

So we need some way of storing the states in a Sequential circuit :o

Enter STORAGE ELEMENTS

SR LATCH

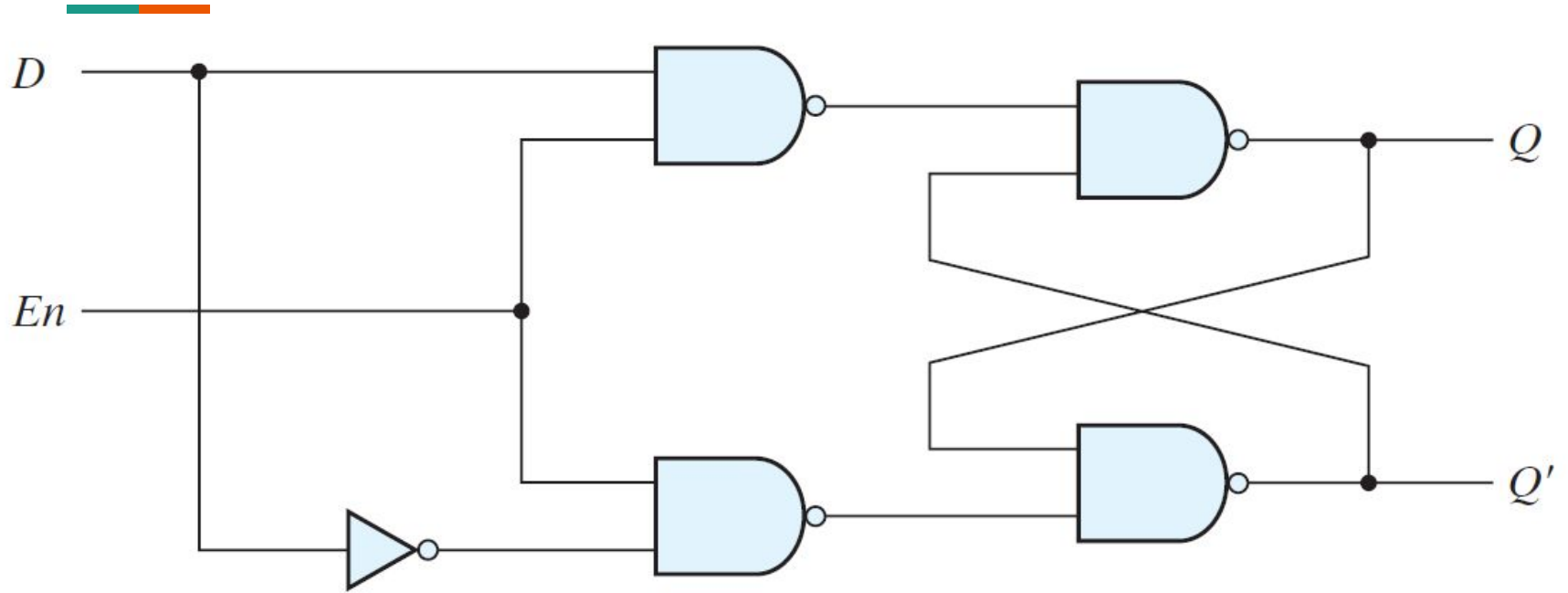



(a) Logic diagram

S	R	Q	Q'
1	0	1	0
0	0	1	0 (after $S = 1, R = 0$)
0	1	0	1
0	0	0	1 (after $S = 0, R = 1$)
1	1	0	0 (forbidden)

(b) Function table

D LATCH



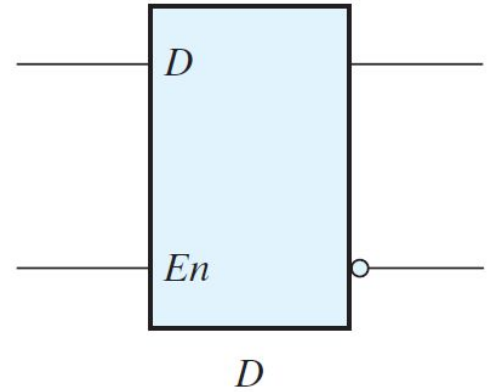
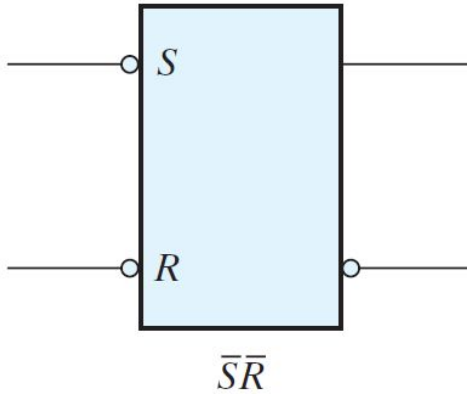
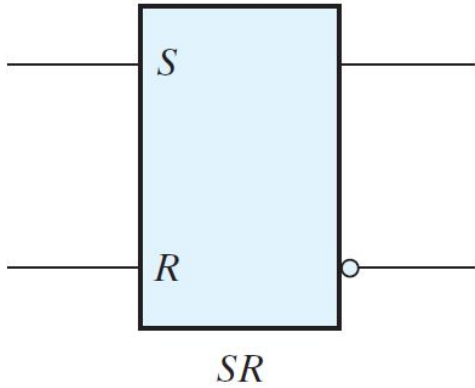


