Computer Architecture

Introduction and Review

Prof. Aliaa Youssif

DIGITAL LOGIC CIRCUITS

- **Logic Gates**
- **Boolean Algebra**
- **Map Specification**
- **Combinational Circuits**
- **Flip-Flops**
- **Sequential Circuits**
- **Memory Components**

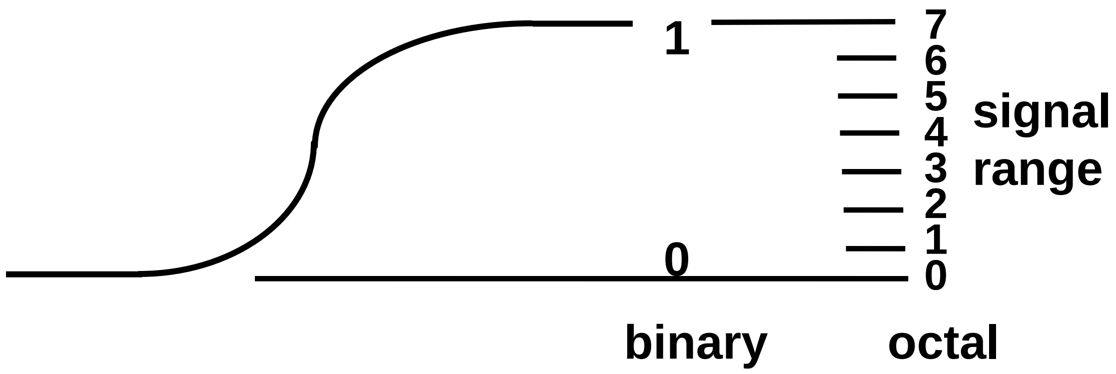
LOGIC GATES

Digital Computers

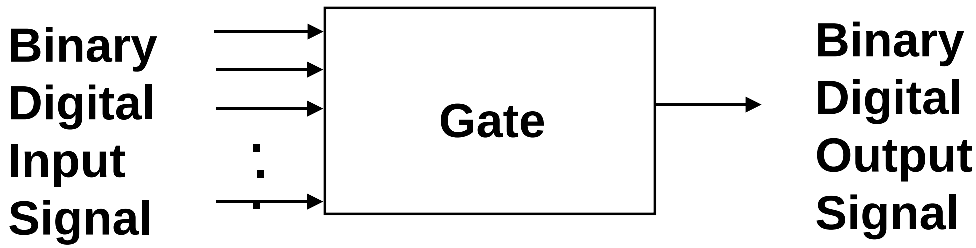
- Imply that the computer deals with digital information, i.e., it deals
with the information that is represented by binary digits

- Why *BINARY* ? instead of Decimal or other number system ?

*** Consider electronic signal**



BASIC LOGIC BLOCK - GATE -



Types of Basic Logic Blocks

- **Combinational Logic Block**

  Logic Blocks whose output logic value depends only on the input logic values

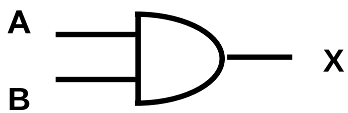
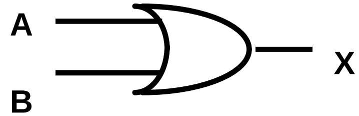
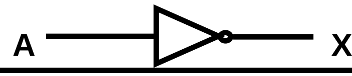
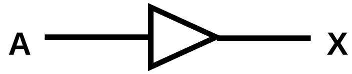
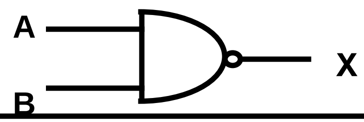
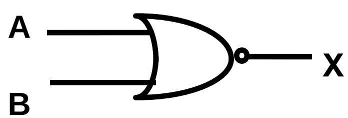
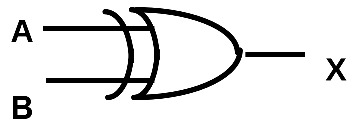
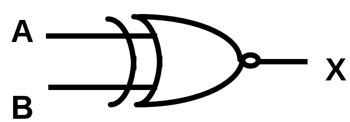
- **Sequential Logic Block**

  Logic Blocks whose output logic value depends on the input values and the state (stored information) of the blocks

Functions of Gates can be described by

- **Truth Table**
- **Boolean Function**
- **Karnaugh Map**

COMBINATIONAL GATES

Name	Symbol	Function	Truth Table
AND		X = A • B or X = AB	A B | X 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		X = A + B	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
I		X = A	A | X 0 | 1 1 | 0
Buffer		X = A	A | X 0 | 0 1 | 1
NAND		X = (AB)’	A B | X 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		X = (A + B)’	A B | X 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR Exclusive OR		X = A ⊕ B or X = A’B + AB’	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
XNOR Exclusive NOR or Equivalence		X = (A ⊕ B)’ or X = A’B’+ AB	A B | X 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

BOOLEAN ALGEBRA

Boolean Algebra

*** Algebra with Binary(Boolean) Variable and Logic Operations**
*** Boolean Algebra is useful in Analysis and Synthesis of Digital Logic Circuits**

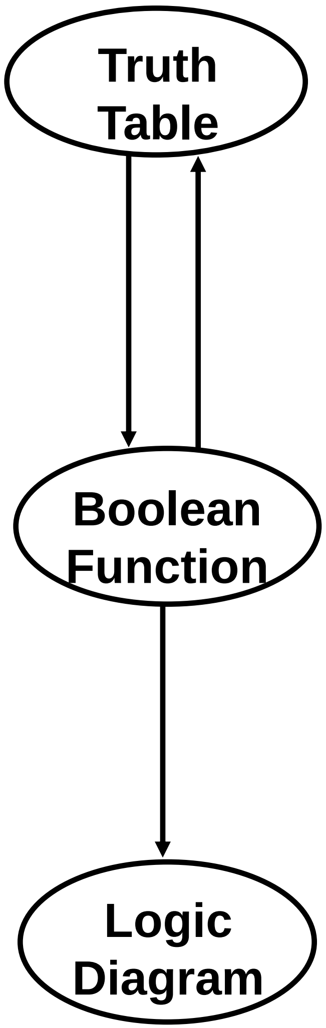
- Input and Output signals can be represented by Boolean Variables, and
- Function of the Digital Logic Circuits can be represented by Logic Operations, i.e., Boolean Function(s)
- From a Boolean function, a logic diagram can be constructed using AND, OR, and I

Truth Table

*** The most elementary specification of the function of a Digital Logic Circuit is the Truth Table**

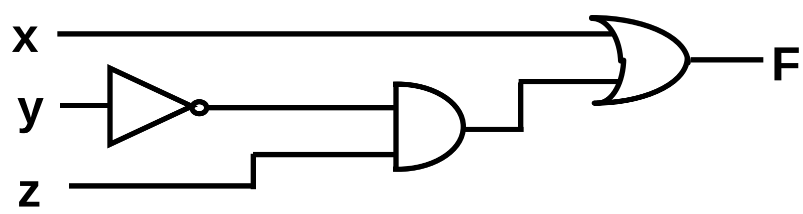
- Table that describes the Output Values for all the combinations of the Input Values, called *MINTERMS*
- n input variables → 2^n minterms

LOGIC CIRCUIT DESIGN



x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

F = x + y'z



BASIC IDENTITIES OF BOOLEAN ALGEBRA

[1] $x + 0 = x$	[2] $x \cdot 0 = 0$
[3] $x + 1 = 1$	[4] $x \cdot 1 = x$
[5] $x + x = x$	[6] $x \cdot x = x$
[7] $x + x' = 1$	[8] $x \cdot X' = 0$
[9] $x + y = y + x$	[10] $xy = yx$
[11] $x + (y + z) = (x + y) + z$	[12] $x(yz) = (xy)z$
[13] $x(y + z) = xy + xz$	[14] $x + yz = (x + y)(x + z)$
[15] $(x + y)' = x'y'$	[16] $(xy)' = x' + y'$
[17] $(x')' = x$	

[15] and [16] : De Morgan's Theorem

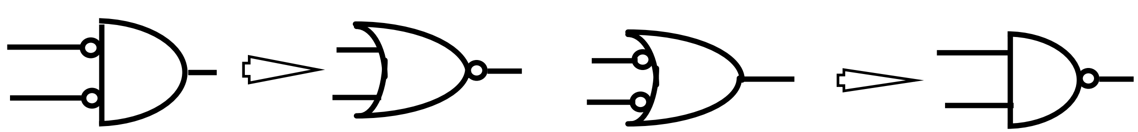
Usefulness of this Table

- **Simplification of the Boolean function**
- **Derivation of equivalent Boolean functions to obtain logic diagrams utilizing different logic gates**
  - **-- Ordinarily ANDs, ORs, and Inverters**
  - **-- But a certain different form of Boolean function may be convenient to obtain circuits with NANDs or NORs**

→ Applications of De Morgans Theorem

$$x'y' = (x + y)' \qquad x' + y' = (xy)'$$

I, AND → NOR   **I, OR → NAND**



EQUIVALENT CIRCUITS

Many different logic diagrams are possible for a given Function

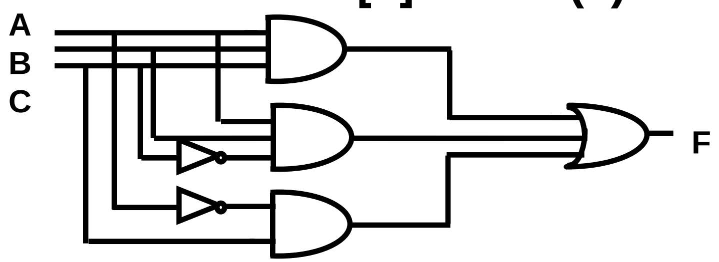
$F = ABC + ABC' + A'C$ .... (1)

$= AB(C + C') + A'C$ [13] …… (2)

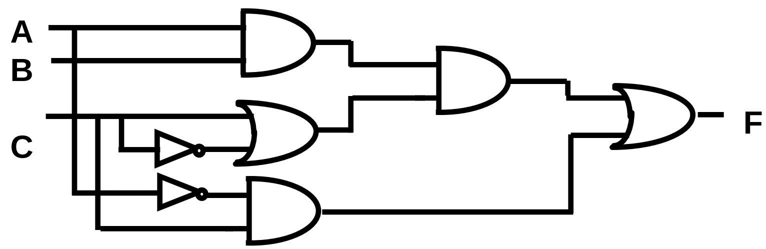
$= AB \cdot 1 + A'C$ [7]

$= AB + A'C$ [4] ……. (3)

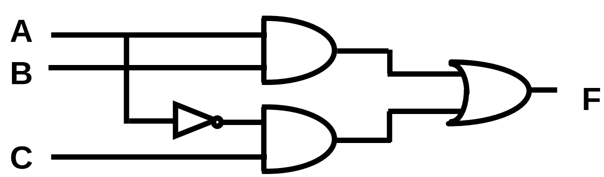
(1)



