Introduction to Computer Systems

LECTURE 6: COMBINATIONAL AND SEQUENTIAL LOGIC CIRCUITS

Objectives

After completing this lesion you will be able to:

- Understand different types of Digital Logic Circuits.
- Understand the characteristics of the DLC.
- Understand the usage of each type.

Digital Logic Circuits

Digital Logic Circuits

Logic circuits can be categorized as

- Combinational Circuits
- Sequential circuits.

A combinational circuit consists of input variables, logic gates and output variables.

- Output depends only on current input
- Has no memory

Combinational Circuits

A combinational circuit consists of input variables, logic gates, and output variables.

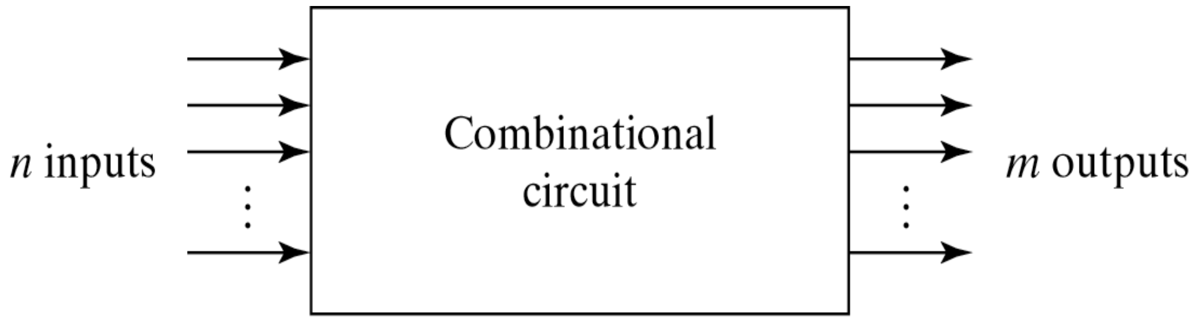
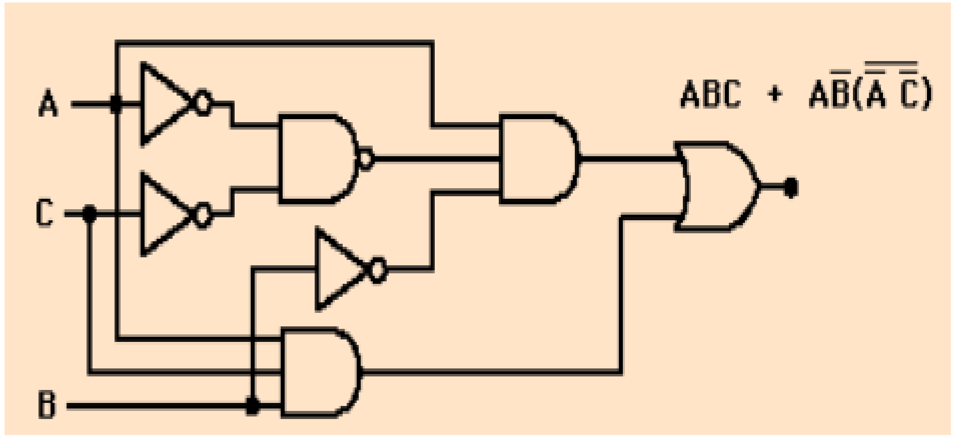
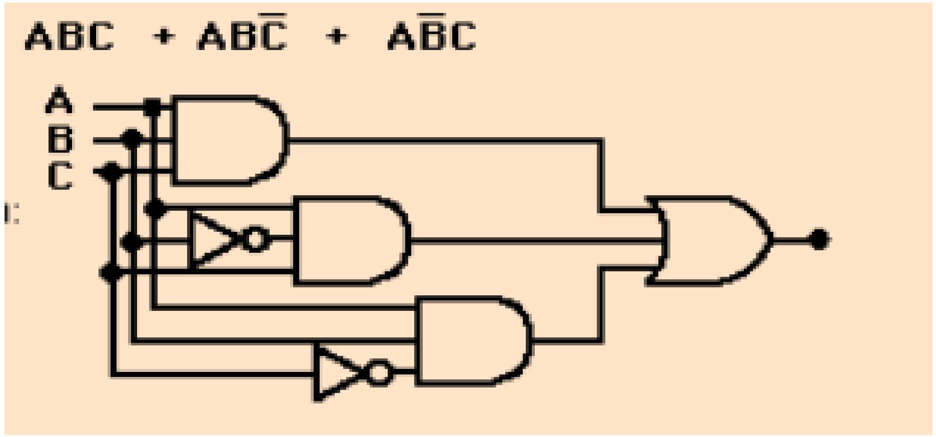


Fig. 4-1 Block Diagram of Combinational Circuit

Combinational Circuits (Examples)


$ABC + A\overline{B}(\overline{\overline{A}\,\overline{C}})$
A
C
B

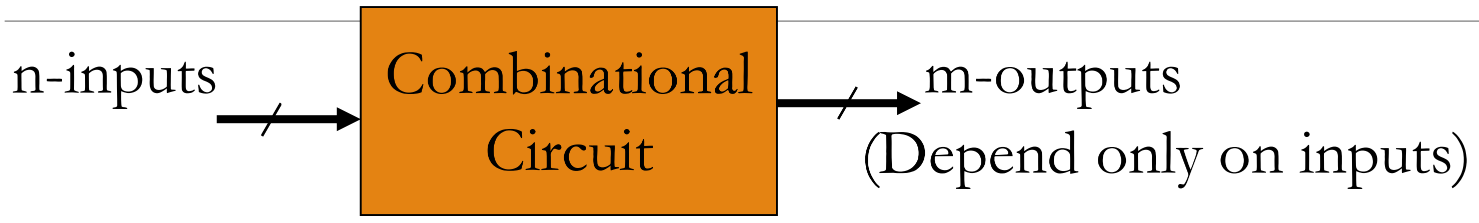


Sequential Logic

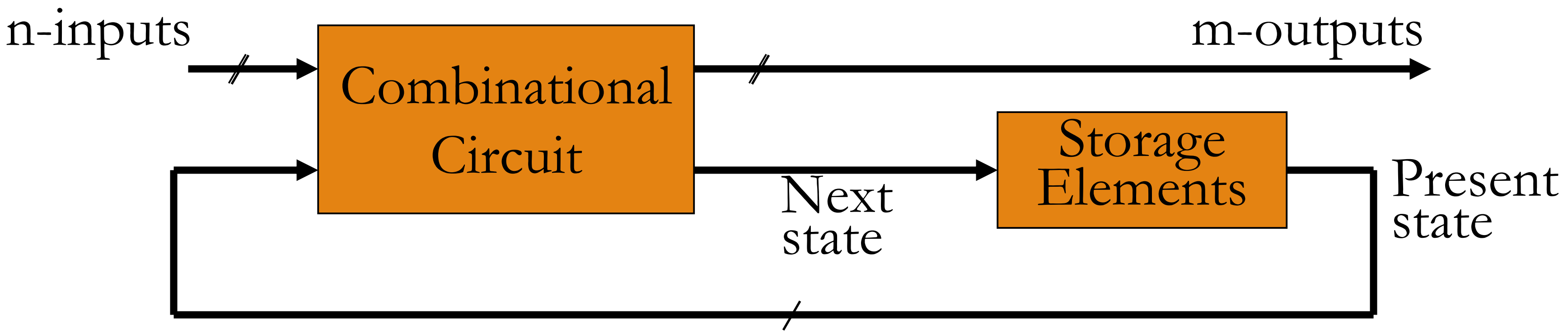
Sequential Logic:

- Output depends not only on current input but also on past input values, e.g., design a counter
- Need some type of memory to remember the past input values

Combinational vs. Sequential Circuits



Combinational Circuit

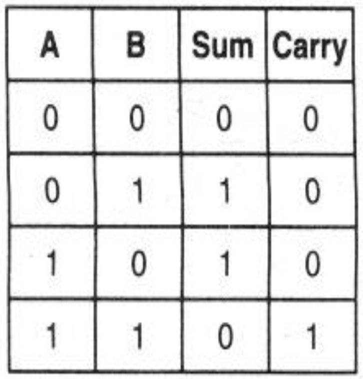


Sequential Circuit

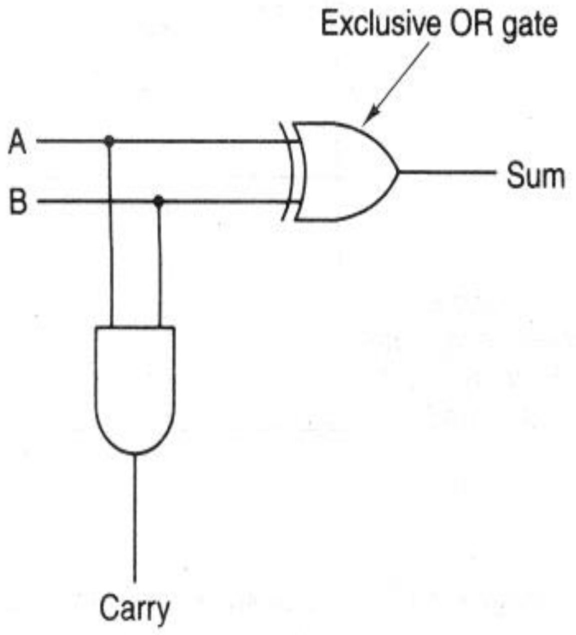
Combinational Circuits

Combinational Circuits
Half Adder

Half Adder : The sum is **XOR** operation and the carry an **AND**



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Sum= A’B+AB’

