Logic circuit

Subject: "Logic circuit"

Lecture 1: Introduction

Ganesh Adhikari

ganeshadhikarireal@gmail.com

ganeshpa@nec.edu.np

Logic circuit

Introduction

Books:

1. A text book of “Logic Circuit and Modern Digital Techniques”.

   by **Ganesh Adhikari.**

2. “Digital Logic and Computer Design”.

   by **M.Mano.**

3. “Computer System Architecture”.

   by **M.Mano.**

Logic circuit

Course Contents:

1. Introduction.
2. Number system and codes.
3. Boolean Algebra and Logic Gates.
4. Simplification of Boolean Function.
5. Combinational logic.
6. MSI and LSI components in combinational logic design.
7. Sequential logic.
8. Register, Counter and Memory unit.
9. Arithmetic Logic Units.

Logic circuit

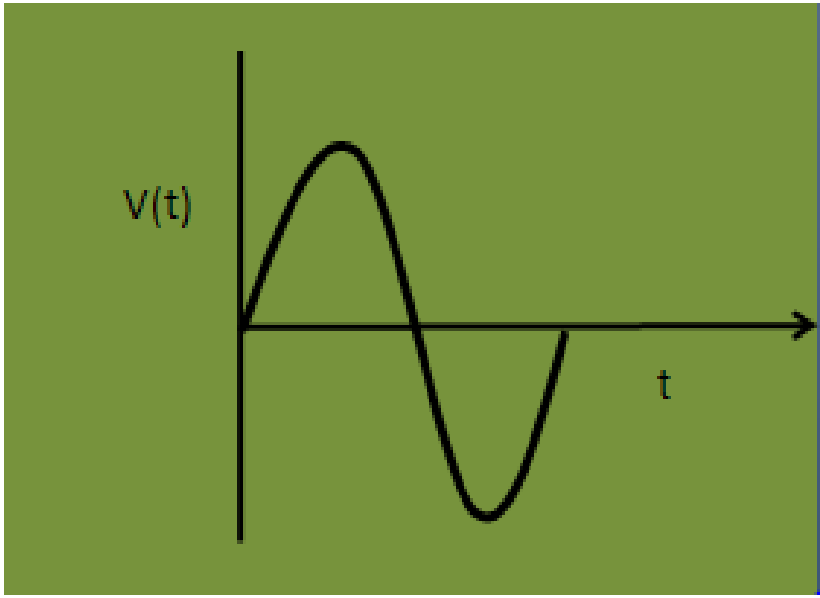
Analog representations

➢Quantity vary over a continuous range of values and any value within limited range implies meaning.

Analog = Continuous

eg.

- ❑ Automobile Speedometer.
- ❑ Audio-microphone.
- ❑ Deflection type Multimeter.
- ❑ Sine wave of Oscilloscope etc.



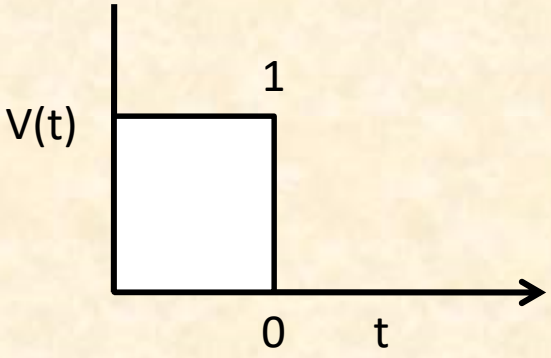
Digital representations

Digital representations

- Quantity changes in discrete steps and deals with only two values, logic high or logic low.
- Any value within limited range does not implies any meaning.

Digital = Discrete (Step by Step)

eg.



- Digital Watch.
- Numerical readout Calculator.
- Numerical readout Multimeter.
- Sand grains on Beach etc.

Digital Techniques Advantages

Important advantages are:

- Easier to design.
- Easier for information storage.
- Operation can be programmed.
- Easier to fabricate on **IC Chips**.
- Posses higher Accuracy and greater precision.
- Less effected by noise.

Binary Numbers and Codes

Binary to Decimal Conversion:

$(10101)_2 = 1x2^4+0x2^3+1x2^2+0x2^1+1x2^0$

$=16+0+4+0+1$

$= 21.$

$= (21)_{10}$

$(1010.01)_2 = 1x2^3+0x2^2+1x2^1+0x2^0+0x2^{-1}+1x2^{-2}$

$= 8+0+2+0+0+(1/4)$

$= (10.25)_{10}$

Binary Numbers and Codes

Decimal to Binary Conversion:

Repeated division method.

$(25)_{10} = (?)_2$

LSB

25/2 = 12 + Remainder of 1.

12/2 = 6 + Remainder of 0.

6/2 = 3 + Remainder of 0.

3/2 = 1+ Remainder of 1.

1/2 = 0 +Remainder of 1.

MSB

1 1 0 0 1

Thus, $(25)_{10} = (11001)_2$.

Binary Numbers and Codes

Octal number system

- Base of eight mean, it has eight possible digits: 0,1,2,3,4,5,6 & 7.

Octal to Binary conversion:

Octal digit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Binar.equiv.	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**

$(351)_8 = (?)_2$

$(351)_8 =$	**3**	**5**	**1**
	↓	↓	↓
	011	**101**	**001**

$(351)_8 = (011101001)_2$

Binary to octal conversion:

- It is the reverse of foregoing process.

Binary Numbers and Codes

Binary to octal conversion:

$$(00111101)_2 = (?)_8$$

$$(00111101)_2 = \underbrace{000}_{\downarrow\ 0} \quad \underbrace{111}_{\downarrow\ 7} \quad \underbrace{101}_{\downarrow\ 5}$$

$$(00111101)_2 = (075)_8$$

Octal to Decimal conversion:

$(235)_8 = (?)_{10}$

$(235)_8 = 2x8^2+3x8^1+5x8^0$

$= 2x64+3x8+5x1$

$= 128+24+5$

$= 157.$

Thus, $(235)_8 = (157)_{10}$

Binary Numbers and Codes

$(13.4)_8 = (?)_{10}$

$(13.4)_8 = 1x8^1+3x8^0+4x8^{-1}$

$= 1x8+3x1+(4/8)$

$= 8+3+(1/2)$

$= 11.5$

Thus, $(13.4)_8 = (11.5)_{10}$

Decimal to Octal Conversion:

$(157)_{10} = (?)_8$

(157)/8 = 19+Remainder of 5.

(19)/8 = 2+ Remainder of 3.

2/8 = 0+ Remainder of 2.

2 3 5

Thus, $(157)_{10} = (235)_8$

Binary Numbers and Codes

Hexadecimal Number System:

- It uses base 16 and has 16 possible digit symbols, the digit 0 through 9 plus letters A, B, C, D, E & F.
- The digits A to F are equivalent to decimal values 10 through 15.

Hex to Binary conversion:

$$(3A5)_{16} = (?)_2$$

$$(3A5)_{16} = \begin{array}{ccc} 3 & A & 5 \\ \downarrow & \downarrow & \downarrow \\ 0011 & 1010 & 0101 \end{array}$$

$$\text{Thus, } (3A5)_{16} = (001110100101)_2$$

Binary to Hex conversion:

$$(01010011110)_2 = (?)_{16}$$

$$\text{Here, } (01010011110)_2 = \begin{array}{ccc} \underbrace{0010} & \underbrace{1001} & \underbrace{1110} \\ \downarrow & \downarrow & \downarrow \\ 2 & 9 & E \end{array}$$

$$\text{Thus, } (01010011110)_2 = (29E)_{16}.$$

Binary Numbers and Codes

Hex to Decimal Conversion:

$(5A3.8)_{16} = (?)_{10}$

$(5A3.8)_{16} = 5X16^2+10X16^1+3X16^0+8X16^{-1}$

$= 5X256+160+3X1+[8/16]$

$= 1280+160+3+[1/2]$

$= 1443.5$

Thus, $(5A3.8)_{16} = (1443.5)10.$

Decimal to Hex Conversion:

$(428)_{10} = (?)_{16}$

$(428)/16 = 26+\text{Remainder of } 12.$ LSD

$(26)/16 = 1+ \text{Remainder of } 10.$

$1/16 = 0 + \text{Remainder of } 1.$ MSD

1 A C

Thus, $(428)_{10} = (1AC)_{16}$.

Binary Coded Decimal (BCD)Code

BCD Code:

- Each decimal digit, 0 through 9 is represented by its four bits binary equivalent.

eg. A decimal number 3059 to BCD.

3	0	5	9	(Decimal)
↓	↓	↓	↓	
0011	0000	0101	1001	(BCD)

BCD to Decimal:

eg. BCD (0111000110001001) to Decimal.

Here,	**0111**	**0001**	**1000**	**1001**
	↓	↓	↓	↓
	7	**1**	**8**	**9**

eg. BCD (001110110101) to Decimal.

Here,	**0011**	**1011**	**0101**
	3	**Forbidden code indicates error in BCD**	**5**

2421 BCD code

9's complement

1's complement

Decimal Equivalent	2	4	2	1
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	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

Thank You...,

2:40 / 2:44

BCD code

Decimal	7 4 2 1	5 4 2 1	3 3 2 1	8 4 $\bar{2}$ $\bar{1}$	7 4 $\bar{2}$ $\bar{1}$
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1	0 1 1 1	0 1 1 1
2	0 0 1 0	0 0 1 0	0 0 1 0	0 1 1 0	0 1 1 0
3	0 0 1 1	0 0 1 1	0 0 1 1	0 1 0 1	0 1 0 1
4	0 1 0 0	0 1 0 0	0 1 0 1	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 0 0	1 0 1 0	1 0 1 1	1 0 1 0
6	0 1 1 0	1 0 0 1	1 1 0 0	1 0 1 0	1 0 0 1
7	1 0 0 0	1 0 1 0	1 1 0 1	1 0 0 1	1 0 0 0
8	1 0 0 1	1 0 1 1	1 1 1 0	1 0 0 0	1 1 1 1
9	1 0 1 0	1 1 0 0	1 1 1 1	1 1 1 1	1 1 1 0

0

Error Detection Codes

- An error detection code used to detect errors during transmission.
- The detected error can't be corrected but it's presence is indicated.
- A parity bit is an extra bit included with message to make total number of 1's either odd or even.

Message	P(Odd)	Message	P(Even)
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

Reflected Code(Gray-Code)

Gray Code

- It becomes convenient to use reflected code to represent the digital data converted from analog data.
- The reflected code changes by one bit as it proceeds from one number to the next.
- A typical application , Analog data represented by Continues change of shaft Position.

Decimal	Binary Code	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Alphanumeric Codes

Alphanumeric Codes

- Used to represents all of various characters (eg. Letter of alphabet, punctuation marks, and other special characters as well as numbers) and function that found on computer keyboard.

A complete alphanumeric code includes:

- 26 lowercase letters.
- 26 uppercase letters.
- 10 numeric digits.
- 7 punctuation marks.
- 20 to 40 other characters such as +, /, %, *, & and so on.

Alphanumeric Codes

ASCII Code

- Most widely used alphanumeric code is the "American Standard Code for Information Interchange (ASCII) Code.
- It is a seven bit code and so it has $2^7 = 128$ possible code groups.
- Used for transfer of alphanumeric information between a computer and external devices such as printer or another computer.

- What actual bit strings would a computer transmit to send message "GOOD" using ASCII with even parity?

Digital Arithmetic Operation

Binary Addition

- 0+0 = 0
- 0+1 = 1
- 1+1 = 10 = 0 + carry of 1 into next position.
- 1+1+1 = 11 = 1+ carry of 1 into next position.

eg. 1100+1101

```
  1100
 +1101
------------------
 11001
```

eg. 101.01

```
  +100.10
------------------
  1001.11
```

Digital Arithmetic Operation

1's Complement Form:

- **1's Complement obtained by changing each 0 to a 1 and each 1 to a 0.**

eg. 1 0 1 0 1 0 1 Original binary number.

↓ ↓ ↓ ↓ ↓ ↓ ↓

0 1 0 1 0 1 0 1's Complement number.

2's Complement Form:

- **2's Complement obtained by adding 1 to least significant bit position to 1's complement number.**

eg. 101101 Original binary number.

010010 1's Complement.

+ 1 Add 1.

010011 2's complement of original binary number.

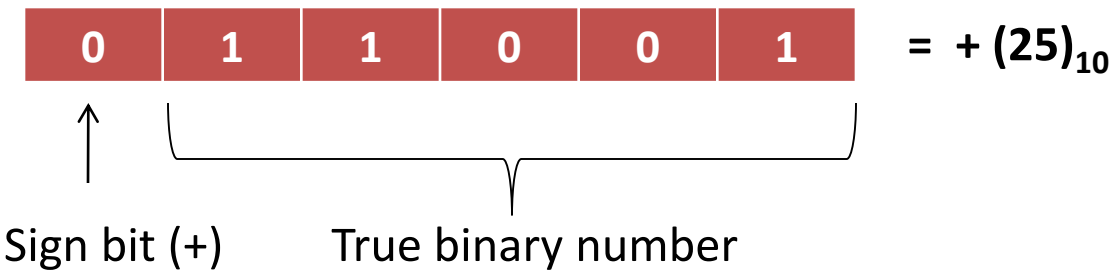
Digital Arithmetic Operation

Signed number representation using 2'S Complement

Rule1

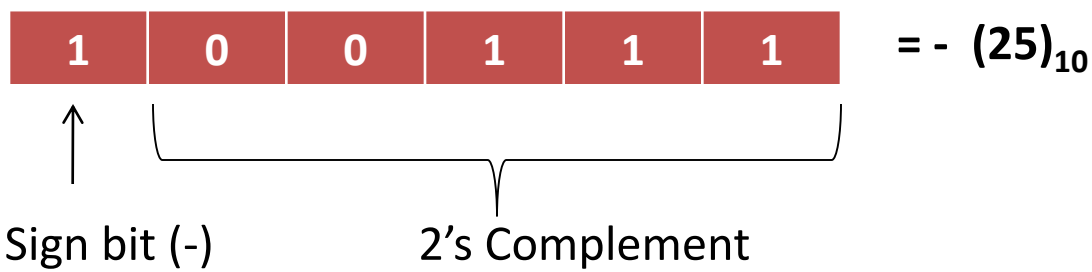
For positive number, the magnitude is represented in its true binary form, and a sign bit 0 attached in front of MSB.

eg.



Rule 2

For negative number, the magnitude must represent in 2's complement form and a sign bit 1 must attach in front of MSB.



Digital Arithmetic Operation

- Illustrate signed number as a signed binary number in 2's complement form.

  + (8) - (8) + (15) - (15) - (12)

Addition in 2's complement system

i) (+9) + (6) ii) (+9) + (-6) iii) (-9) + (+6) iv) (+9) + (-9)

iii) (-9) + (+6)

+(9) = 01001

10110 1's complement

+ 1 (Add 1)

-(9)= 10111 (2's complement of -9)

(-9)	10111
(+6)	00110

(-3)	11101

Digital Arithmetic Operation

iv) (+9) + (-9)

(+9) = 01001

(-9) = 10111

0 = 1(00000)

Disregarded

Binary subtraction

A B	BR	D
0 0	0	0
0 1	1	1
1 0	0	1
1 1	0	0

1001

- 0110

0011

Digital Arithmetic Operation

Subtraction in 2's complement

Step1: Negate the subtrahend.

Step2: Add this to the minuend.

eg. Let minuend = 8 and subtrahend = 5.

Binary Multiplication

0x1 = 0

1x1 = 1

eg. 101 x 101

```
   101
 x 101
-------
   101
  000
 101
-------
 11001
```

Digital Arithmetic Operation

Binary Division

0 /1 = 0

1/1 = 1

```
          101
101) 11001
     101
      10
      101
      101
      000
```

Digital Arithmetic Operation

BCD Addition

Each decimal digit represent it by a four bit code ranging from 0000 to 1001.

i) Sum (< or =) to 9:

eg. 5 (0101)+4 (0100)

5	0101	BCD for 5.
+4	0100	BCD for 4.
9	1001	

ii) Sum [41(0100 0001) + 14(0001 0100)]

41	0100	0001	BCD for 41.
+14	0001	0100	BCD for 14.
55	0101	0101	BCD for 55.

Digital Arithmetic Operation

ii) **Sum > 9:**

The sum must required to corrected by addition of six (0110), in doing so the sum get converted it into valid BCD sum.

eg.

9	1001	
6	0110	
15	1111	Invalid BCD sum.

Add six to convert it into valid sum.

	1111	
	+0110	
0001	0101	Valid BCD sum.

Digital Arithmetic Operation

BCD addition of (36 + 54)

36	0011	0110	
54	0101	0100	
90	1000	1010	Invalid sum.
	1	0110	Add 6.
	1001	0000	Valid BCD sum.

Digital Arithmetic Operation

Hexadecimal Addition

- Insert decimal equivalent for digit >9.
- For sum ≤ 15, then directly represent as hex digit.
- For sum ≥ 16, then Subtract 16 and carry a 1 to next digit position.

Decimal	Hexadec	Decimal	Hexadec	Decimal	Hexadec
0	0	16	10	32	20
1	1	17	11	33	21
2	2	18	12	34	22
3	3	19	13	35	23
4	4	20	14	36	24
5	5	21	15	37	25
6	6	22	16	38	26
7	7	23	17	39	27
8	8	24	18	40	28
9	9	25	19	41	29
10	A	26	1A	42	2A
11	B	27	1B	43	2B
12	C	28	1C	44	2C
13	D	29	1D	45	2D
14	E	30	1E	46	2E
15	F	31	1F	47	2F

Digital Arithmetic Operation

Hex addition of (58+36)

$$\begin{array}{r} 58 \\ 36 \\ \hline 8E \end{array}$$

Hex addition of (5AE+31E)

$$\begin{array}{r} 5AE \\ 31E \\ \hline 8CC \end{array}$$

Hex subtraction

- Take 2's complement of hex subtrahend.
- Add it to minuend.
- Any carry out of MSD position will be disregarded.

Digital Arithmetic Operation

Subtract 3C2 from 781.

Convert subtrahend (3C2) to its 2's complement.

	3C2	
0011	1100	0010
1100	0011	1110
	C 3 E	

	F	F	F
-	3	C	2
	C	3	D
			+ 1
	C	3	E

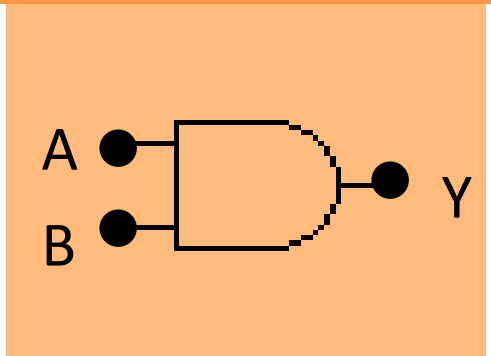
Add it to minuend (781).

```
    7 8 1
  + C 3 E
  -------
  1 3 B F
```

Disregard carry.

Logic gates and Boolean Algebra

- Logic gates constructed by using switches, relays, transistors, diode or Ics.
- Logic gates are building blocks for any digital circuit.

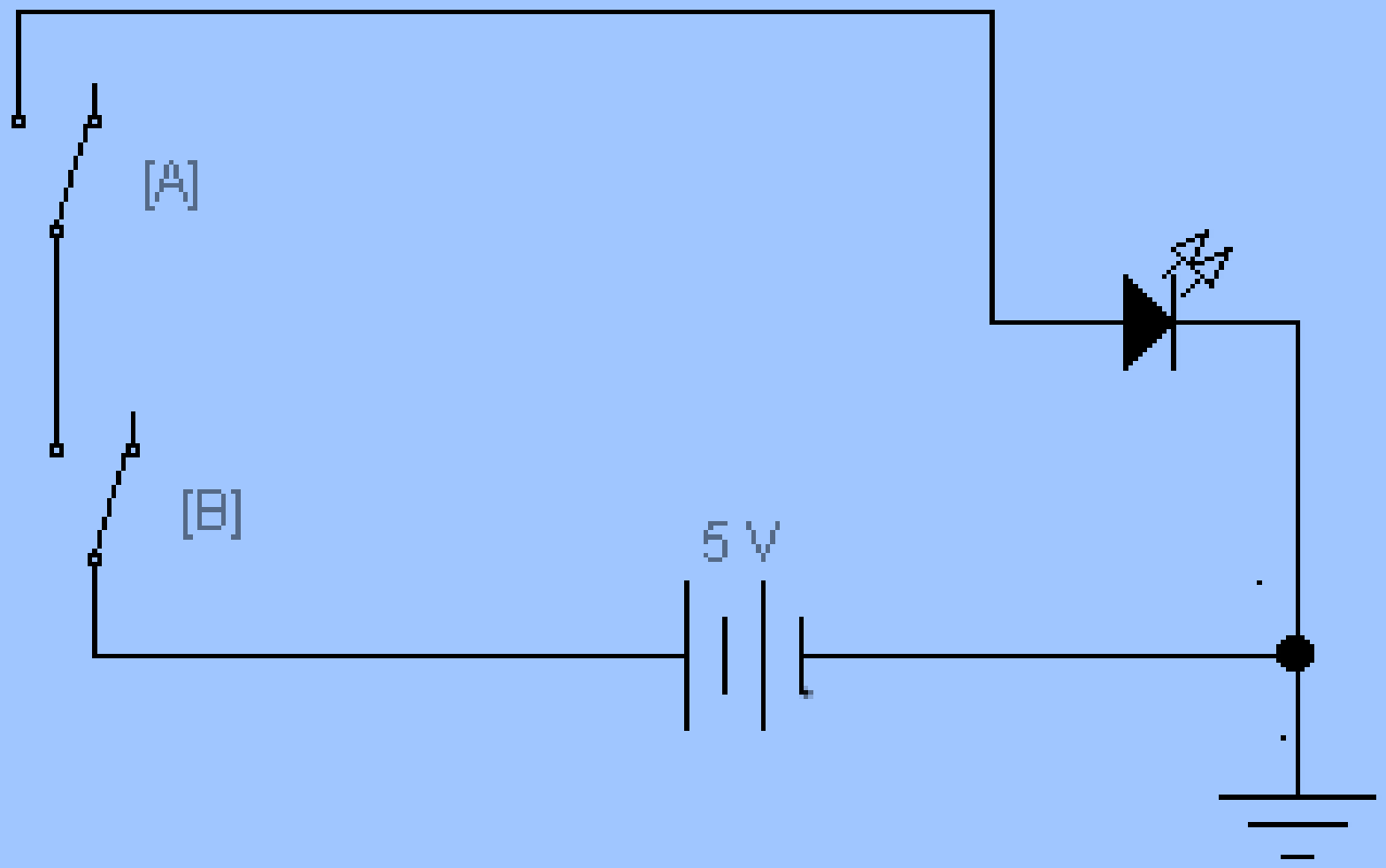


AB	Y=AB
00	0
01	0
10	0
11	1

AND gate

- The symbol for AND gate and their corresponding truth table is shown above.
- Sometimes it is called all or nothing gate.
- Output becomes high, if all inputs are at high.
- Output becomes low, if any one input is at low.

Logic gates and Boolean Algebra



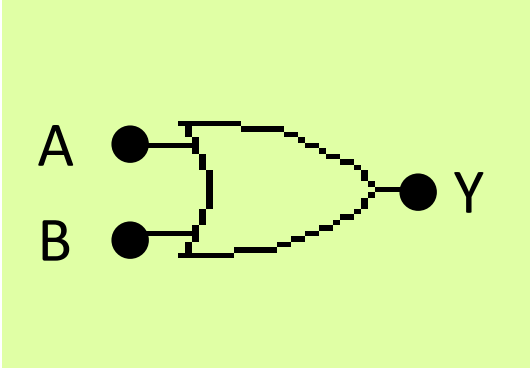
Logic gates and Boolean Algebra

OR Gates:

The logic symbol for OR gate and their corresponding truth table is shown in diagram.

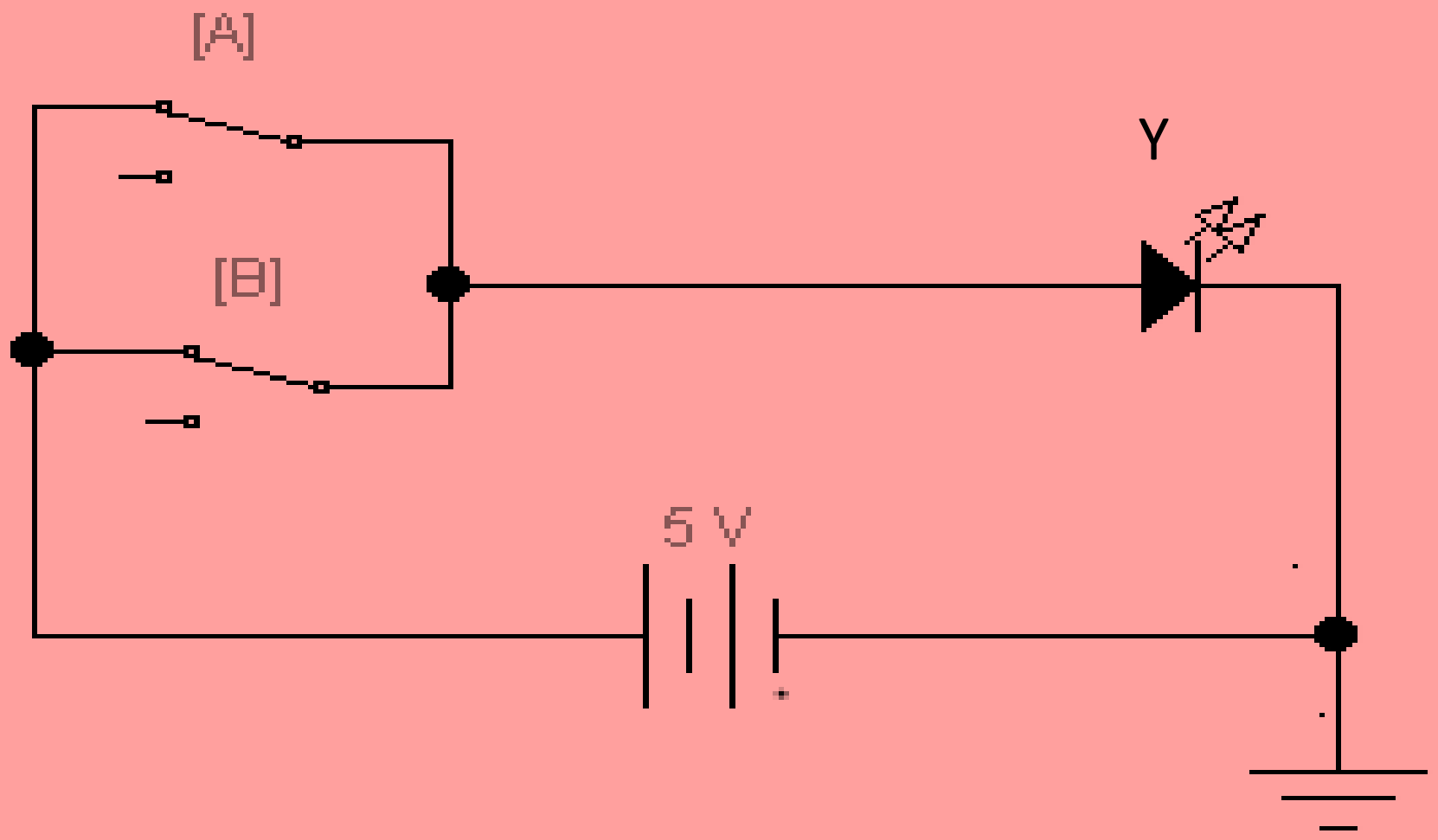
The output becomes high, if at least any one input is at high.

The output becomes low, if all input becomes at low.