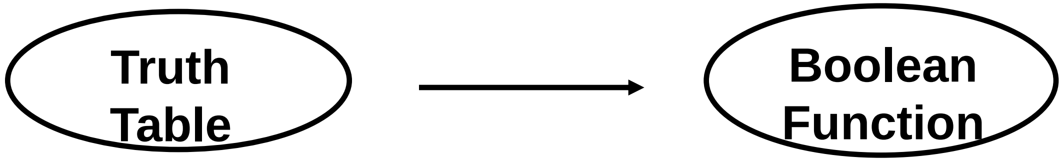
(2)



(3)



SIMPLIFICATION

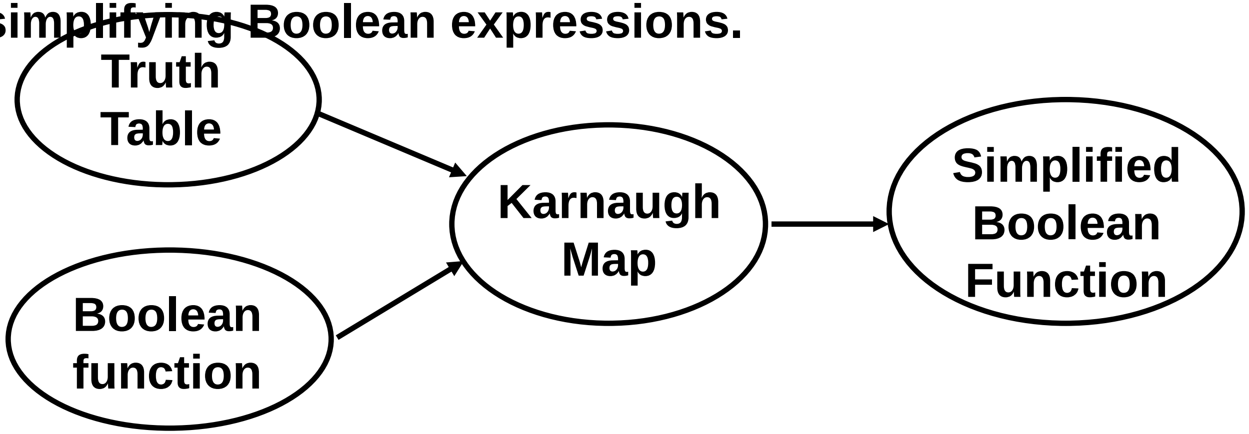


Unique **Many different expressions exist**

Simplification from Boolean function

- Finding an equivalent expression that is least expensive to implement
- For a simple function, it is possible to obtain a simple expression for low cost implementation
- But, with complex functions, it is a very difficult task

Karnaugh Map (K-map) is a simple procedure for simplifying Boolean expressions.



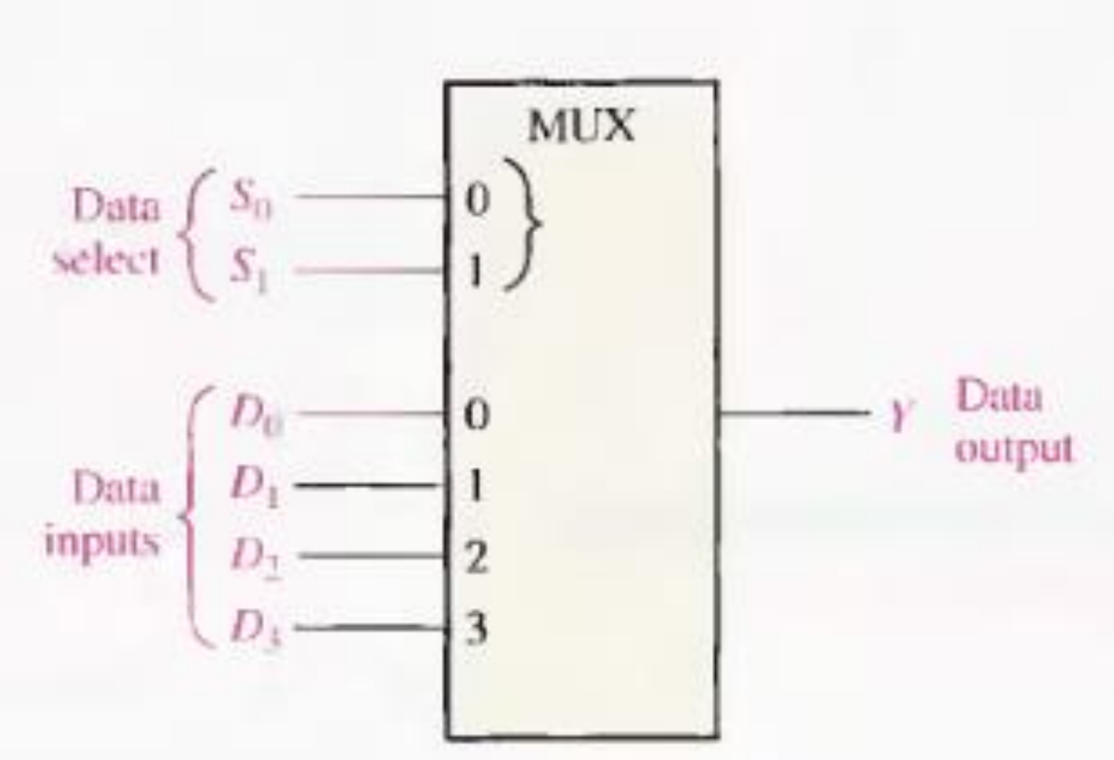
COMBINATIONAL LOGIC CIRCUITS

- **Multiplexer**
- **Encoder**
- **Decoder**
- **Parity Checker**
- **Parity Generator**
- **etc**

Selecting

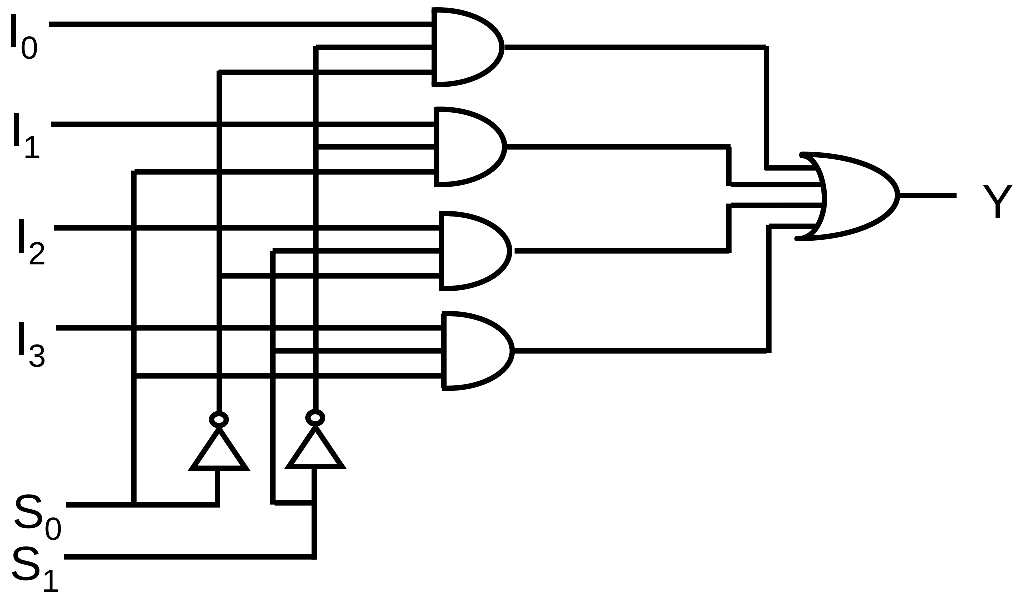
- Selecting of data or information is a critical function in digital systems and computers
- Circuits that perform selecting have:
  - A set of information inputs from which the selection is made
  - A single output
  - A set of control lines for making the selection
- Logic circuits that perform selecting are called *multiplexers*

4-to-1-line Multiplexer



4-to-1-line Multiplexer

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

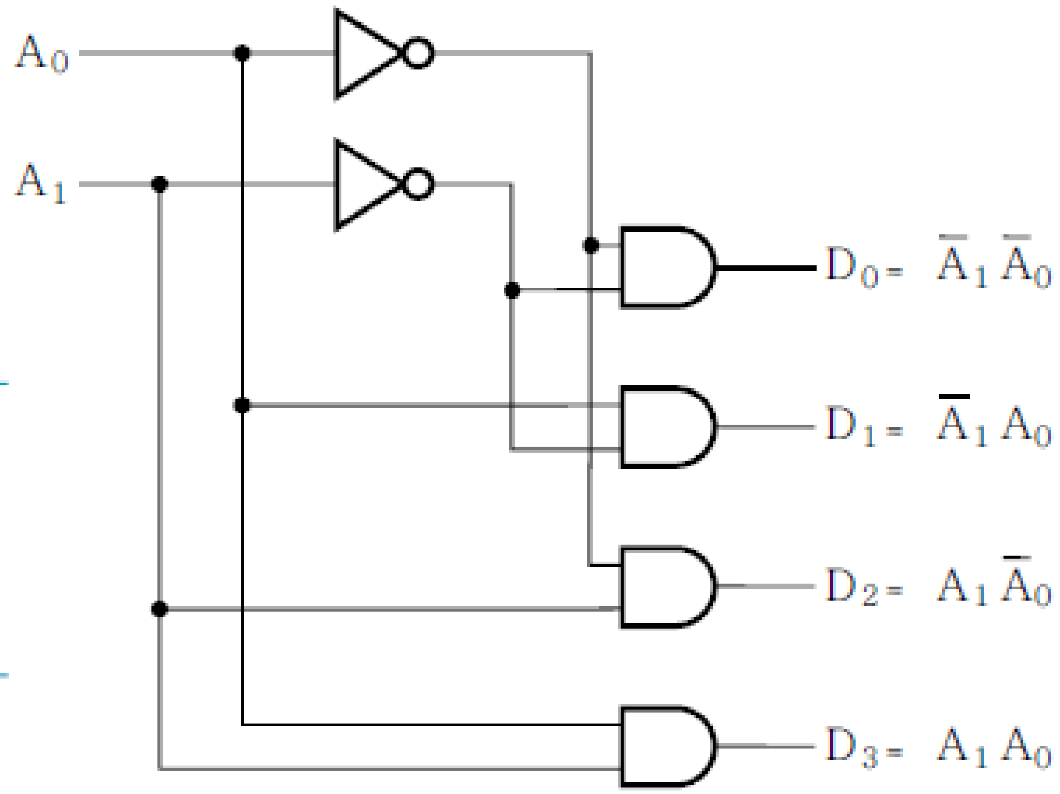


Decoding

- Decoding - the conversion of an *n*-bit input code to an *m*-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
- Circuits that perform decoding are called *decoders*