$A \oplus B$

Carry=AB

Figure 4-23 (a) Truth table for 1 bit addition. (b) A circuit for a half adder

Combinational Circuits – Full Adder

5. Draw a diagram

Input			Output	
x	y	z	C	S
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

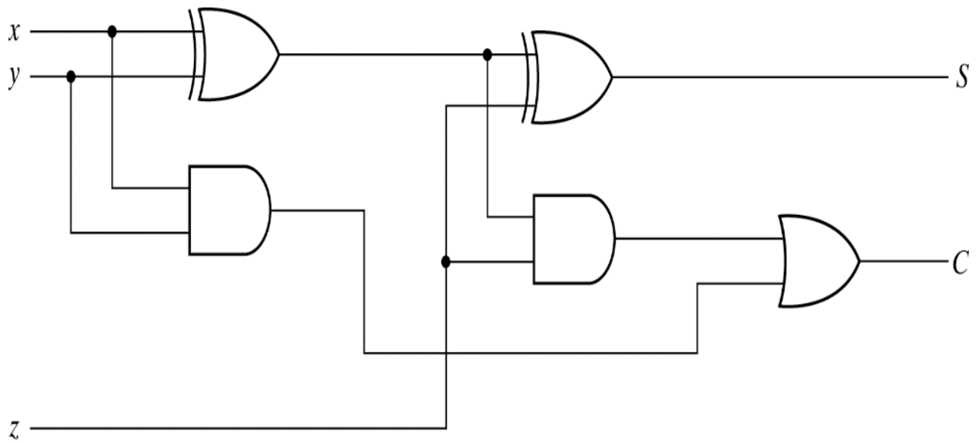


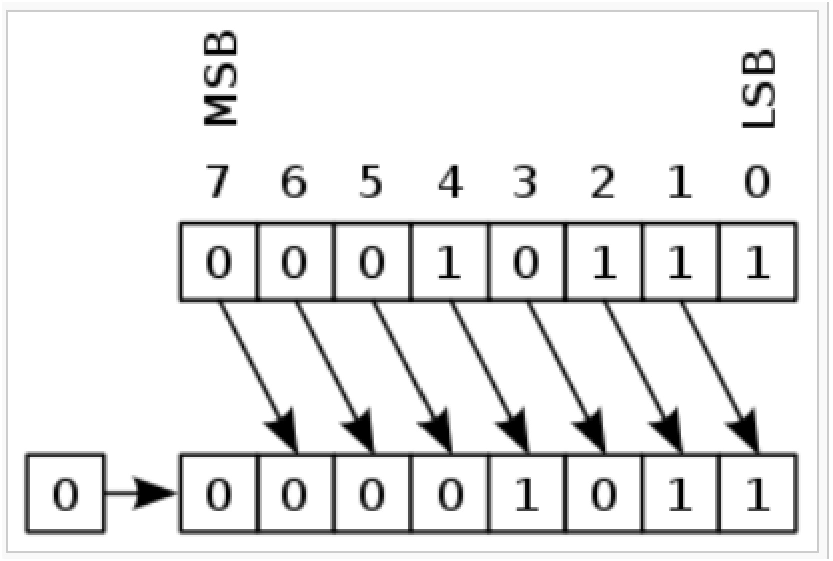
Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

Binary Multiplication

- E.g: 15 * 5 = 75, 15=1111,5=$\underset{A}{1}\overset{B}{0}\underset{C}{1}$<-multiplier

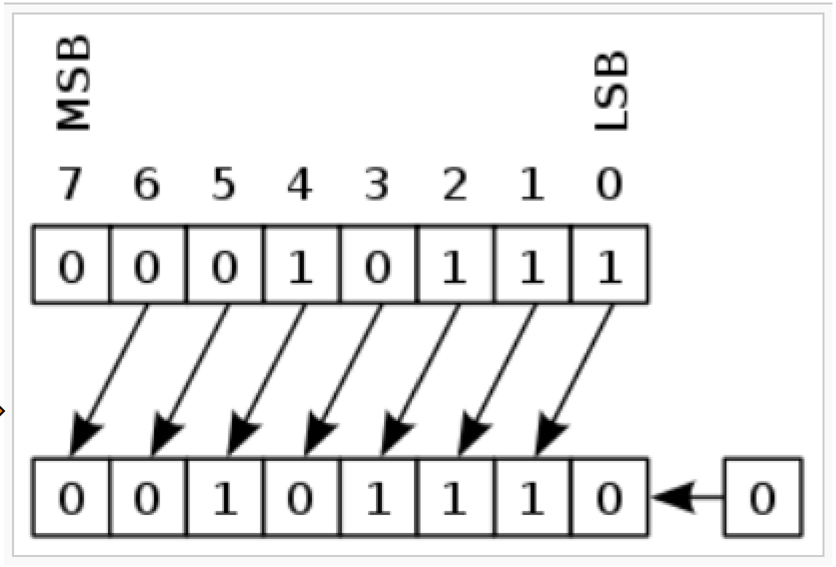
0000 0000	initialize to 0
0000 1111	A =1
0000 1111	sum
0001 1110	shift ← (left)
0000 0000	B = 0
0001 1110	sum
0011 1100	shift ← (left)
0000 1111	C = 1
0100 1011	sum, no shift

Shifter



Right shift 1-bit

Left shift 1-bit



Shifter

Figure shows an eight-input, eight-output, 1-bit left/right shifter.

- An eight-bit input is applied onto D_0, D1…D7, and the shifted output is taken out of S_0, S1…S7.
- The shift direction is controlled with the control signal C.

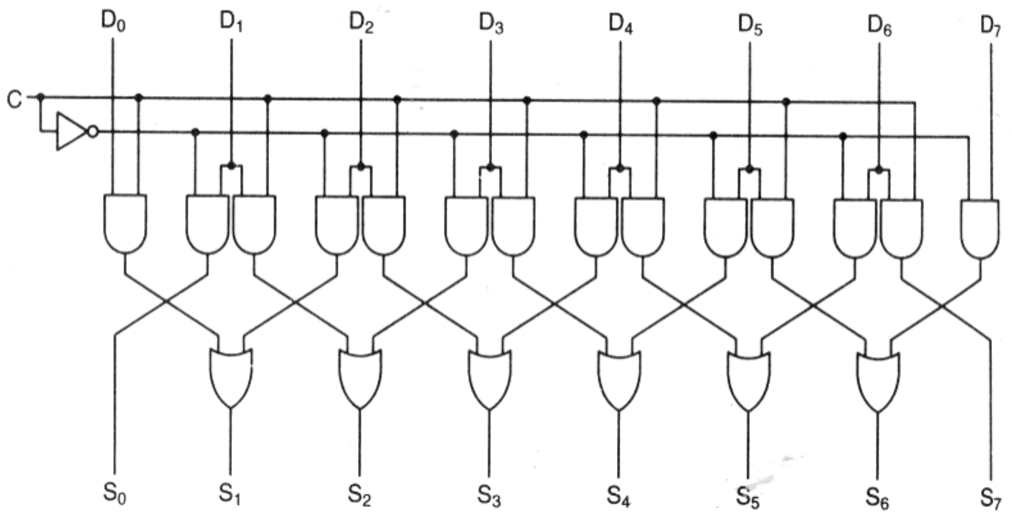


Figure 4-22. A 1-bit left/right shifter.

- A logic one on C enables the right hand side AND gates and pass each input bit to the OR gate to its right and hence, a one-bit right shift is performed.
- Similarly, a logic zero on C performs a left shift.

Comparator

A comparator circuit compares two input words.

The simplest comparator circuit is an exclusive NOR gate, which compares two individual bits:

- produces a zero if the two bits are unequal
- produces a logic one if they are equal

Comparator

When two words of length *n* are to be compared,

- *n* XOR gates and
- a NOR gate can be combined

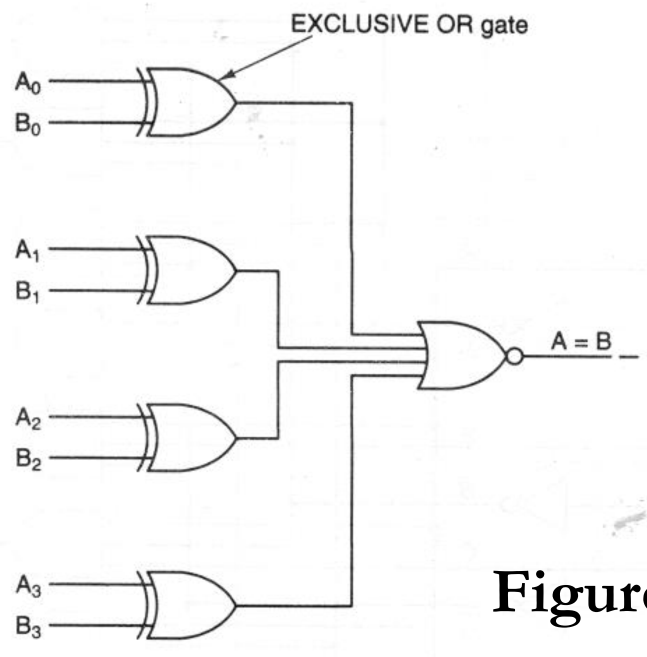


Figure: A simple 4-bit comparator

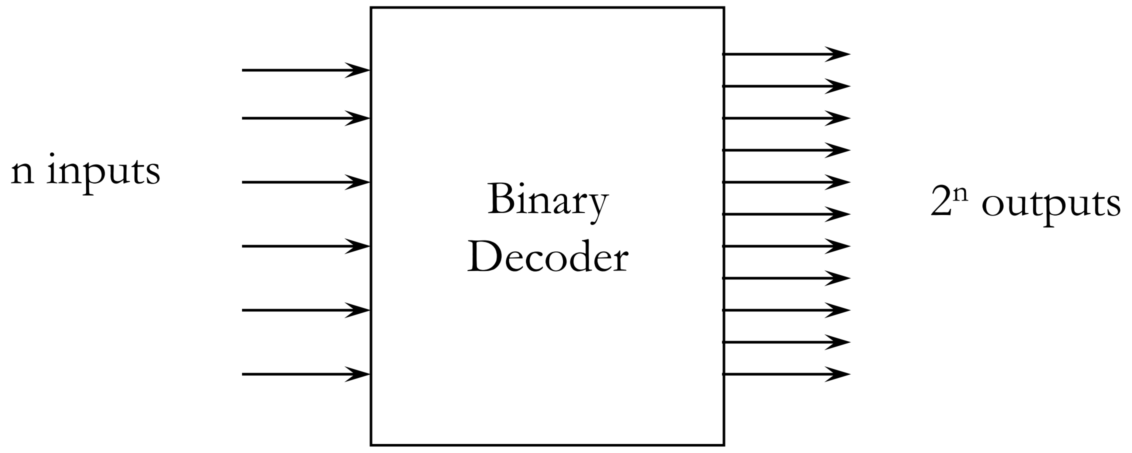
Decoder/Encoder

Decoder

- A combinational circuit that converts binary information from the *n* coded inputs to a maximum of 2^n unique outputs
- ***n***-to-***m*** line decoder = ***n x m*** decoder Ex: Octal to Binary line decoder
  - ***n*** inputs, ***m*** outputs (3 inputs, 8 outputs)
- If the n-bit coded information has unused bit combinations, the decoder may have less than 2^n outputs
  - $m \leq 2^n$