AB	Y=A+B
00	0
01	1
10	1
11	1

Logic gates and Boolean Algebra



Logic gates and Boolean Algebra

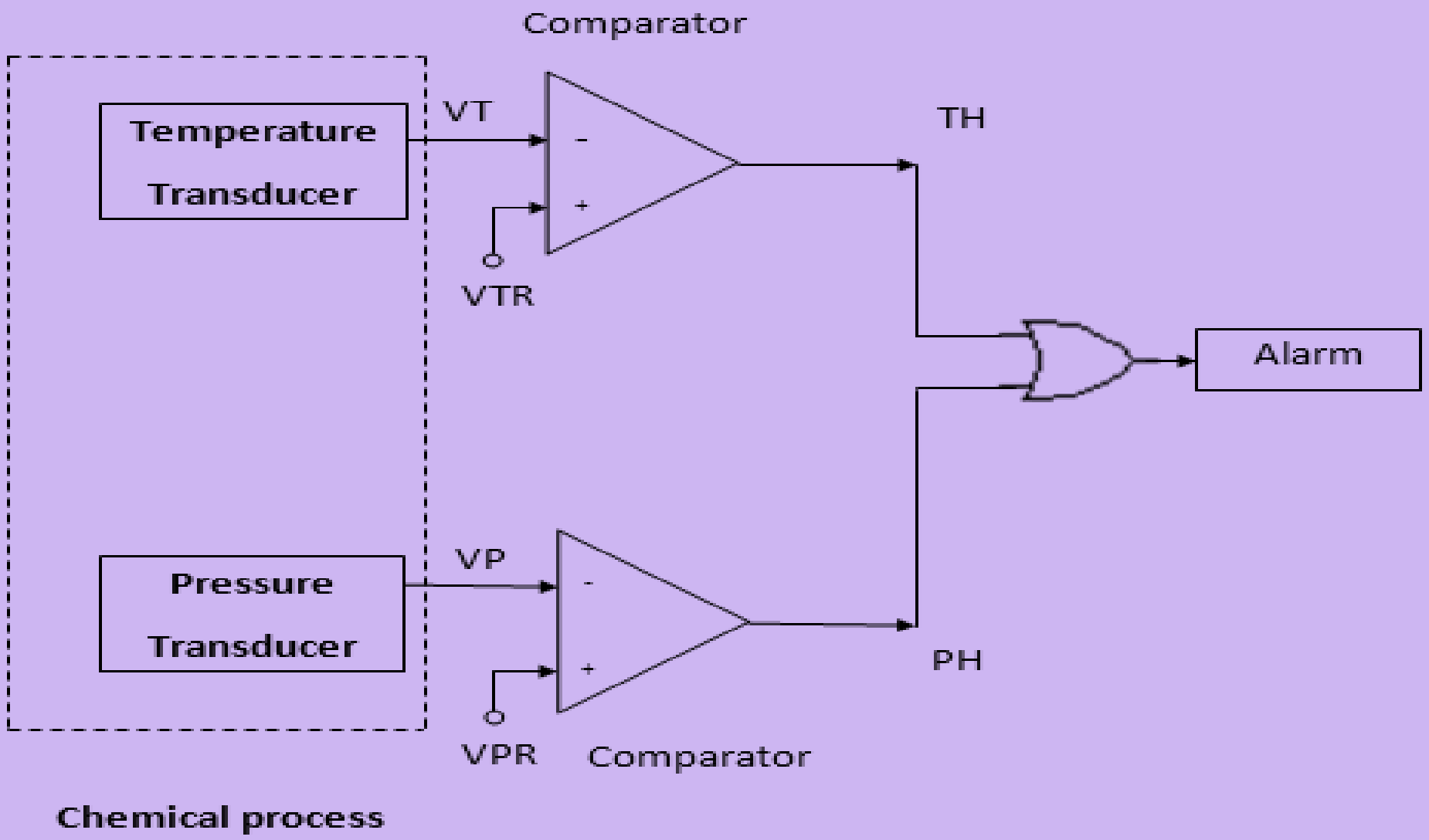


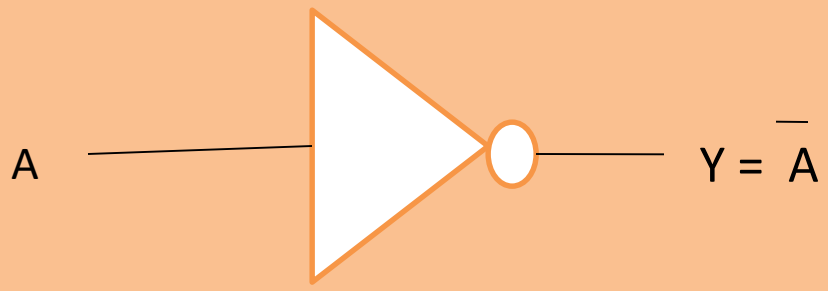
Diagram OR Gate in alarm system

Logic gates and Boolean Algebra

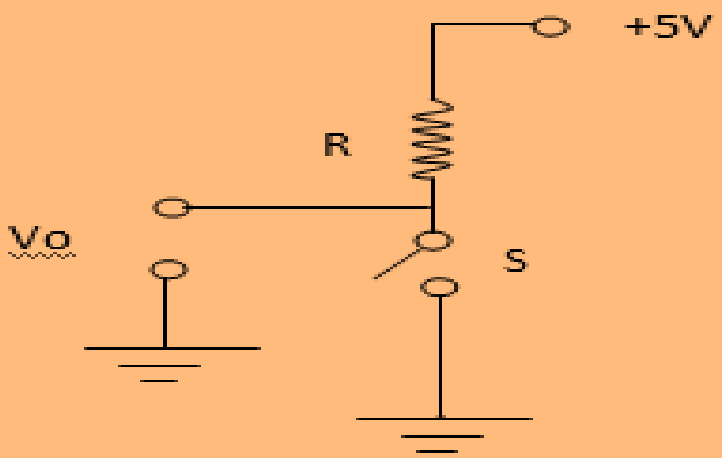
NOT Gate Operation:

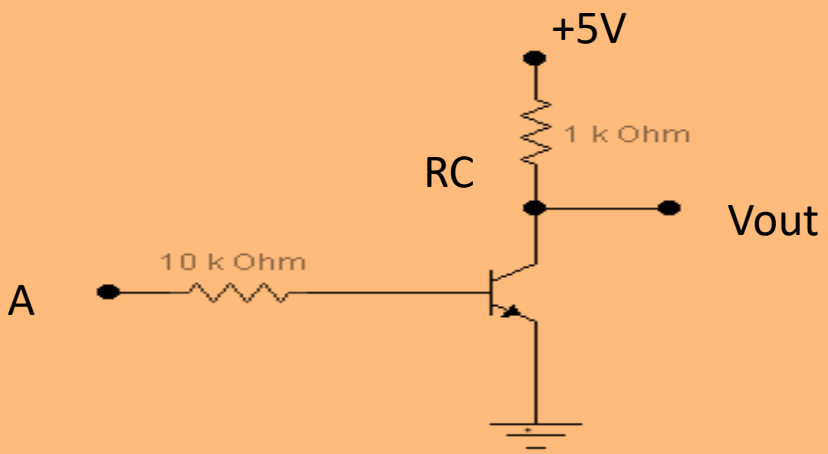
The basic symbol and truth table for NOT gate is as shown in diagram.

A NOT Gate i.e. also called inverter has one input and one output.



A	Y
0	1
1	0

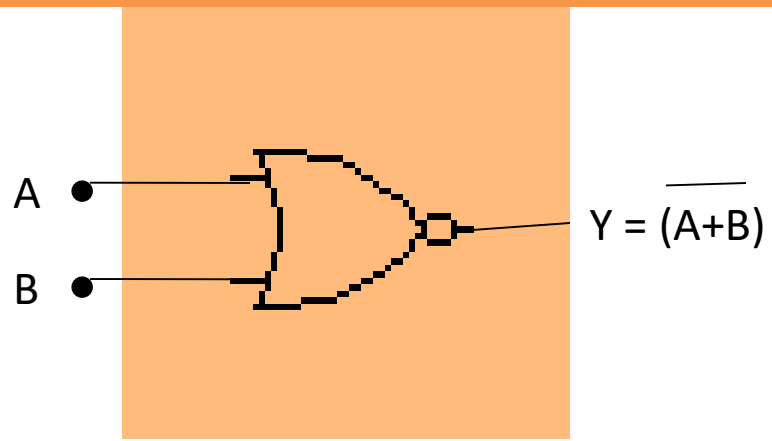




Logic gates and Boolean Algebra

NOR Gate:

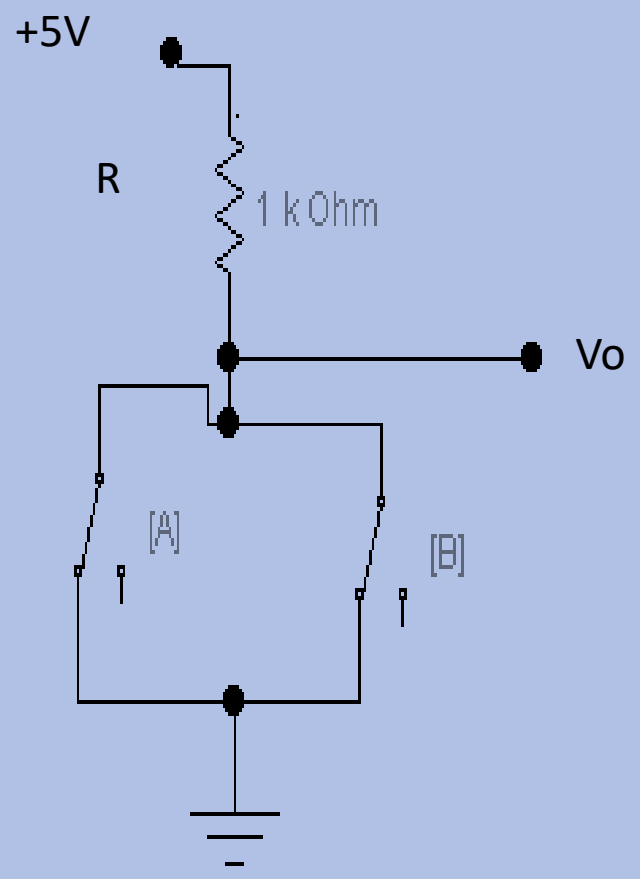
- The NOR gate and it's corresponding truth table is as shown in diagram.
- The small circle at the tip represents inversion operation.

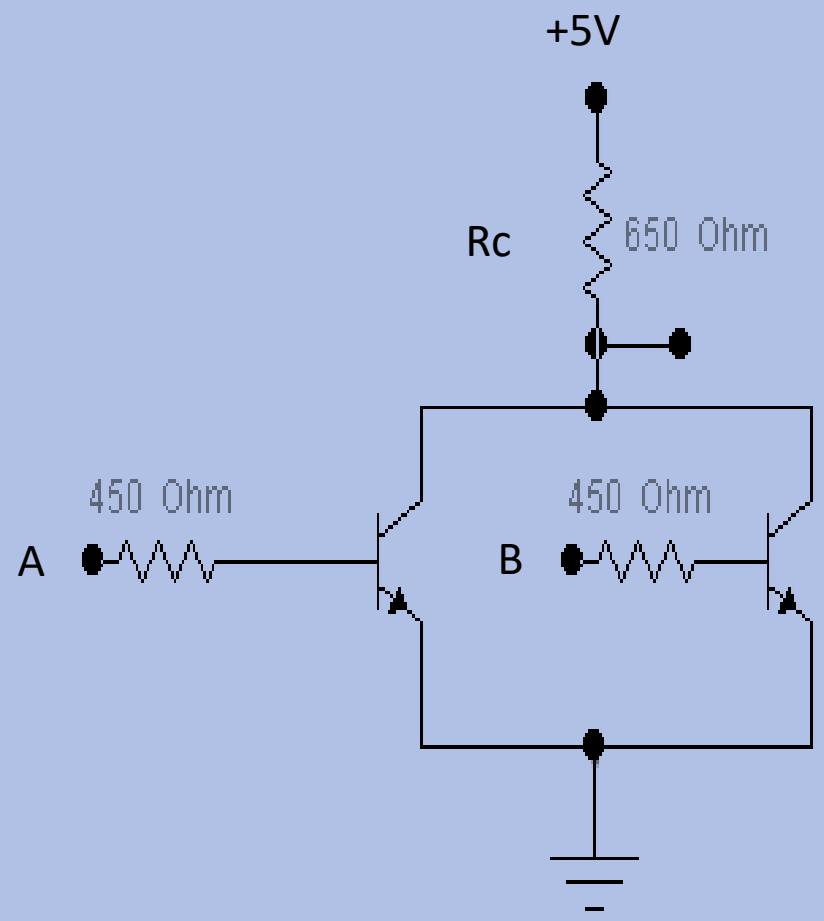


AB	Y
00	1
01	0
10	0
11	0

The NOR Gate equivalent circuit can be realized via several means as shown in diagram.

Logic gates and Boolean Algebra

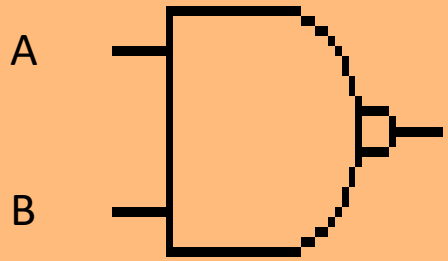




Logic gates and Boolean Algebra

NAND Gate:

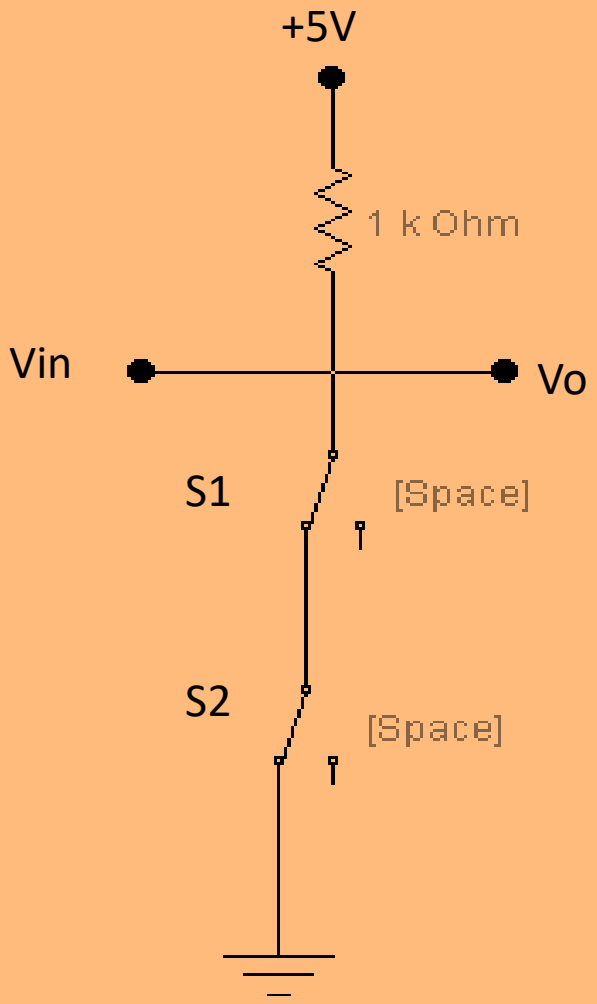
- The NAND Gate and it's corresponding truth table is as shown in diagram.
- The small circle at the tip of the gate represents inversion operation.
- The NAND gate equivalent circuit can be realized via several means as shown in diagram.

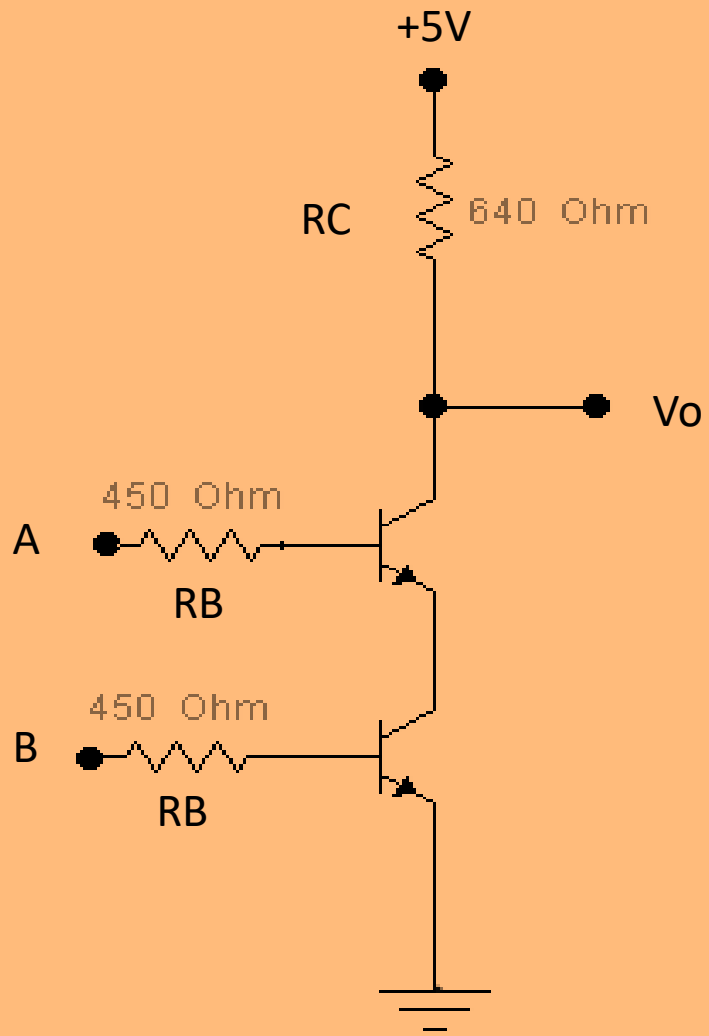


$y = \overline{(AB)}$

AB	Y
00	1
01	1
10	1
11	0

Logic gates and Boolean Algebra

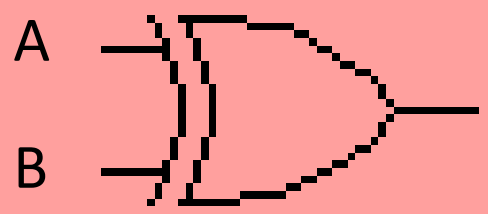




Logic gates and Boolean Algebra

Exclusive-OR gate:

- The logic symbol and corresponding truth table for Exclusive OR gate is as shown in diagram.
- Sometimes it referred to as “any but not all gate”.
- It also represented as ‘XOR’ gate and symbol $\oplus$ indicated in output expression.



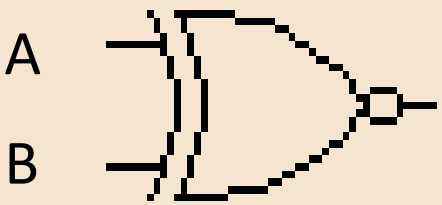
$$Y = A \oplus B$$
$$= \overline{A}\,B + A\,\overline{B}$$

AB	Y
00	0
01	1
10	1
11	0

Logic gates and Boolean Algebra

Exclusive NOR gate:

- The logic symbol and corresponding truth table for Exclusive NOR gate is as shown in diagram.
- The term exclusive NOR gate be often represented as 'XNOR' gate.
- The output of 'XNOR' gate is the complement of 'XOR' truth table, due to circle at tip of 'XOR' gate.



$$Y = A \odot B$$

$$= A\,B + \overline{A}\,\overline{B}$$

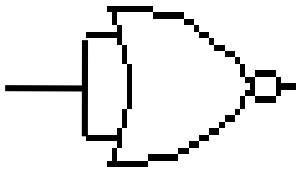
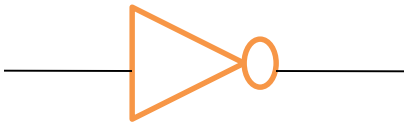
AB	Y
00	1
01	0
10	0
11	1

Logic gates and Boolean Algebra

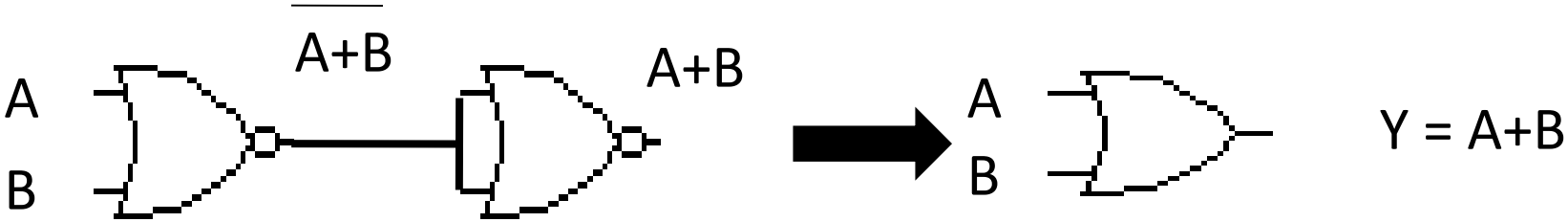
NAND and NOR gate as a universal gate:

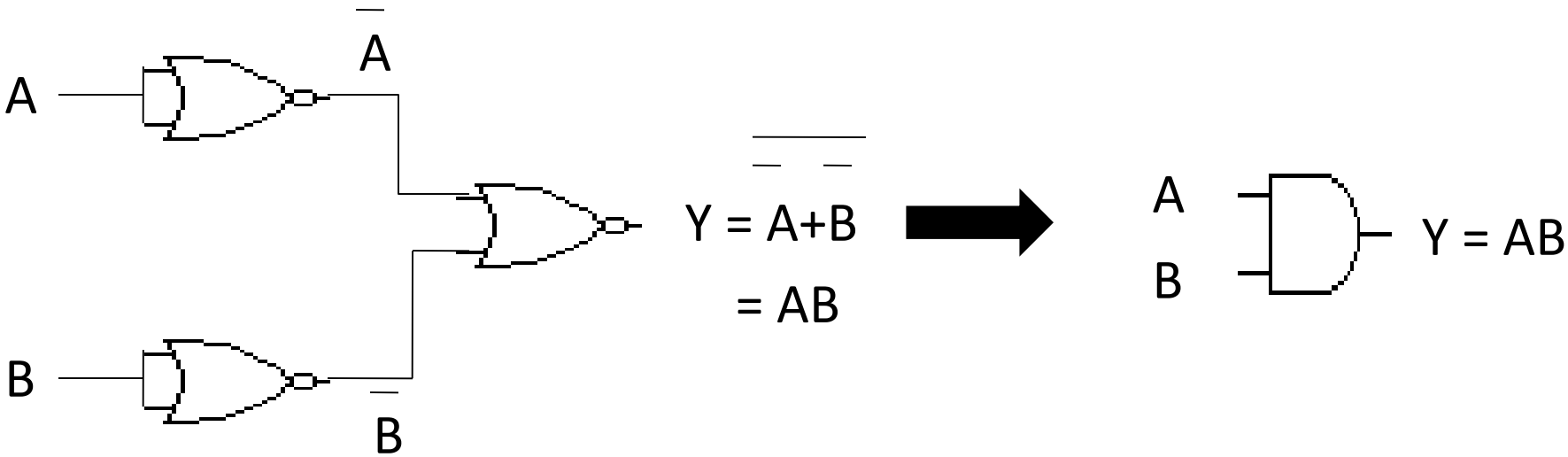
- As AND, OR, NOT gate can be realized using NAND and NOR gate, these gates are called Universal gate.
- The entire logic system can be implemented by using any of these two gates.

NOR gate as Universal gate:

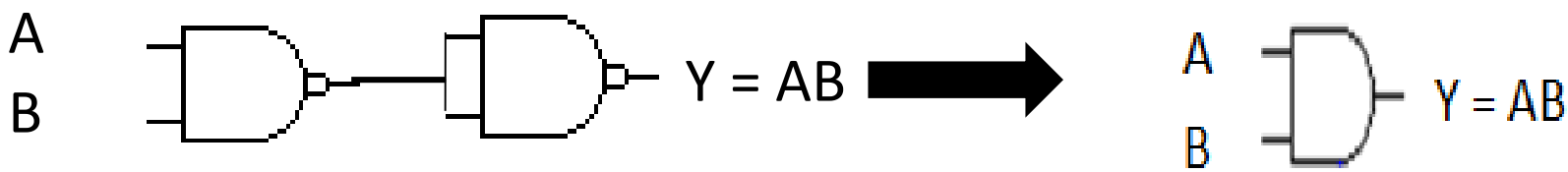
A  $Y = \overline{A + A} = \overline{A}$ ➡  $Y = \overline{A}$

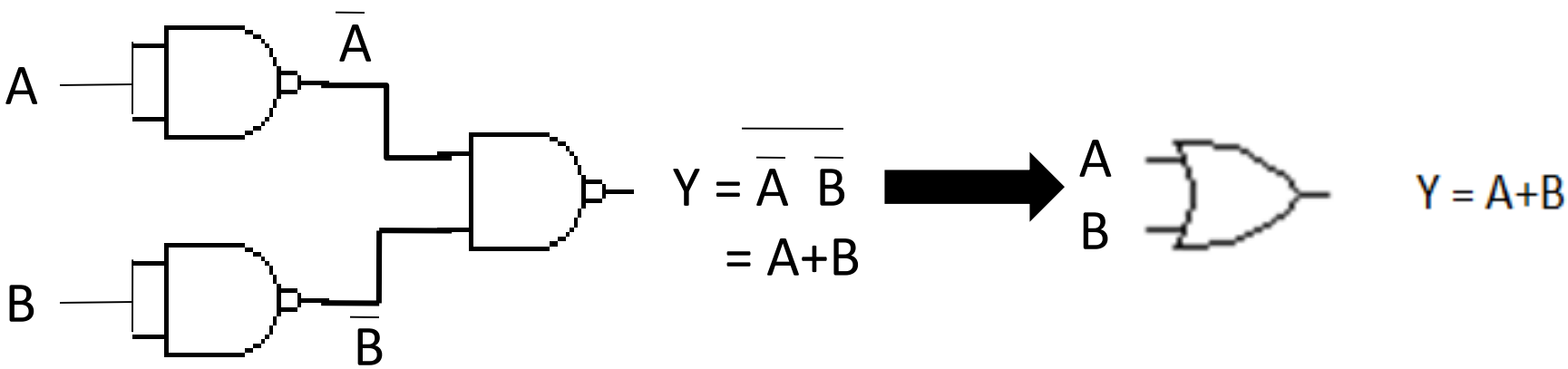
Logic gates and Boolean Algebra





Logic gates and Boolean Algebra





Logic gates and Boolean Algebra

DEMORGAN'S THEOREMS:

De Morgan contributed two important theorem for Boolean algebra.

Theorem1:

$$\overline{A+B} = \overline{A} \cdot \overline{B}$$

Theorem2:

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

Simplify the following expressions using De Morgan's Theorem.

1. $\overline{\overline{ABCD}}$

2. $\overline{\overline{A + B} + \overline{C + D}}$

Logic gates and Boolean Algebra

Min-term and Max-terms:

- In engineering text, sum of products form is called minterm.
- eg. Y= AB+CD+DEFG.
- Similarly, product of sum is called maxterm.
- eg. Y= (A+B) (C+D) (E+G).

ABC	Minterm	Maxterm
000	$\overline{A}\,\overline{B}\,\overline{C}$	$A + B + C$
001	$\overline{A}\,\overline{B}\,C$	$A + B + \overline{C}$
010	$\overline{A}\,B\,\overline{C}$	$A + \overline{B} + C$
011	$\overline{A}\,B\,C$	$A + \overline{B} + \overline{C}$
100	$A\,\overline{B}\,\overline{C}$	$\overline{A} + B + C$
101	$A\,\overline{B}\,C$	$\overline{A} + B + \overline{C}$
110	$A\,B\,\overline{C}$	$\overline{A} + \overline{B} + C$
111	$A\,B\,C$	$\overline{A} + \overline{B} + \overline{C}$

Logic gates and Boolean Algebra

Boolean Algebra Laws:

1. **OR Laws:**

   $A+A = A$

   $A+1 = 1$

   $A+0 = A$

   $A + \overline{A} = 1$

2. **AND Laws:**

   $A.A = A$

   $A.1 = A$

   $A.0 = 0$

   $A.\overline{A} = 0$

3. **Double inversion:**

   $\overline{\overline{A}} = A.$

4. Commutative laws:

$A+B = B+A$

5. Associative laws:

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

$A.(BC) = (AB).C$

$(A+B)+(C+D) = A+B+C+D.$

6. Distributive laws:

$A (B+C) = AB + AC$

$A+BC = (A+B) (A+C)$

$A + \overline{A} B = A+B$

Logic gates and Boolean Algebra

7. **Absorptive laws:**

$A+AB = A$

$A(A+B) = A$

$A(\overline{A} + B) = AB.$

Prove the Boolean expression mentioned below.

- **$(A+B)(A+C) = A+BC$**

LHS = $(A+B)(A+C)$

$= AA+AC+AB+BC$

$= A+AC+AB+BC$

$= A(1+C)+AB+BC$

$= A+AB+BC$

$= A(1+B)+BC = A+BC$ Proved.

Logic gates and Boolean Algebra

- $A + \overline{A}\,B = A+B$

LHS $= A + \overline{A}\,B$

$= A.1 + \overline{A}\,B$

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

$= A + A\,B + \overline{A}\,B$

$= A + B\,(A + \overline{A})$

$= A + B$. Proved

Simplify the expression mentioned below, and draw the logic circuit for simplified expression.

- $\overline{(\overline{A}\,B + A\,\overline{B})} + \overline{(A\,B + \overline{A}\,\overline{B})} = \overline{(A \oplus B)} + \overline{(A \odot B)}$

- $ABC + \overline{(C\,\overline{D})}\;\overline{(\overline{A}\,\overline{C})}$

- $A\,B\,\overline{C} + A\,B\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + \overline{A}\,B\,C$

Logic gates and Boolean Algebra

Design for typical logic circuit:

- Design a logic circuit that has three inputs A, B & C, and whose output will be high only when majority of inputs becomes high. Draw circuit for simplified expression.