2-to-4 Decoder

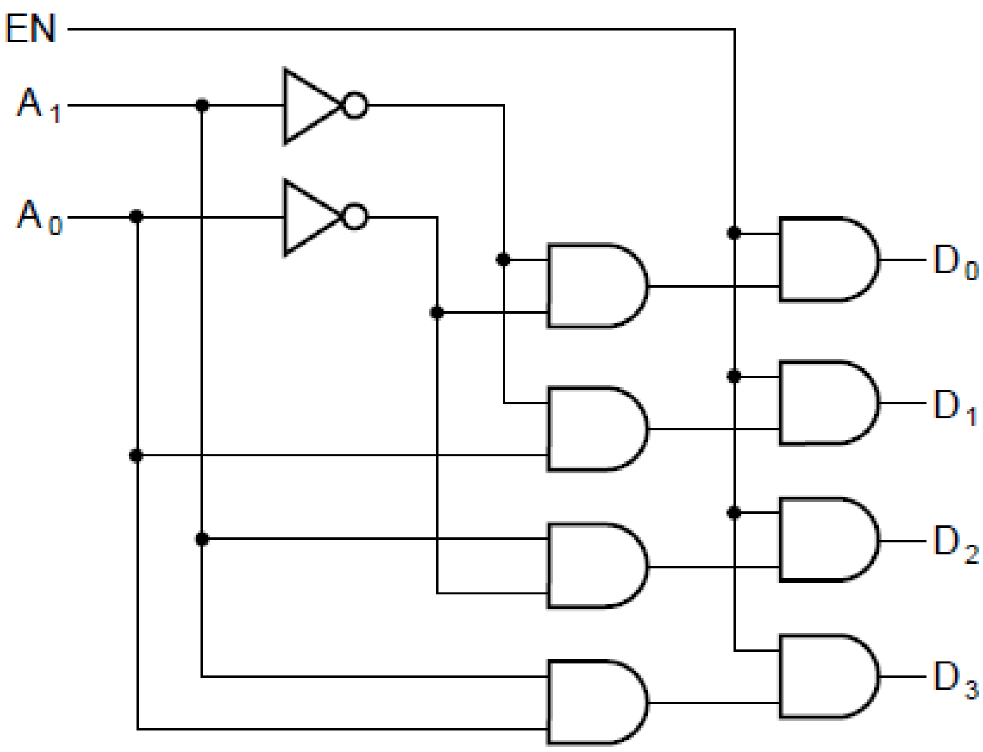
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



2-to-4 Decoder with enable

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

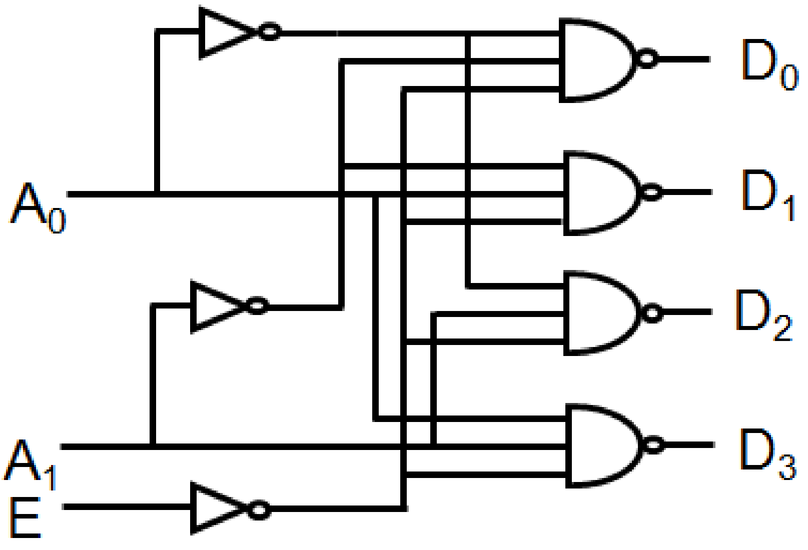
(a)



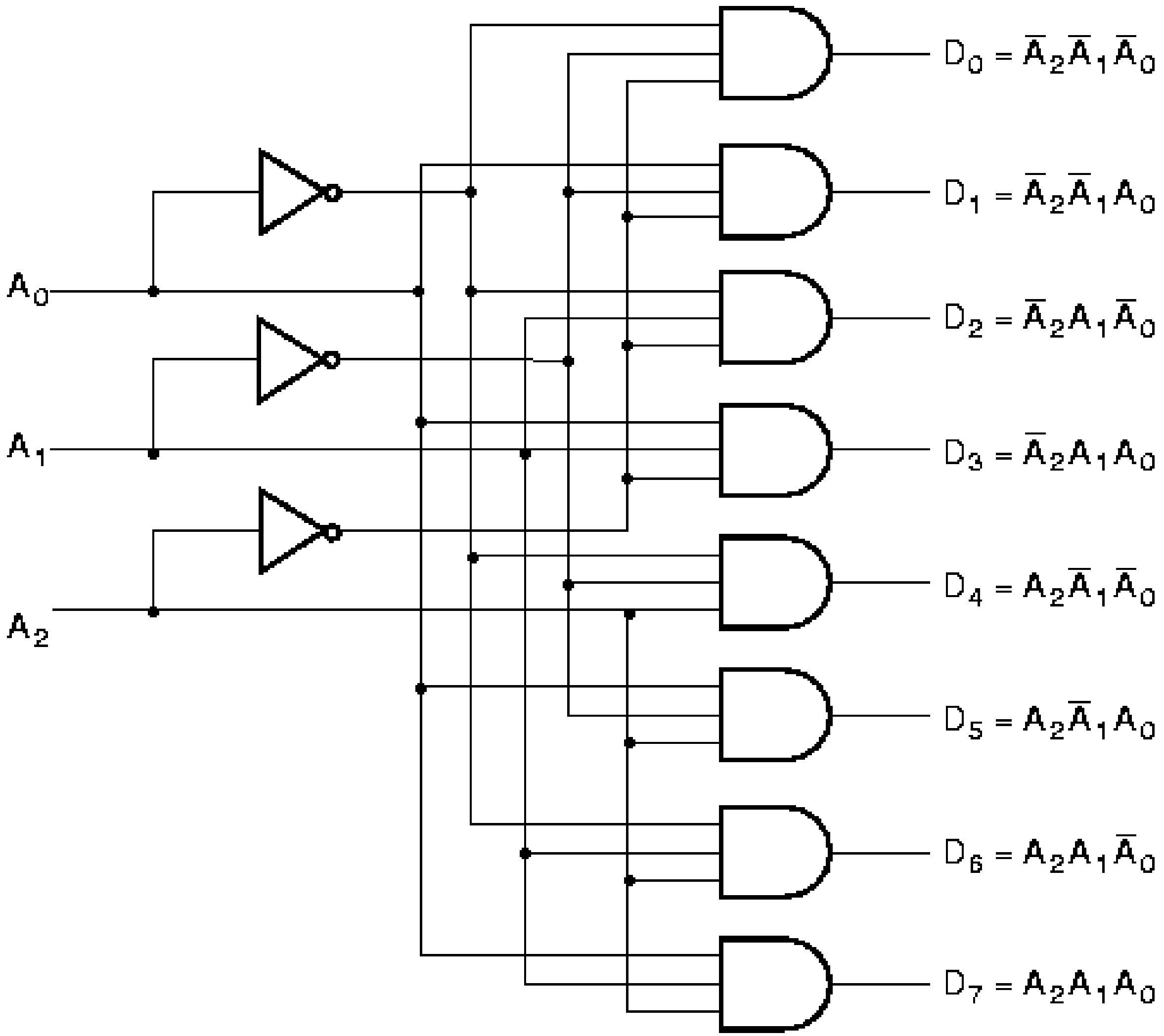
(b)

2-to-4 Decoder with enable

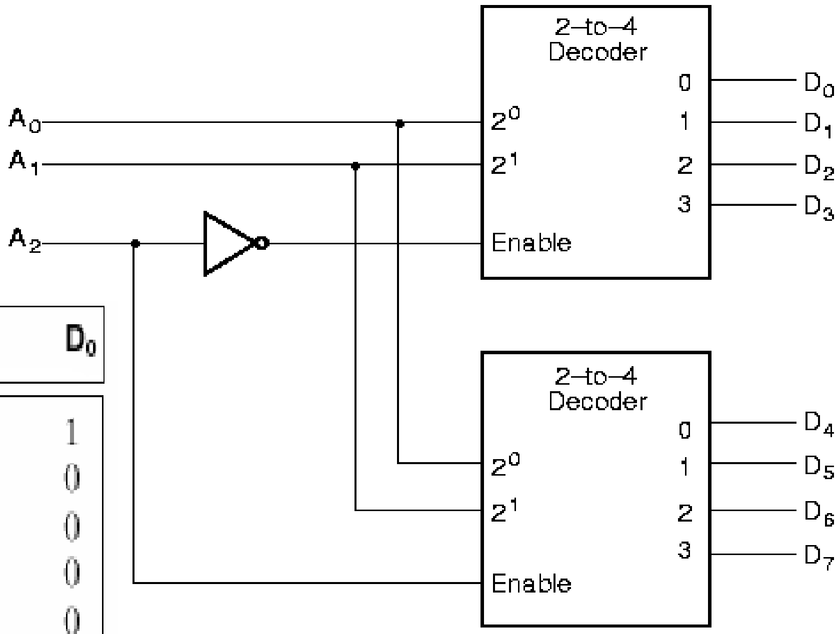
E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	d	d	1	1	1	1