En	D	Next state of Q
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

This situation provides a path from input D to the output, and for this reason, the circuit is often called a transparent latch.



SYMBOLS OF LATCHES





LATCHES

- Level Sensitive Devices
- Basic Elements used to construct flip flops
- Useful in the case of Asynchronous sequential circuits.

FLIP FLOPS

- Edge Sensitive Devices
- More reliable as there are no unpredictable states
- Useful in the case of Synchronous sequential circuits.



(a) Response to positive level



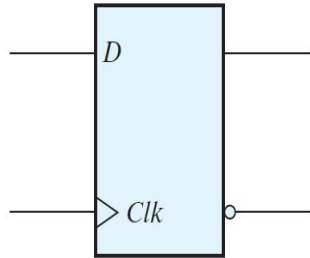
(b) Positive-edge response



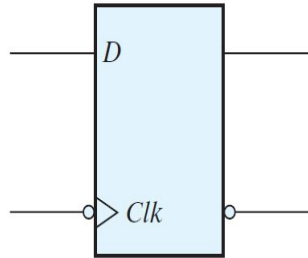
(c) Negative-edge response

D - FF

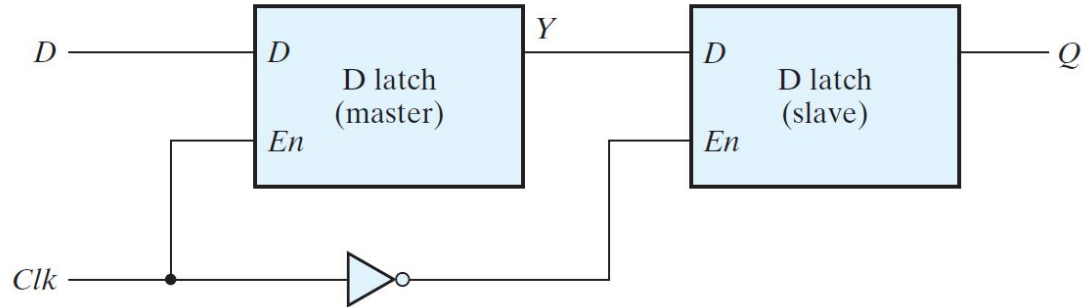
- A change in the output of the flip-flop can be triggered only by and during the transition of the clock from 1 to 0.



(a) Positive-edge



(a) Negative-edge



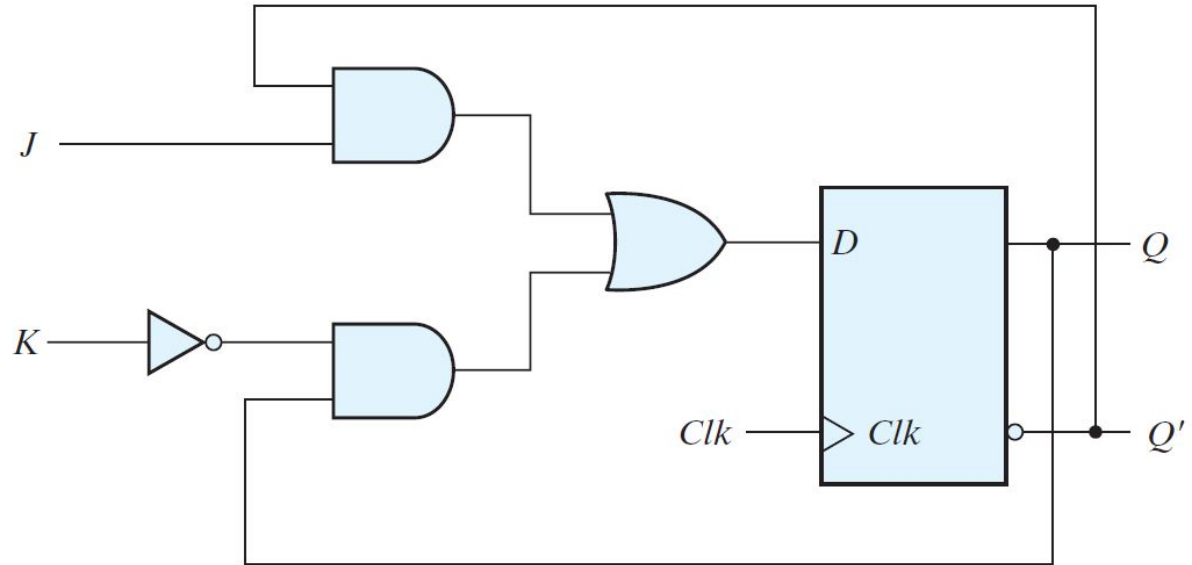
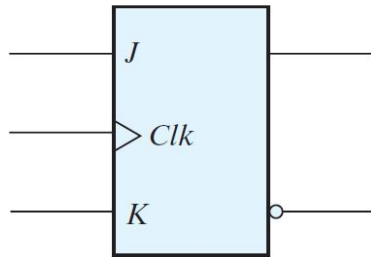