ABC	Y
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

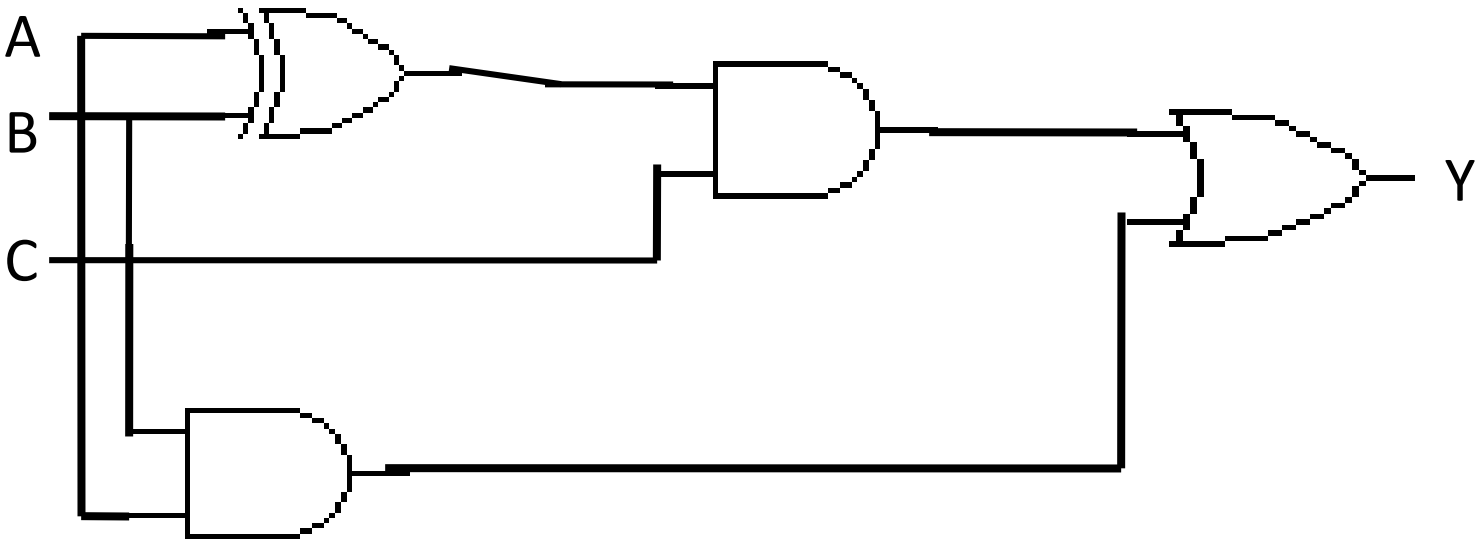
- The output expression can be written as

$$y = \bar{A}\,B\,C + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C\,.$$

$$Y = C\,(\bar{A}\,B + A\,\bar{B}) + A\,B\,(\bar{C} + C).$$

$$Y = C\,(A \oplus B) + A\,B.$$

Logic gates and Boolean Algebra



- Design a logic circuit that has three inputs A, B & C and whose output will be high only when input A becomes high while input B C have different value. Draw circuit for simplified expression.

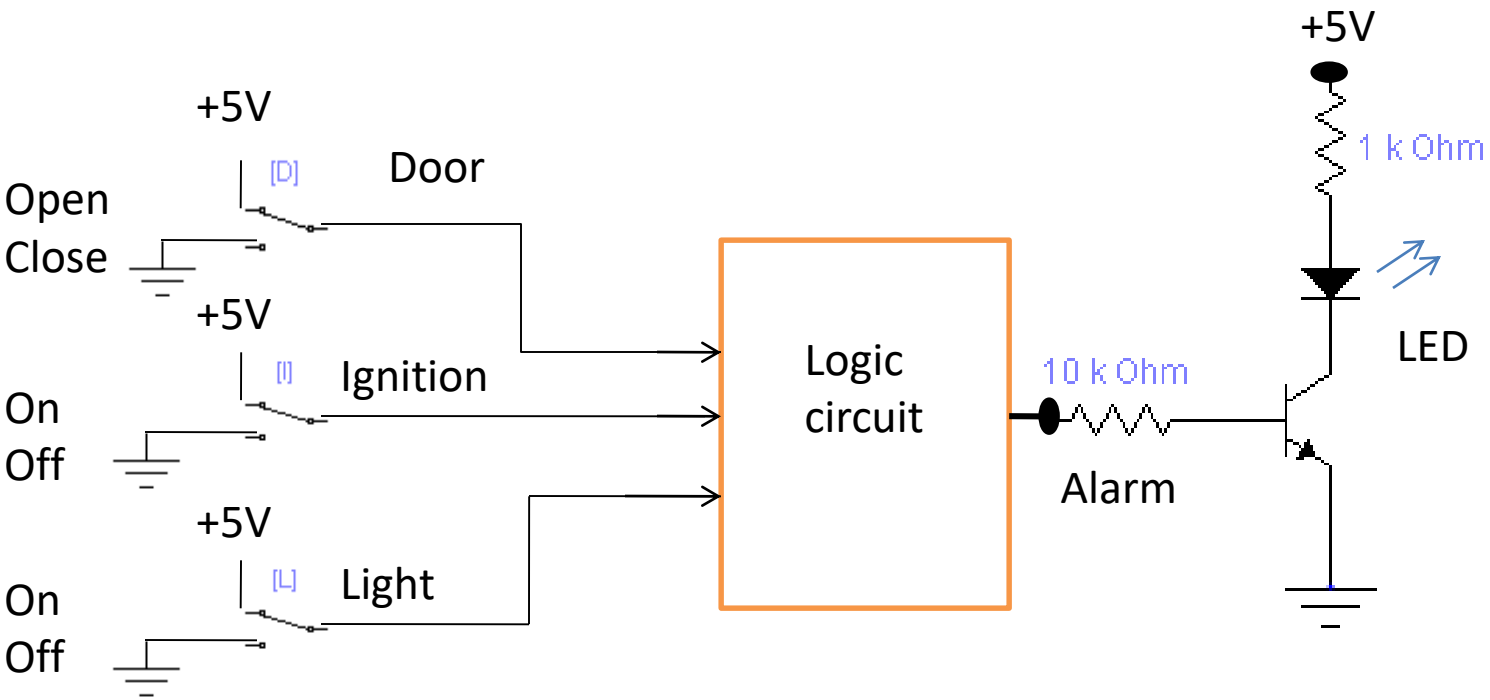
Logic gates and Boolean Algebra

- Design a logic circuit that has four inputs A,B,C & D and the circuit output becomes high only for the input greater than 0010 and less than 1100. Draw the circuit for simplified expression.
- Design a logic circuit using A, B, C & D inputs, whose output will be high only when the two binary numbers AB and CD are equal in magnitude.
- Design a logic circuit whose output is high whenever A & B are both high as long as C & D are either both low or both high.
- Design a logic circuit for an automobile alarm circuit used to detect certain undesirable conditions. The three switches are used to indicate the status of the door by the driver's seat, the ignition and the headlights respectively. These three switches acts as inputs so that the alarm would be activated whenever either of the following conditions exists:

  a) The headlight are ON while the ignition is OFF.

  b) The door is open while the ignition is ON.

Logic gates and Boolean Algebra



Logic gates and Boolean Algebra

Karnaugh Map Method:

It is a graphical tool for simplifying logic equation or to convert a truth table to its corresponding logic circuit.

K-Map standard format:

A \ B	0	1
0	0	1
1	2	3

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Logic gates and Boolean Algebra

- Design a logic circuit that has three inputs A, B & C and whose output will be high only when input A becomes high while input B C have different value. Draw circuit for simplified expression.

ABC	Y
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	0

A \ BC	00	01	11	10
0				
1		1		1

$$y = A\,\overline{B}\,C + A\,B\,\overline{C}$$

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

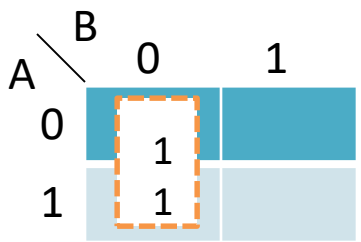
$$Y = A\,(B \oplus C)$$

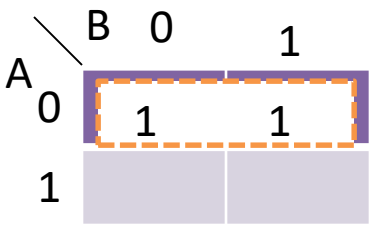
Logic gates and Boolean Algebra

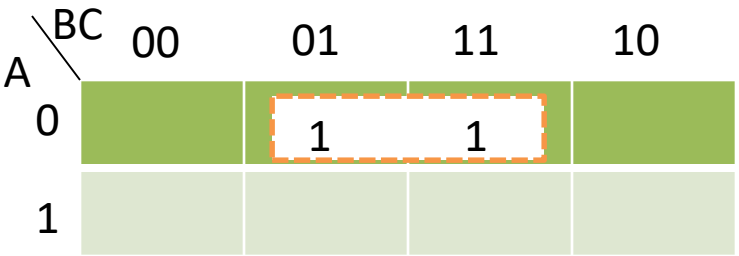
Looping concept:

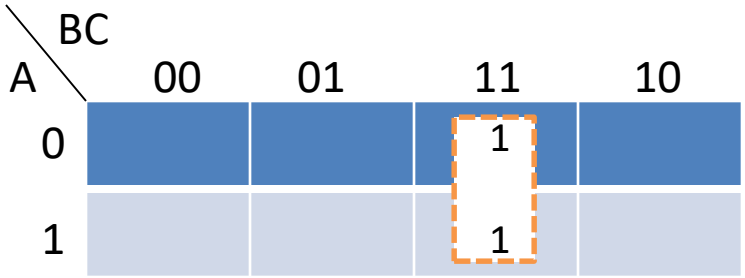
- The process of combining squares that contains 1's in the K-map is called looping.
- Proper looping becomes very important in K-Map to get simplified expression.

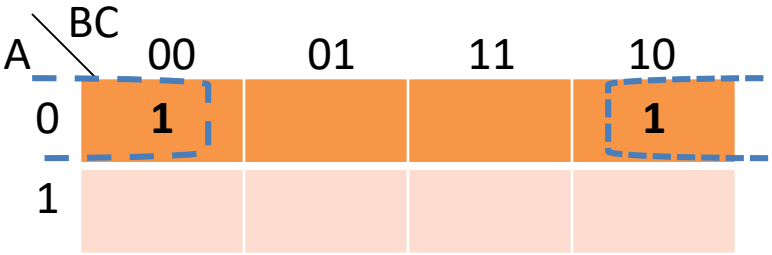
Looping for two:











Logic gates and Boolean Algebra

AB \ CD	00	01	11	10
00				1
01				
11				
10	1	1		1

Looping for four:

AB \ CD	00	01	11	10
00			1	
01			1	
11			1	
10			1	

AB \ CD	00	01	11	10
00				
01	1	1	1	1
11				
10				

Logic gates and Boolean Algebra

AB \ CD	00	01	11	10
00	1			**1**
01				
11				
10	1			1

AB \ CD	00	01	11	10
00				
01		1	1	
11		1	1	
10				

AB \ CD	00	01	11	10
00				
01	1			1
11	1			1
10				

AB \ CD	00	01	11	10
00			**1**	**1**
01				
11				
10			1	1

Logic gates and Boolean Algebra

Looping for Eight:

AB \ CD	00	01	11	10
00		**1**	**1**	
01		1	1	
11		1	1	
10		1	1	

AB \ CD	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

AB \ CD	00	01	11	10
00	**1**			**1**
01	1			1
11	1			1
10	1			1

AB \ CD	00	01	11	10
00	**1**	**1**	**1**	**1**
01				
11				
10	1	1	1	1

Logic gates and Boolean Algebra

Simplify the expression mentioned below using K-Map.

- $f(A,B,C) = \sum (1,3,5,7)$.
- $f(A,B,C,D) = \sum (0,2,5,8,11,13,15)$.
- $f(A,B,C,D) = \sum (1,3,6,9,12,13,15)$.
- $f(A,B,C,D) = \sum (0,2,8,10,12,14,15)$.

Using K-Map, Simplify the Boolean expression mentioned below.

- $Y = \bar{A}(B\,C\,\bar{D} + C\,D) + \bar{C}\,\bar{D} + C\,\bar{D}$.

  $Y = \bar{A}\,B\,C\,\bar{D} + \bar{A}\,C\,D + \bar{C}\,\bar{D} + C\,\bar{D}$.

AB \ CD	00	01	11	10
00	**1**		**1**	**1**
01	1		1	1
11	1			1
10	1			1

$Y = \bar{D} + \bar{A}C$.

Logic gates and Boolean Algebra

Simplify the following Boolean function using different variables K-Map.

- $\overline{A}\,B + A\,\overline{B} + \overline{B}\,C$.
- $A\,\overline{B}\,C + \overline{B}\,C + \overline{A}\,C\,\overline{D}$.
- $A\,B\,C\,D + \overline{A}\,\overline{B} + B\,C + \overline{C}\,\overline{D}$.

- $f(A,B,C,D) = \sum(0,2,8,10,12,13,14,15)$.

AB \ CD	00	01	11	10
00	1			1
01				
11	1	1	1	1
10	1			1

$$Y = AB + \overline{B}\,\overline{D}$$

Logic gates and Boolean Algebra

Simplify the expression mentioned below using K-Map.

$f(A\ B\ C\ D) = \sum (0,1,2,3,7,8,9,10,11,15)$.

AB \ CD	00	01	11	10
00	**1**	**1**	**1**	**1**
01			1	
11			1	
10	1	1	1	1

$$Y = \overline{B} + CD$$

Logic gates and Boolean Algebra

Find SOP and POS for four variable function mentioned below.

$f(A,B,C,D) = \sum (1,3,5,10,11,12,13,14,15)$.

$f(A,B,C,D) = \prod (0,2,4,6,7,8,9)$.

AB \ CD	00	01	11	10
00		1	1	
01		1		
11	1	1	1	1
10			1	1

AB \ CD	00	01	11	10
00	0			0
01	0		0	0
11				
10	0	0		

$f(A,B,C,D) = AB+AC+\bar{A}\,\bar{B}\,D+B\,\bar{C}\,D$.

$f(A,B,C,D) = (A+D)(A+B+C)(A+\bar{B}+\bar{C})$.

Logic gates and Boolean Algebra

Don't care conditions:

- For certain input conditions, when there is no specified output levels, under such circumstances "Don't care'' is used.
- Normally, the symbol "X" is used for "Don't care".

❖ **Design a code converter circuit which convert "BCD to Excess-3 code". Use "don't care" condition for unpredictable output.**

- Since each code uses four bits to represent a decimal digit, there must be four input variables and four output variables.
- Let four input binary variables as A,B,C & D and four output variables as S,T,U & V.

Logic gates and Boolean Algebra

BCD				Excess- 3			
A	B	C	D	S	T	U	V
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
				x	x	x	x
				x	x	X	x
				x	x	x	x
				x	x	x	x
				x	x	x	x
				x	x	x	x

K-Map for S:

AB \ CD	00	01	11	10
00				
01		1	1	1
11	x	x	x	x
10	1	1	x	x

$$S = A + BD + BC$$

Logic gates and Boolean Algebra

Five variable K-Map:

- It consists two copies of four variable map, one of which reflected or flipped horizontally.

AB \ CDE	000	001	011	010	110	111	101	100
00	**0**	**1**	**3**	**2**	**6**	**7**	**5**	**4**
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

Logic gates and Boolean Algebra

Simplify the given function of five variables using K-Map.

$f(A,B,C,D,E) = \sum (0,4,6,7,8, 11,12,16,20,22,23,24,26,27,28,30,31).$

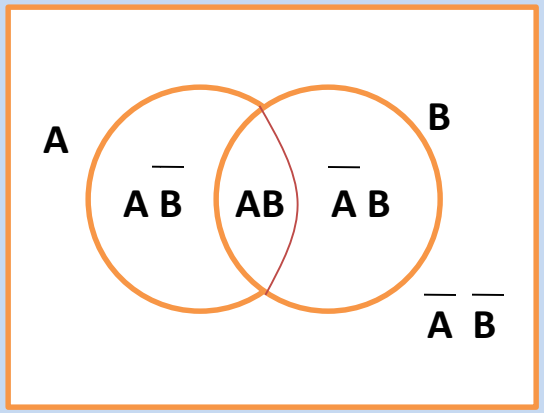
AB \ CDE	000	001	011	010	110	111	101	100
00	1				1	1		1
01	1		1					1
11	1		1	1	1	1		1
10	1				1	1		1

$f(A,B,C,D,E) = \overline{D}\,\overline{E} + A\,B\,D + \overline{B}\,C\,D + B\,\overline{C}\,D\,E\,.$

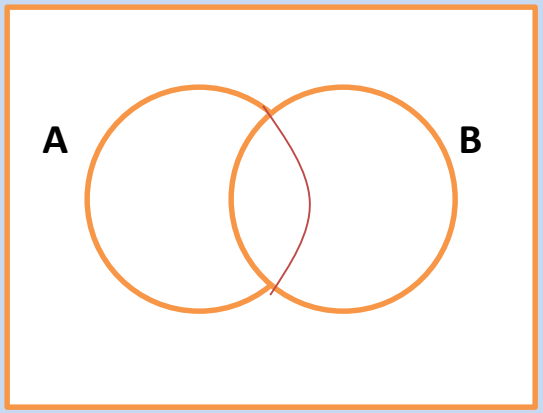
Logic gates and Boolean Algebra

Venn diagram:

- Helpful illustration used to visualize the relationship among variables of Boolean expression is the Venn diagram.
- For A = 1 indicates that one is inside the circle and for A = 0, indicates one outside the circle.



Venn diagram for two variables.



A = (AB+A)

Combinational Logic Circuit

- Combinational circuit output at any instant depend on present combination of inputs without regard to previous inputs.

Figure2.1 Block diagram for Combinational circuit.

- Combinational circuit performs a specific information processing operation fully specified logically by a set of Boolean functions.

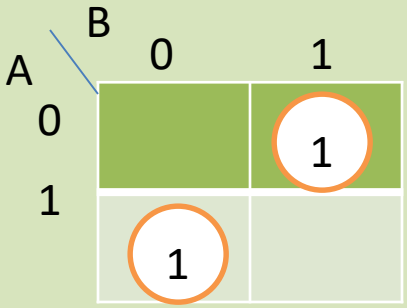
Combinational Logic Circuit

Adders

a) Half adder

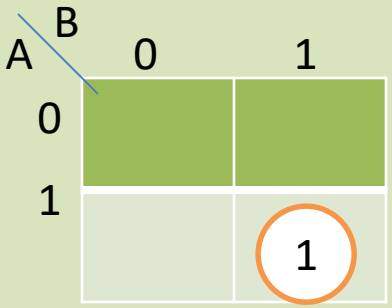
- Combinational circuit that performs addition of two bits is called half adder.
- The input variables designate as augends and addend bits.
- The output variables produce the sum and carry.

AB	C	S
00	0	0
01	0	1
10	0	1
11	1	0



K-Map for Sum.

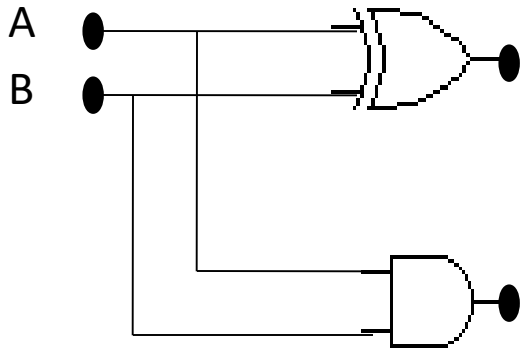
$S = \overline{A}\,B + A\,\overline{B}$



K-Map for Carry.

$C = AB$

Combinational Logic Circuit



$S = \overline{A}\,B + A\,\overline{B}$

$C = AB$

Full Adder

➢A combinational circuit that perform binary addition for three bits.

ABC	C	S
000	0	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	1	1

Combinational Logic Circuit

K-Map for Sum:

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

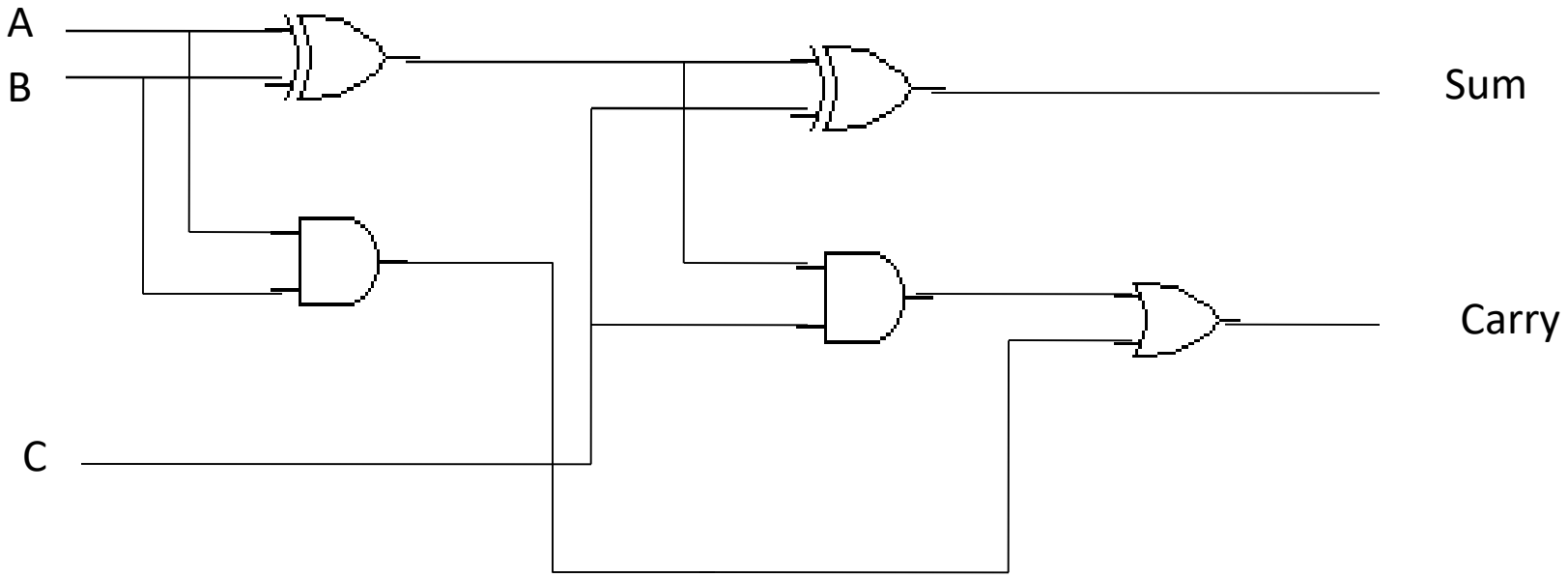
$$S = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C.$$

K-Map for Carry:

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

$$C = A\,C + A\,B + B\,C.$$

Combinational Logic Circuit



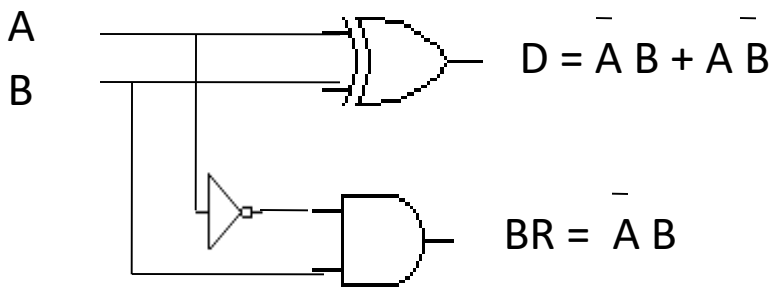
Combinational Logic Circuit

Subtractors

a) Half subtractor:

Perform binary subtraction for two bits and produces their differences.

AB	BR	D
00	0	0
01	1	1
10	0	1
11	0	0



Combinational Logic Circuit

Full subtractor:

ABC	BR	D
000	0	0
001	1	1
010	1	1
011	1	0
100	0	1
101	0	0
110	0	0
111	1	1

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

$$D = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C.$$

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

$$BR = \bar{A}\,B + \bar{A}\,C + B\,C.$$

Combinational Logic Circuit

Code conversion:

- Code convertor circuit makes two systems compatible with each other even though each uses a different binary.
- Sometimes, output of one system use as input to other system.
- Each system uses different code for same information.

4-bit Binary	Gray
A B C D	S T U V
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

Combinational Logic Circuit

AB \ CD	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

$S = A$

AB \ CD	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$T = \bar{A}B + A\bar{B}$

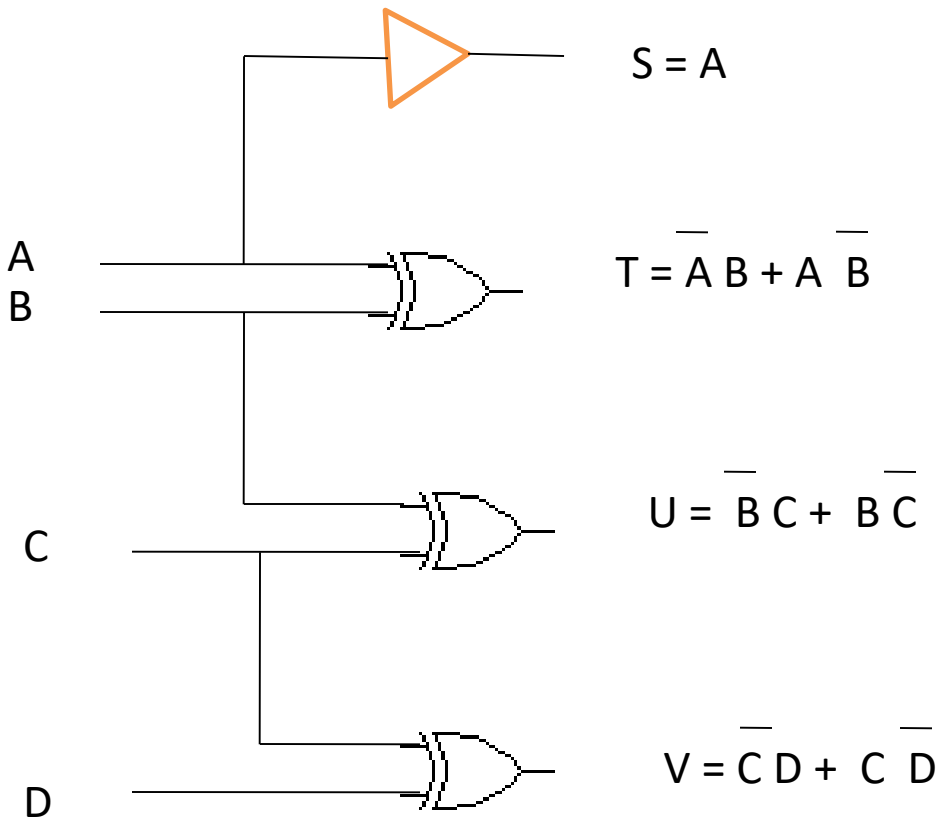
AB \ CD	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

$U = \bar{B}C + B\bar{C}$

AB \ CD	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

$V = \bar{C}D + C\bar{D}$

Combinational Logic Circuit



$S = A$

$T = \overline{A}\,B + A\,\overline{B}$

$U = \overline{B}\,C + B\,\overline{C}$

$V = \overline{C}\,D + C\,\overline{D}$

Combinational Logic Circuit

BCD to Excess -3 Code:

Refer previous slide for don't care condition, already solved.

Parity generation and checking:

- The circuit that generates parity bit at transmitter is called "parity generator" and circuit that checks parity bit at receiver is called a "parity checker".
- Parity bit is an extra bit included with message to make total number of 1's either odd or even.
- An error is detected if checked parity does not correspond with the one transmitted.