3-to-8-line Decoder

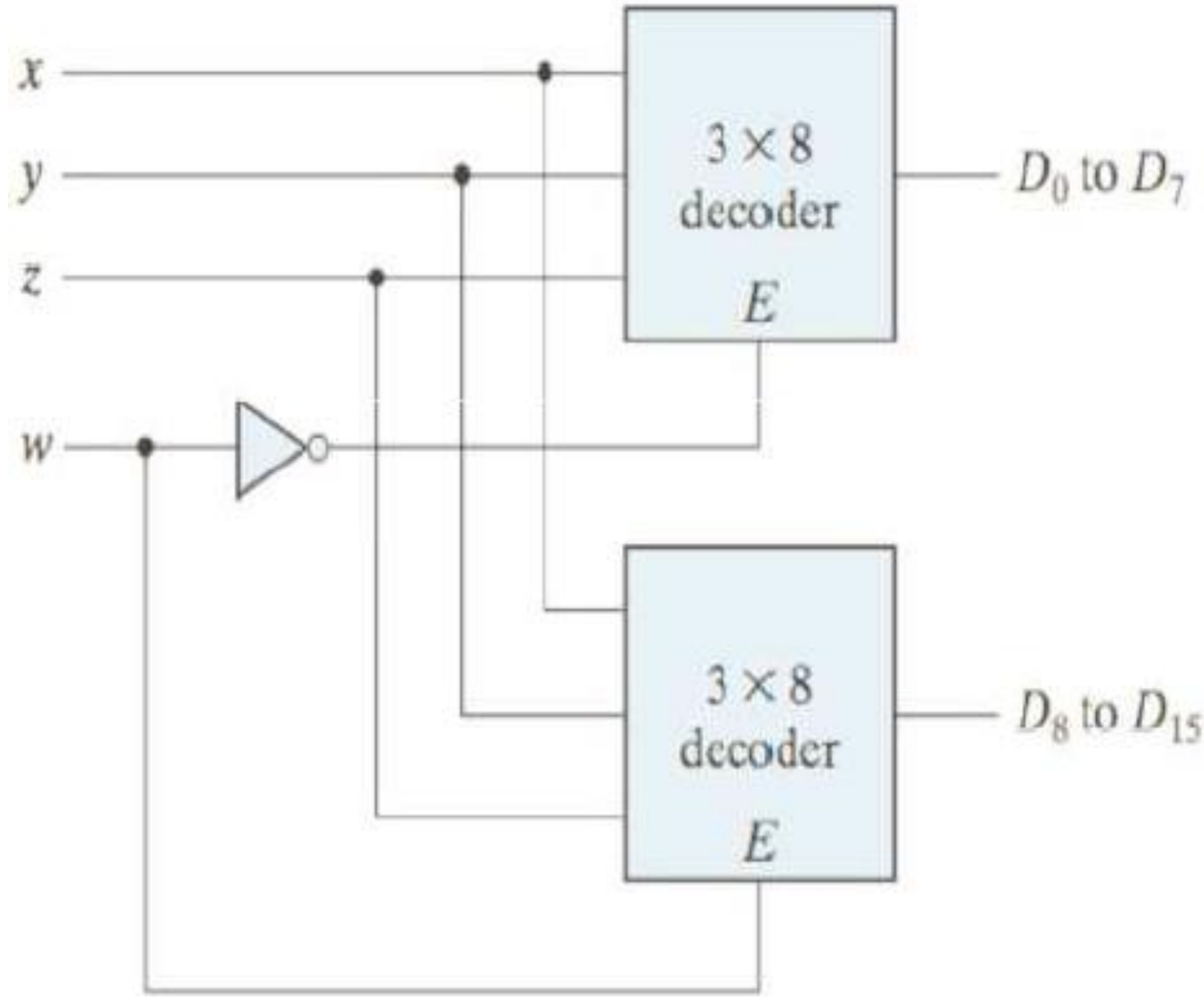


Decoder Expansion



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

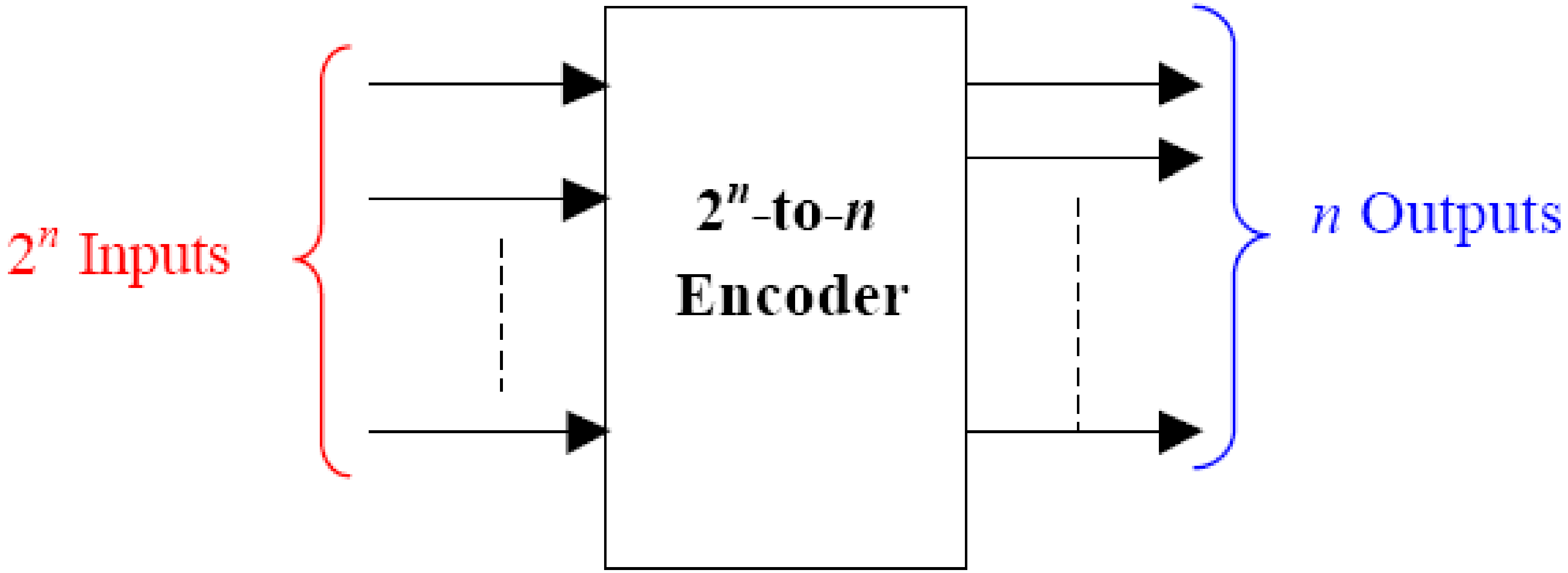
Decoder Expansion



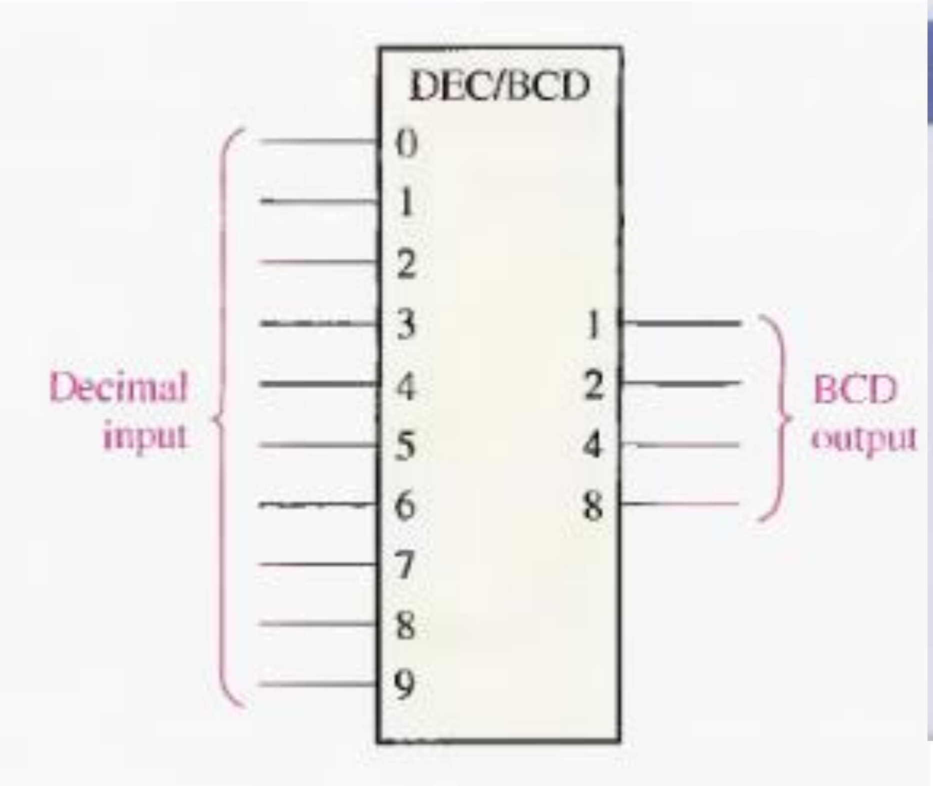
Encoding

- Encoding - the opposite of decoding - the conversion of an m-bit input code to a n-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
- Circuits that perform encoding are called *encoders*
- An encoder has 2^n (or fewer) input lines and n output lines which generate the binary code corresponding to the input values

Encoder

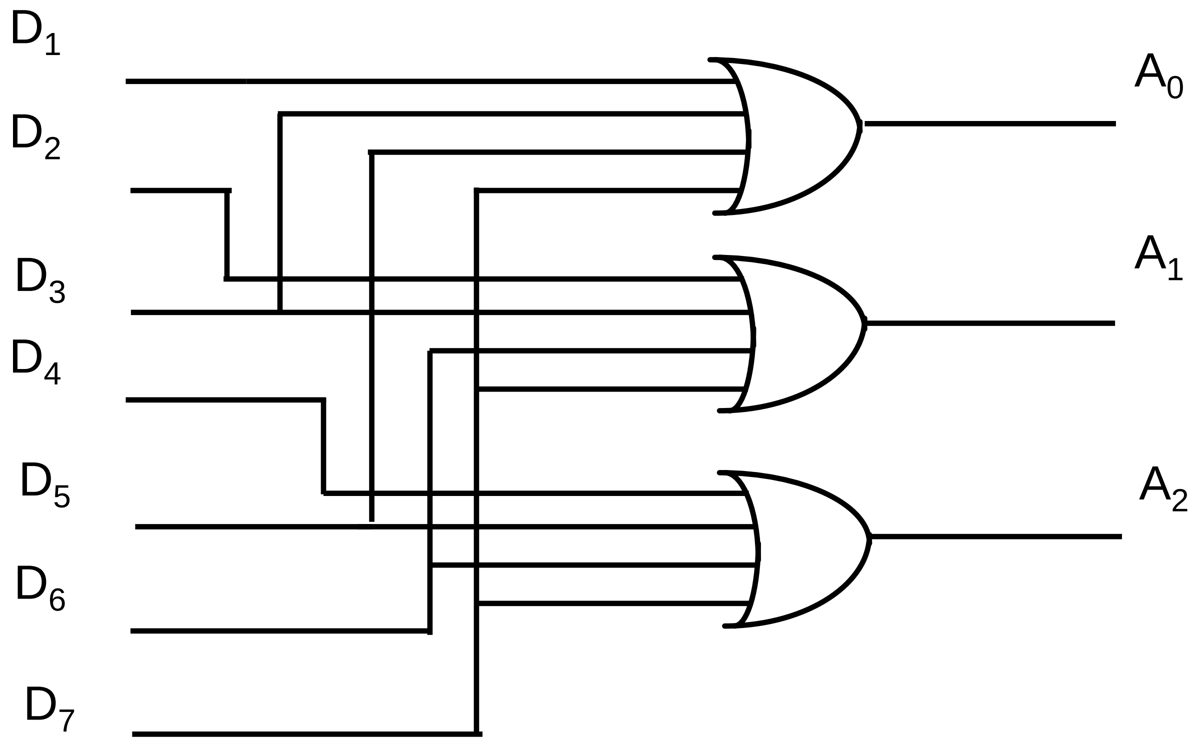


Decimal-to-BCD Encoder



DECIMAL DIGIT	BCD CODE A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Octal-to-Binary Encoder



Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

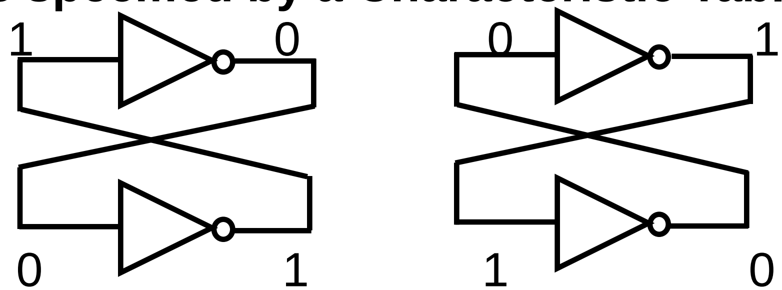
$x = D_4 + D_5 + D6 + D7 \quad y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$

FLIP FLOPS

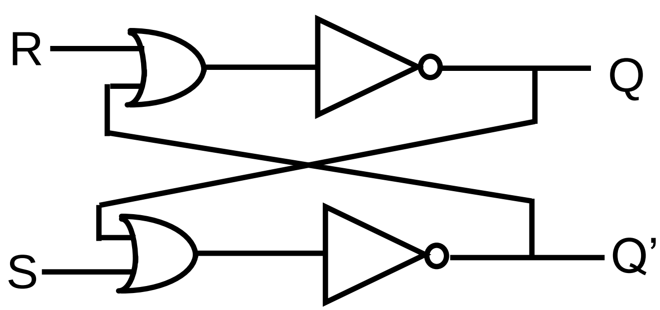
Characteristics

- **2 stable states**
- **Memory capability**
- **Operation is specified by a Characteristic Table**



0-state **1-state**

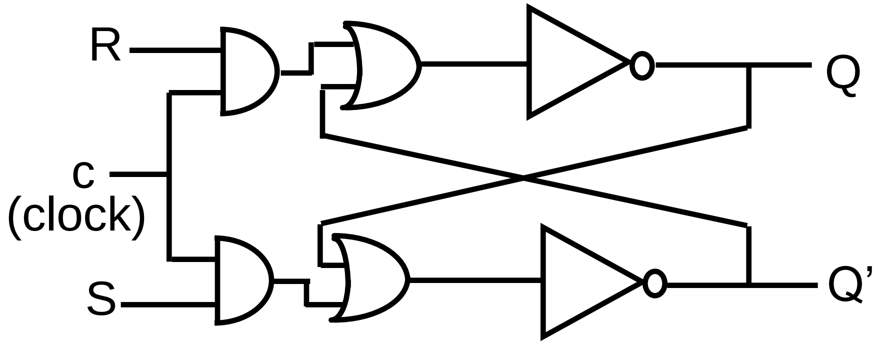
In order to be used in the computer circuits, state of the flip flop should have input terminals and output terminals so that it can be set to a certain state, and its state can be read externally.



S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	indeterminate (forbidden)

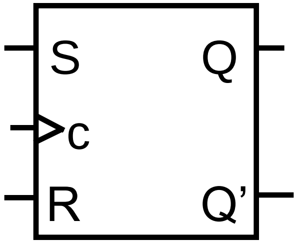
CLOCKED FLIP FLOPS

In a large digital system with many flip flops, operations of individual flip flops are required to be synchronized to a clock pulse. Otherwise, the operations of the system may be unpredictable.

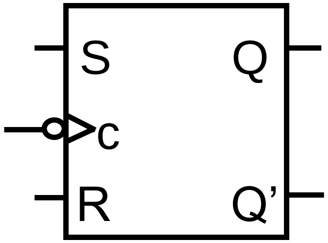


Clock pulse allows the flip flop to change state only when there is a clock pulse appearing at the c terminal.

We call above flip flop a Clocked RS Latch, and symbolically as

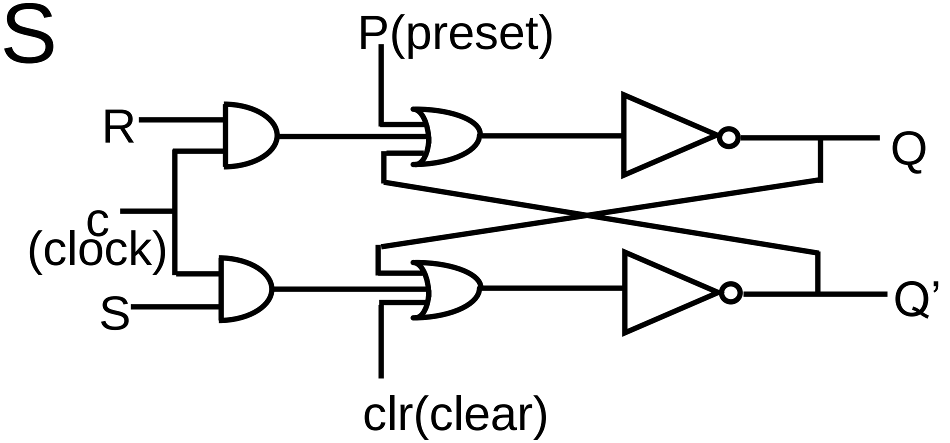


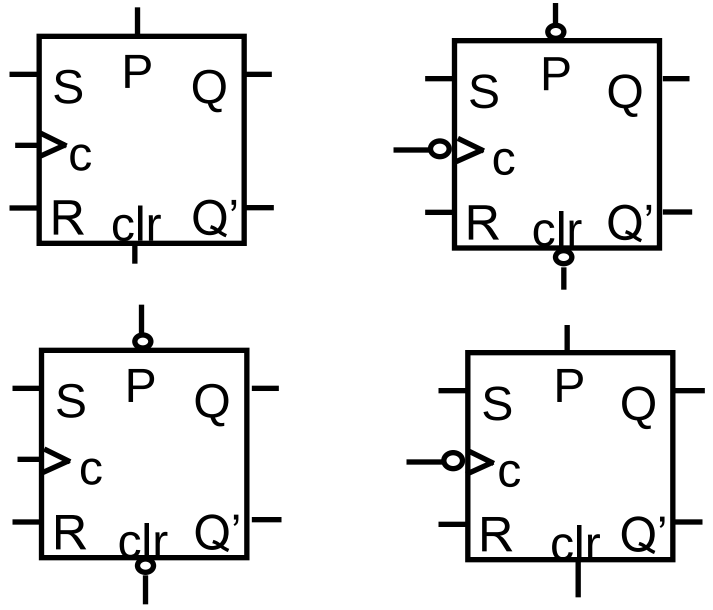
operates when clock is high



operates when clock is low

RS-LATCH WITH PRESET AND CLEAR INPUTS

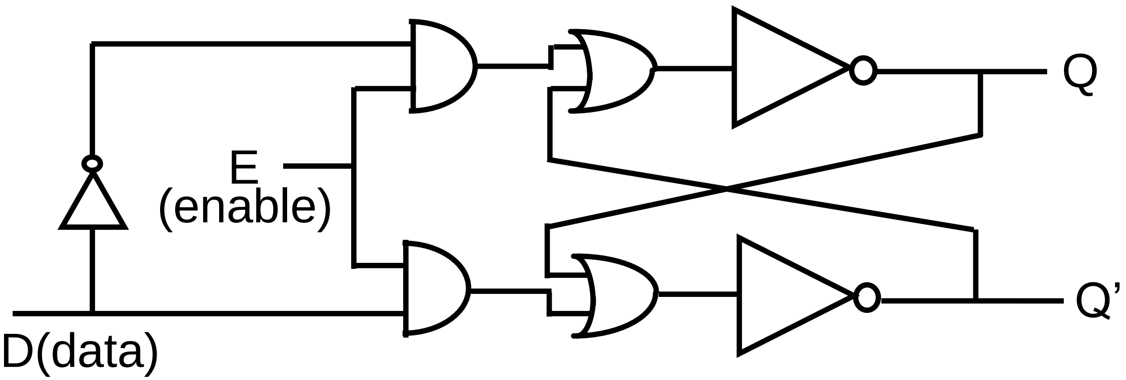


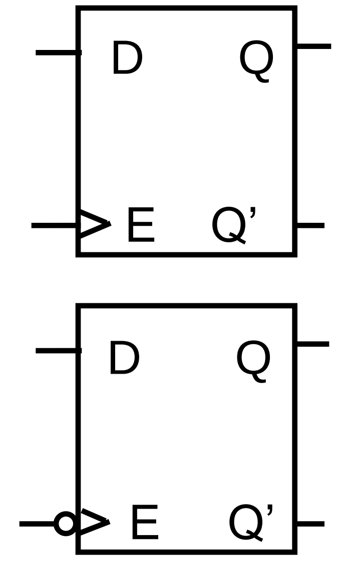


D-LATCH

D-Latch

Forbidden input values are forced not to occur by using an inverter between the inputs





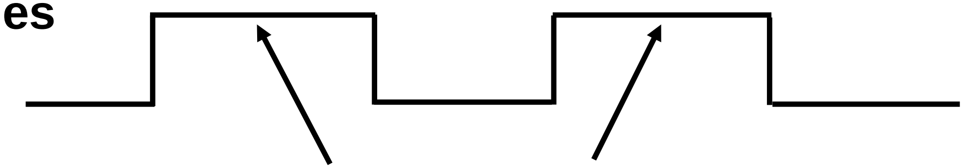
D	Q(t+1)
0	0
1	1

EDGE-TRIGGERED FLIP FLOPS

Characteristics

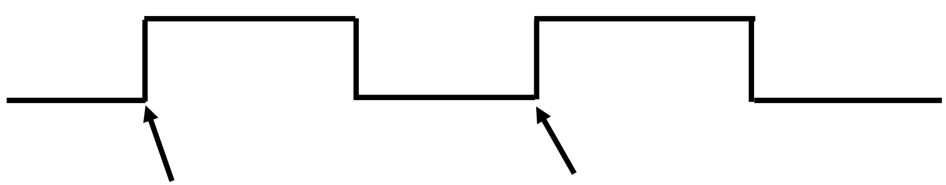
- **State transition occurs at the rising edge or falling edge of the clock pulse**

Latches



respond to the input only during these periods

Edge-triggered Flip Flops (positive)