Binary Decoder

- Black box with n input lines and 2^n output lines

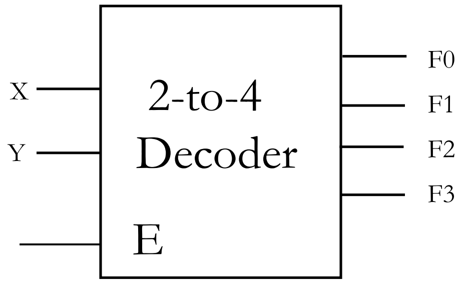


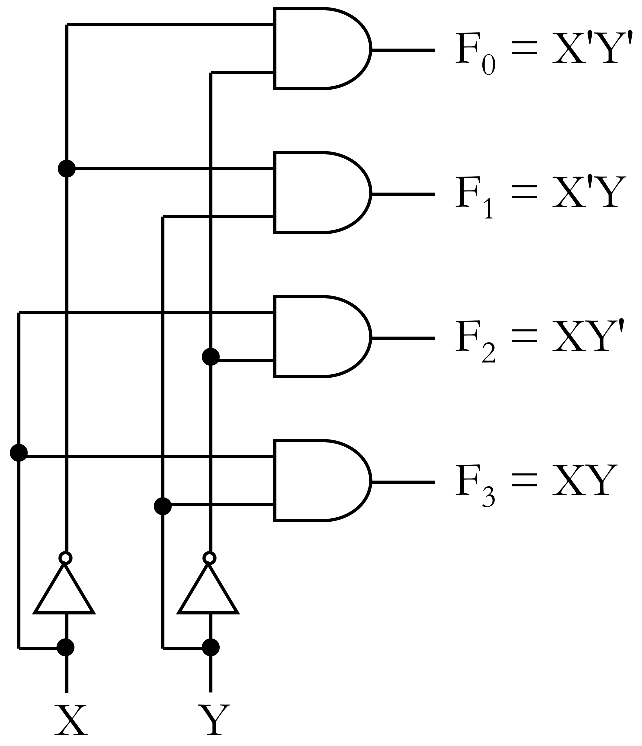
2-to-4 Binary Decoders

Truth Table:

X	**Y**	$\mathbf{F_0}$	$\mathbf{F_1}$	$\mathbf{F_2}$	$\mathbf{F_3}$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

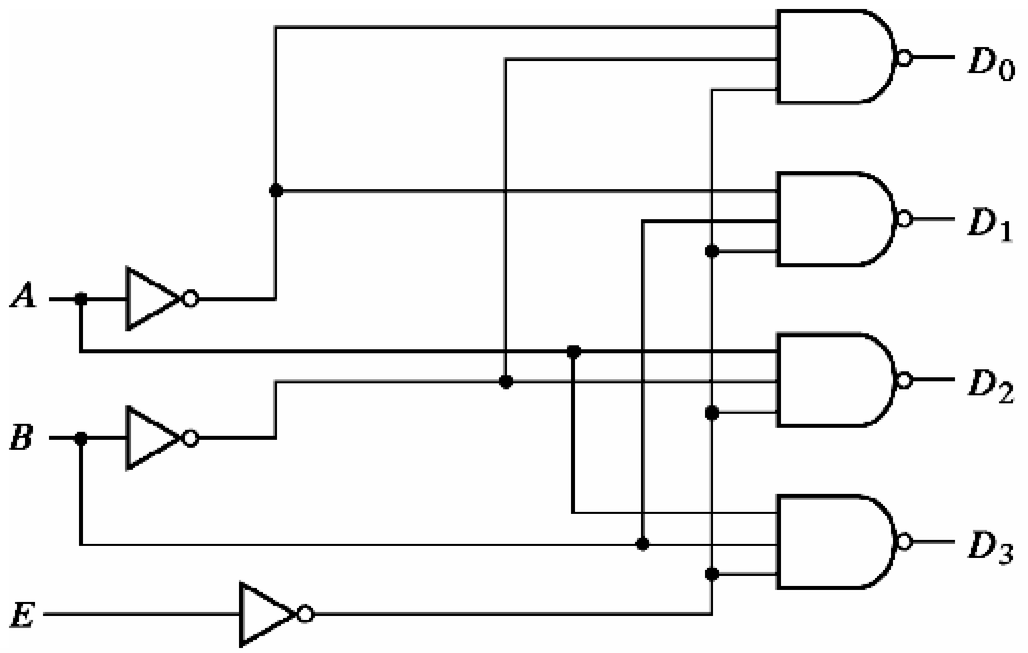
- From truth table, circuit for 2x4 decoder is:
- Note: Each output is a 2-variable minterm (X'Y', X'Y, XY' or XY)





2-to-4 Decoders : NAND Implementation

Decoder is enabled when E=0 and an output is active if it is 0



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

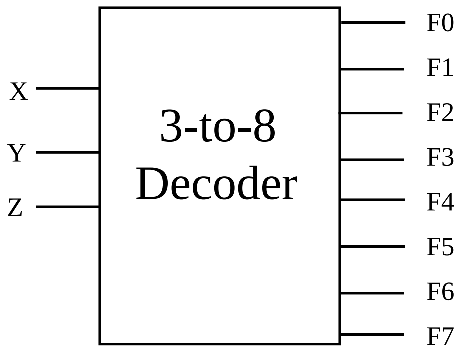
(b) Truth table

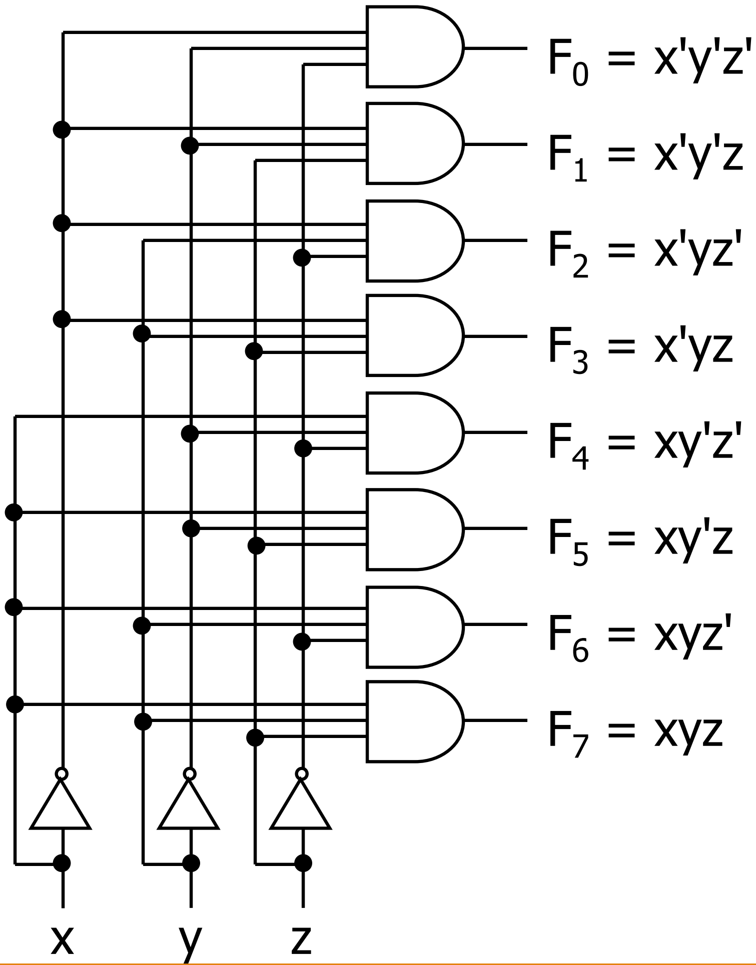
2-to-4-Line Decoder with Enable Input

3-to-8 Binary Decoder

Truth Table:

x	y	z	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_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