Combinational Logic Circuit

3-bit message parity bit:

ABC	P(even)
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

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

$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

$$P = \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C + \bar{A}\,\bar{B}\,C$$

$$P = \bar{C}(\bar{A}\,B + A\,\bar{B}) + C(A\,B + \bar{A}\,\bar{B})$$

$$P = \bar{C}(\bar{A}\,B + A\,\bar{B}) + C\overline{(\bar{A}\,B + A\,\bar{B})}$$

$$P = (A \oplus B) \oplus C$$

Combinational Logic Circuit

Parity checker:

Four bit received parity error check:

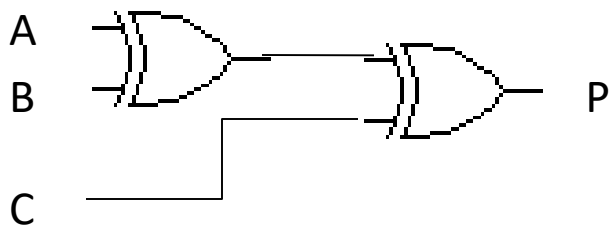
AB \ CP	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

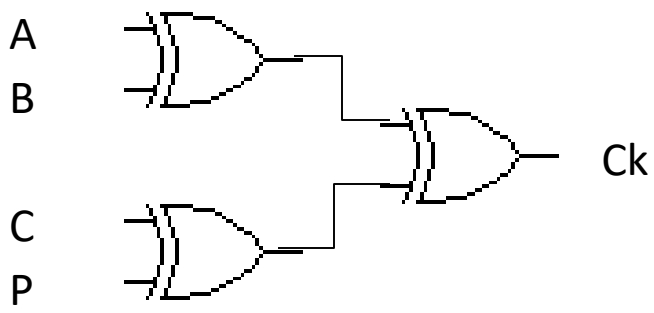
$$Ck = \bar{A}\,\bar{B}\,\bar{C}\,P + \bar{A}\,\bar{B}\,C\,\bar{P} + A\,\bar{B}\,\bar{C}\,\bar{P} + \bar{A}\,B\,C\,P + \bar{A}\,B\,\bar{C}\,P + A\,B\,C\,\bar{P} + A\,\bar{B}\,\bar{C}\,\bar{P} + A\,\bar{B}\,C\,P.$$

$$Ck = A + B + C + P.$$

ABCP	Ck(even)
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	0

Combinational Logic Circuit





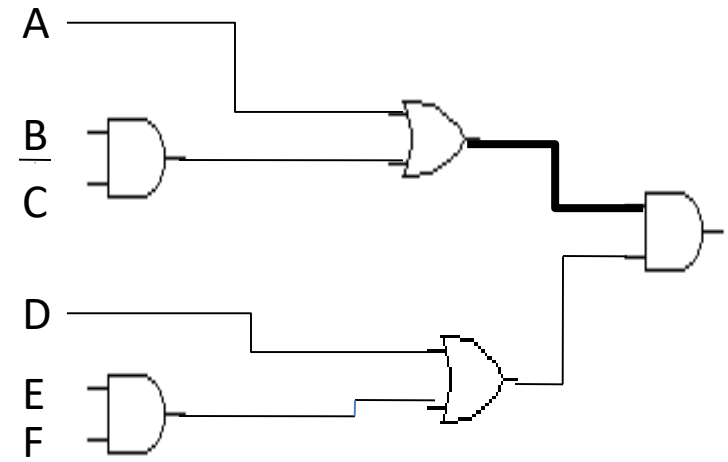
Combinational Logic Circuit

Multilevel NAND Gates:

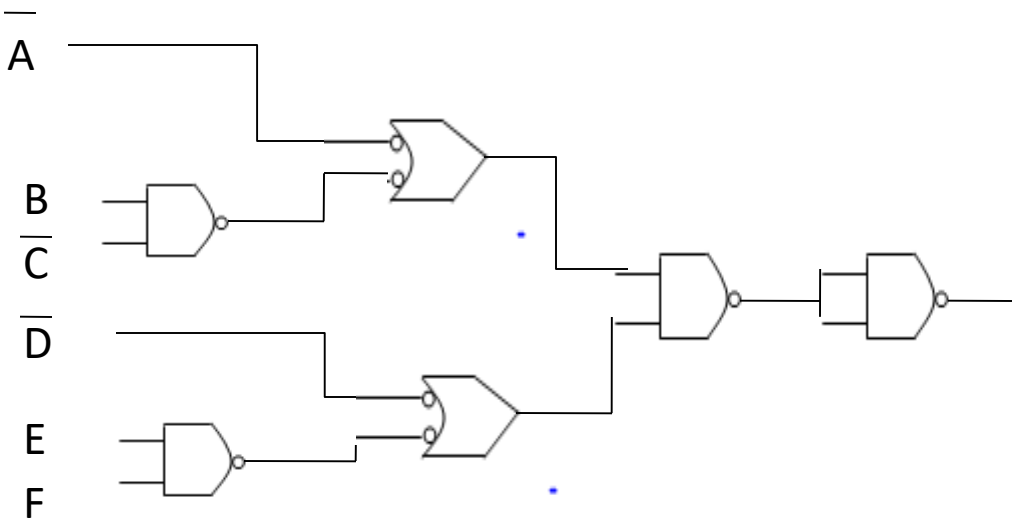
- For given Boolean function, draw logic diagram with AND, OR and NOT gates.
- Convert all AND gates to NAND gates with AND invert graphic symbol.
- Convert all OR gates to NAND gates with invert OR graphic symbols.
- Check all small circles in the diagram.

- eg. $Y = (A + B\,\overline{C})(D + E\,F)$

Combinational Logic Circuit

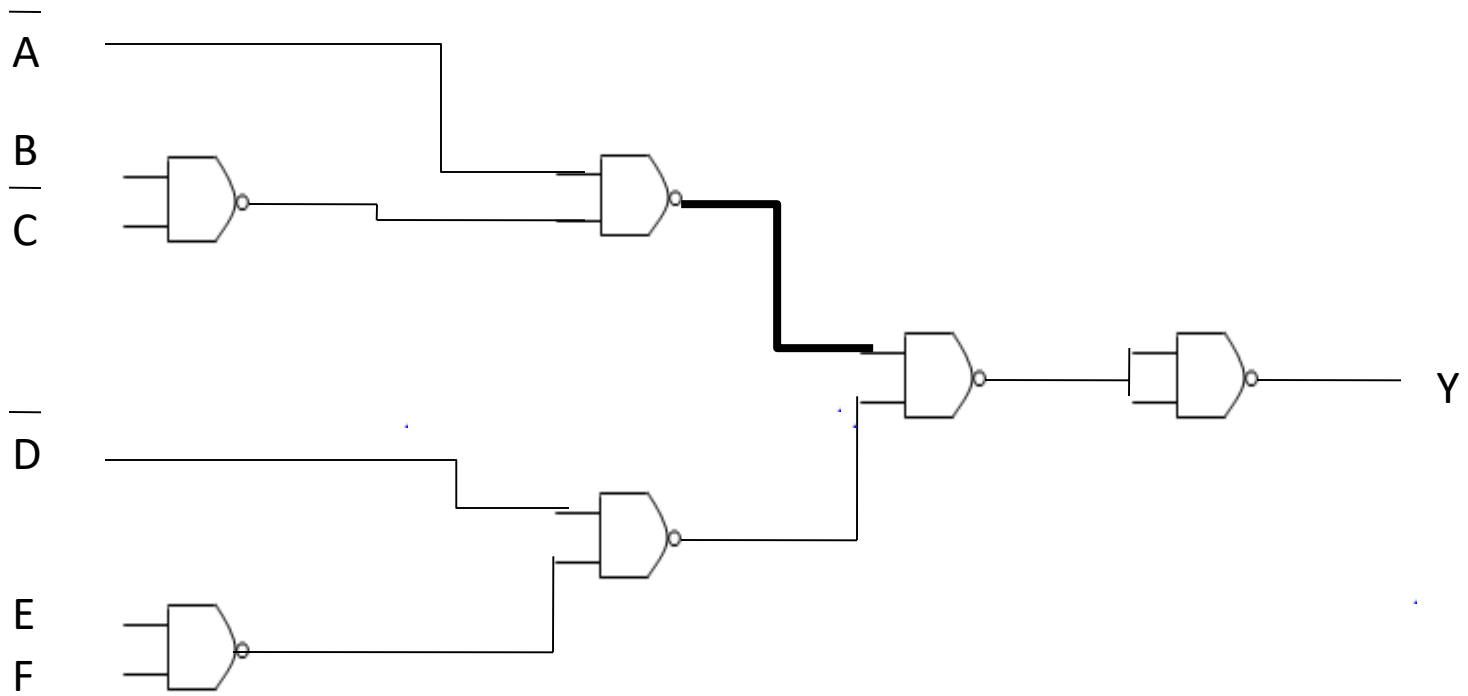


$$Y = (A + B\overline{C})\,(D + E\,F)$$



Combinational Logic Circuit

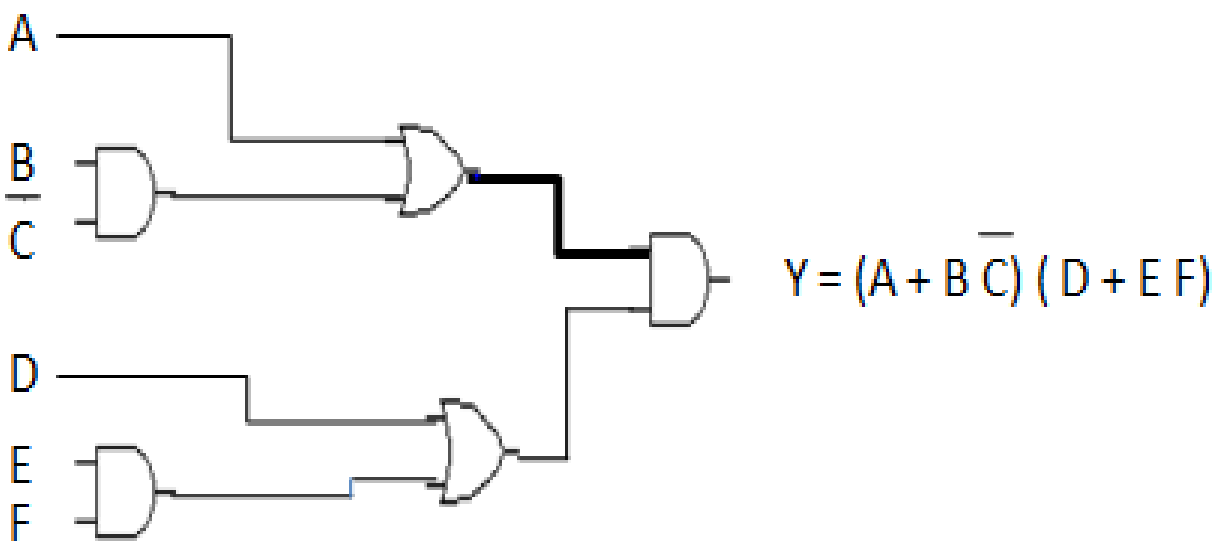
Multilevel NAND gates:



Combinational Logic Circuit

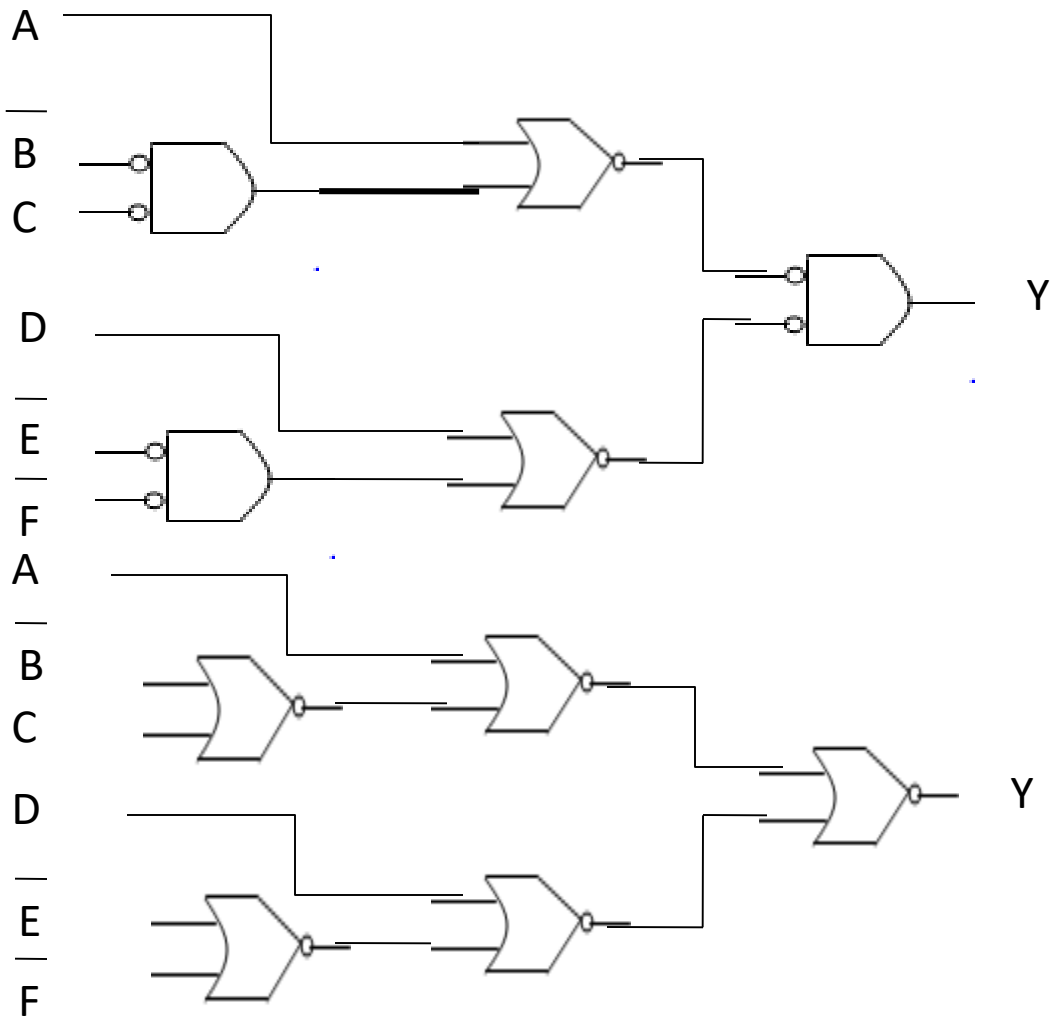
Multilevel NOR gates:

- Draw the AND-OR logic diagram from the given algebraic expression.
- Convert all OR gates to NOR gates with OR-invert graphic symbols.
- Convert all AND gates to NOR gates with invert AND graphic symbols.
- Any small circle that is not compensated by another small circle along the same line needs an inverter.
- eg. $Y = (A+B\overline{C})(D+EF)$.



Combinational Logic Circuit

Multilevel NOR gate:



MSI Combinational Logic Circuit

MSI Components perform specific digital functions commonly needed in the design of digital systems.

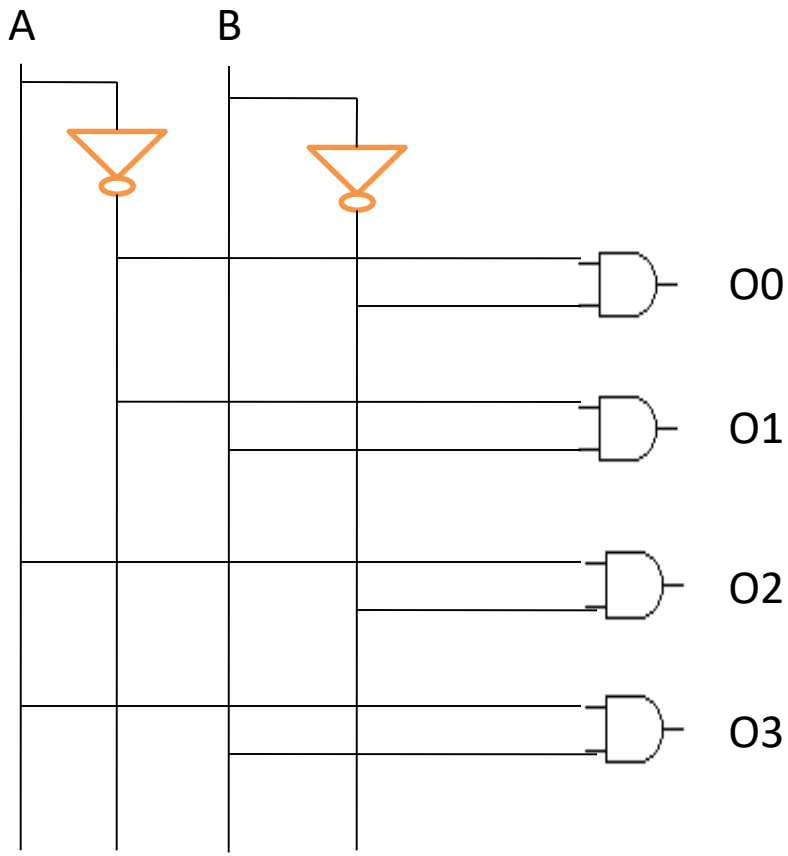
Decoder

- A combinational circuit that converts binary information from n-input lines to a maximum of 2^n unique output lines.

Inputs		Outputs			
A	B	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

MSI Combinational Logic Circuit

Decoder:



MSI Combinational Logic Circuit

- Design a combinational Full adder circuit with a decoder and two OR gates.

Encoder:

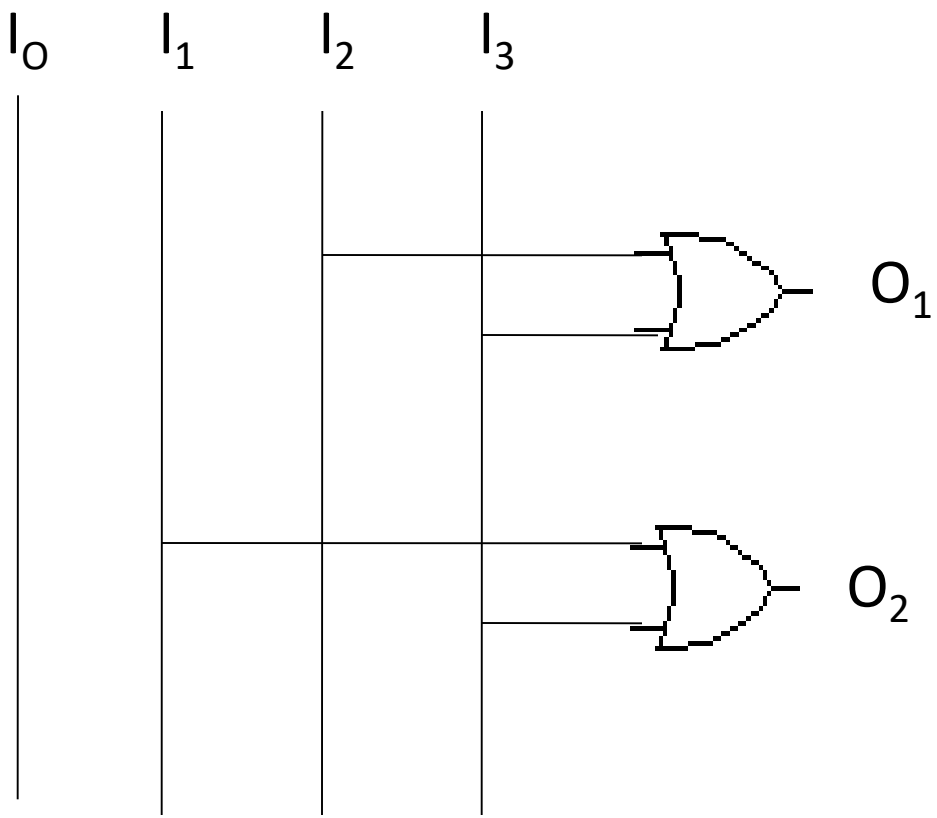
- It performs the inverse operation of a decoder.
- It convert binary information from **2^n** (or fewer)input lines to n-output lines.

Inputs				Outputs	
I0	I1	I2	I3	O1	O2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

MSI Combinational Logic Circuit

O1 = I2 + I3

O2 = I1 + I3



MSI Combinational Logic Circuit

Multiplexer:

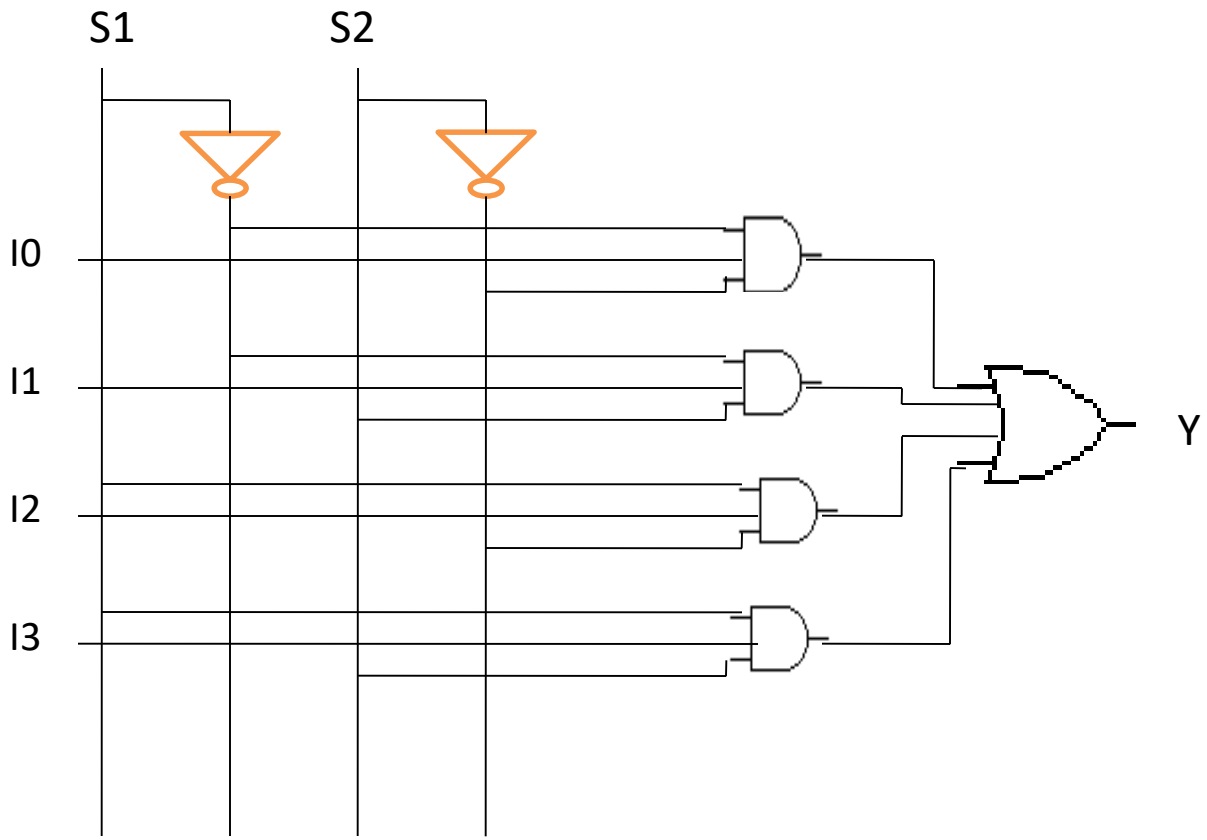
- A combinational circuit that selects binary information from one of many inputs lines and directs it to a single output line.
- The selection of particular input line is controlled by a set of selection lines.
- It transmits a large number of information over a smaller number of channels or lines.
- In Multiplexer, For 2^n input lines there becomes n-selection lines and single output line.

Inputs	Selection lines	Output
2^2	2	1
2^3	3	1
2^4	4	1
.	.	.
2^n	n	1

MSI Combinational Logic Circuit

Multiplexer:

S1	S2	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3



MSI Combinational Logic Circuit

De-multiplexer:

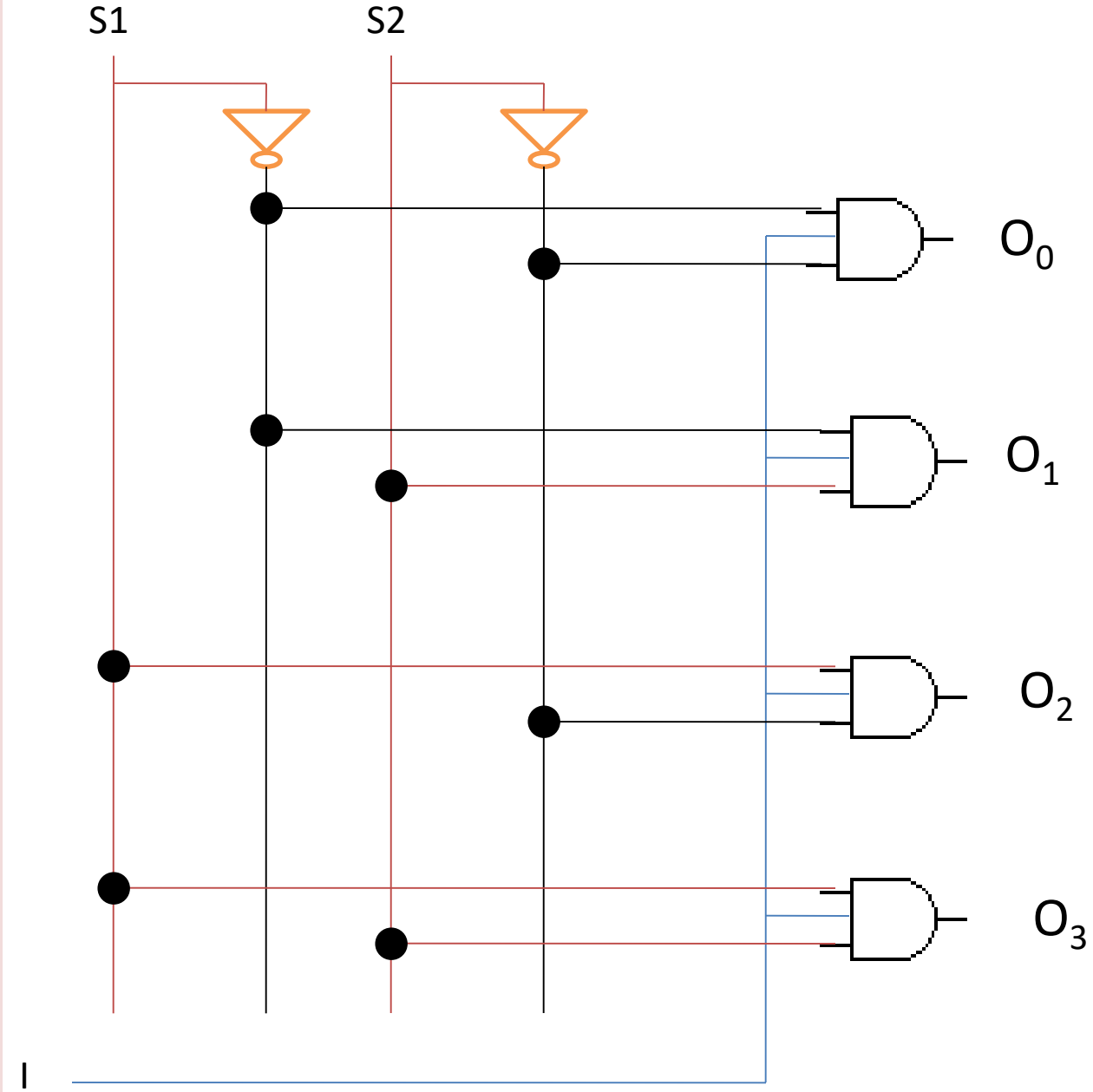
- A combinational circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.
- De-multiplexer performs the inverse operation of Multiplexer.
- The selection of particular output line is controlled by the bit values of n selection lines.
- It transmits single information over a large number of channels or lines.

Input	Selection lines	Outputs
1	2	2^2
1	3	2^3
1	4	2^4
.	.	.
1	n	2^n

MSI Combinational Logic Circuit

De-multiplexer:

S1	S2	Y
0	0	O0
0	1	O1
1	0	O2
1	1	O3



MSI Combinational Logic Circuit

Binary parallel adder:

It produces the arithmetic sum of two binary numbers in parallel.

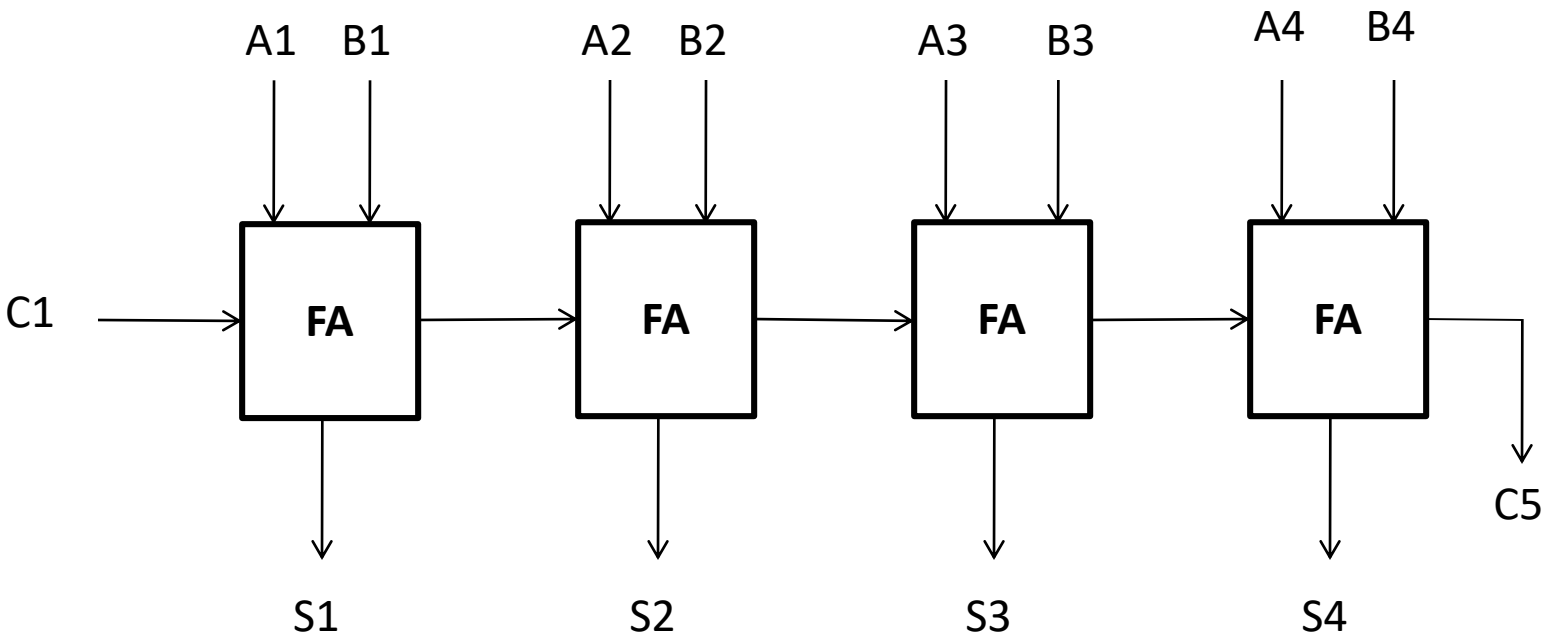


Diagram 4-bit parallel adder.