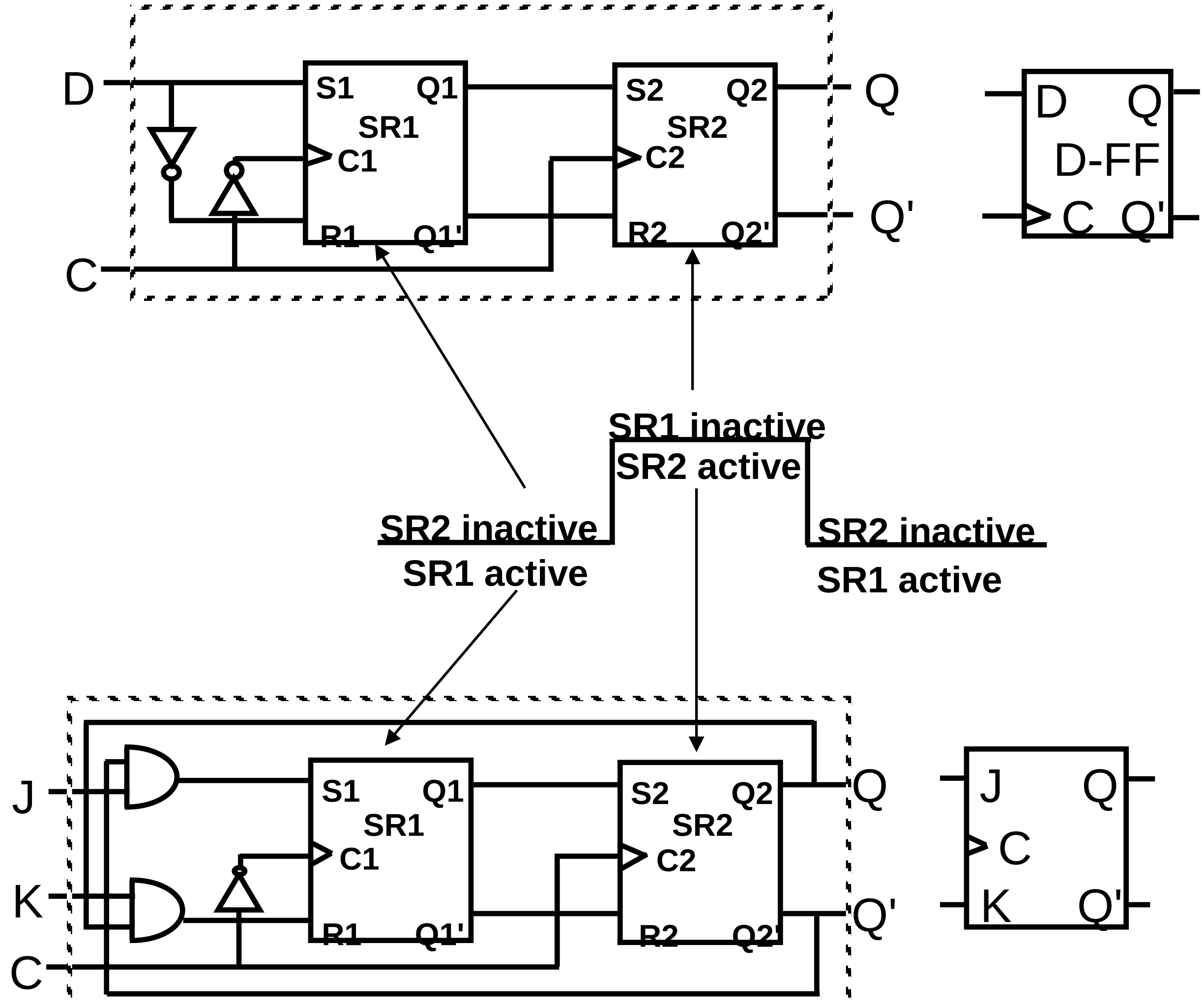


respond to the input only at this time

POSITIVE EDGE-TRIGGERED

D-Flip Flop

JK-Flip Flop

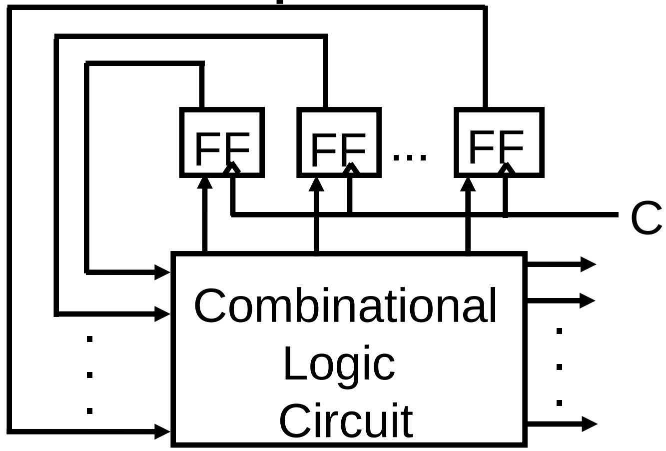


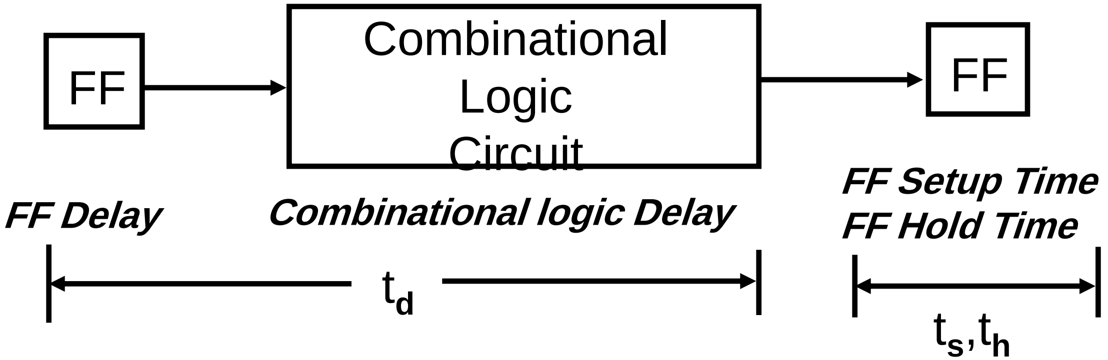
T-Flip Flop: JK-Flip Flop whose J and K inputs are tied together to make T input. Toggles whenever there is a pulse on T input.

CLOCK PERIOD

Clock period determines how fast the digital circuit operates.
How can we determine the clock period ?

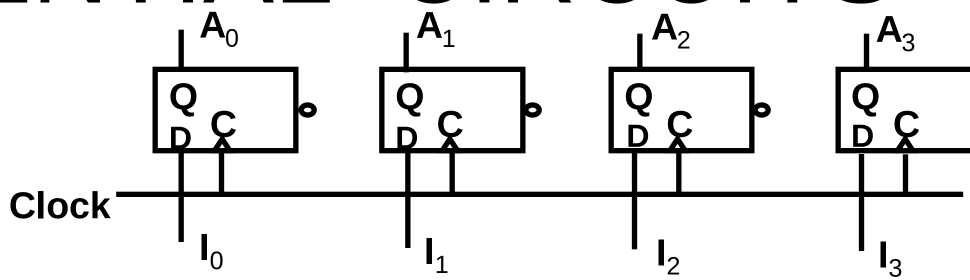
Usually, digital circuits are sequential circuits which has some flip flops