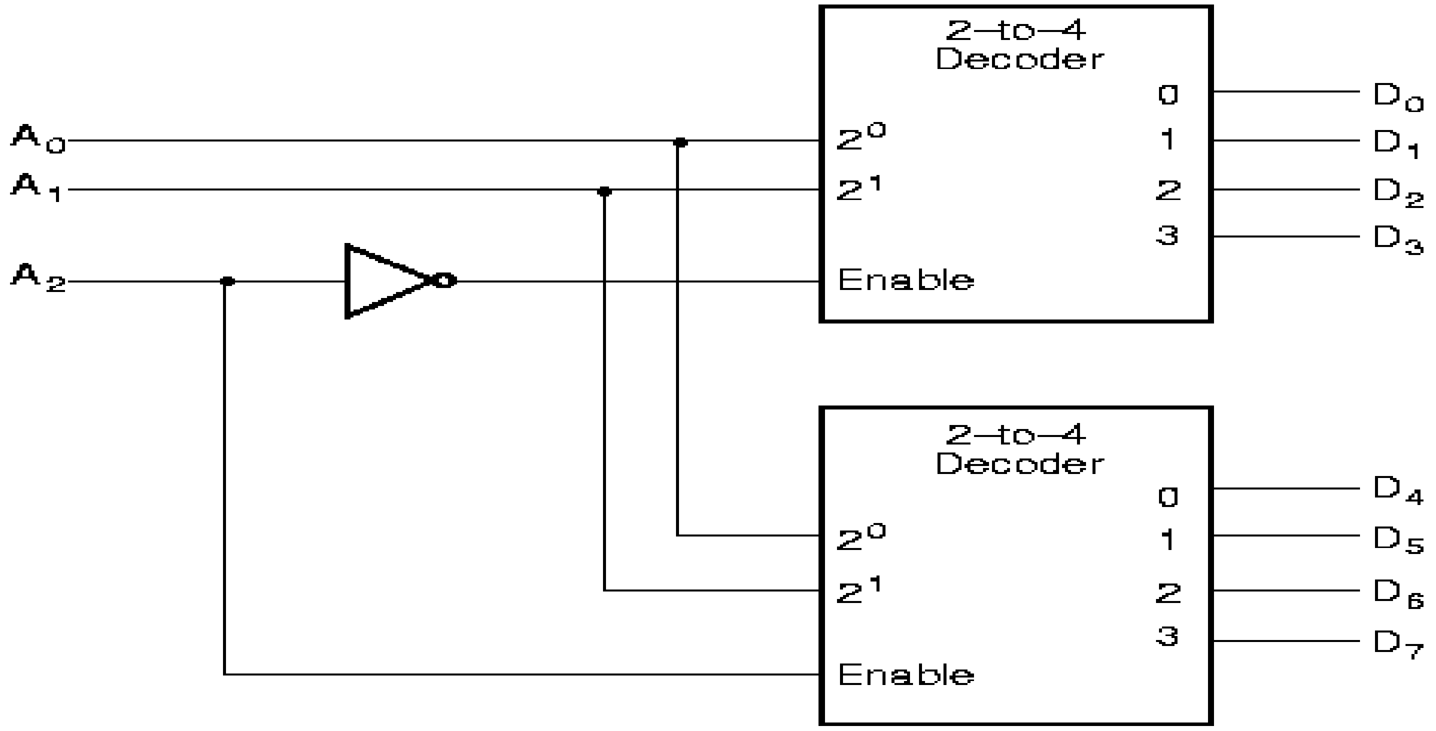
Decoder Expansion

Decoder expansion

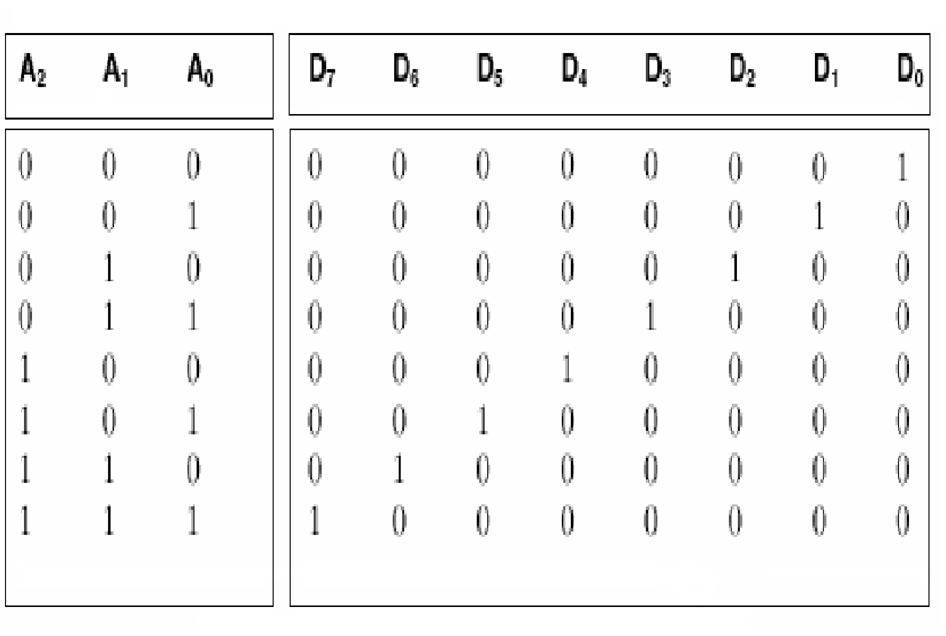
- Combine two or more small decoders with enable inputs to form a larger decoder

- 3-to-8-line decoder constructed from two 2-to-4-line decoders
  - **The MSB is connected to the enable inputs**
  - **if A_2=0, upper is enabled; if A_2=1, lower is enabled.**

Decoder Expansion(Contd.)

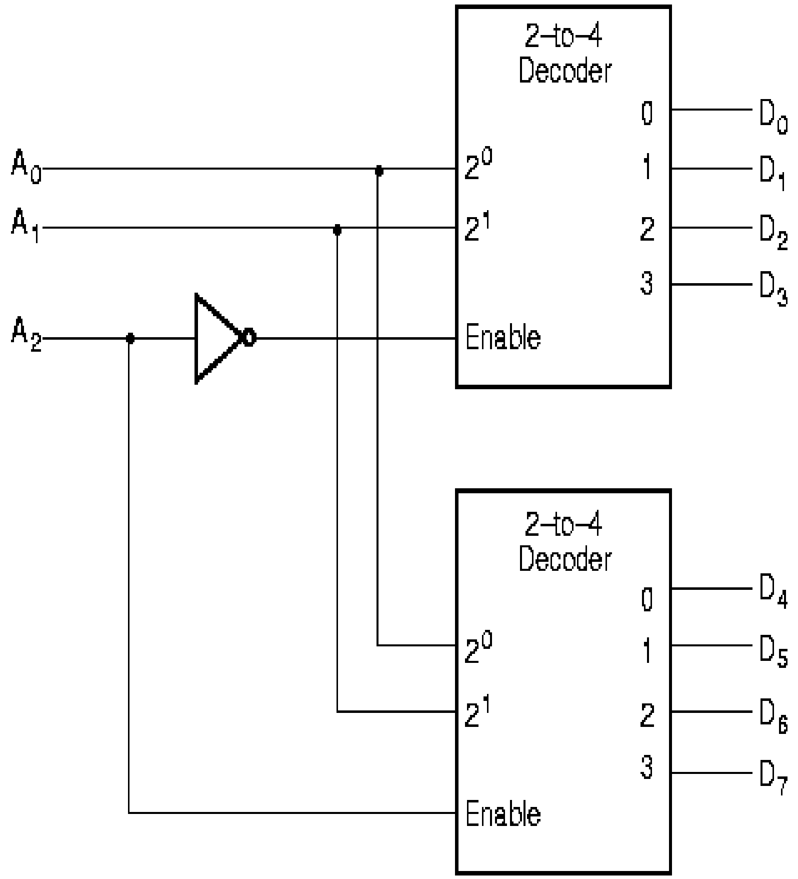


Combining two 2-4 decoders to form one 3-8 decoder using enable switch



A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

The highest bit is used for the enables



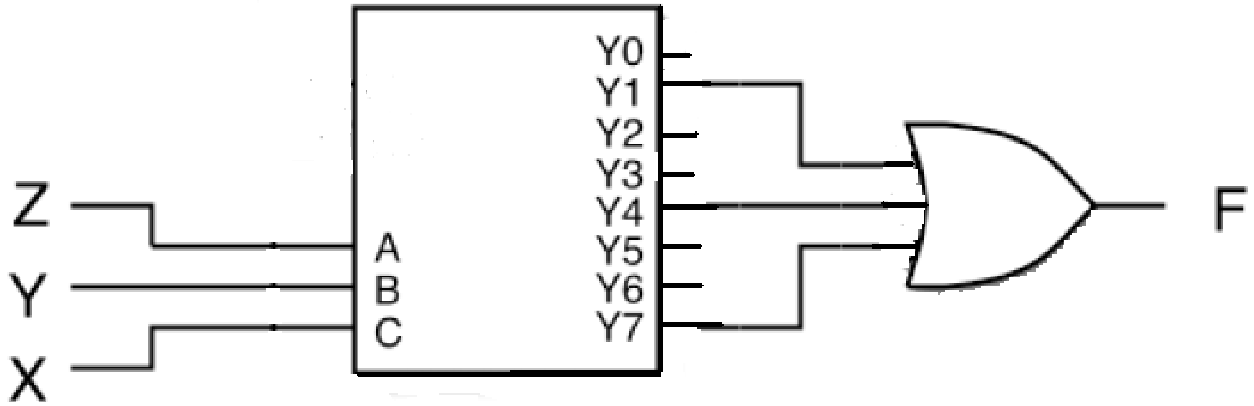
Combinational Circuit Design with Decoders

Combinational circuit implementation with decoders

- A decoder provide 2^n minterms of n input variables
- Since any Boolean function can be expressed as a sum of minterms, one can use a decoder and external OR gates to implement any combinational function.

Combinational Circuit Design with Decoders

Example Realize F (X,Y,Z) = Σ (1, 4, 7) with a decoder:



Encoder

Inverse Operation of a decoder

2^n input, n output

Truth Table

3 OR Gates Implementation

- A0 = D1 + D3 + D5 + D7
- A1 = D2 + D3 + D6 + D7
- A2 = D4 + D5 + D6 + D7

Inputs								Outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Truth Table for Encoder

Multiplexer (MUX)

- A combinational circuit that receives binary information from one of 2^n input data lines and directs it to a single output line
- A 2^n -to 1 multiplexer has *2^n input data lines* and *n input selection lines*
- 4-to-1 multiplexer Diagram
- 4-to-1 multiplexer Function Table

Function Table for 4-to-1 line Multiplexer

Select		Output
S1	S0	Y
0	**0**	I_0
0	**1**	I_1
1	**0**	I_2
1	**1**	I_3

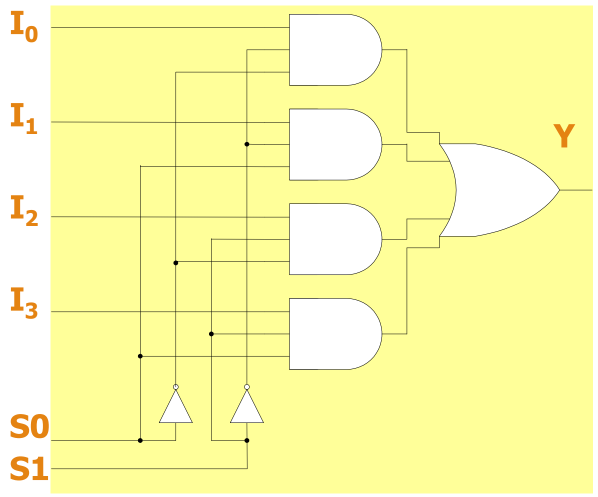
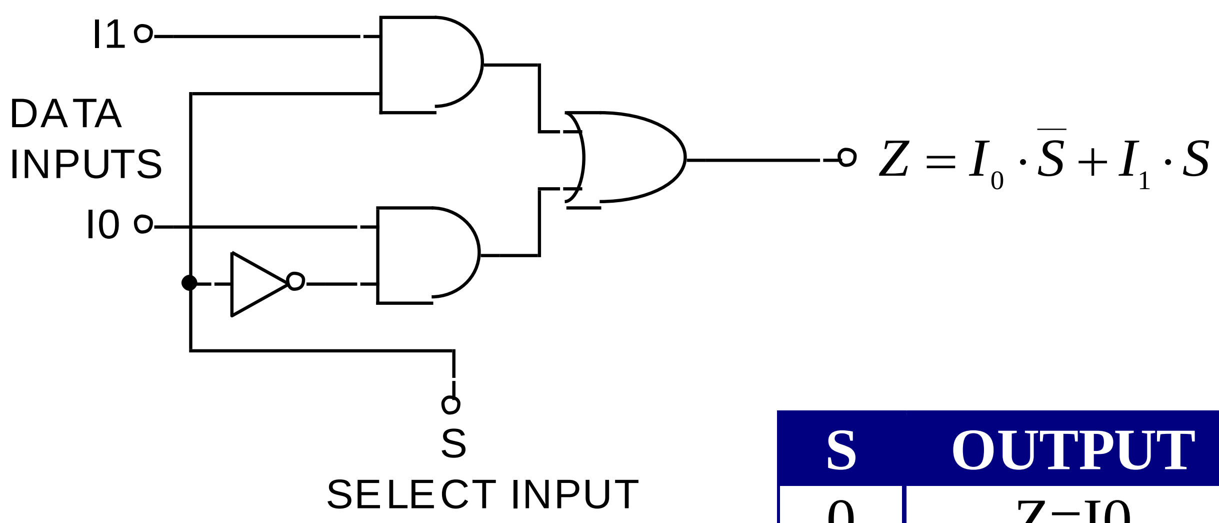


Fig. 4-to-1 Line Multiplexer

Usage: Data routing, Parallel-to-serial conversion, Operation sequencing, Implement logic function of a truth table