MSI Combinational Logic Circuit

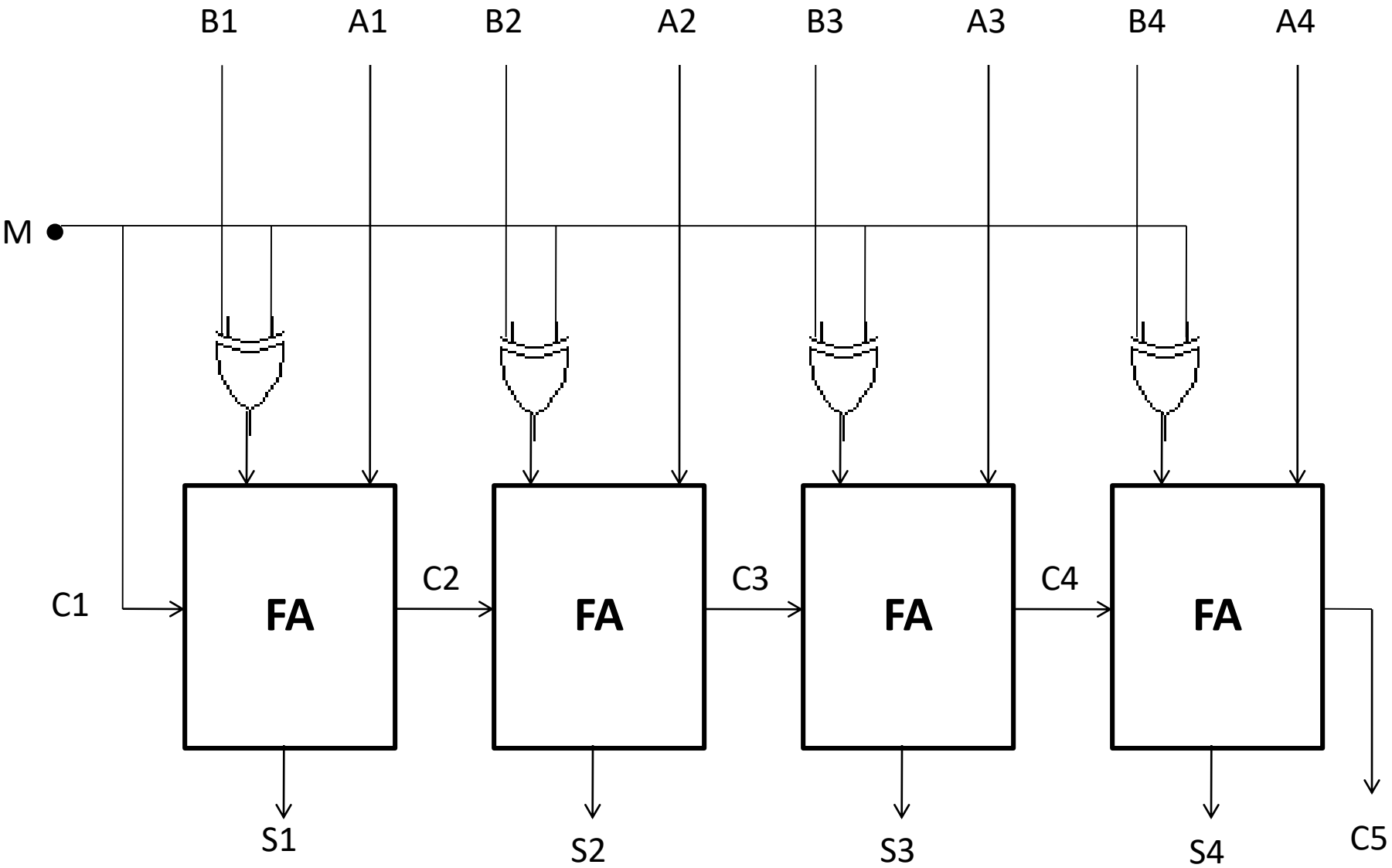
Binary adder – subtractor :

- Design an binary adder/subtractor circuit with one selection variable M and two inputs A and B. For M = 0, the circuit need to perform addition i.e. (A+B) and for M = 1, the circuit must perform subtraction i.e. (A - B) by taking 2's complement of B.

For M = 0, it becomes $B \oplus 0 = B$. The full adder receive the value of B, the input carry is 0, and the circuit performs (A + B).

For M = 1, it becomes $B \oplus 1 = \overline{B}$ and C1 = 1. The input B are complemented and a 1 is added via the input carry. The circuit performs the operation A + 2'S complement of B i.e. (A – B) operation.

MSI Combinational Logic Circuit



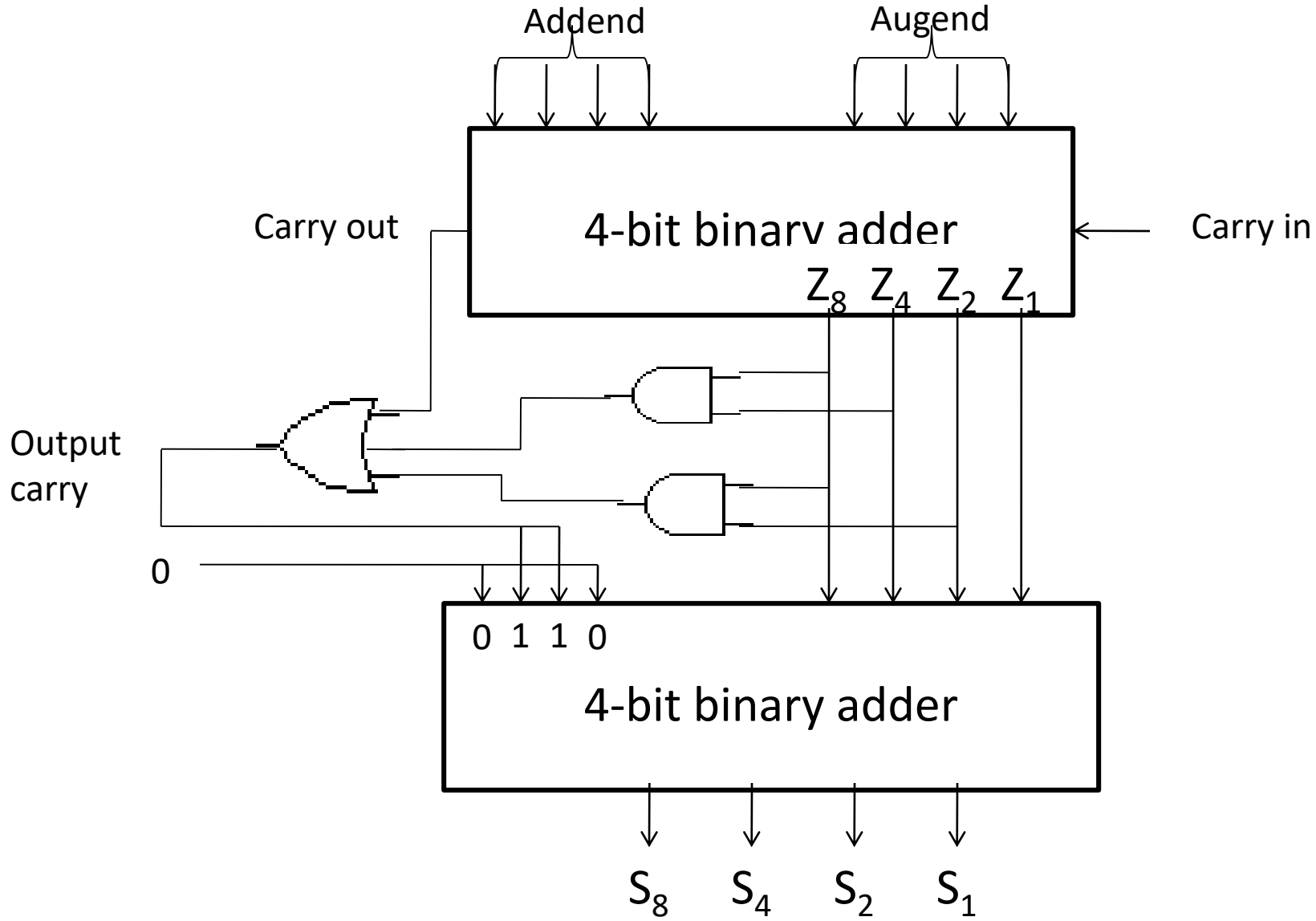
MSI Combinational Logic Circuit

BCD Adder:

- It adds two BCD digits in parallel and produces sum digit also in BCD.
- To add 0110 to binary sum, one can use a second 4-bit binary adder as shown in diagram.
- When output carry becomes equal to zero, nothing is added to the binary sum.
- When output carry becomes equal to one, the binary 0110 is added to the binary sum via bottom 4-bit binary adder.
- The output carry generated from the bottom binary adder can be ignored, since it supplies information already available at output carry terminal.

MSI Combinational Logic Circuit

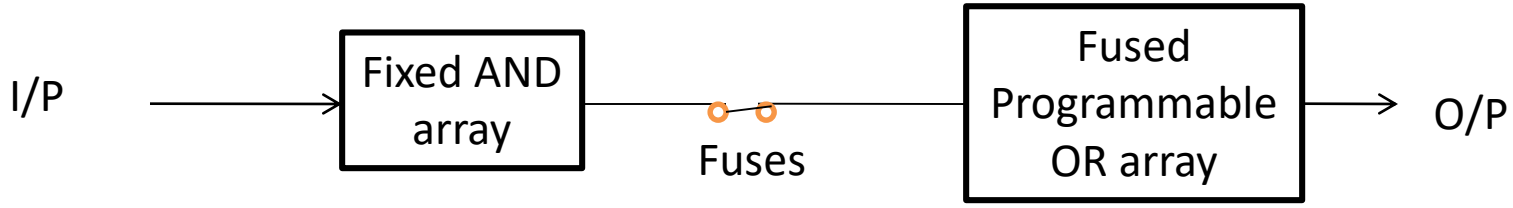
BCD Adder:



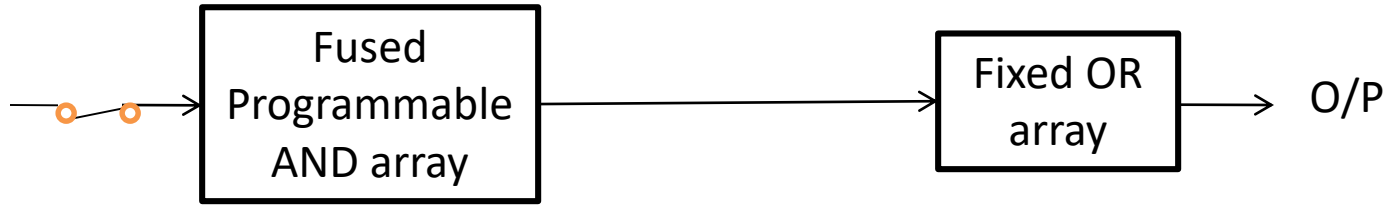
Programmable Logic Device

PLD:

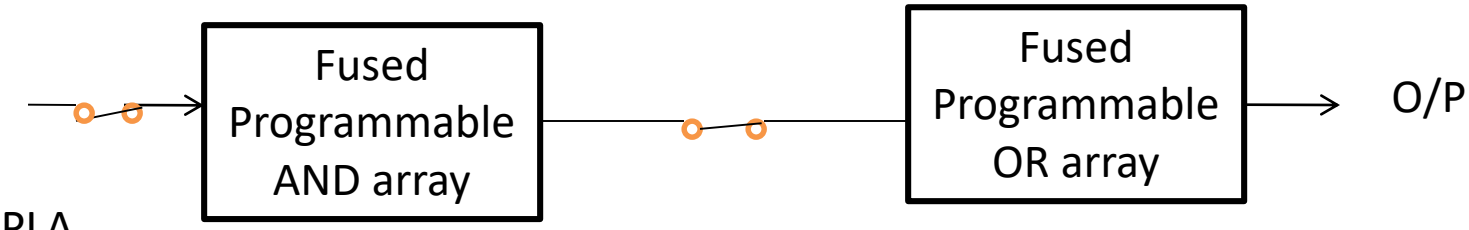
It plays very important role in the design for digital system.



Diag.a PROM



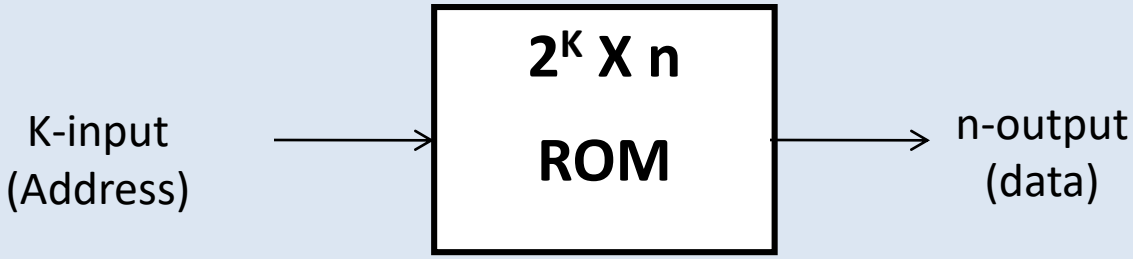
Diag.b PAL



Diag.c PLA

Programmable Logic Device

ROM:

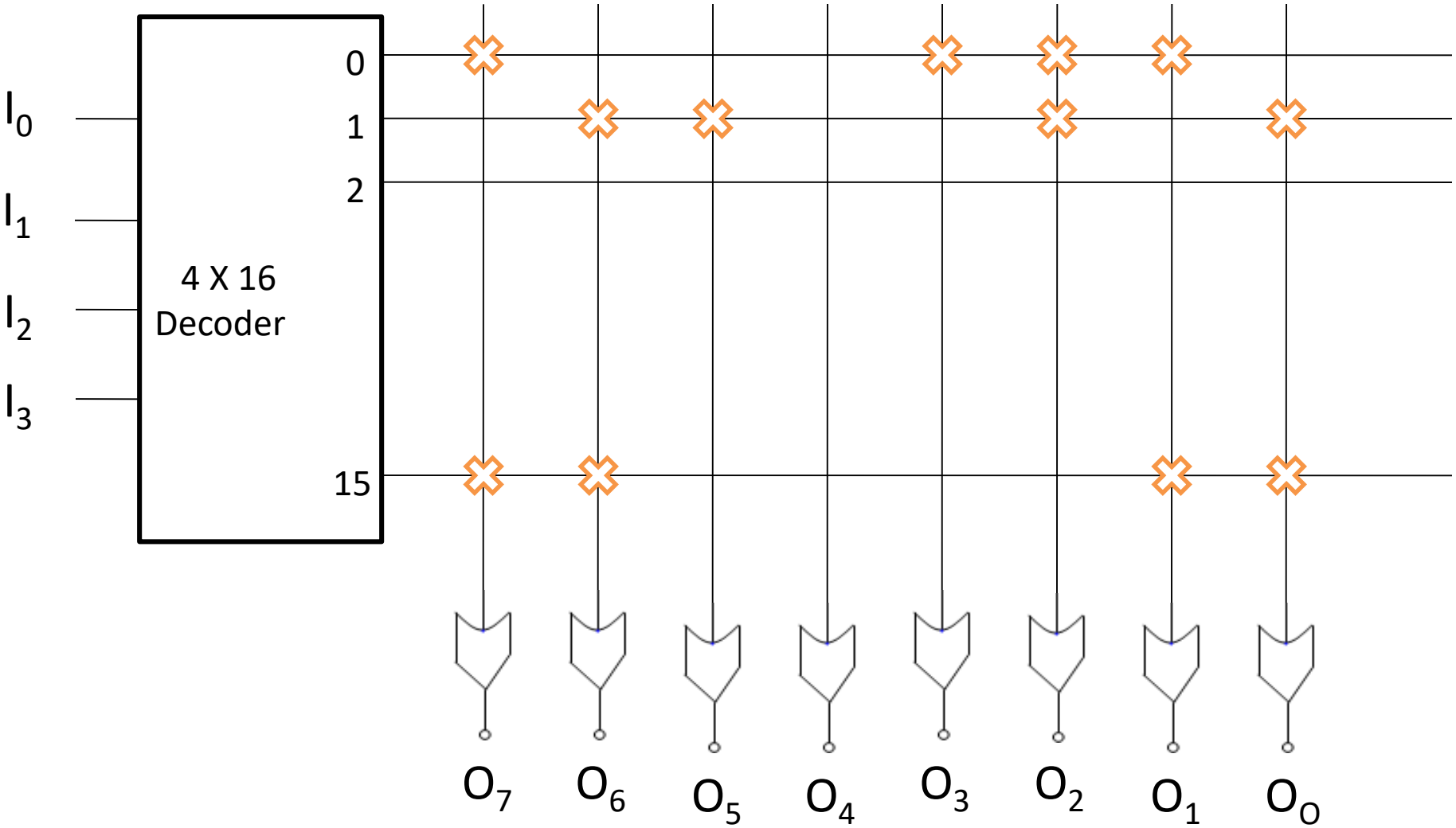


Block diagram of ROM

Consider a 16 x 8 ROM, i.e. it has four inputs, eight outputs and ROM contains 16x8 = 128 programmable connections.

Inputs				Outputs							
I3	I2	I1	I0	O7	O6	O5	O4	O3	O2	O1	O0
0	0	0	0	1	0	0	0	1	1	1	0
0	0	0	1	0	1	1	0	0	1	0	1
.	.	.	.								
1	1	1	1	1	1	0	0	0	0	1	1

Programmable Logic Device



O7 (I3,I2,I1,I0) = ∑m(0, ……,15)

Programmable Logic Device

PLA:

- Similar in concept to PROM, except that PLA does not provide full decoding of variables and does not generate all the min-terms.
- The decoder is replaced by an array of AND gates that can be programmed to generate product terms of input variables.
- The product terms are then selectively connected to OR gates to provide the sum of products for the required Boolean functions.

- Design a combinational circuit using PLD device as PLA (4x8x4), and that is used to implement the full subtractor functions in which difference represented Di and borrow represented as Br.

Programmable Logic Device

Truth table

Inputs				Outputs			
I3	I2 A	I1 B	I0 C	O3	O2	O1 D	O0 Br
X	0	0	0	X	X	0	0
X	0	0	1	X	X	1	1
X	0	1	0	X	X	1	1
X	0	1	1	X	X	0	1
X	1	0	0	X	X	1	0
X	1	0	1	X	X	0	0
X	1	1	0	X	X	0	0
X	1	1	1	X	X	1	1

Programmable Logic Device

K-map:

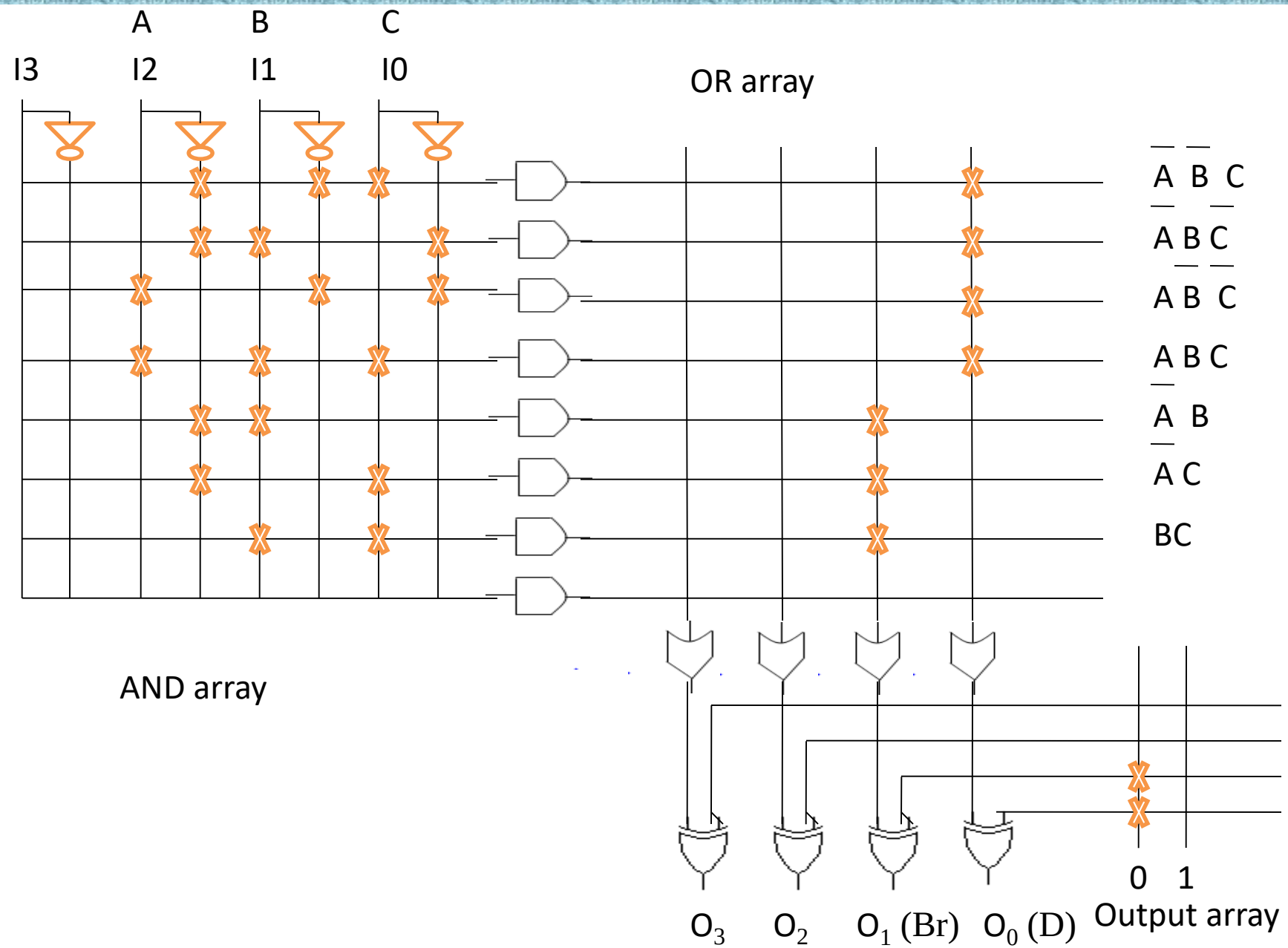
A \ BC	00	01	11	10
0		1		1
1	1		1	

$$D = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + ABC.$$

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

$$Br = \overline{A}\,B + \overline{A}\,C + B\,C$$

Programmable Logic Device



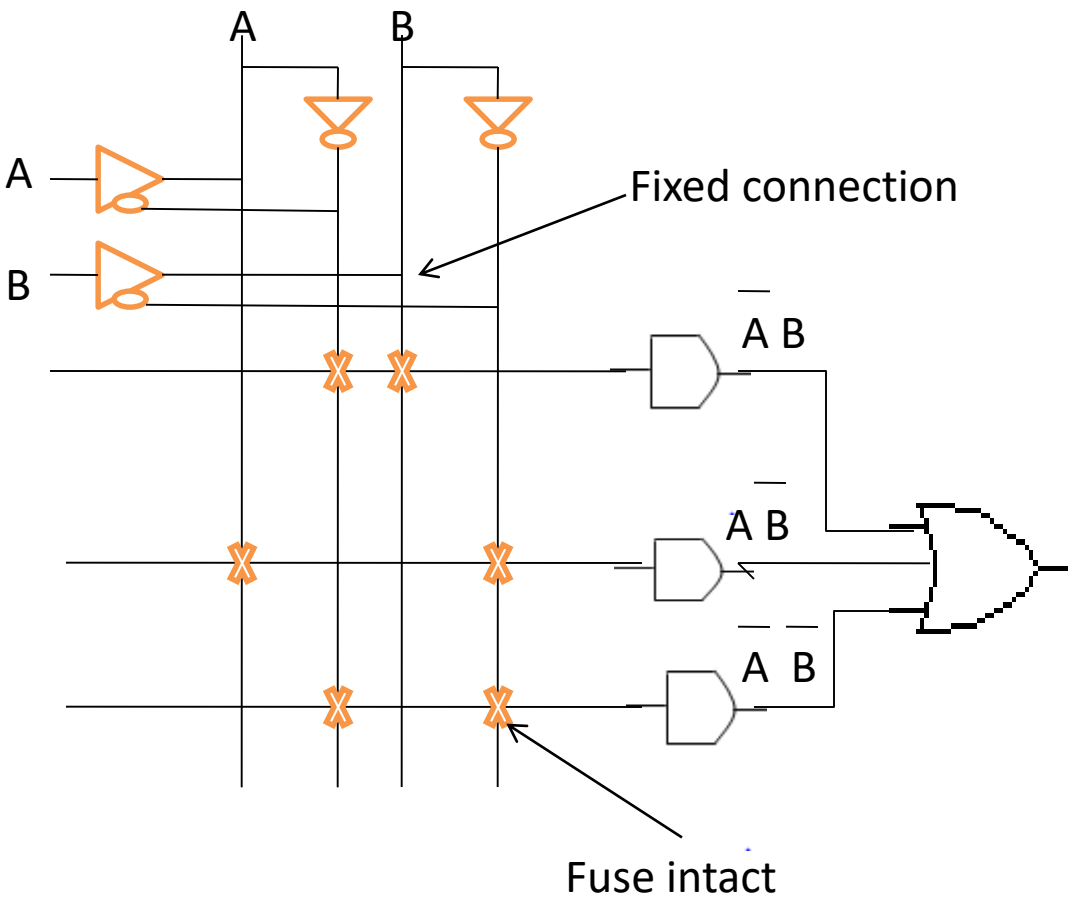
Programmable Logic Device

PAL:

- It consists of programmable array of AND gates that connects to a fixed array of OR gates.
- Suitable for sum of products.
- Input variables are buffered to prevent loading by large number of AND gates inputs.

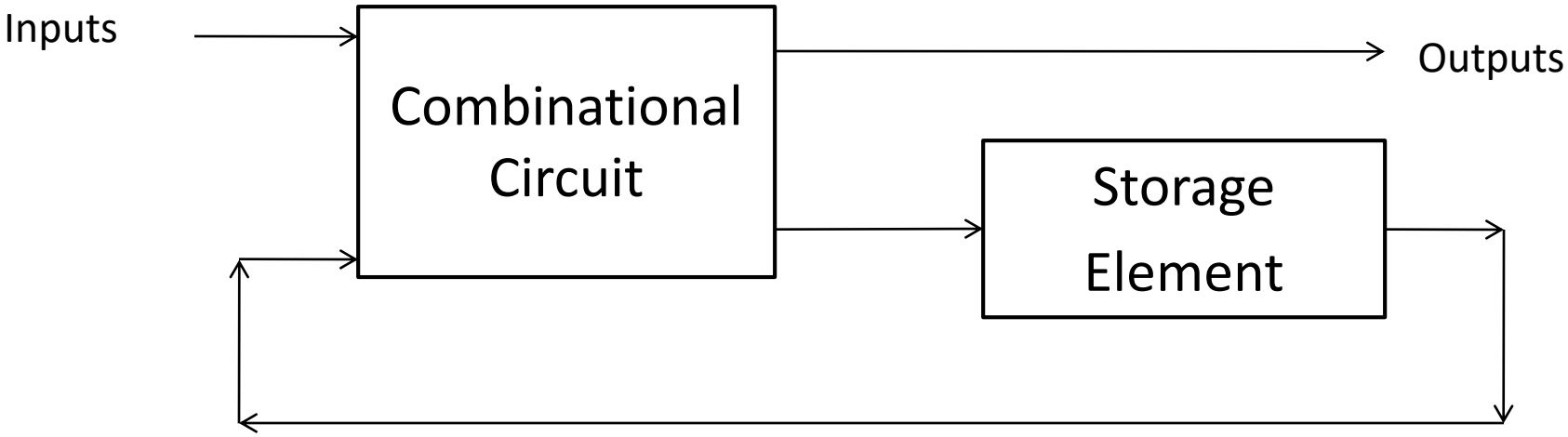
- Design a combinational circuit using PLD device as PAL to implement the expression mentioned below.
- $Y = \overline{A}\,B + A\,\overline{B} + \overline{A}\,\overline{B}$.

Programmable Logic Device



$$Y = \bar{A}\,\bar{B} + A\,\bar{B} + \bar{A}\,\bar{B}$$

Sequential Logic Circuit



Block diagram of sequential circuit

Sequential Logic Circuit

Latches and Flip-flop:

- The major difference between the various types of Latches and flip-flops are the number of inputs they posses and the number in which inputs effect the binary state.

- Latches are most often used within flip-flops.

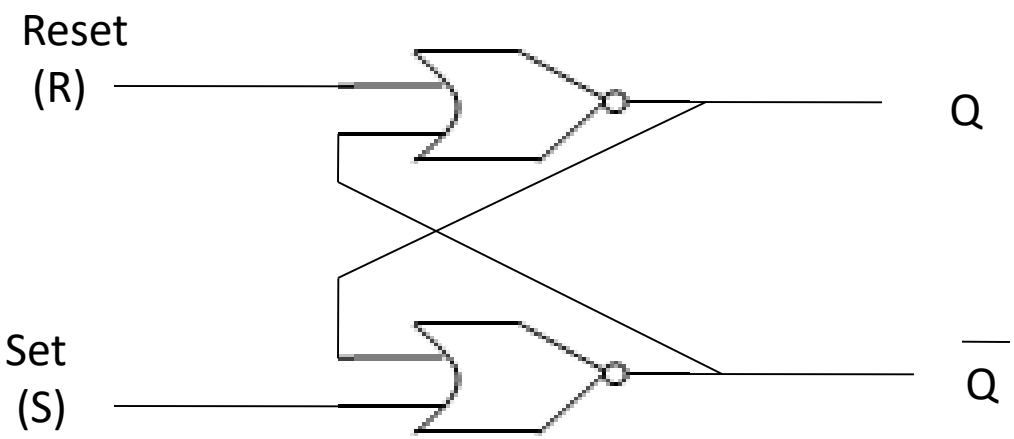
- Latches assumed to be transparent .