CHAR EQN AND TABLE

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

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

JK - FF



CHAR EQN AND TABLE

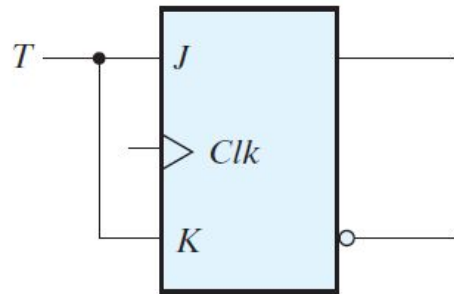
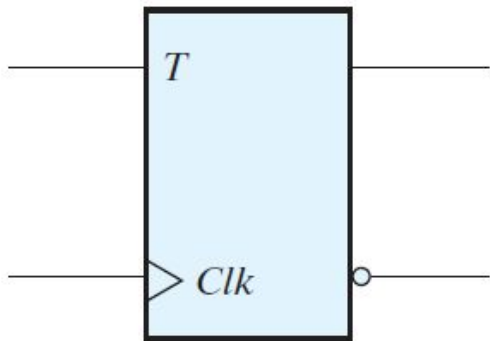
***JK* Flip-Flop**

<i>J</i>	<i>K</i>	$Q(t + 1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

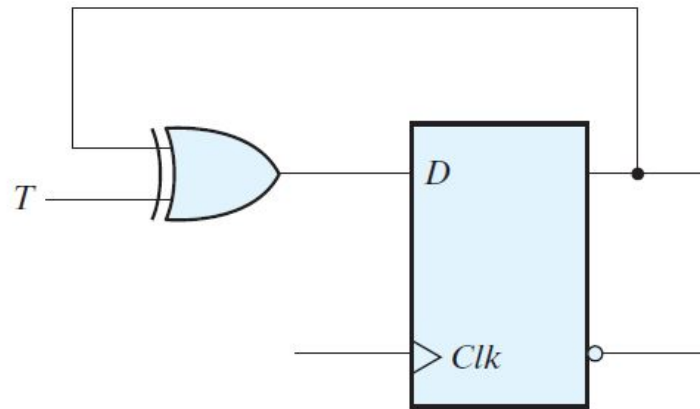
$$Q(t + 1) = JQ' + K'Q$$



T - FF



(a) From JK flip-flop



(b) From D flip-flop

CHAR EQN AND TABLE

***T* Flip-Flop**

<i>T</i>	$Q(t + 1)$
0	$Q(t)$ No change