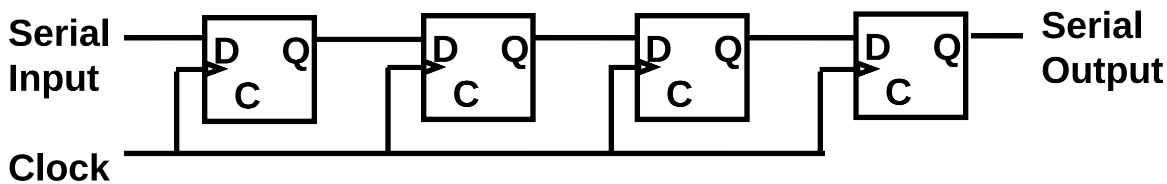


clock period T = $t_d + t_s + t_h$

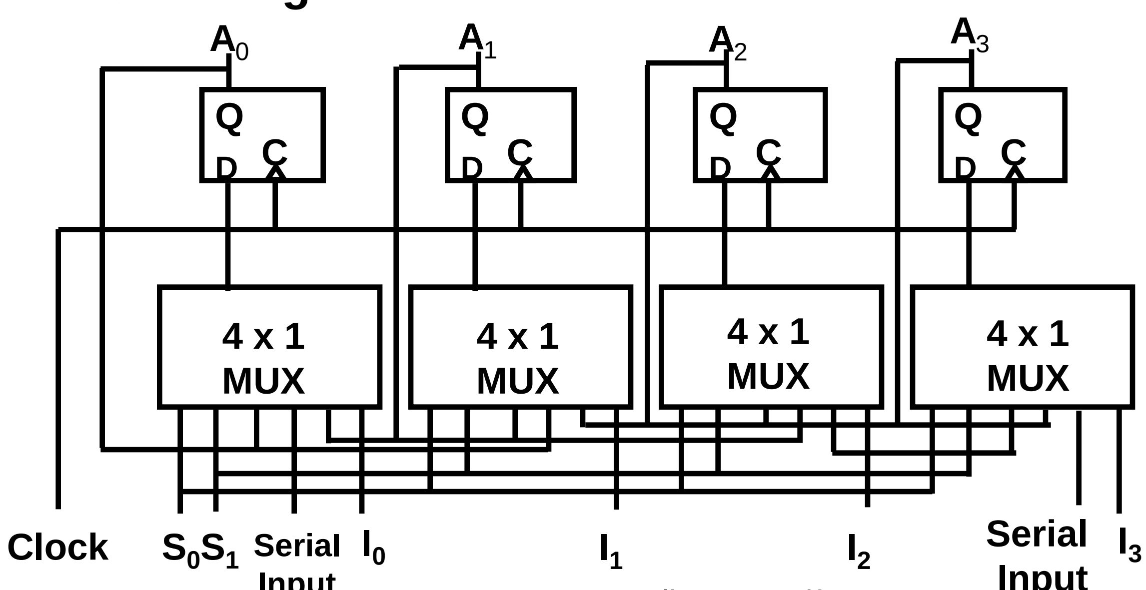
SEQUENTIAL CIRCUITS - Registers



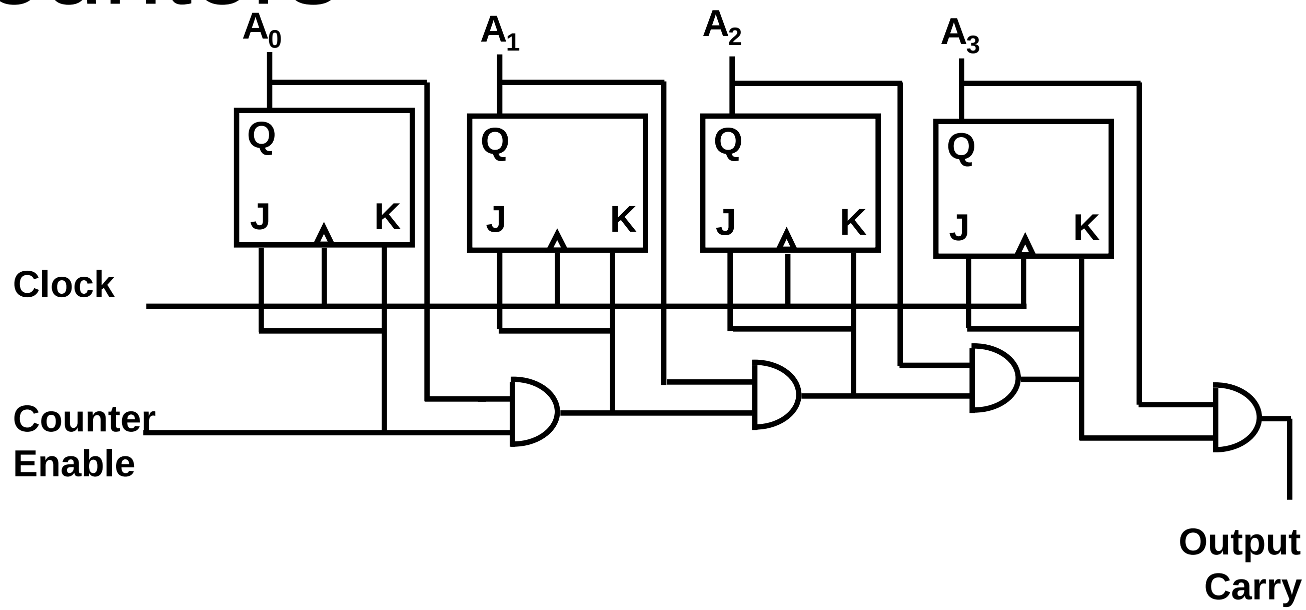
Shift Registers



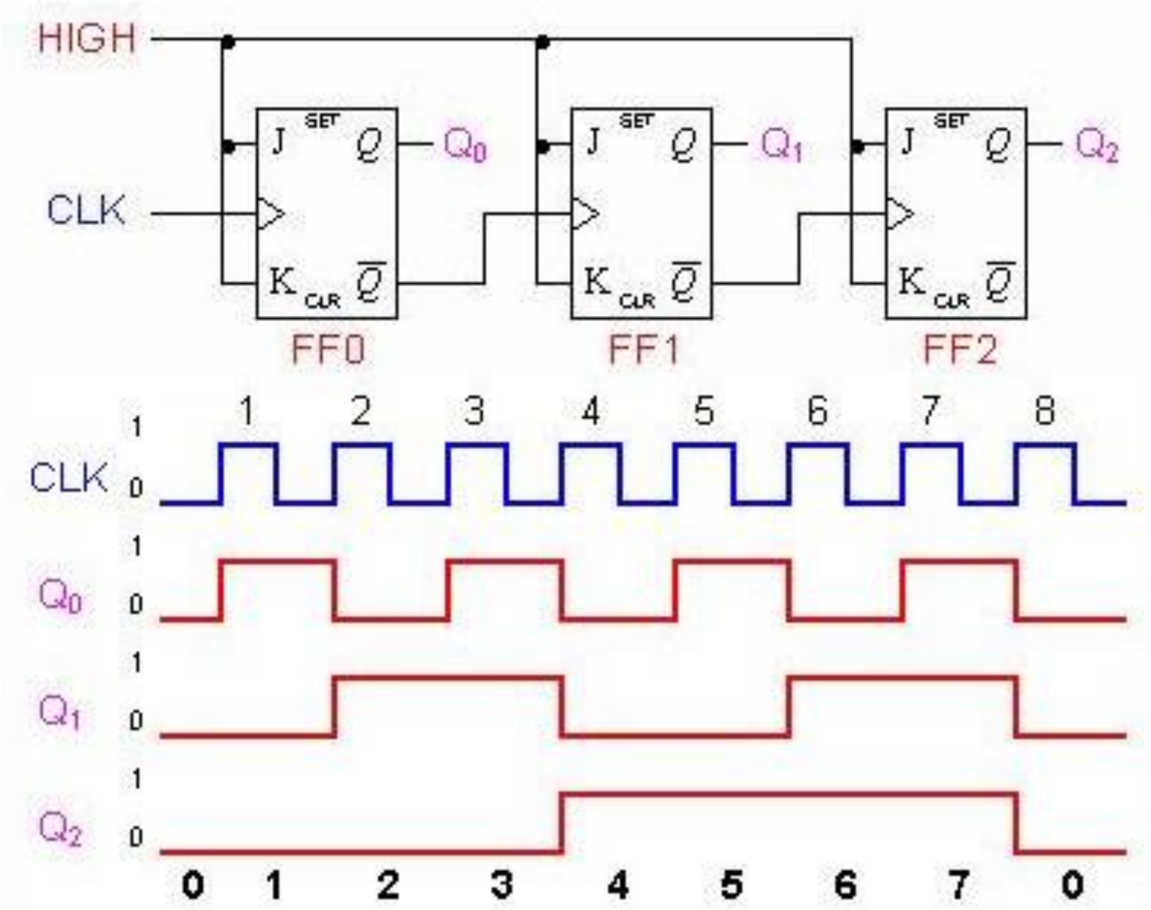
Bidirectional Shift Register with Parallel Load



SEQUENTIAL CIRCUITS - Counters

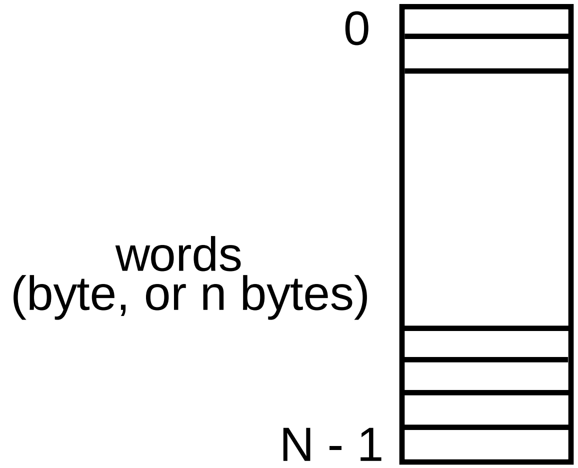


SEQUENTIAL CIRCUITS - Counters



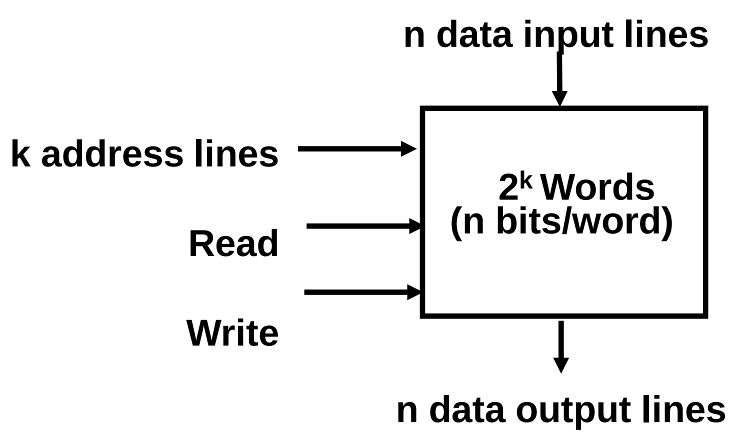
MEMORY COMPONENTS

Logical Organization

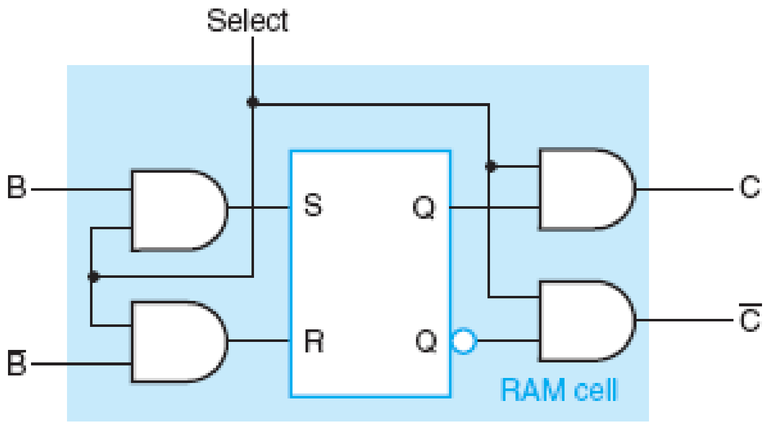


Random Access Memory

- **Each word has a unique address**
- **Access to a word requires the same time independent of the location of the word**
- **Organization**



MEMORY COMPONENTS

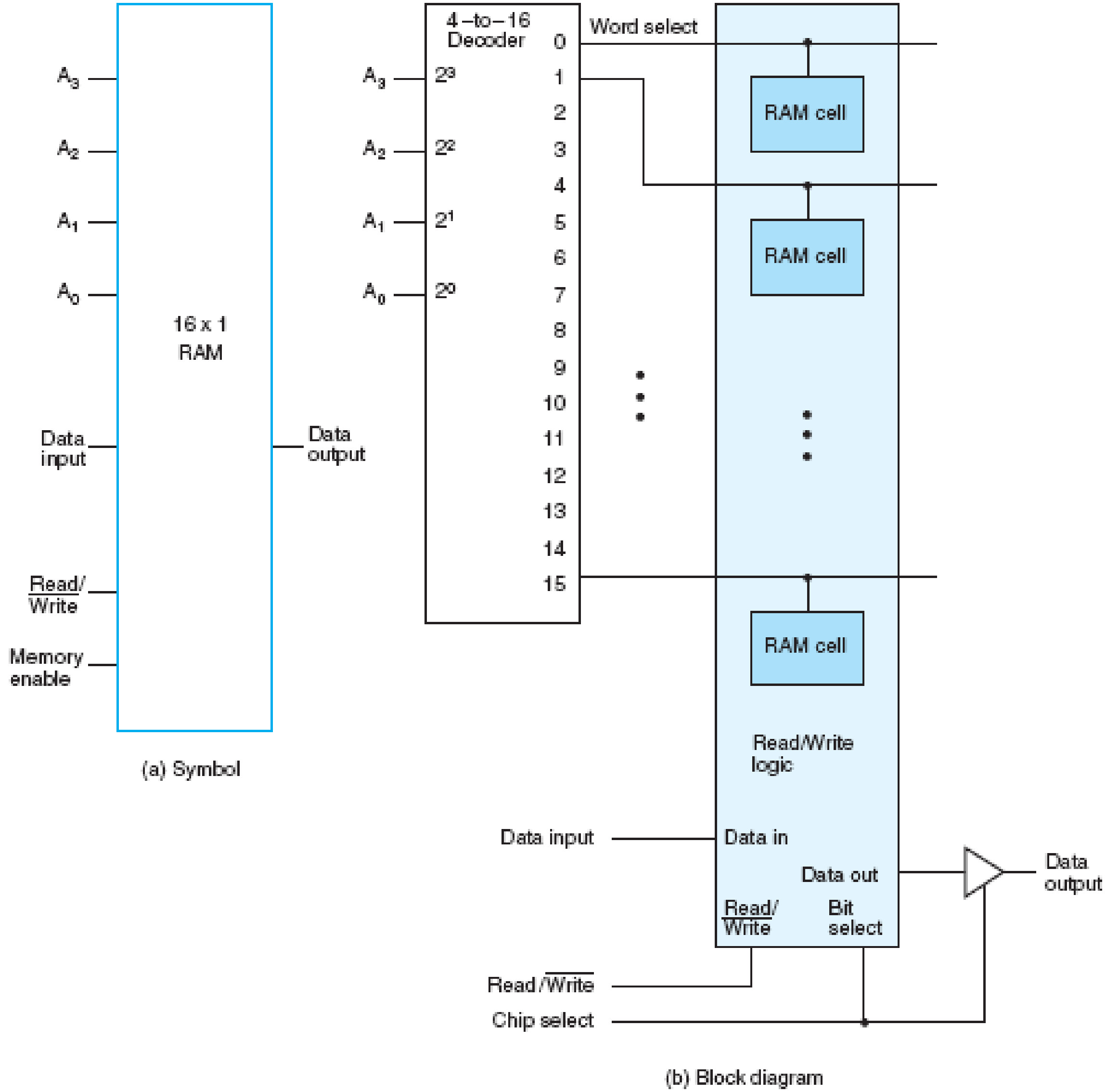


Static RAM Cell

Memory address: Binary	Memory address: Decimal	Memory contents
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
	•	•
	•	•
	•	•
	•	•
	•	•
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