- Flip-flops, assumed to be not transparent.

Sequential Logic Circuit

SR Latches:

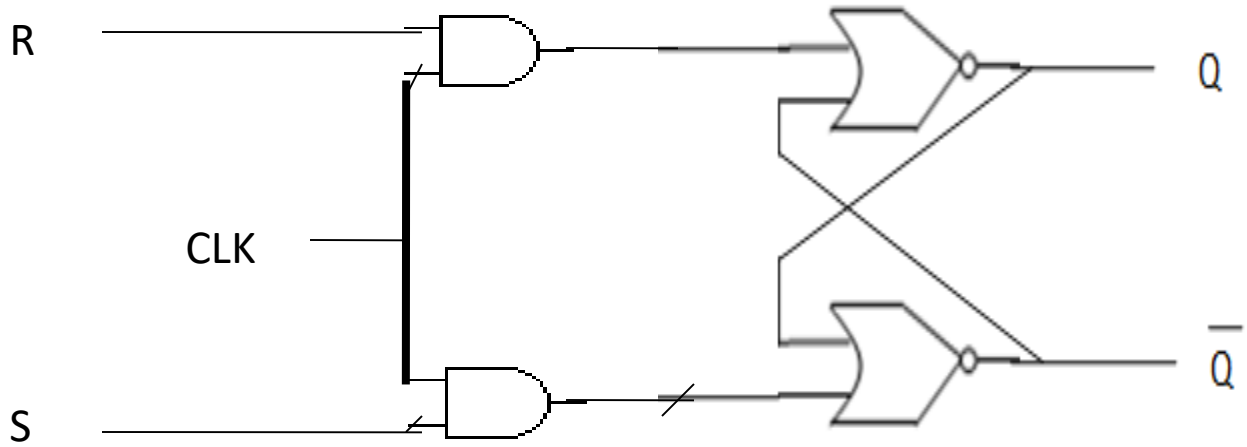
S	R	Q	$\overline{Q}$	State
1	0	1	0	Set state
0	0	1	0	Set state
0	1	0	1	Reset state
0	0	0	1	Reset state
1	1	0	0	Undefined



Sequential Logic Circuit

Clocked S R Flip-flop:

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

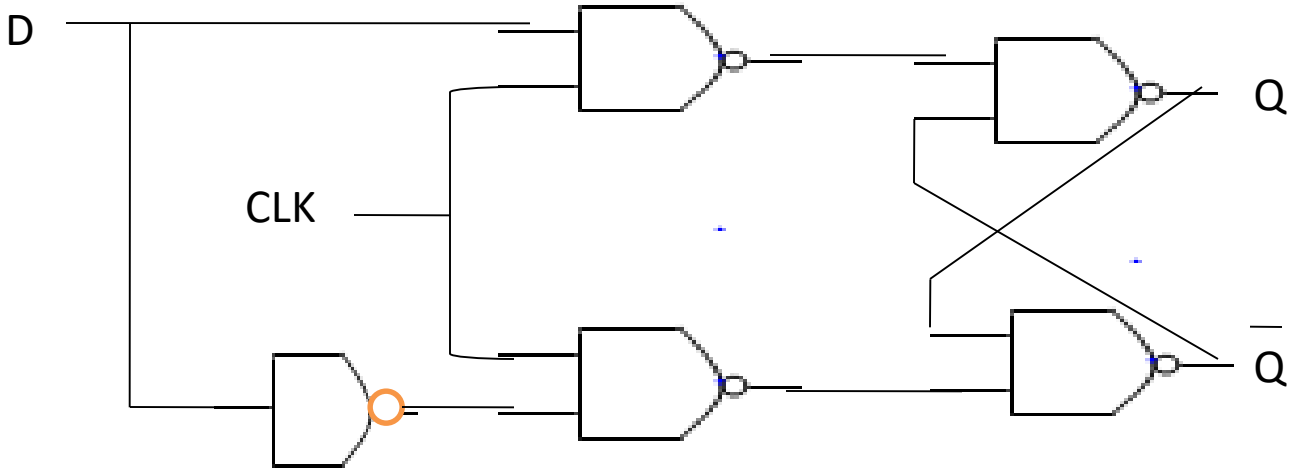


Q \ SR	00	01	11	10
0			X	1
1	1		X	1

$$Q(t+1) = S + \overline{R}\ Q$$

Sequential Logic Circuit

D Flip-flop:



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

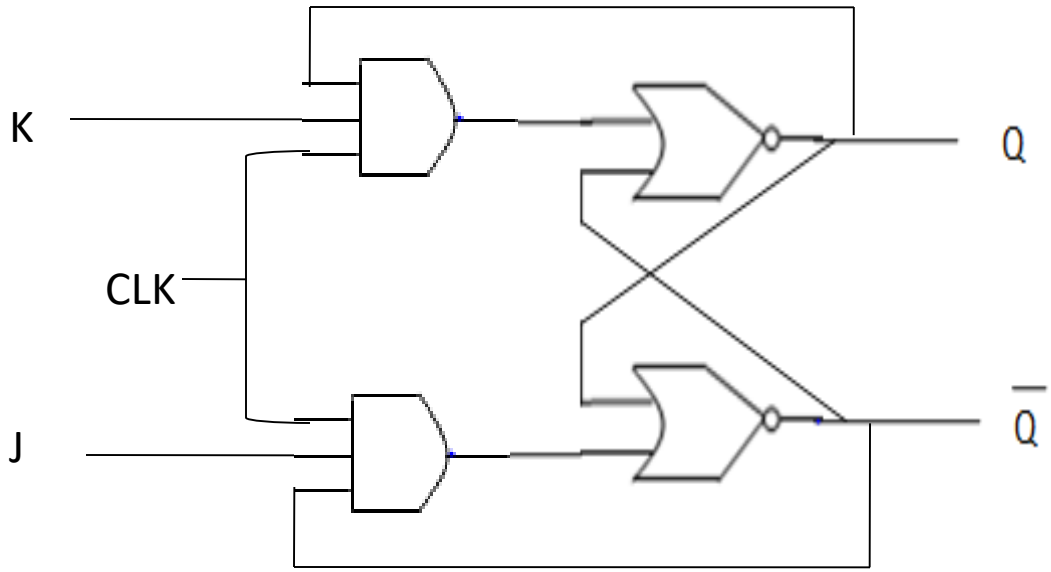
Q \ D	0	1
0		1
1		1

Q(t+1) = D

Sequential Logic Circuit

J K Flip-flop:

- It is the refinement form of the RS flip-flop wherein in-determinant state of RS type is eliminated.



Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

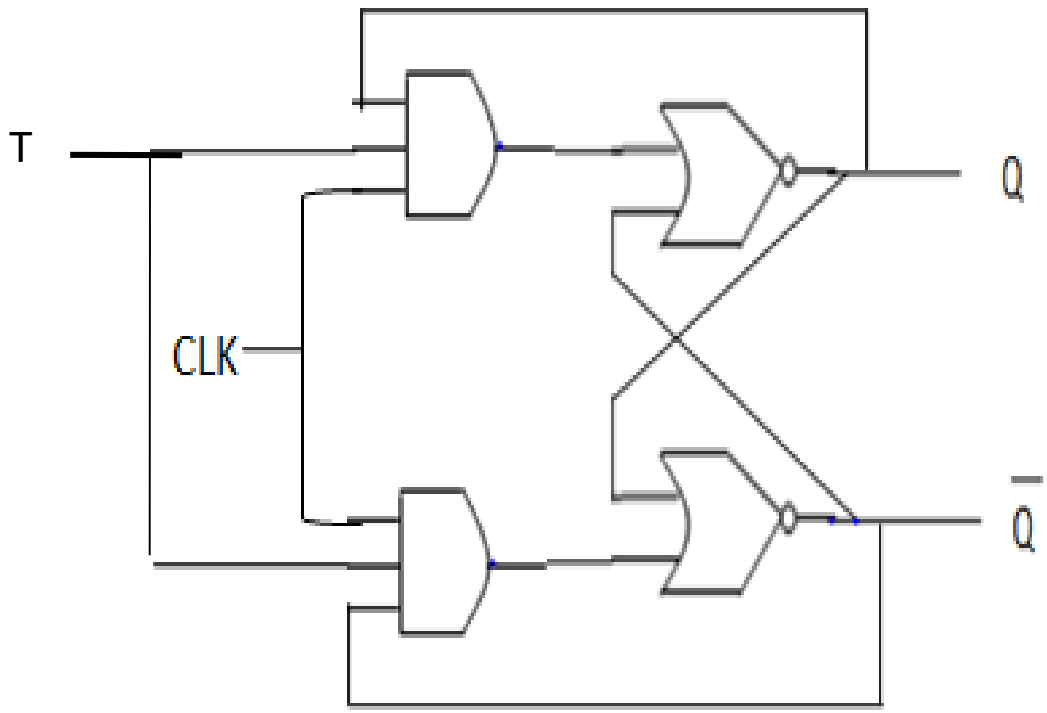
Sequential Logic Circuit

Q \ JK	00	01	11	10
0			1	1
1	1			1

$$Q(t+1) = \overline{Q}\ J + Q\ \overline{K}$$

Sequential Logic Circuit

T Flip-flop:



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

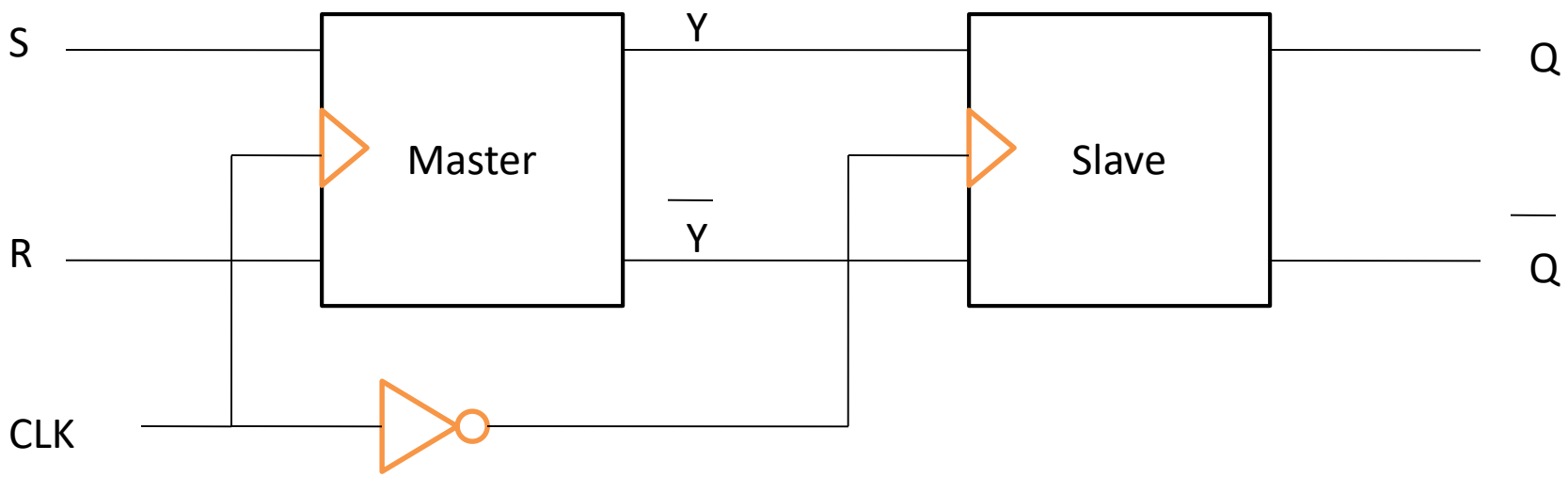
Q \ T	0	1
0		1
1	1	

$$Q(t+1) = \overline{Q}\,T + Q\,\overline{T}$$

Sequential Logic Circuit

Master Slave Flip-flop:

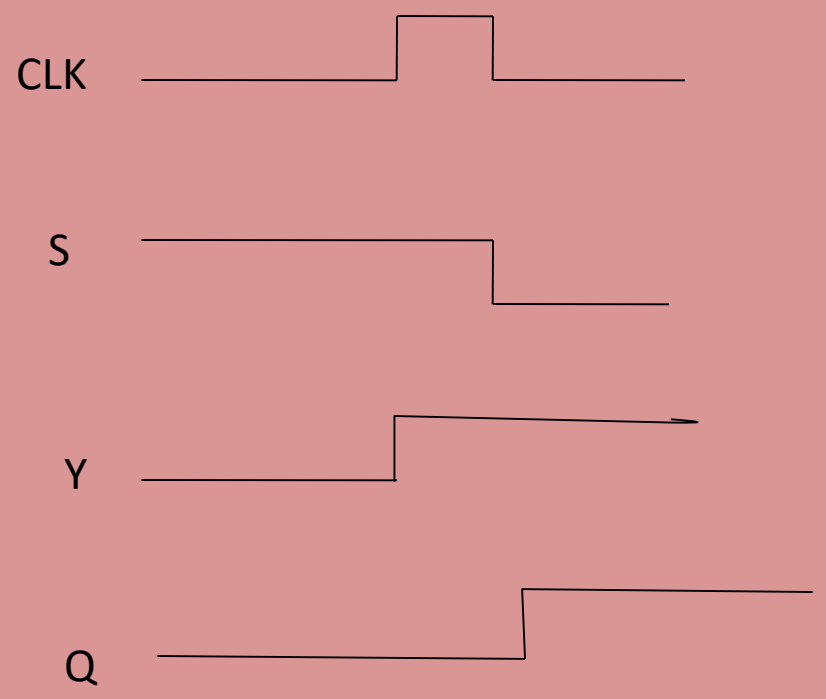
It constructed from two separate flip-flops wherein one circuit serves as a master and other as a slave and overall circuit is referred to as a master slave flip-flop.



Logic diagram for Master Slave Flip-flop.

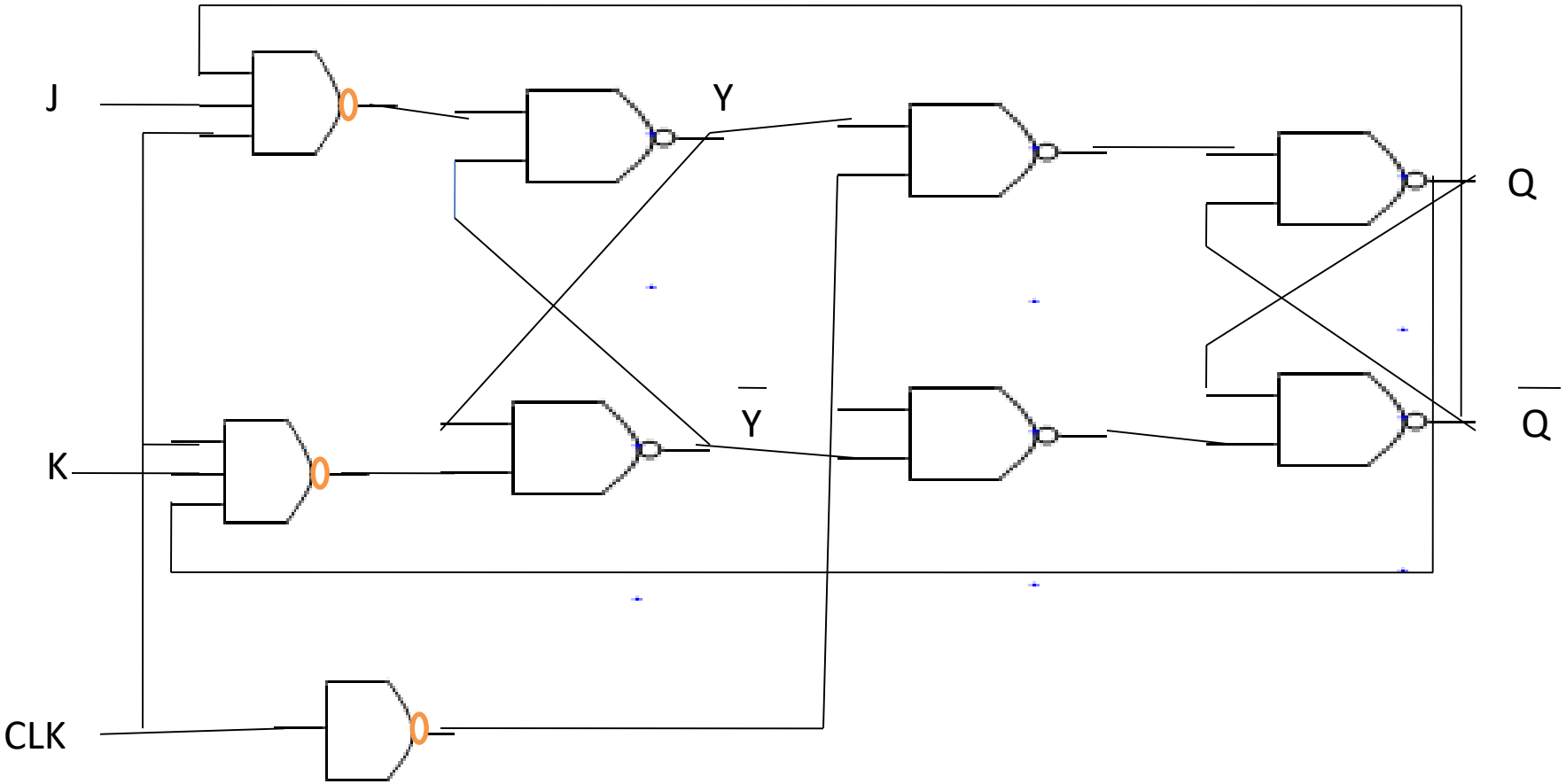
Sequential Logic Circuit

Timing relationships:



Sequential Logic Circuit

Master Slave Flip-flop:

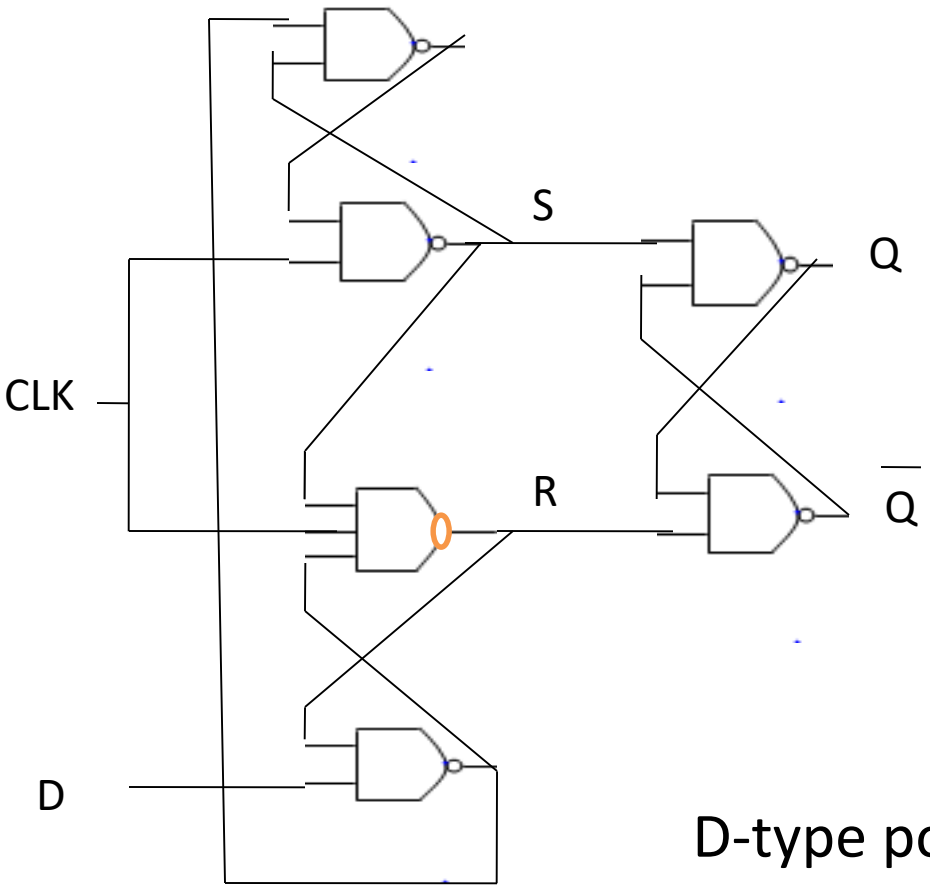


Clocked Master Slave JK Flip-flop

Sequential Logic Circuit

Edge-triggered flip-flop:

- Flip-flop that synchronizes state changes during clock pulse transition is termed as Edge triggered flip-flop.

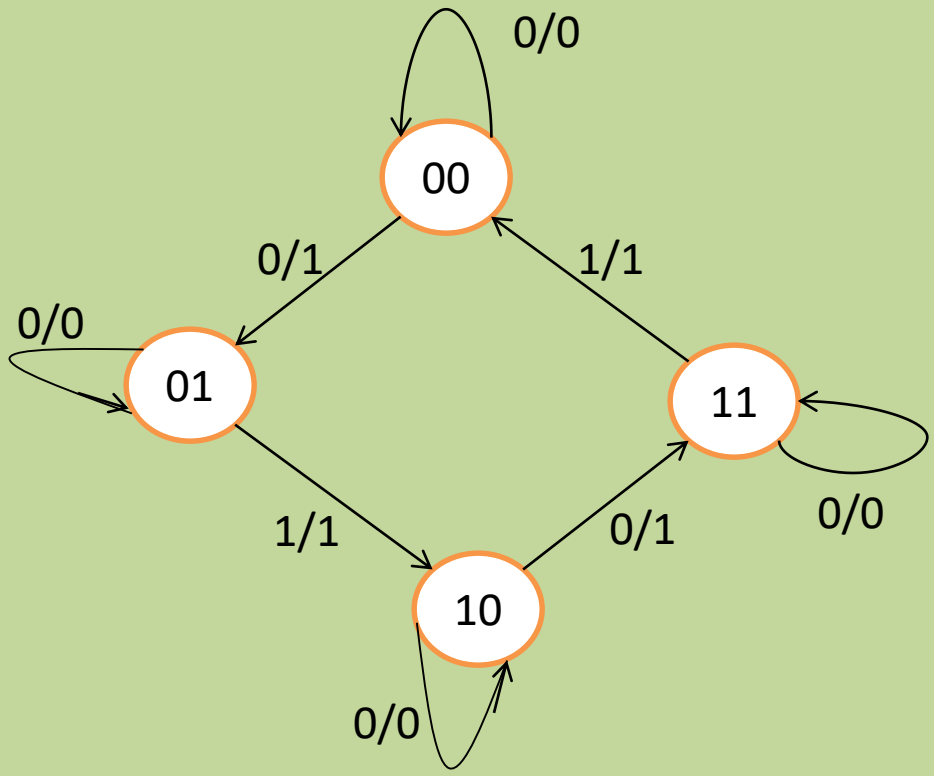


D-type positive edge triggered flip-flop

Sequential Logic Circuit

Analysis of clocked sequential circuit:

State Diagram:



- **Design a sequential circuit from given state diagram using D flip-flops.**

Sequential Logic Circuit

Excitation table:

Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

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

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

Sequential Logic Circuit

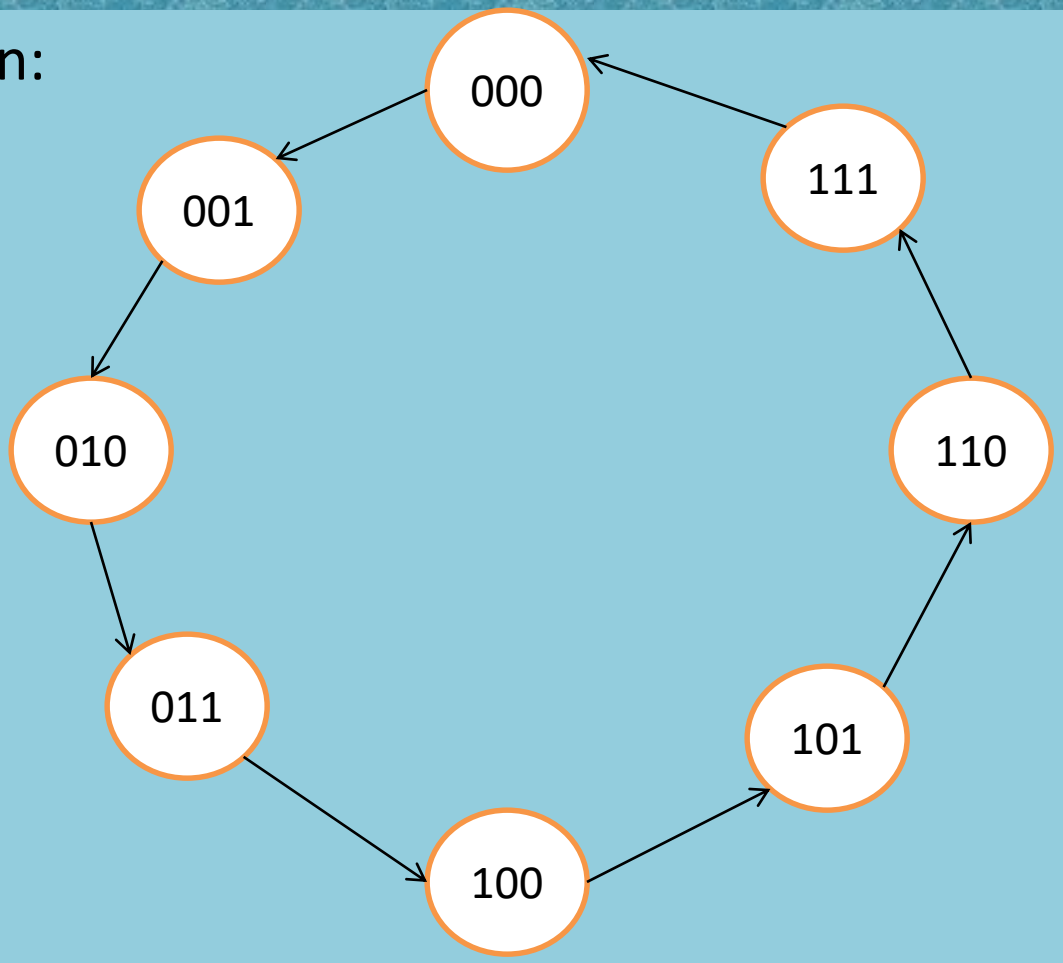
Counter Design:

- A sequential circuit that goes through a prescribed sequence of state upon the application of clock pulses is called a counter.
- A counter that follows the binary sequence is called binary counter.
- An n-bit binary counter consists of n flip-flops and can count from 0 to 2^n-1.

- Design a counter using T Flip-flop that counts the binary sequence from 000 to 111 and returns to 000 to repeat the sequence.

Sequential Logic Circuit

Counter Design:



Sequential Logic Circuit

Counter design:

Count sequence(PS)			Next state			Flip-flop inputs		
A2	A1	A0	A2	A1	A0	TA2	TA1	TA0
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

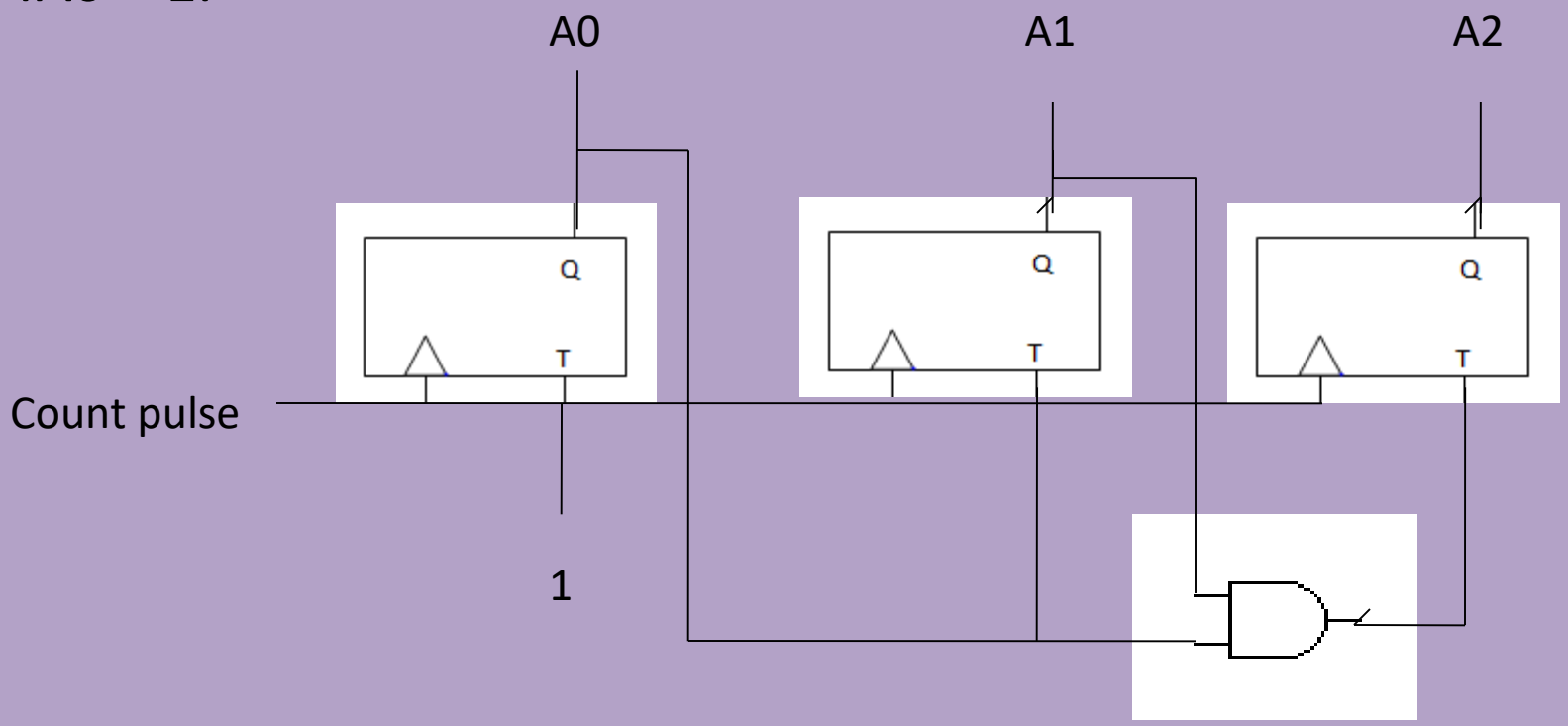
Sequential Logic Circuit

From K-map:

TA2 = A1A0.