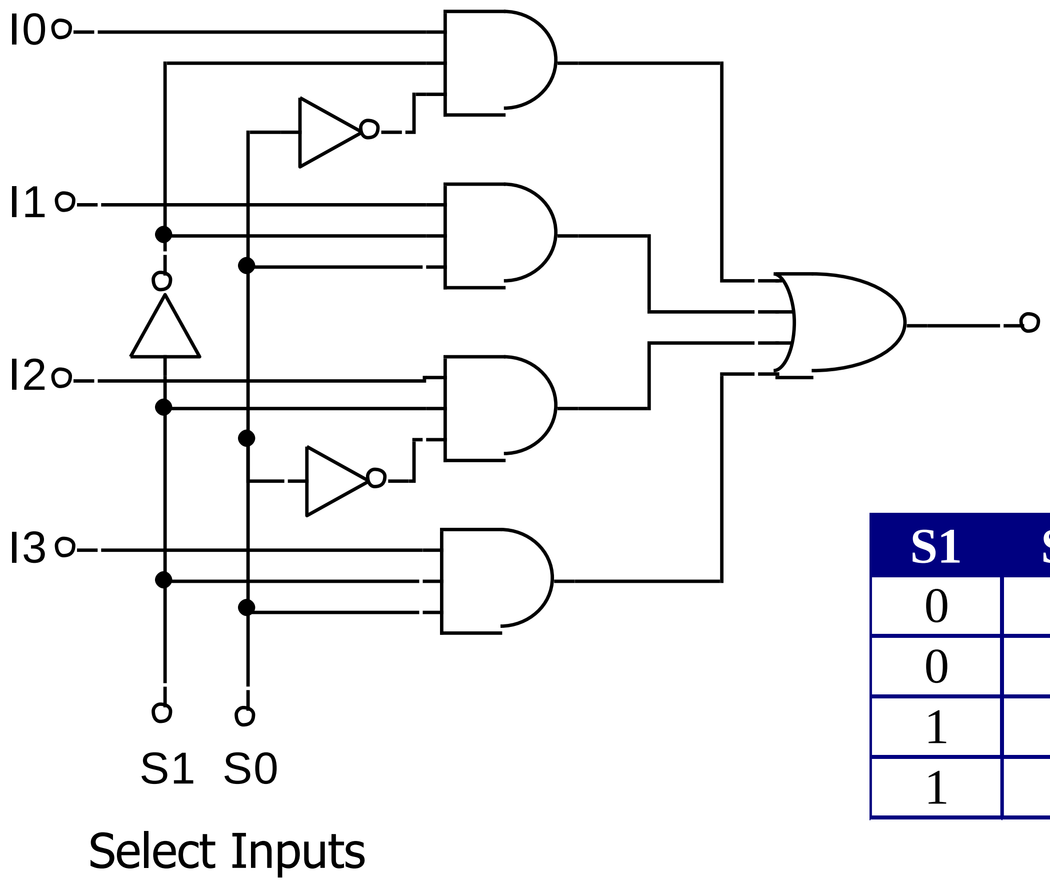
Basic 2-Input Multiplexer



$$Z = I_0 \cdot \overline{S} + I_1 \cdot S$$

S	OUTPUT
0	Z=I0
1	Z=I1

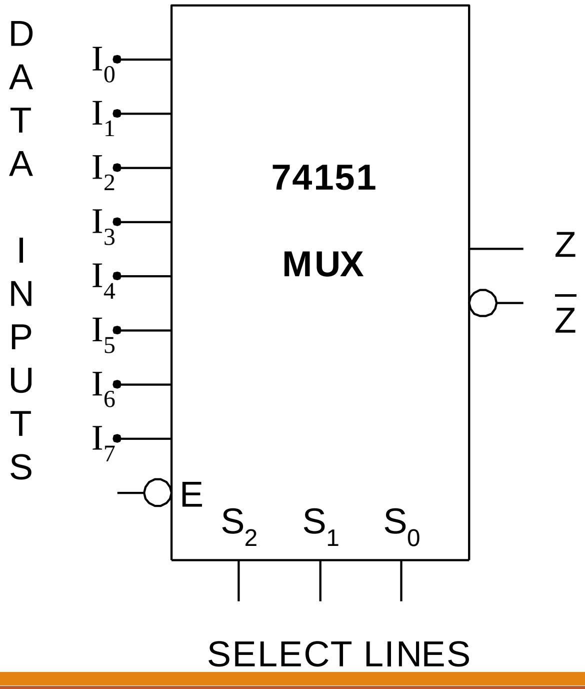
4-Input Multiplexer



S1	S0	OUTPUT
0	0	Z=I0
0	1	Z=I1
1	0	Z=I2
1	1	Z=I3

Multiplexer Logic Diagram

- ❖ Takes one of many inputs and funnels it to an output Z.
- ❖ Take the selector lines convert to a decimal number and this is the input funneled to the output.
- ❖ Strobe is active low enable



S2	S1	S0	E	Z
0	0	0	0	I0
0	0	1	0	I1
0	1	0	0	I2
0	1	1	0	I3
1	0	0	0	I4
1	0	1	0	I5
1	1	0	0	I6
1	1	1	0	I7

Multiplexer

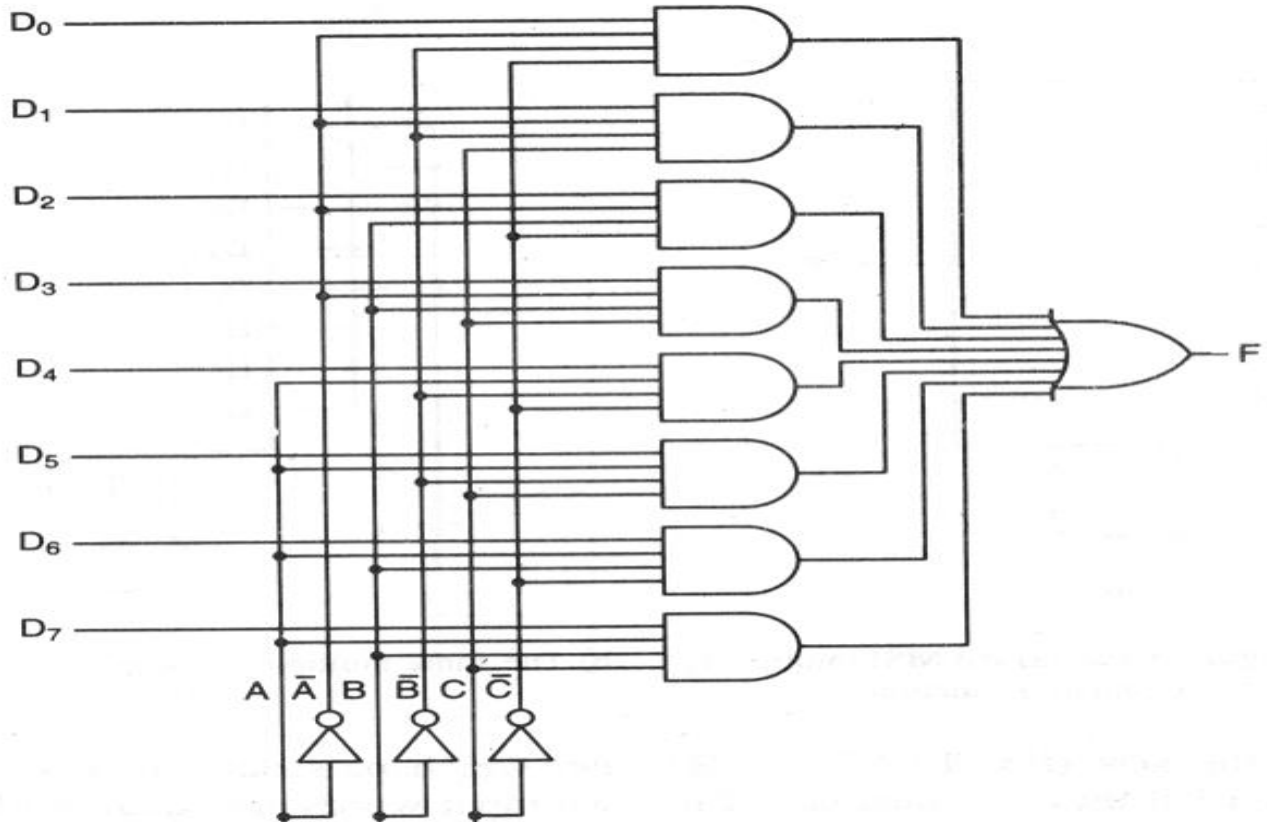


Figure :A 8-line to 1-line multiplexer

Programmable Logic Arrays (PLA)

Programmable logic arrays (PLA) are very general chips that provide a quick and easy way to implement Boolean functions expressed in the sum-of-products form.

- A PLA comprises of an array of AND gates and another array of OR gates.
  - Each one of the AND gate outputs is tapped through tiny fuses to the inputs of each OR gate.
  - Similarly, the input lines of the PLA chip are tapped to the inputs of each AND gate.
- Applying a high voltage to the chip a designer can blow selected fuses leaving only the desired circuit connections intact.
- Therefore, by choosing the proper set of fuses to blow, a designer can implement or “program” the desired Boolean functions rapidly.

Programmable Logic Arrays (PLA)