1	$Q'(t)$ Complement

$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$



REGISTERS AND COUNTERS



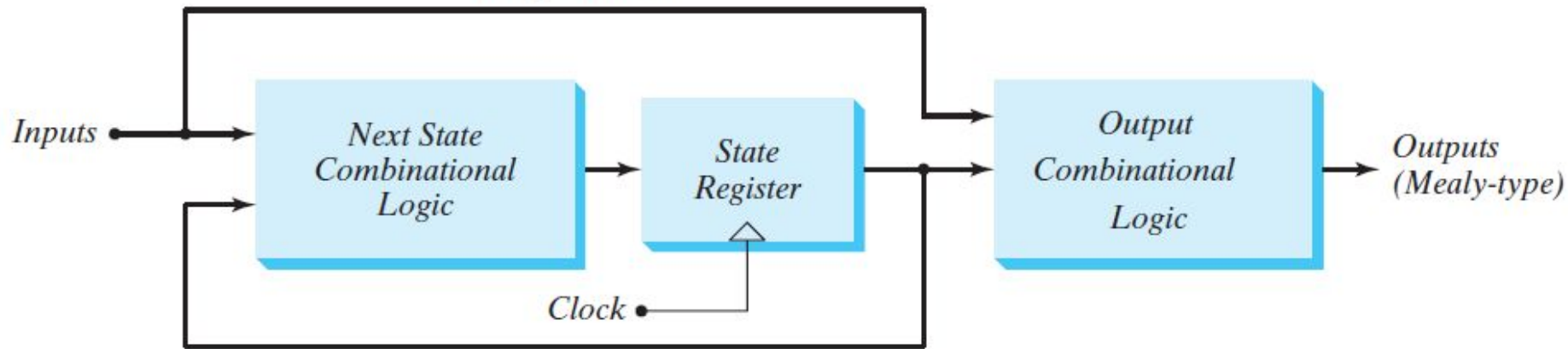
MEALY MACHINES

- The output is a function of both the present state and the input.
- The output is the value that is present immediately before the active edge of the clock.

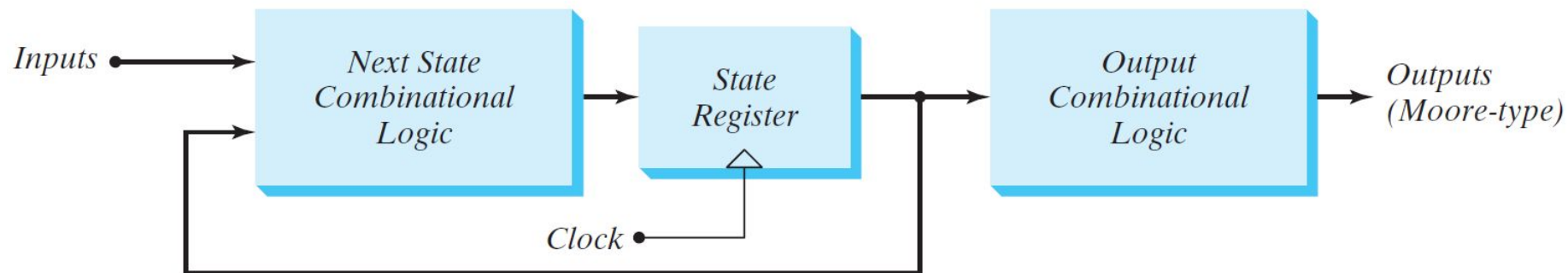
MOORE MACHINES

- The output is a function of only the present state.
- The outputs are synchronized with the clock, because they depend only on flip-flop outputs that are synchronized with the clock.