TA1 = A0.

TA0 = 1.



Logic diagram for 3-bit binary counter

Sequential Logic Circuit

- Design a counter using D Flip-flops that counts the binary sequence from 00 to 11 and returns to 00 to repeat the sequence.

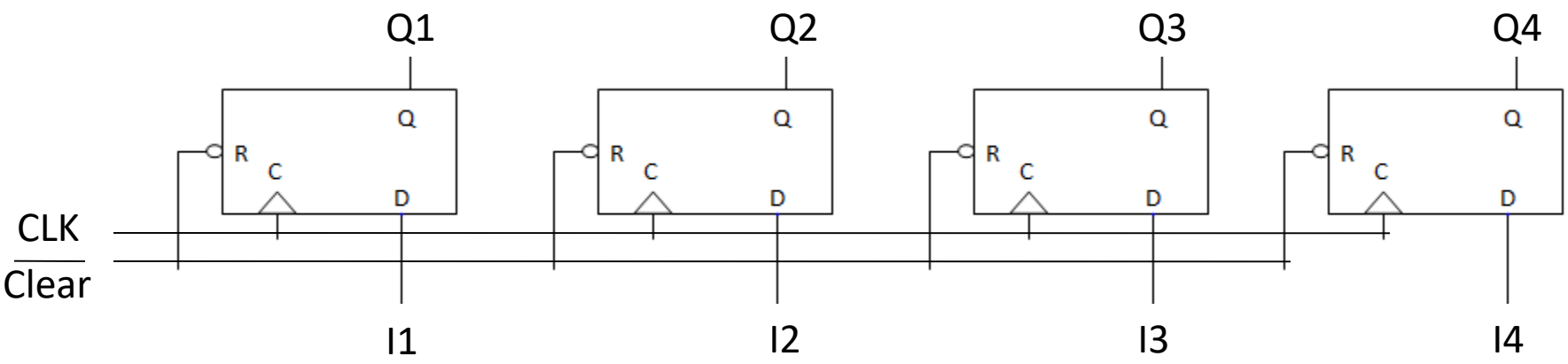
- Design a counter using D Flip-flops that counts the binary sequence from 111 to 000 and returns to 111 to repeat the sequence.

- Design a BCD counter using T Flip-flops that counts the binary sequence from 0000 to 1001 and returns to 0000 to repeat the sequence.

Sequential Logic Circuit

Registers:

- A register consists of a group of binary storage cells suitable for holding binary information.
- An n-bit register include n flip-flops, and capable to store n-bit information.
- A group of flip-flops constitute a register.

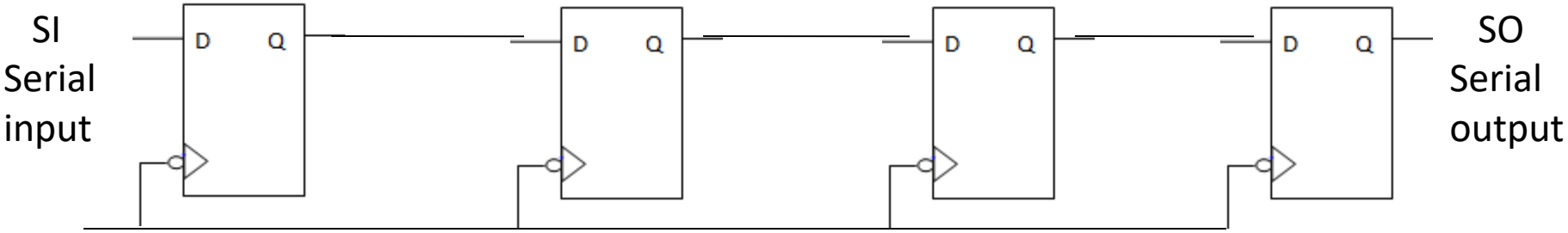


Diag. 4-bit register

Sequential Logic Circuit

Shift register:

- A register capable of shifting its binary information either to the right or to the left is called a shift register.
- It consists chain of flip-flop connected in cascade with output of one flip-flop connected to input of next flip-flop.
- All flip-flops receive a common clock pulse which causes a shift from one stage to the next.

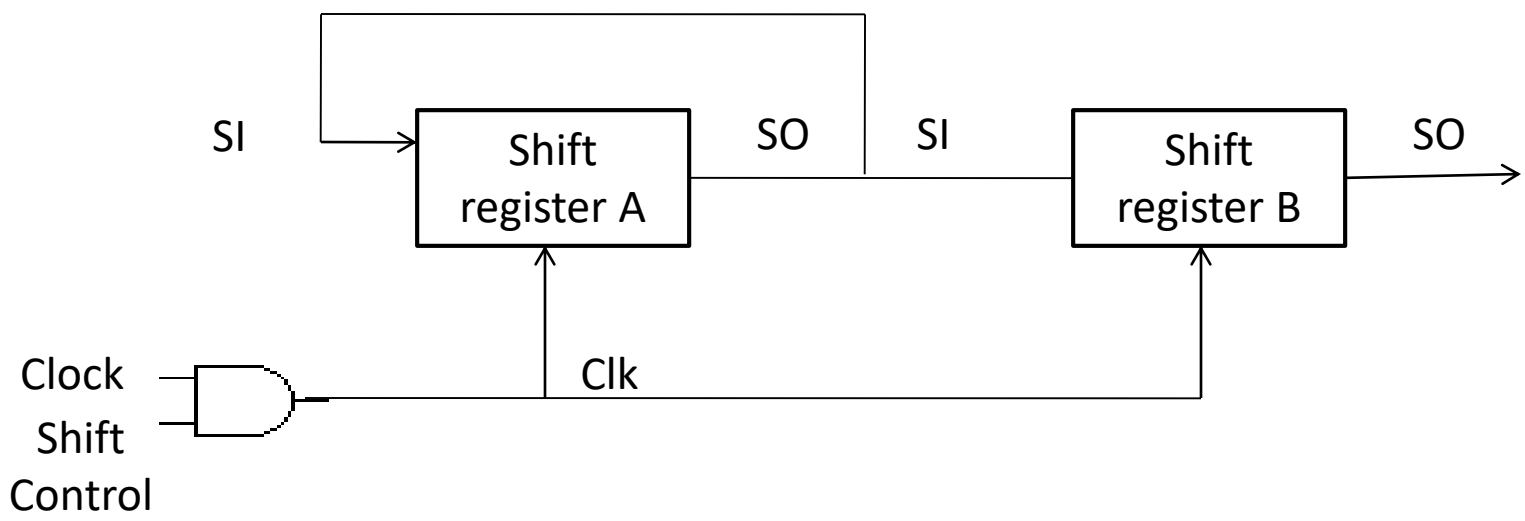


Diag. Shift register

Sequential Logic Circuit

Serial Transfer:

- A digital system is said to operate in serial mode when information is transferred and manipulated one bit at a time.
- The content of one register is transferred to another by shifting the bits from one register to the other.
- Information is transferred one bit at a time by shifting the bit out of the source register into destination register.



Sequential Logic Circuit

Timing pulse	Shift register A	Shift register B	Serial output of B
Initial value	1 0 0 1	0 1 0 1	1
After T1	1 1 0 0	1 0 1 0	0
After T2	0 1 1 0	0 1 0 1	1
After T3	0 0 1 1	0 0 1 0	0
After T4	1 0 0 1	1 0 0 1	1

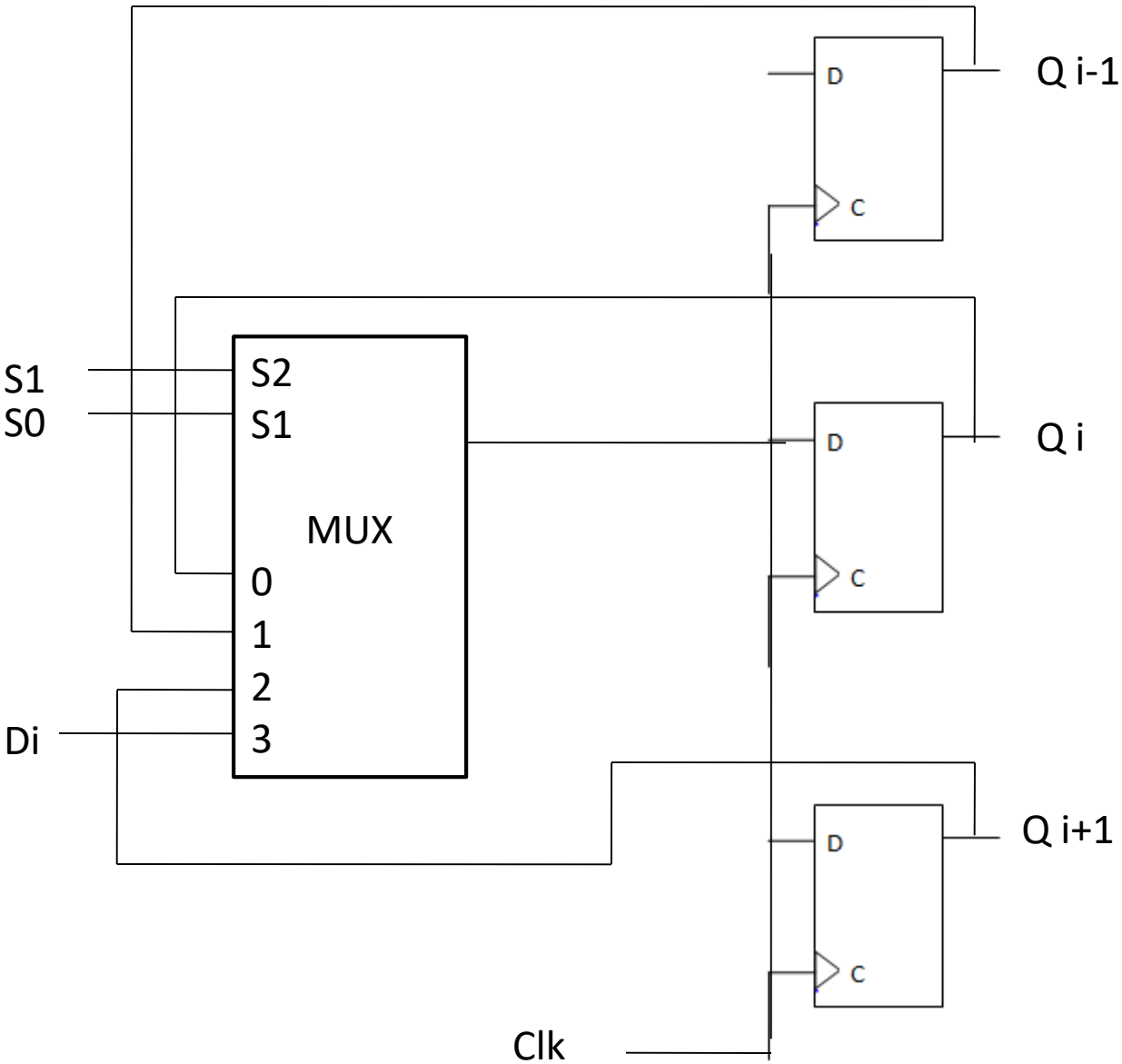
Sequential Logic Circuit

Bidirectional shift register:

- A register capable of shifting binary data in both directions is termed as a bidirectional shift register.

Mode S2	Control S3	Register operation
0	0	No Change
0	1	Shift down
1	0	Shift up
1	1	Parallel load

Sequential Logic Circuit

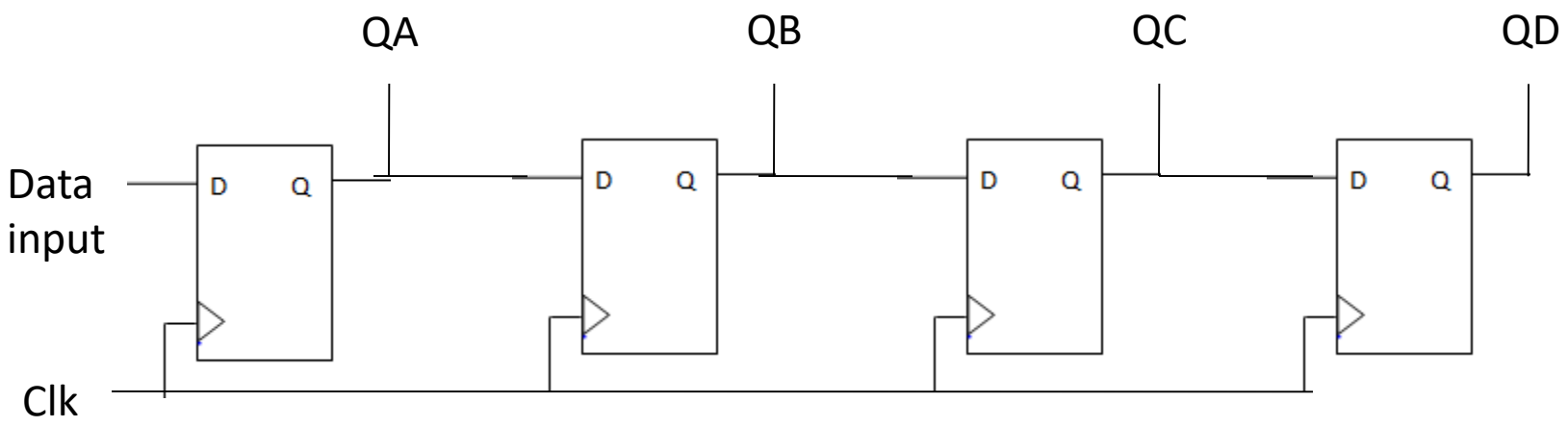


Sequential Logic Circuit

Types of shift register:

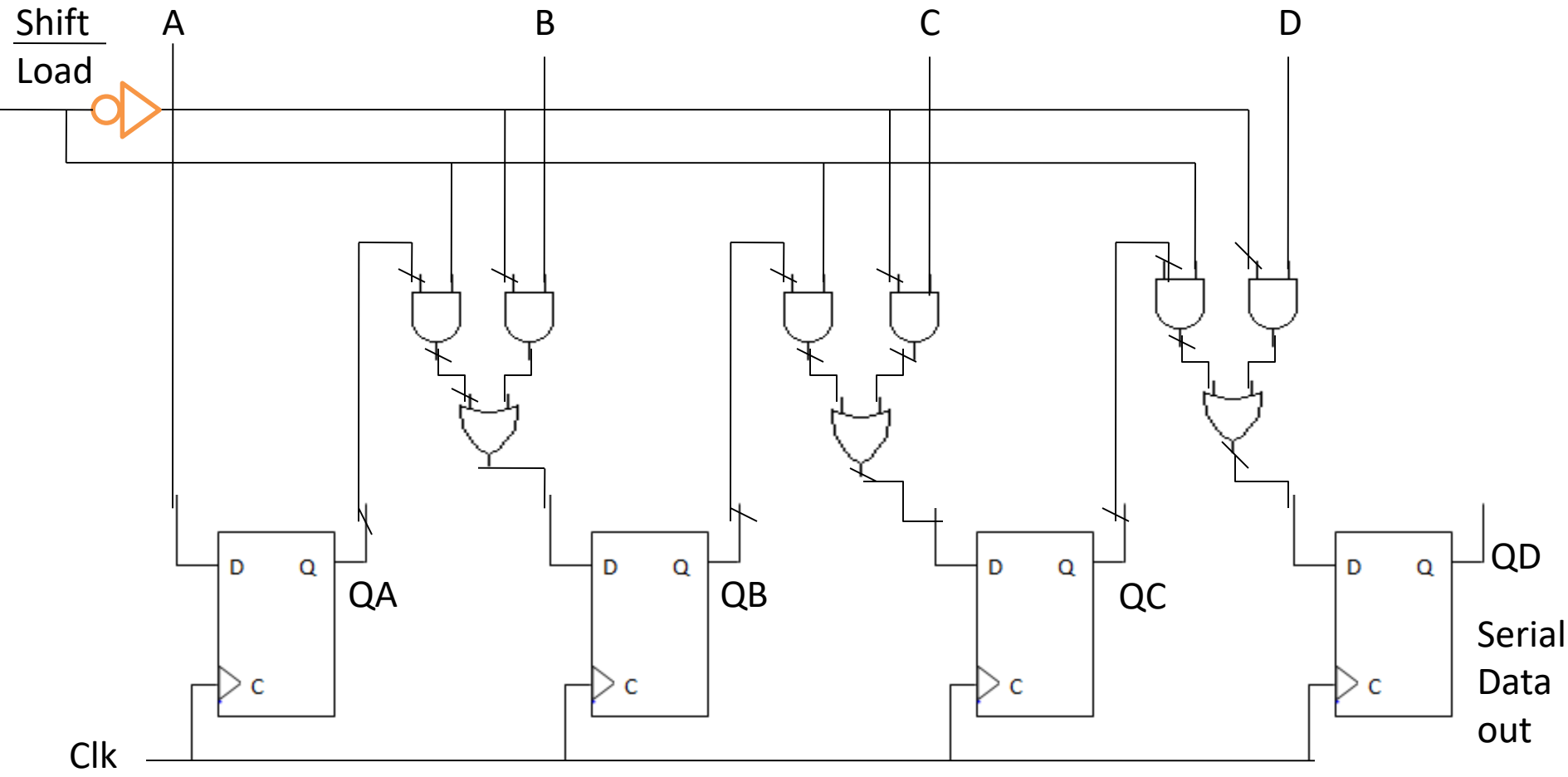
1. Serial in serial out shift register.
2. Serial in parallel out shift register.
3. Parallel in serial out shift register.
4. Parallel in parallel out shift register.

Serial in parallel out shift register:



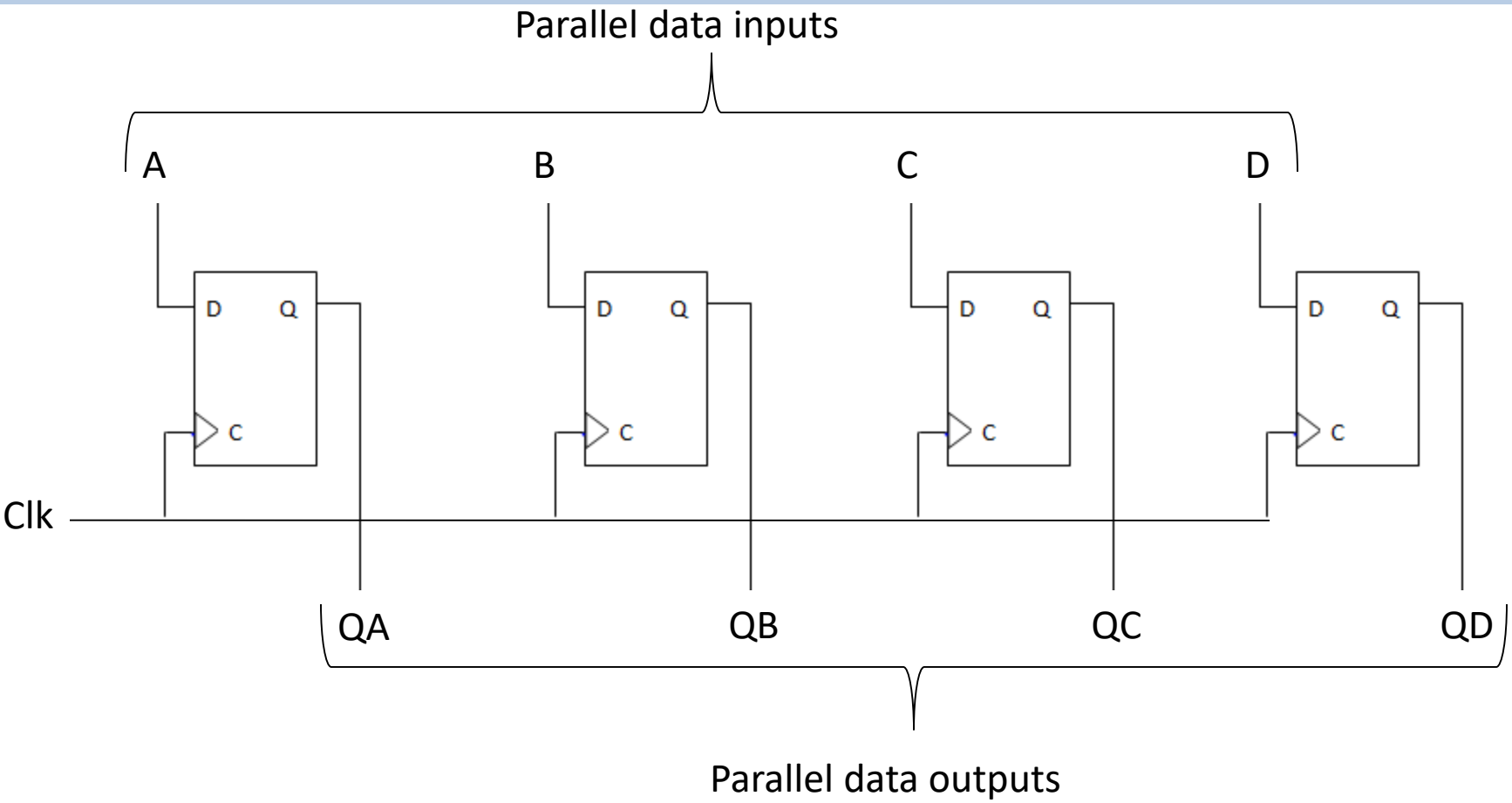
Sequential Logic Circuit

Parallel in serial out shift register:



Sequential Logic Circuit

Parallel in parallel out shift register:



Processor Logic Design

Processor unit:

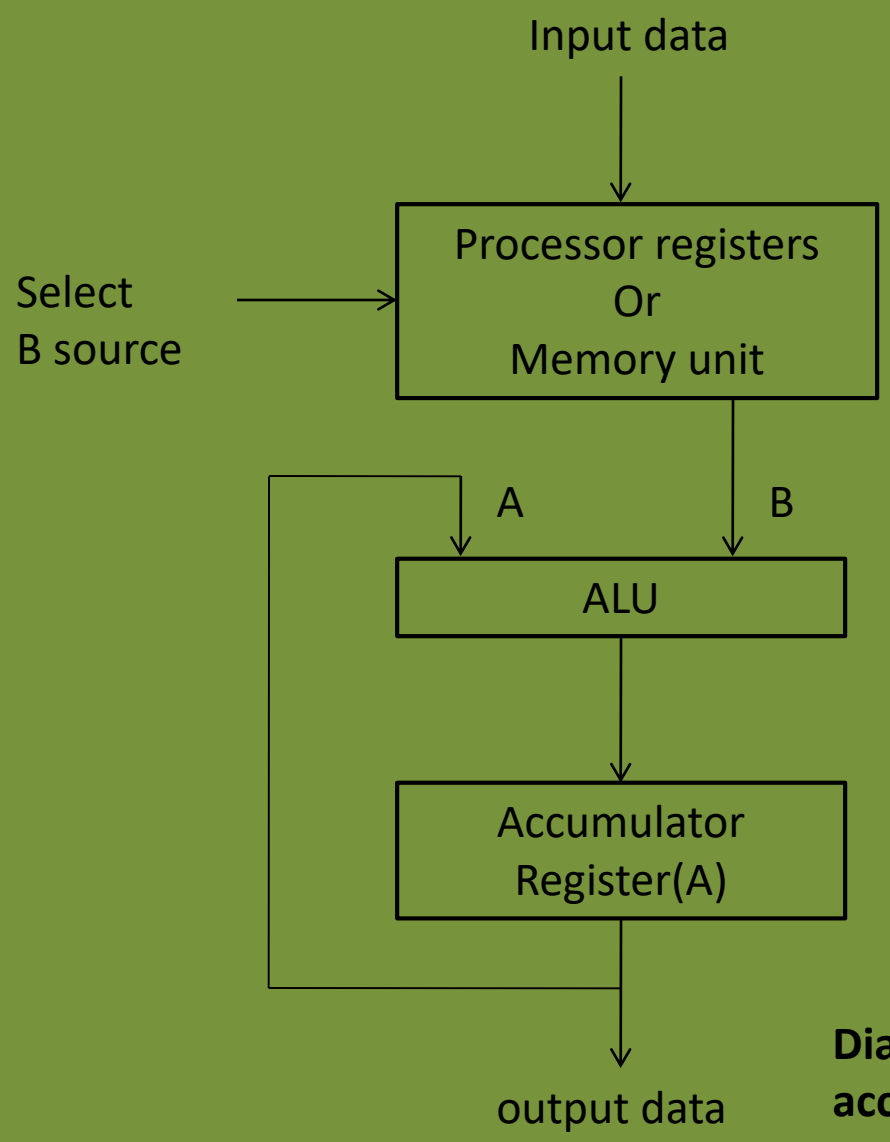
- It implements the operation in the system.
- It comprised of a number of registers and digital functions that implement arithmetic, logic, shift and transfer micro-operations.
- Processor unit on combining with a control unit used to supervise the sequence of micro-operation that is also termed as CPU.

Accumulator Register:

A processor unit that separate one register from all others is termed as an accumulator register, abbreviated as AC or A register.

Processor Logic Design

Accumulator Register:



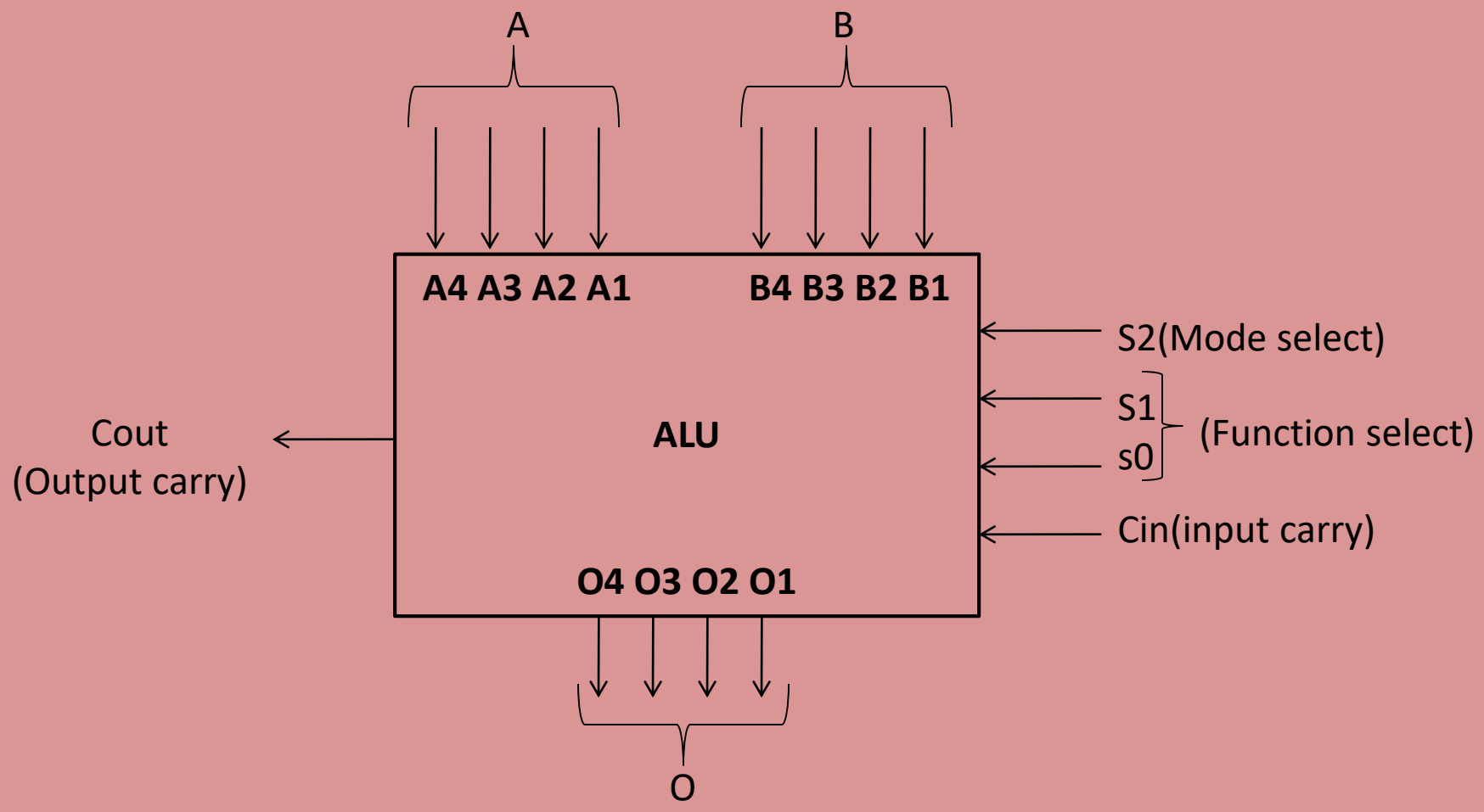
Diag. processor unit with an accumulator register

Processor Logic Design

T1: $A \leftarrow 0$	Clear A
T2: $A \leftarrow A+R1$	Transfer R1 to A
T3: $A \leftarrow A+R2$	Add R2 to A.