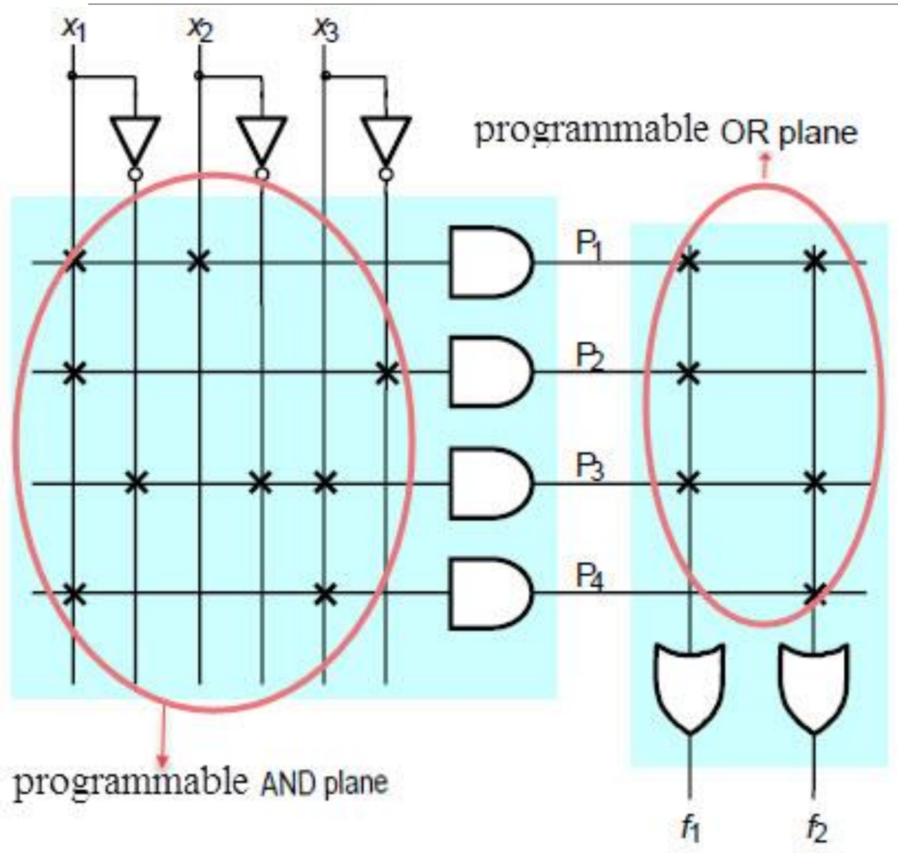


Figure: Customary schematic for the PLA

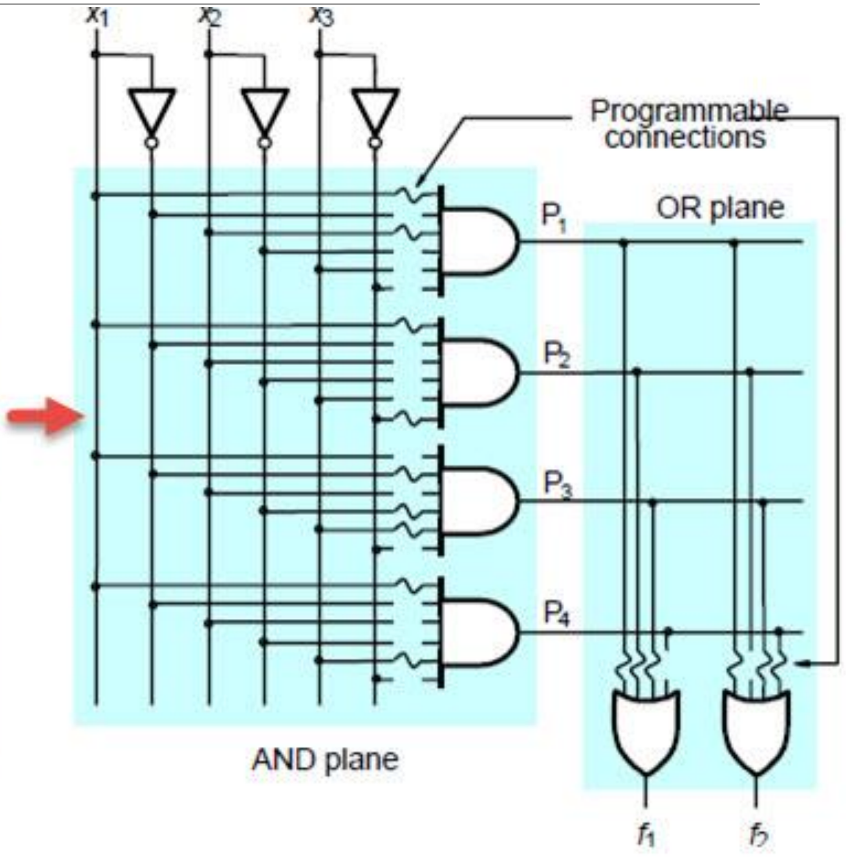


Figure: Gate-level diagram of a PLA

Arithmetic Logic Unit (ALU)

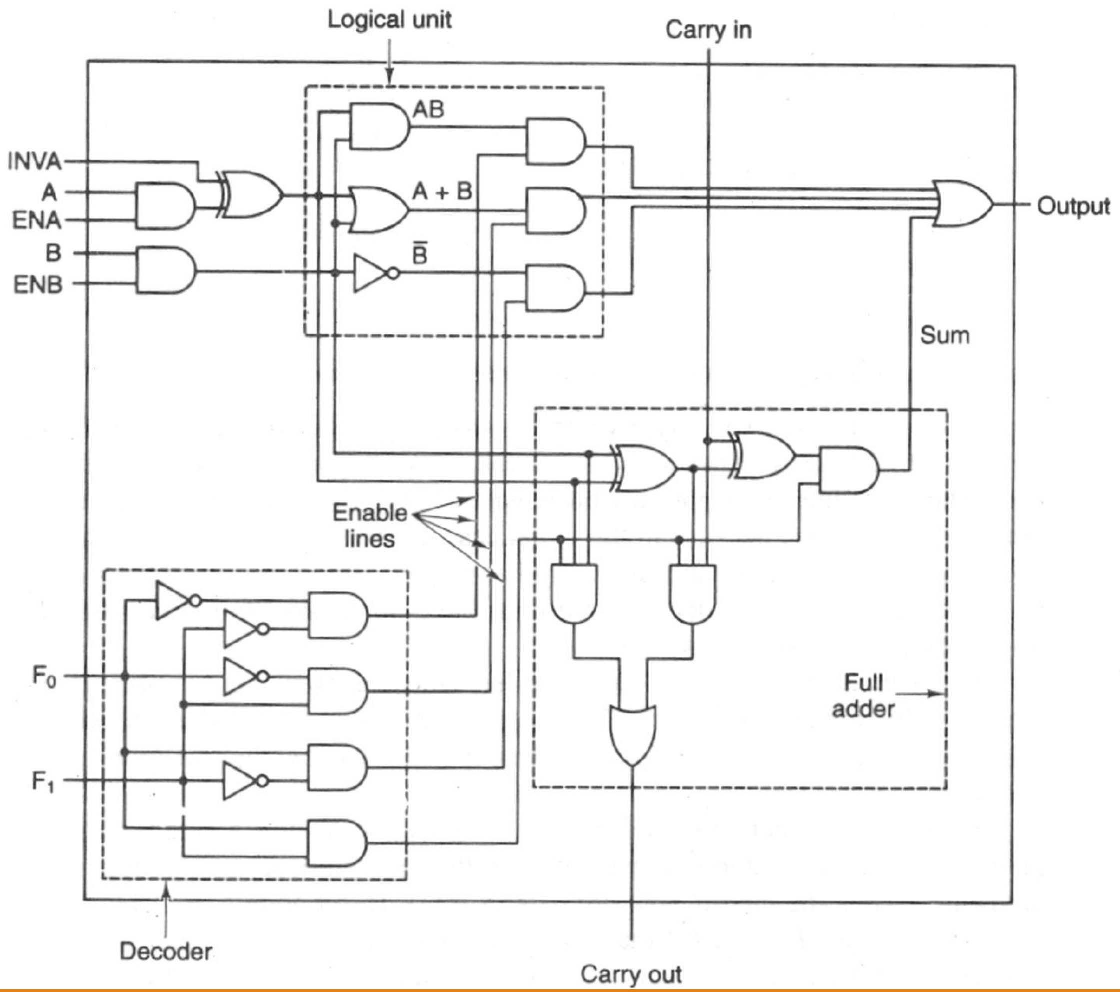
At the heart of any computer CPU is the arithmetic logic unit or the ALU.

It is a circuit that performs AND, OR, and sum operations of two input words.

AND and OR logical operations are easily implemented using the corresponding logic gates.

The sum operation is achieved using the full adder circuit described earlier.

Arithmetic Logic Unit (ALU)



Arithmetic Logic Unit (ALU)

If the CPU uses *n*-bit machine words, then *n* such ALU blocks must be connected together.

While the AND and OR operations only need the *n* ALUs to be placed in the *n* bit positions, the SUM operation requires the Carry-in and Carry-out signals of adjacent units cascaded

Arithmetic Logic Unit (ALU)