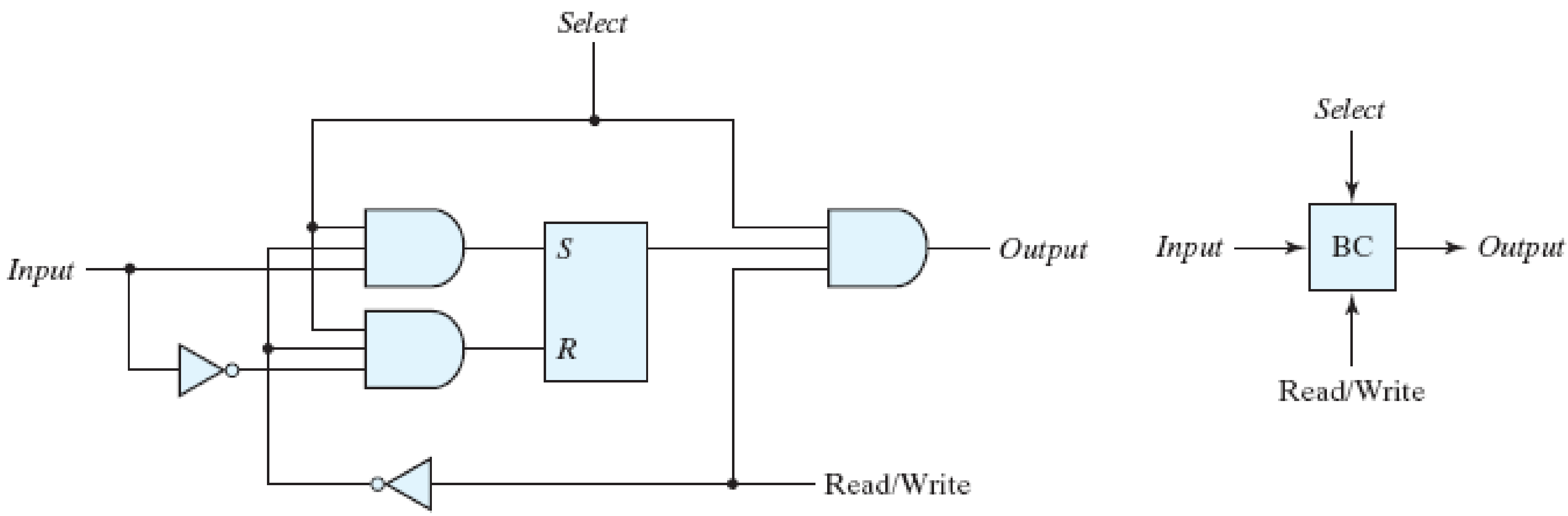
Contents of a 1024 × 16 Memory

MEMORY COMPONENTS



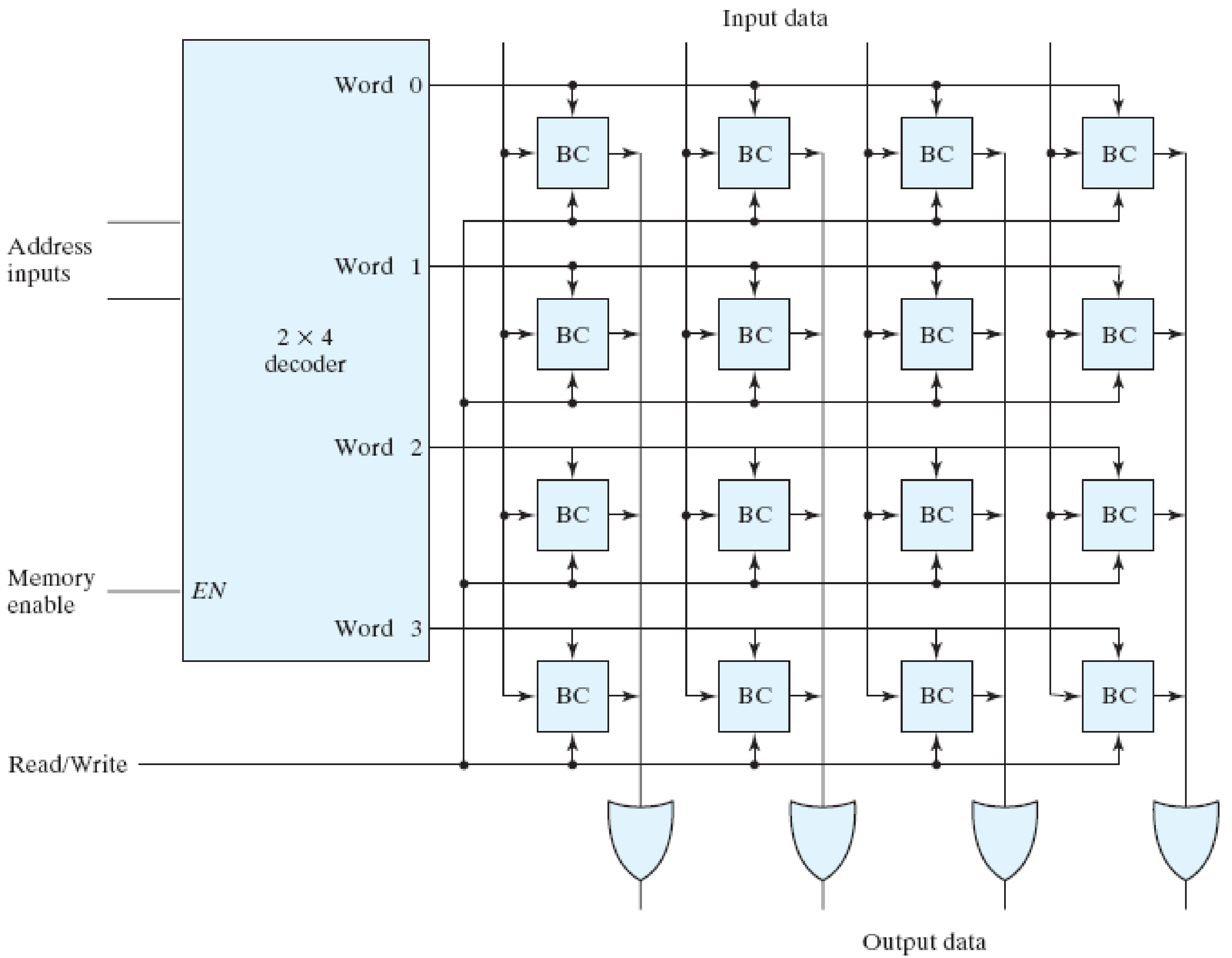
16-Word by 1-Bit RAM Chip

MEMORY COMPONENTS



(a) Logic diagram (b) Block diagram

MEMORY COMPONENTS



READ ONLY MEMORY(ROM)

Characteristics

- **Perform read operation only, write operation is not possible**
- **Information stored in a ROM is made permanent during production, and cannot be changed**
- **Organization**

k address input lines

m x n ROM
($m=2^k$)

n data output lines

Information on the data output line depends only on the information on the address input lines.

--> Combinational Logic Circuit

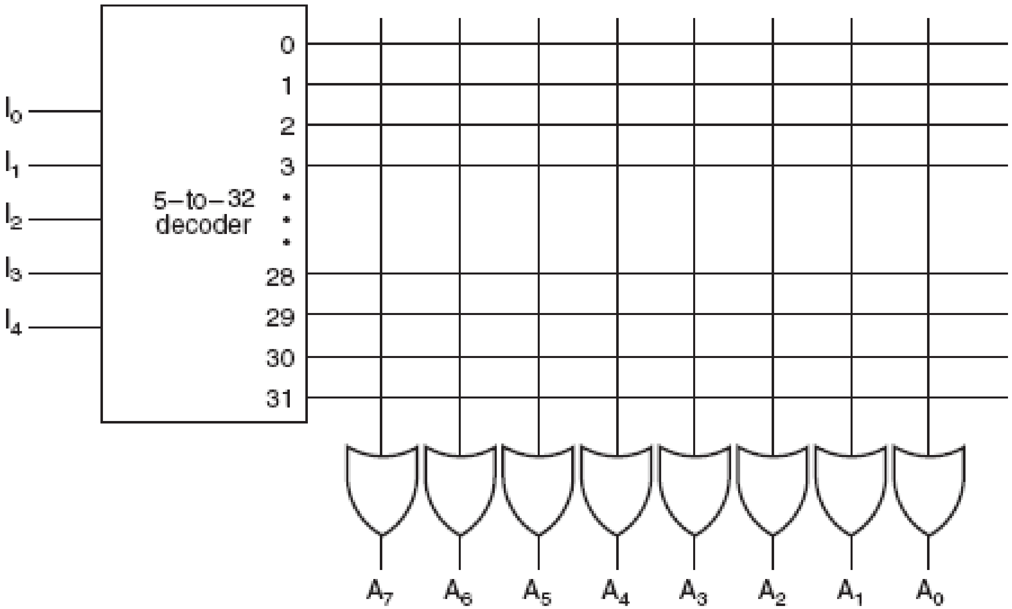
X_0=A'B' + B'C
X_1=A'B'C + A'BC'
X_2=BC + AB'C'
X_3=A'BC' + AB'
X_4=AB

Canonical minterms

X_0=A'B'C' + A'B'C + AB'C
X_1=A'B'C + A'BC'
X_2=A'BC + AB'C' + ABC
X_3=A'BC' + AB'C' + AB'C
X_4=ABC' + ABC

address	*Output*				
ABC	X_0	X_1	X_2	X_3	X_4
000	1	0	0	0	0
001	1	1	0	0	0
010	0	1	0	1	0
011	0	0	1	0	0
100	0	0	1	1	0
101	1	0	0	1	0
110	0	0	0	0	1
111	0	0	1	0	1

READ ONLY MEMORY(ROM)



Internal Logic of a 32 × 8 ROM

Block Diagram of ROM

TYPES OF ROM

ROM

- **Store information (function) during production**
- **Mask is used in the production process**
- **Unalterable**
- **Low cost for large quantity production --> used in the final products**

PROM (Programmable ROM)

- **Store info electrically using PROM programmer at the user's site**
- **Unalterable**
- **Higher cost than ROM -> used in the system development phase**
  -> Can be used in small quantity system

EPROM (Erasable PROM)

- **Store info electrically using PROM programmer at the user's site**
- **Stored info is erasable (alterable) using UV light (electrically in some devices) and rewriteable**
- **Higher cost than PROM but reusable --> used in the system development phase. Not used in the system production due to reusability**