Mealy Machine



Moore Machine



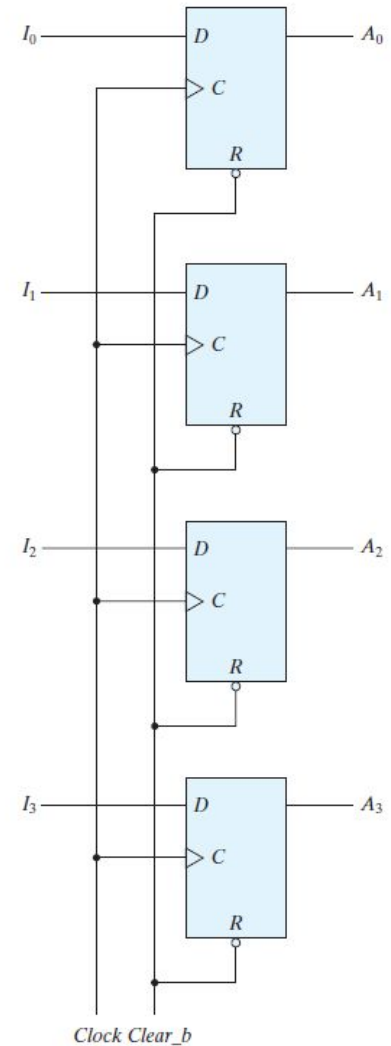


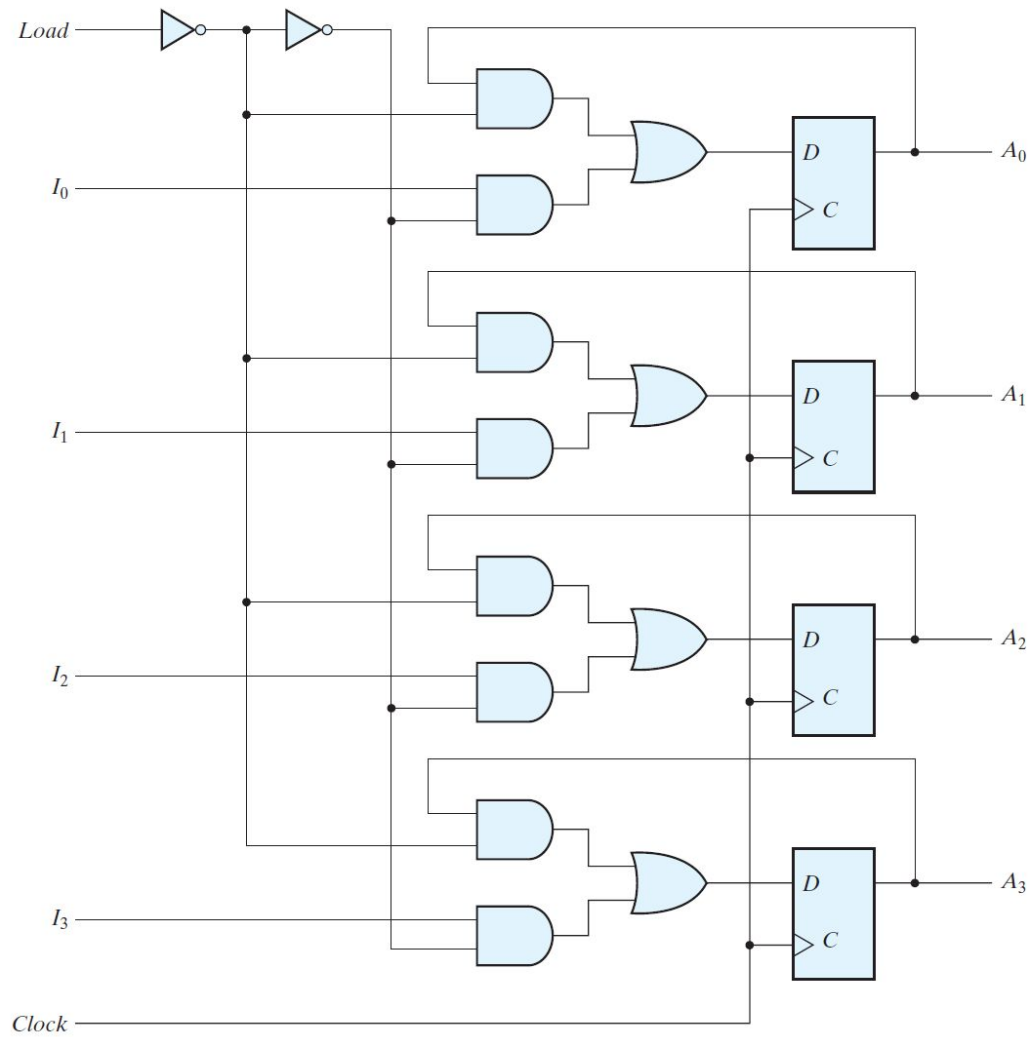
REGISTER

- A register is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information.
- In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.
- The flip-flops hold the binary information, and the gates determine how the information is transferred into the register

Register with Parallel Loads

- The transfer of new information into a register is referred to as loading or updating the register.
- If all the bits of the register are loaded simultaneously with a common clock pulse, we say that the loading is done in parallel.
- Wait. How to Stop Loading?