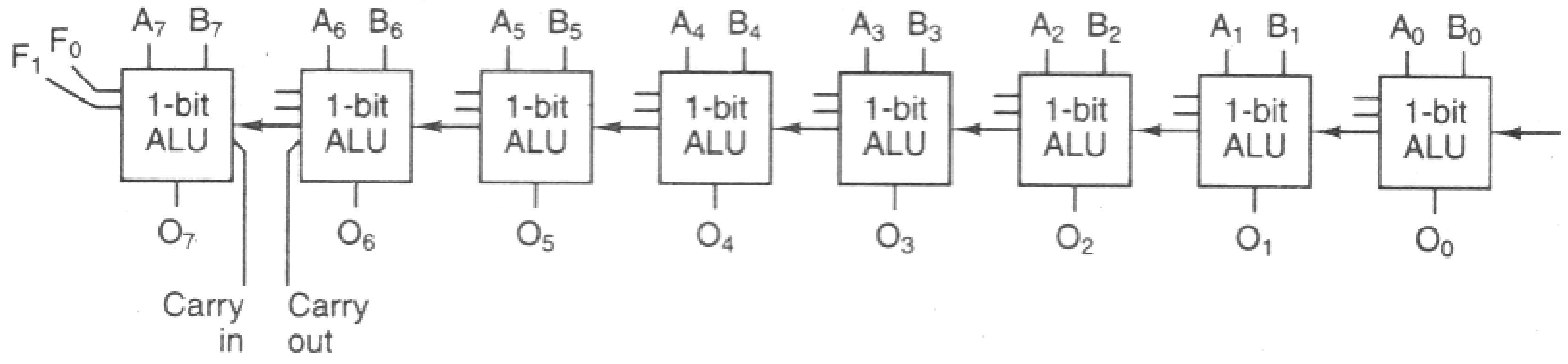


Figure 4-27. Cascading eight 1-bit ALUs to form an 8-bit ALU

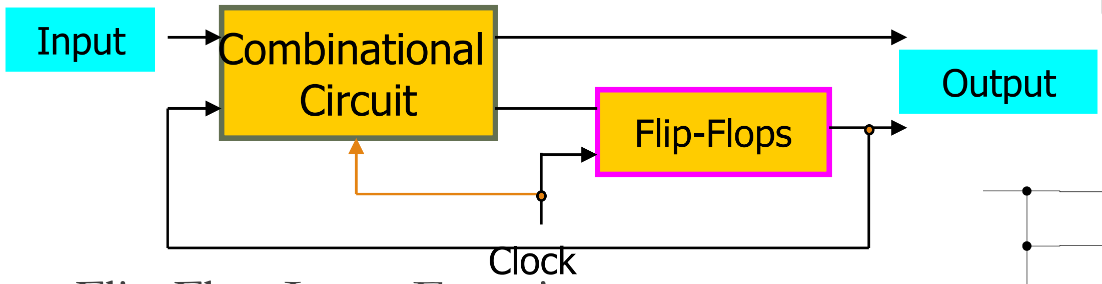
Sequential Circuits

Sequential Circuits

A sequential circuit is an interconnection of F/F and Gate

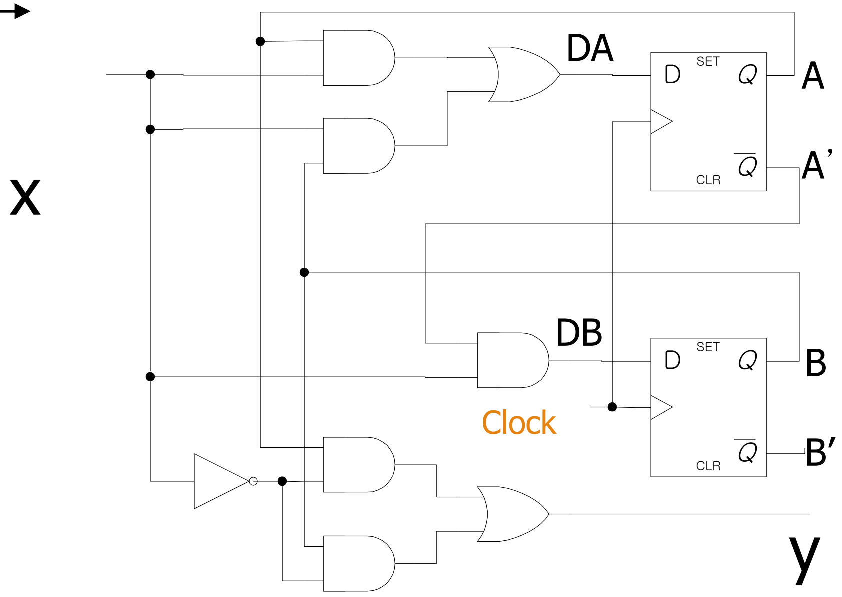
Clocked synchronous sequential circuit

Combinational Circuit = Gate
Sequential Circuit = Gate + F/F



Flip-Flop Input Equation

- Boolean expression for F/F input
- Input Equation
  - $D_A = Ax + Bx$, $D_B = A'x$
- Output Equation
  - $y = Ax' + Bx'$
- *Example of a sequential circuit*



Sequential Circuits

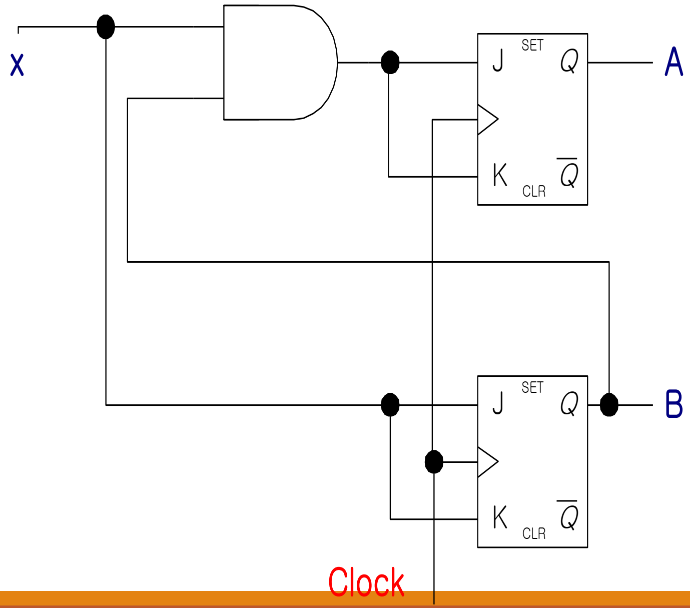
Sequential Circuit Design Procedure

1. The Problem is stated
2. I/O variables are assigned
3. Truth table(I/O relation)
4. Simplified Boolean Functio
5. Logic circuit diagram

Recall Combinational Circuit Design

Sequential Circuit: State diagram, State table

F/F : 2^{m+n} (m - State , n - Input)



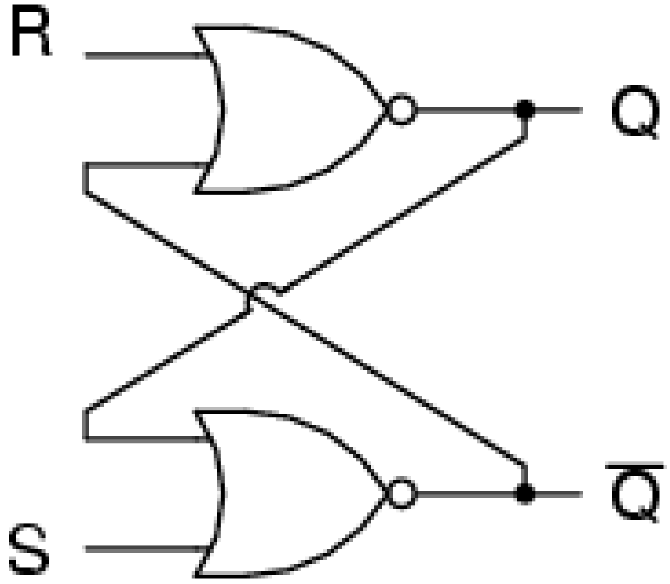
◆ Logic Diagram

Latches

SR Latch

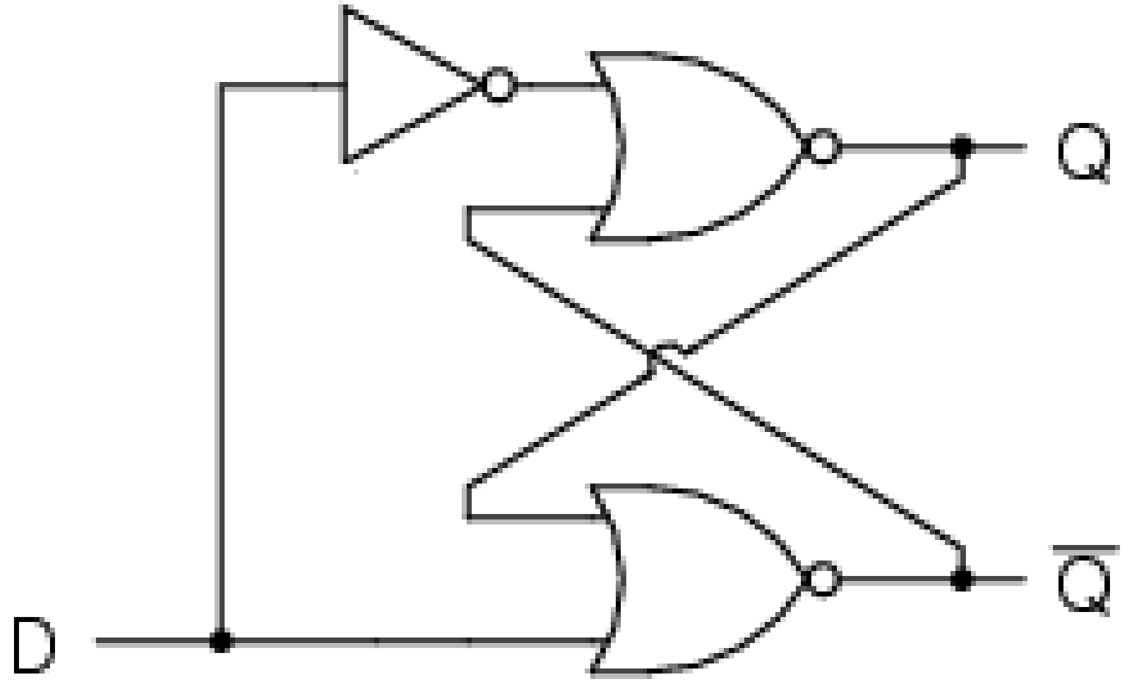
S	R	Action
0	0	Keep state
0	1	Q=0
1	0	Q=1
1	1	Restricted Combination

Characteristic Table



Latches

D Latch



Clock Signals

Clock signals are used to maintain the desired timing in the circuits.

- Clock circuits emit pulse trains of precise repetition interval and width.
- Sometimes it is necessary to have one clock pulse train trail another by a fixed time.
- A circuit with the appropriate delay may be inserted to achieve the desired phase shift

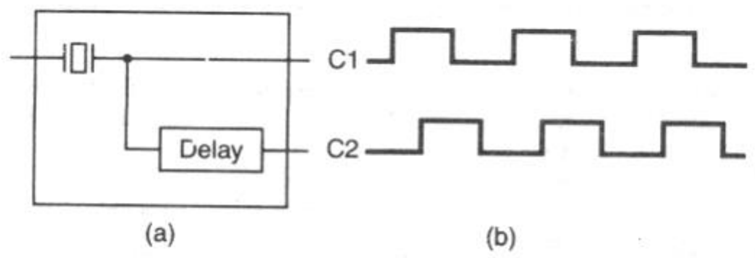
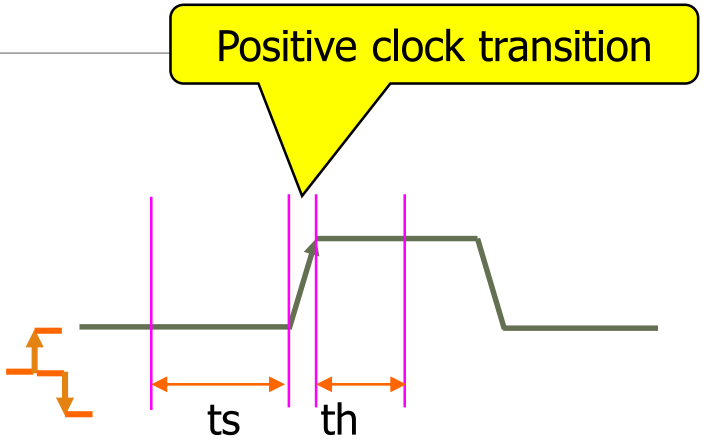


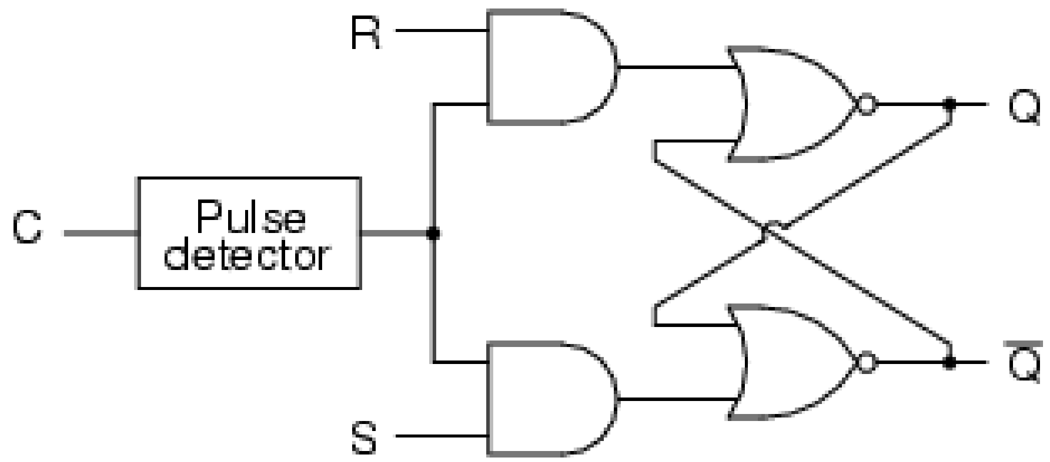
Figure 4-28. Clock circuit and the clock waveforms