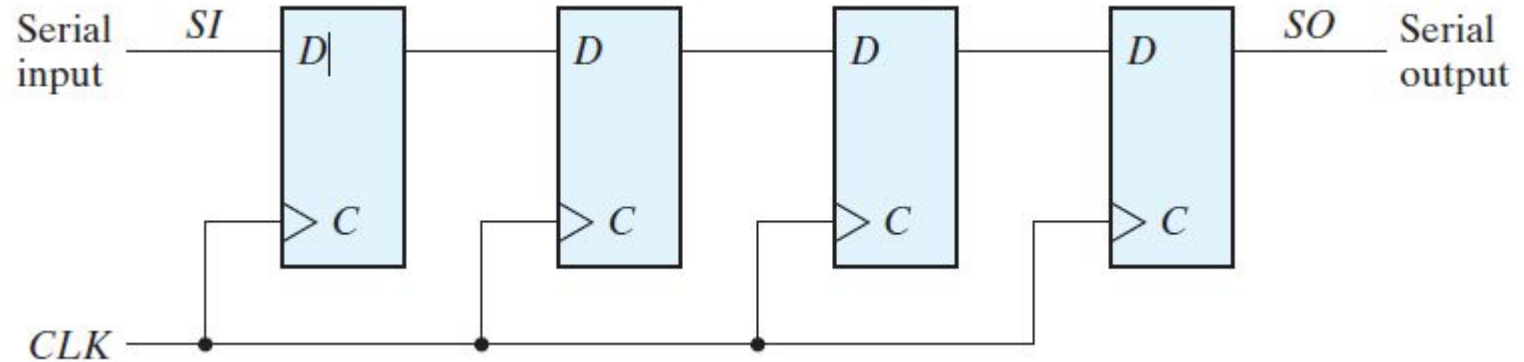


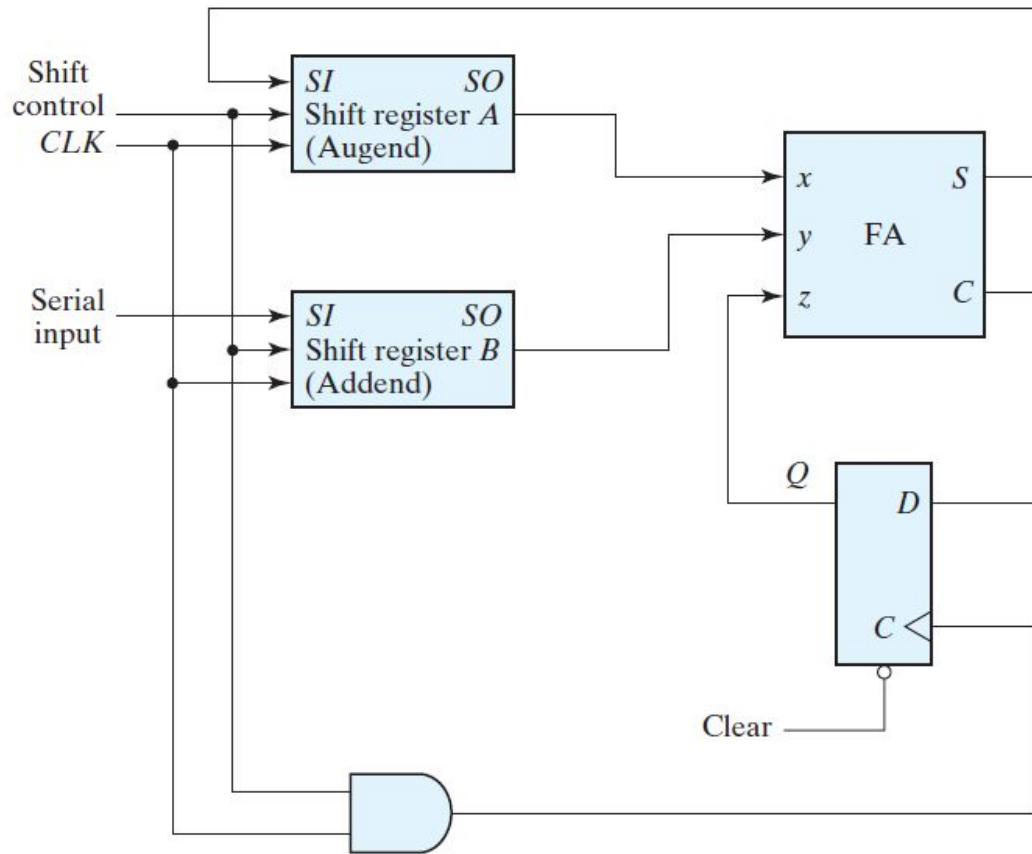


Shift Registers

- A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a shift register.
- The logical configuration consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop.
- And all the FFs are synchronized using common clock pulses.

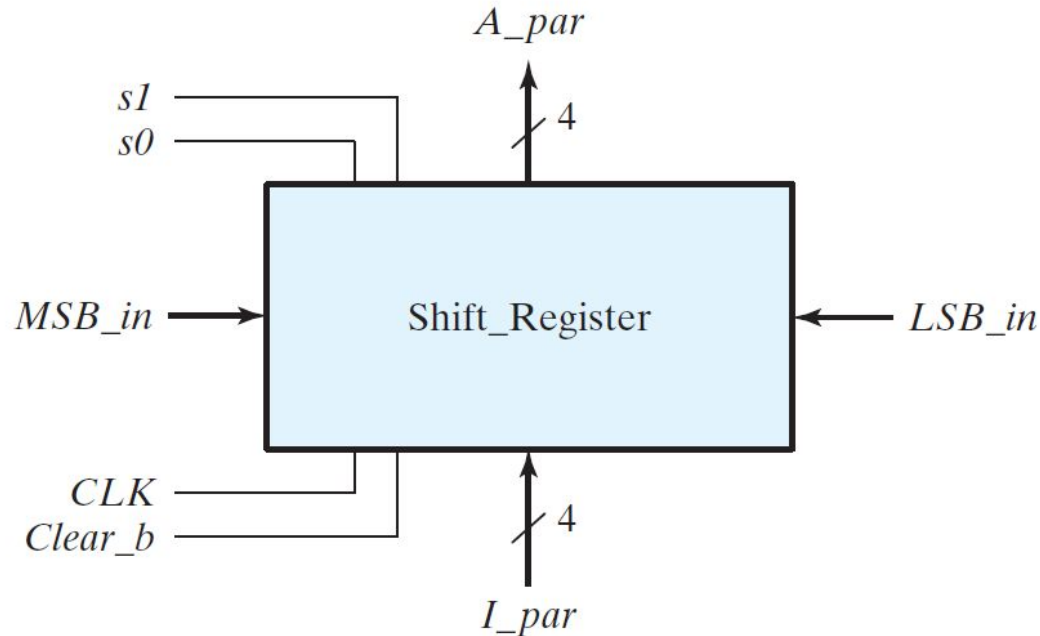
Four bit Shift Register





SERIAL ADDER

Universal Shift Register



Mode Control		Register Operation
s_1	s_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load



COUNTER

- A register that goes through a prescribed sequence of states upon the application of input pulses is called a counter.
- The gates in the counter are connected in such a way as to produce the prescribed sequence of states.
- A counter that follows the binary number sequence is called a binary counter.

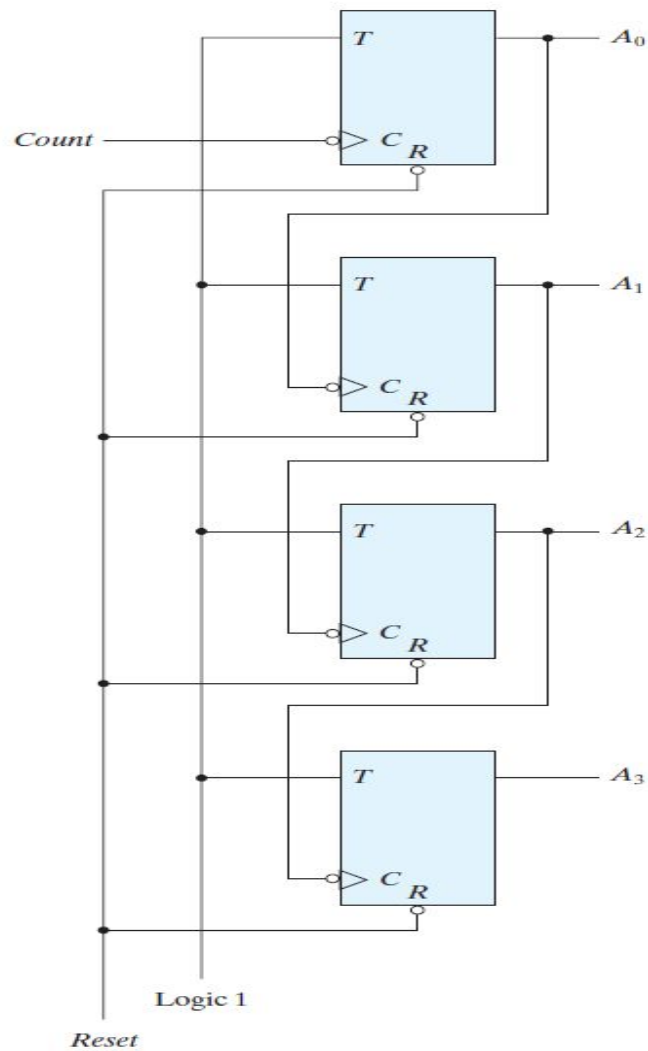
RIPPLE COUNTER



- In a ripple counter, a flip-flop output transition serves as a source for triggering other flip-flops.
- Eg : Binary Ripple Counter, BCD Ripple Counter.