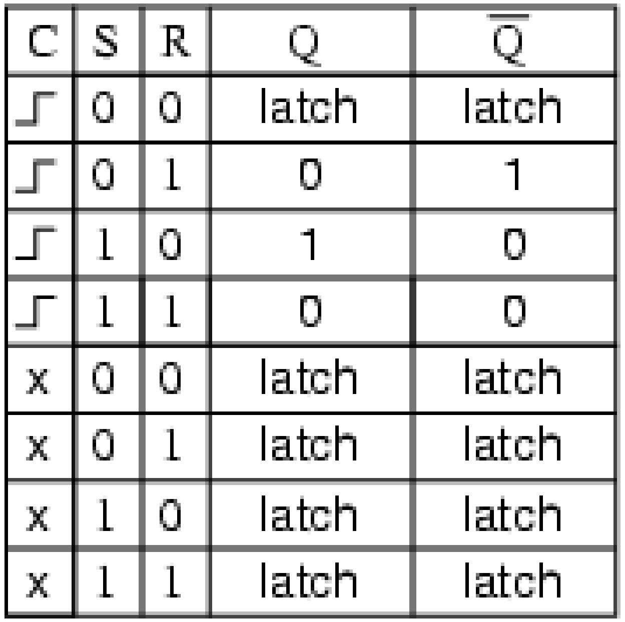
Edge-Triggered Flip Flops



- State Change : *Clock Pulse*
  - Rising Edge(positive-edge transition)
  - Falling Edge(negative-edge transition)
- Setup time(20ns)
  - Minimum time that D input must remain at constant value before the transition.
- Hold time(5ns)
  - Minimum time that D input must not change after the positive transition.
- Propagation delay(max 50ns)
  - time between the clock input and the response in Q

SR Flip-flop

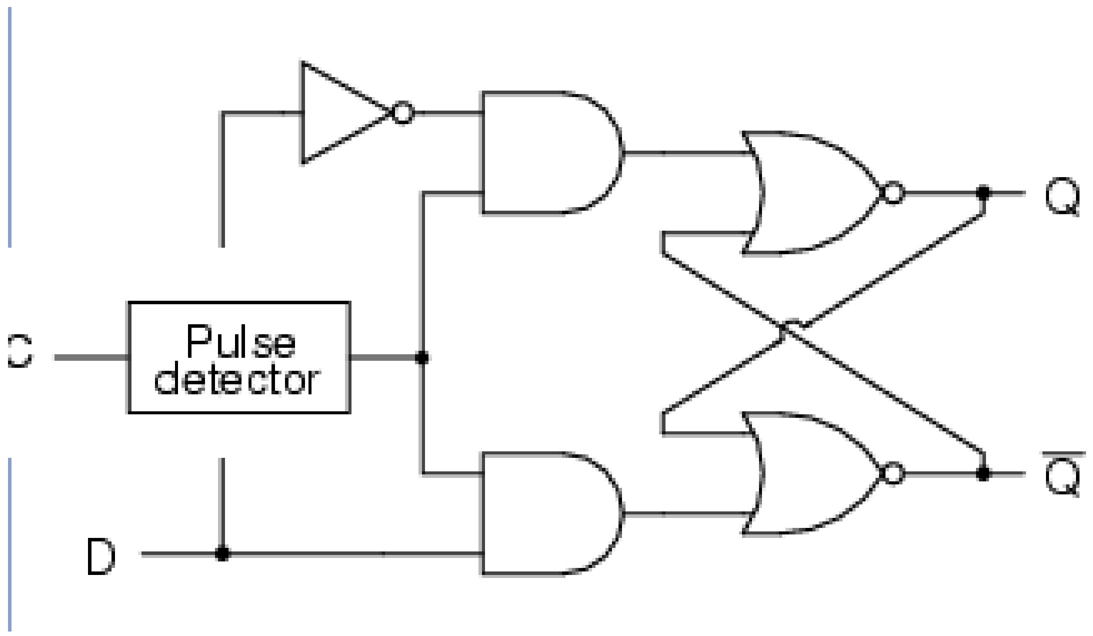


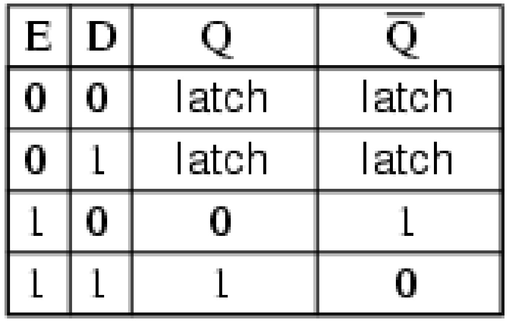


C	S	R	Q	$\overline{Q}$
↑	0	0	latch	latch
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	0	0
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

Characteristic Table

D Flip-flop





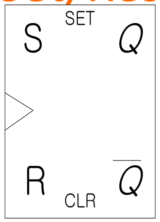
E	D	Q	$\overline{Q}$
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

Flip-flops

The *storage elements* employed in clocked *sequential circuit*

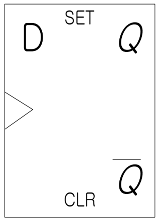
A binary cell capable of storing one bit of information

SR*(Set/Reset)* F/F



S	R	Q(t+1)	
0	0	Q(t)	no change
0	1	0	clear to 0
1	0	1	set to 1
1	1	?	Indeterminate

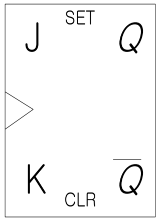
n D*(Data)* F/F



D	Q(t+1)	
0	0	clear to 0
1	1	set to 1

"no change" condition: Q(t+1)=D

n JK*(Jack/King)* F/F

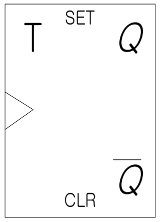


J	K	Q(t+1)	
0	0	Q(t)	no change
0	1	0	clear to 0
1	0	1	set to 1
1	1	Q(t)'	Complement

- JK F/F is a refinement of the SR F/F
- The indeterminate condition of the SR type is defined in complement

1) Disable Clock 2) Feedback output into input

n T*(Toggle)* F/F



T	Q(t+1)	
0	Q(t)	no change
1	Q'(t)	Complement

- T=1(J=K=1), T=0(J=K=0) JK F/F
- $Q(t+1)= Q(t) \oplus T$

Flip-flops

Excitation Table

- Required input combinations for a given change of state
- Present State Next State

SR F/F			
Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	0	1
1	0	1	0
1	1	X	1

JK F/F			
Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D F/F		
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

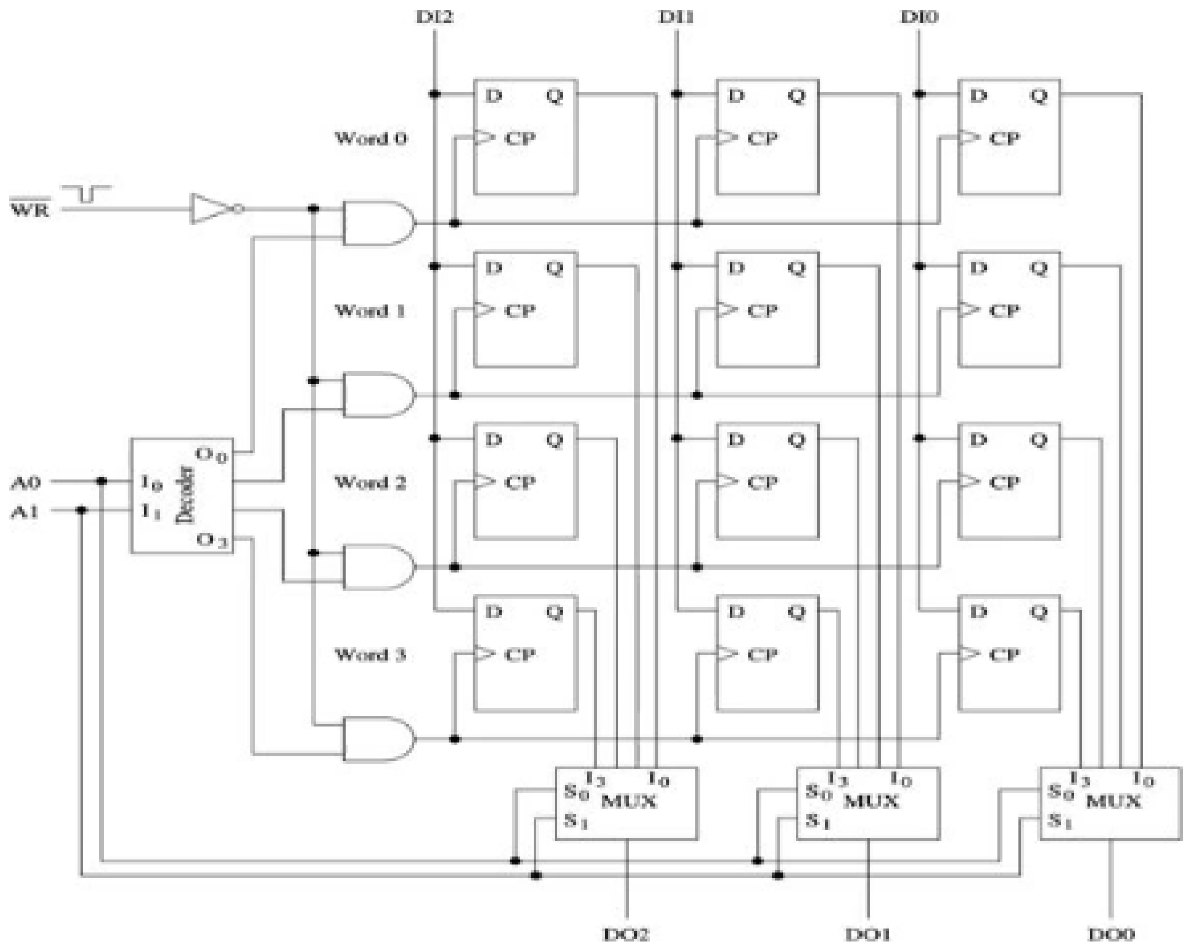
T F/F		
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Don't Care

1 : Set to 1
0 : Complement

1 : Clear to 0
0 : No change

Use of Circuits: Memory



Thank You

End of CF from ICS