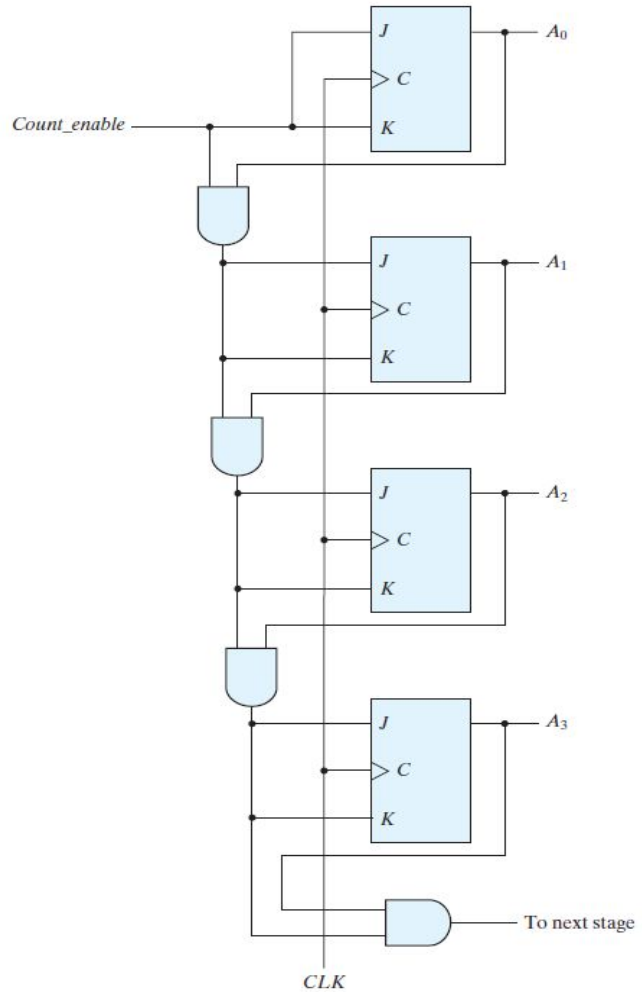
SYNC. COUNTER

- In a synchronous counter, the C inputs of all flip-flops receive the common clock.
- The decision whether a flip-flop is to be complemented is determined from the values of the data inputs.
- Eg : Synchronous Binary Counter



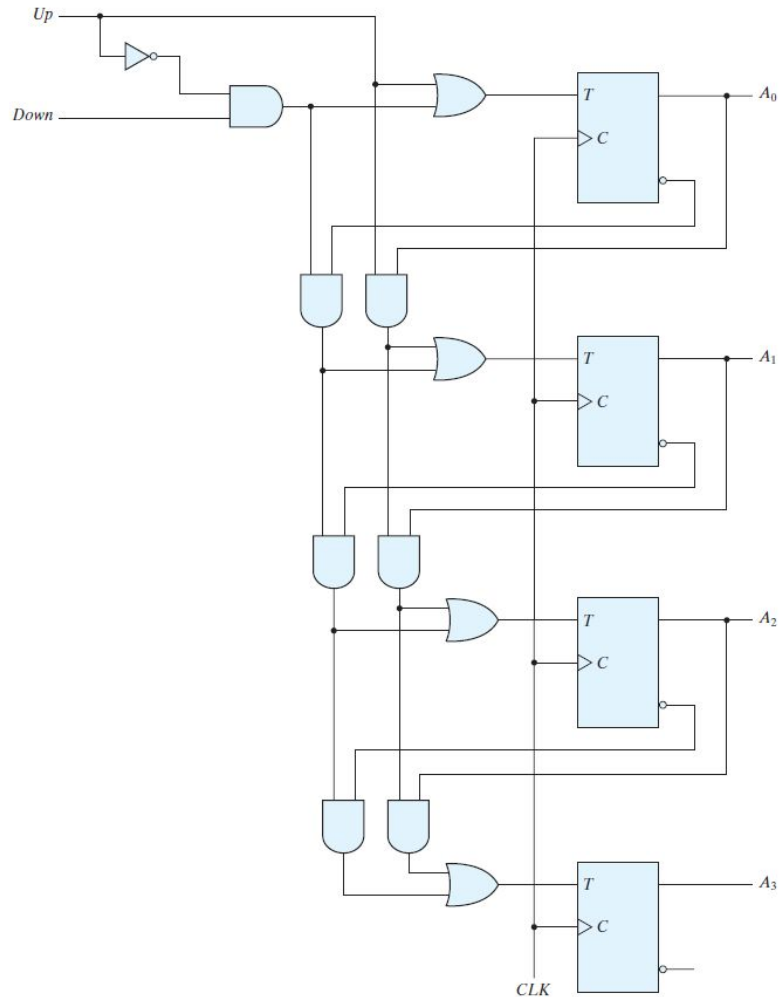
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

FOUR BIT
BINARY
RIPPLE
COUNTER



A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

FOUR BIT
BINARY
SYNC.
COUNTER




A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

UP DOWN
BINARY
SYNC.
COUNTER



STATE REDUCTION AND HAZARDS

- 
- A state table with fewer rows are easier to realize.
 - Equivalent States : Those which produce same outputs for a given input and go to equivalent next states.
 - State table can be reduced by removing equivalent states.
 - Two ways :
 1. Row Reduction
 2. Implication Table



IMPLICATION TABLE