Processor Logic Design

Arithmetic Logic Unit(ALU):

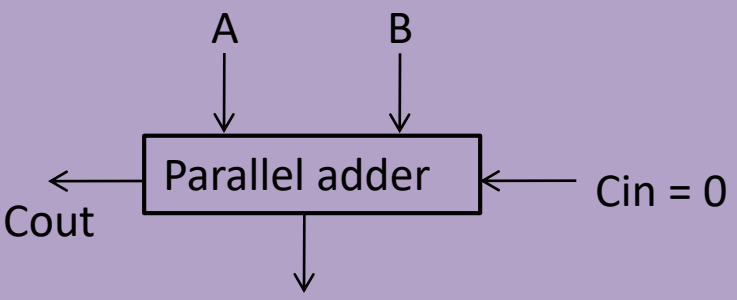


S2 distinguishes between arithmetic logic operations.

S1, So specify particular arithmetic or logic operation.

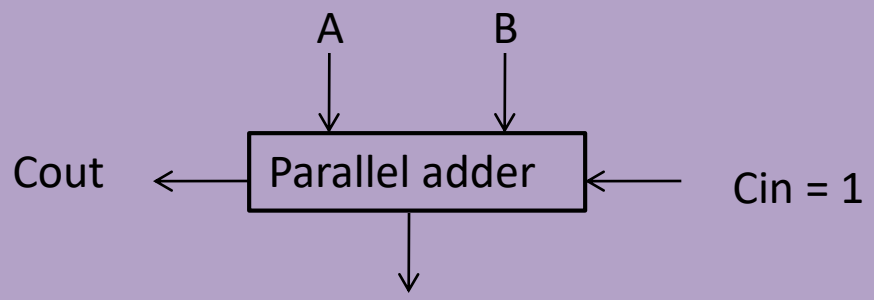
Processor Logic Design

Arithmetic Circuit Design:



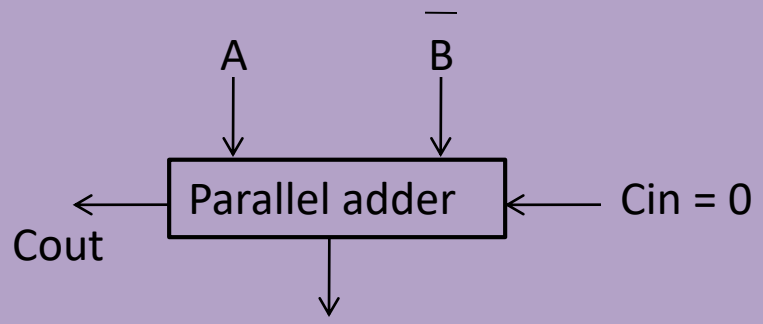
0 = A+B

(a) Addition



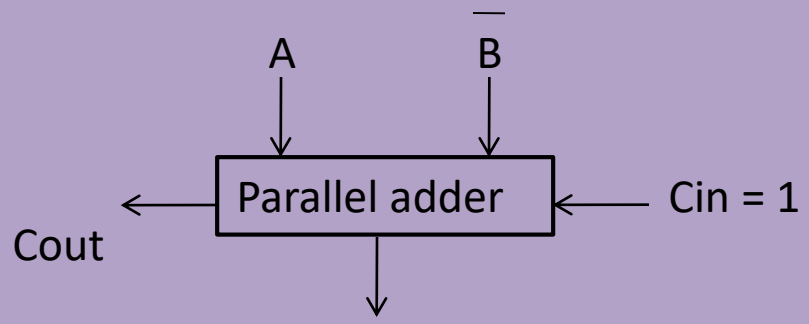
0 = A+B+1

(b) Addition with carry



$0 = A + \overline{B}$

(c) A plus 1's complement

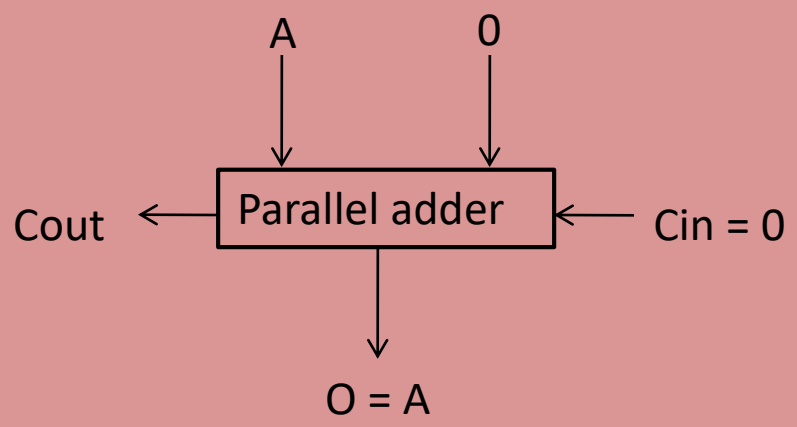


$0 = A + \overline{B} + 1$

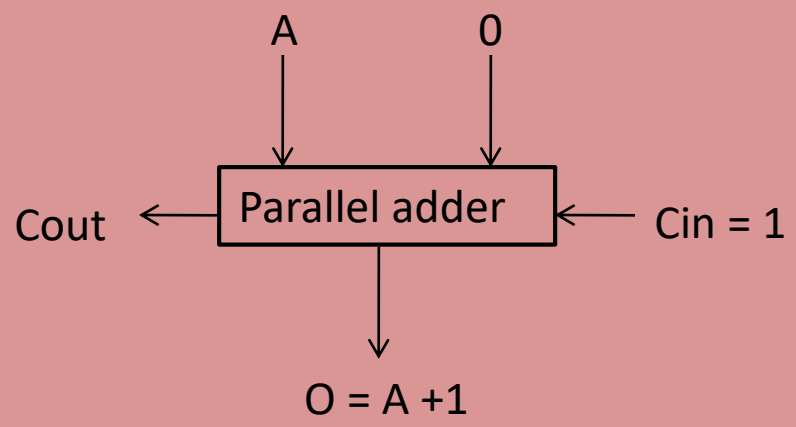
(c) Subtraction

Processor Logic Design

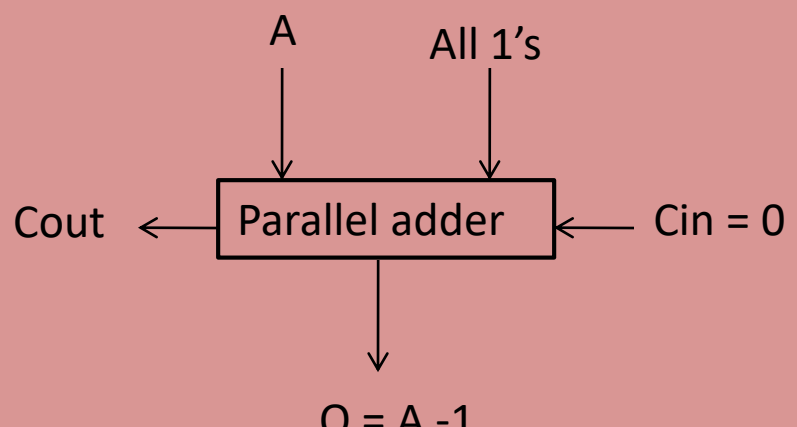
Arithmetic circuit design:



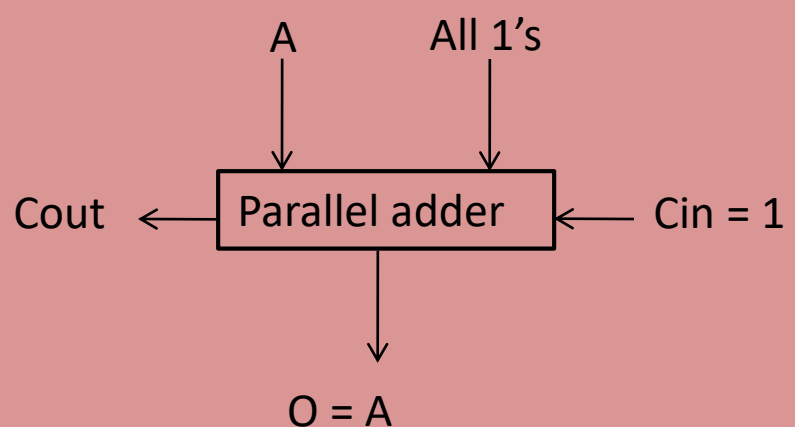
(e) Transfer A



(f) Increment A



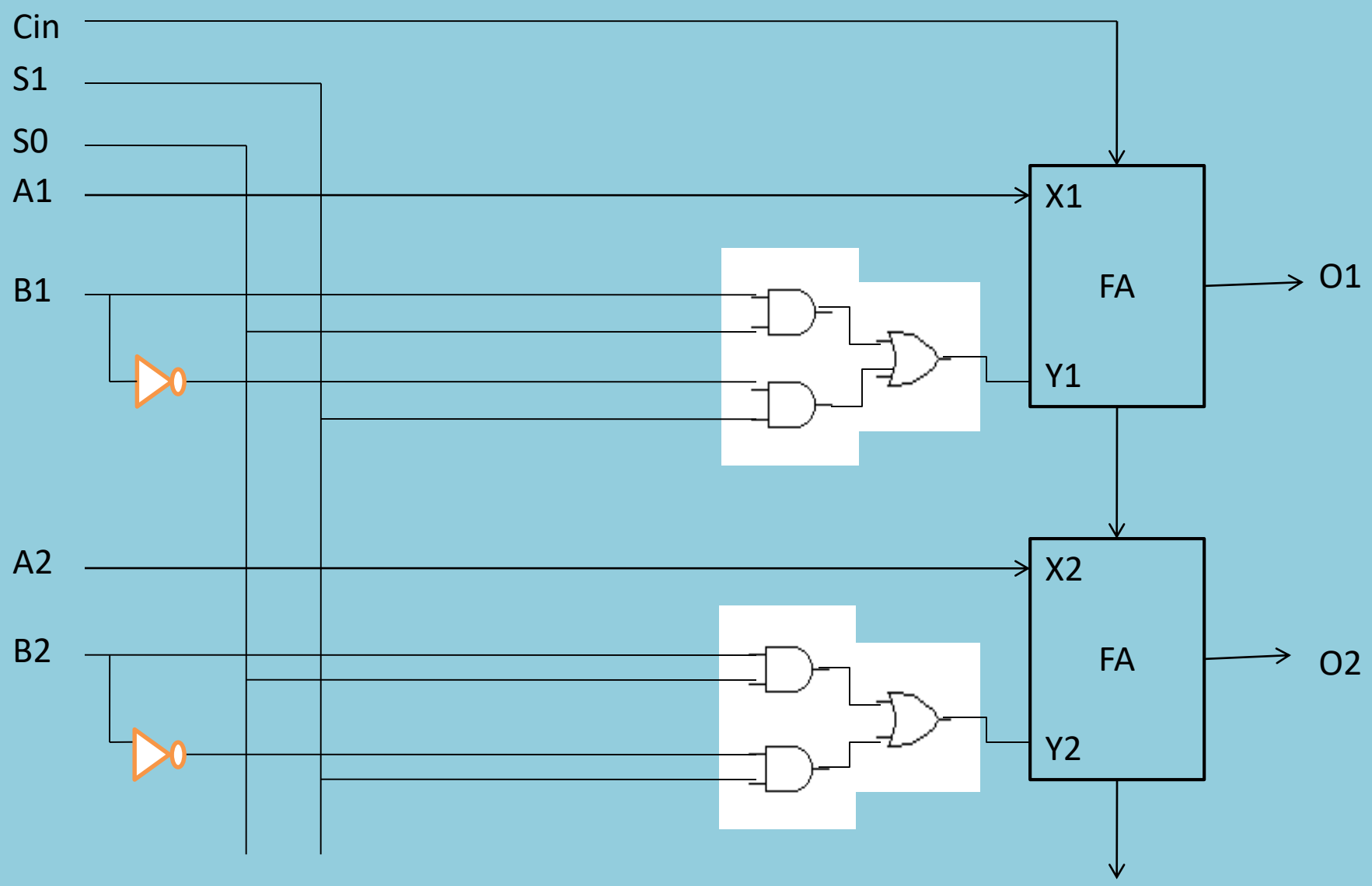
(g) Decrement A



(h) Transfer A

Processor Logic Design

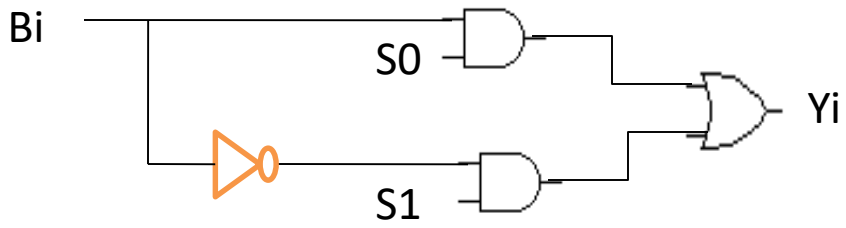
Arithmetic circuit design:



Processor Logic Design

Arithmetic circuit design:

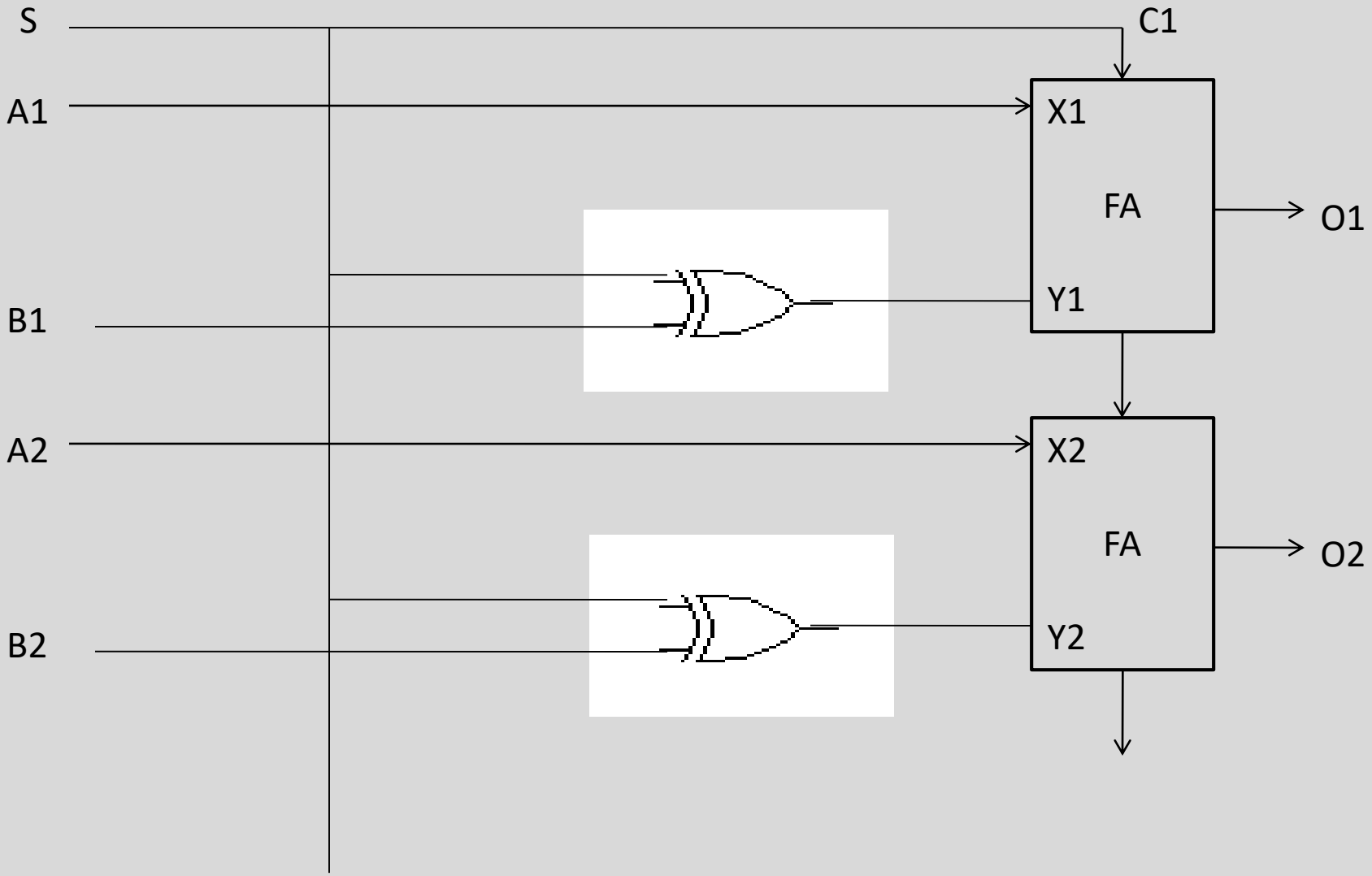
S1 S0	Yi
0 0	0
0 1	Bi
1 0	$\overline{Bi}$
1 1	1



- Design an adder/subtractor circuit with one selection variable S and two inputs A and B. For S = 0, the circuit need to perform addition i.e. (A+B) AND FOR S = 1, the circuit must perform subtraction (A –B) by taking 2's complement of B.

Processor Logic Design

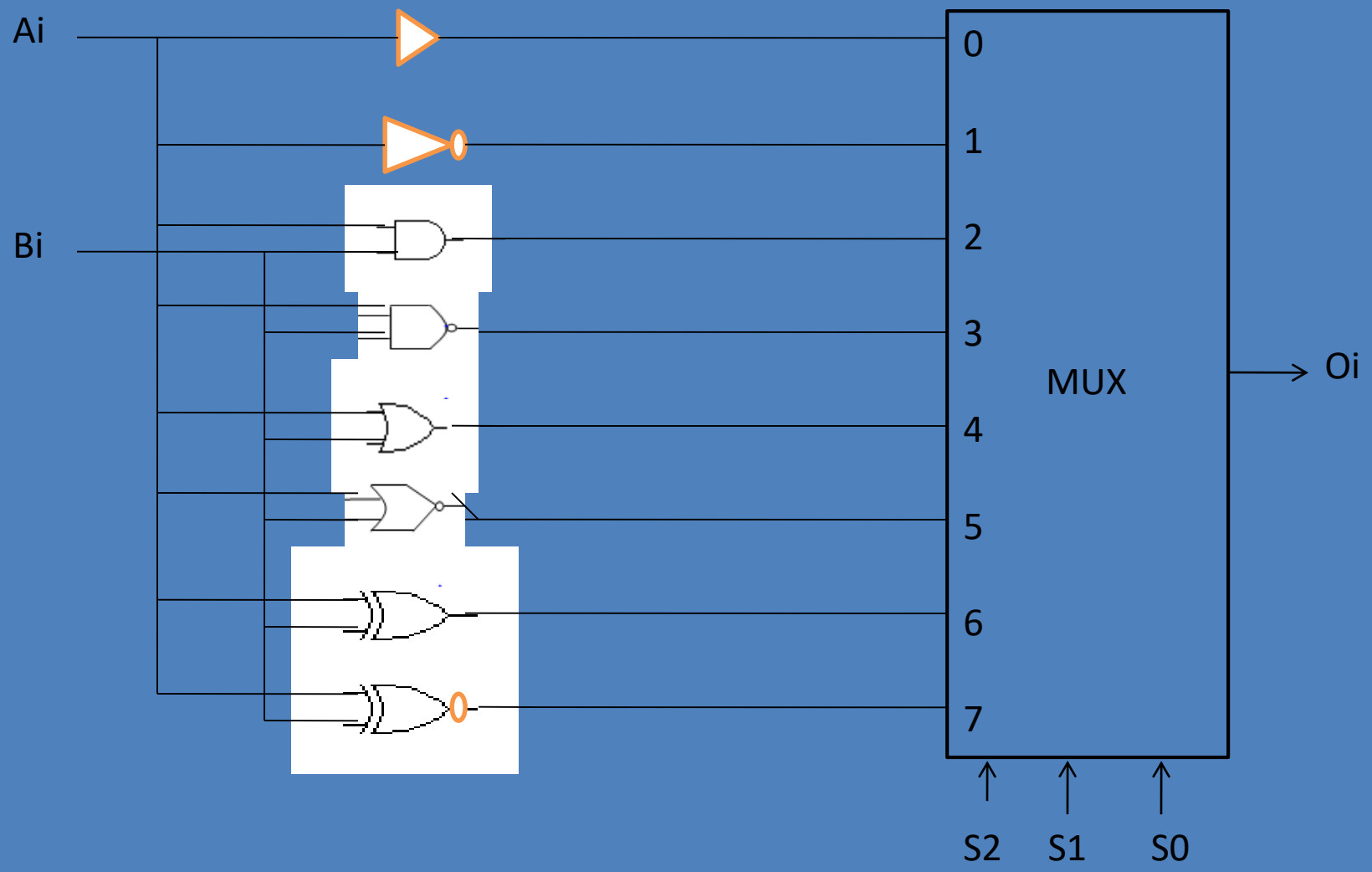
Design:



Processor Logic Design

Logic Circuit Design:

- Design a sequential logic circuit that could perform eight different logical gate operation. Use the concept of function table, multiplexer, different gates to obtained the desired result.



Processor Logic Design

Logic Design:

S2 S1 S0	Output	Operation
0 0 0	$O_i = A_i$	Buffer
0 0 1	$O_i = \overline{A_i}$	NOT
0 1 0	$O_i = A_i B_i$	AND
0 1 1	$O_i = \overline{A_i B_i}$	NAND
1 0 0	$O_i = A_i + B_i$	OR
1 0 1	$O_i = \overline{A_i + B_i}$	NOR
1 1 0	$O_i = A_i \oplus B_i$	EX-OR
1 1 1	$O_i = A_i \odot B_i$	EX-NOR

Counter

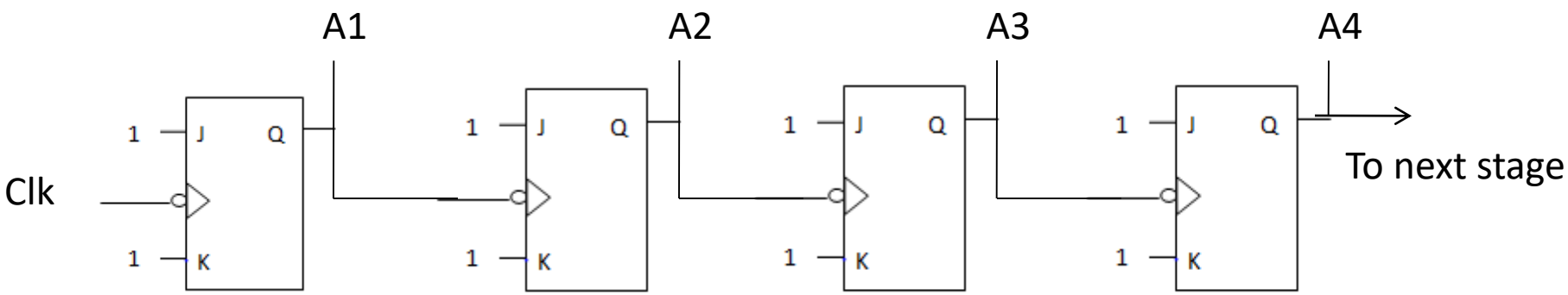
Classification of counter:

Synchronous	Asynchronous (Ripple)
➢ The common clock pulse triggers all the flip- flop simultaneously.	➢The first flip-flop is clocked by external clock pulse and then each successive flip-flop is clocked by output of previous flip-flop.
➢Comparatively fast, as propagation delay involved is less.	➢ Comparatively slow, as propagation delay involved is large.
➢Event occurs at same time.	➢Event do not occurs at same time.
➢Flip-flop within a counter made to change state at exactly same time.	➢Flip-flop within a counter not made to change state at exactly same time.

Counter

Ripple Counter:

a) Binary Ripple Counter:



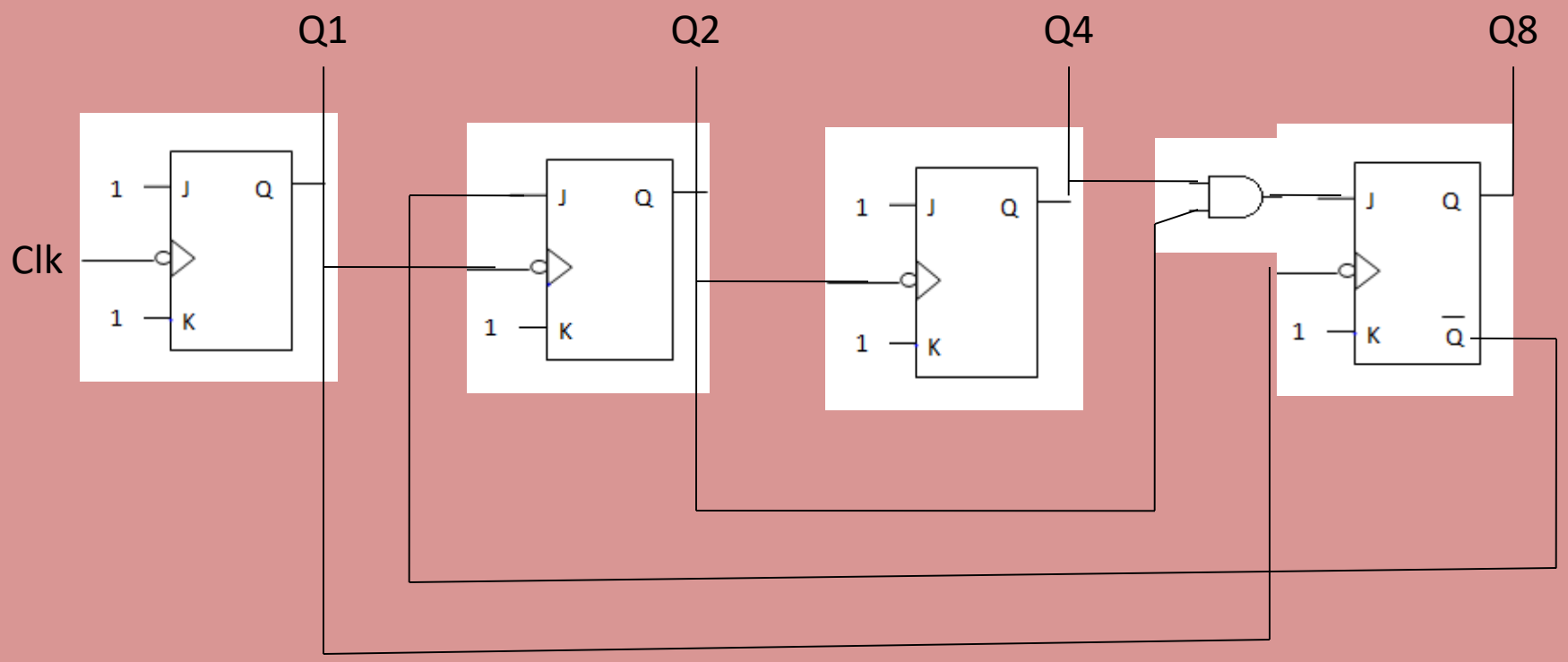
A 4-bit binary ripple counter

Counter

Count sequence	Condition for complementing flip-flop
A4 A3 A2 A1	
0 0 0 0	Complement A1.
0 0 0 1	Complement A1, A1 will go from 1 to 0 and complement A2.
0 0 1 0	Complement A1.
0 0 1 1	Complement A1, A1 will go from 1 to 0 and complement A2. A2 will go from 1 to 0 and complement A3.
0 1 0 0	Complement A1.
0 1 0 1	Complement A1, A1 will go from 1 to 0 and complement A2.
0 1 1 0	Complement A1.
0 1 1 1	Complement A1. , A1 will go from 1 to 0 and complement A2. A2 will go from 1 to 0 and complement A3. A3 will go from 1 to 0 and complement A4.
1 0 0 0	And so on.

Counter

BCD Ripple Counter:



Diag. BCD Ripple Counter

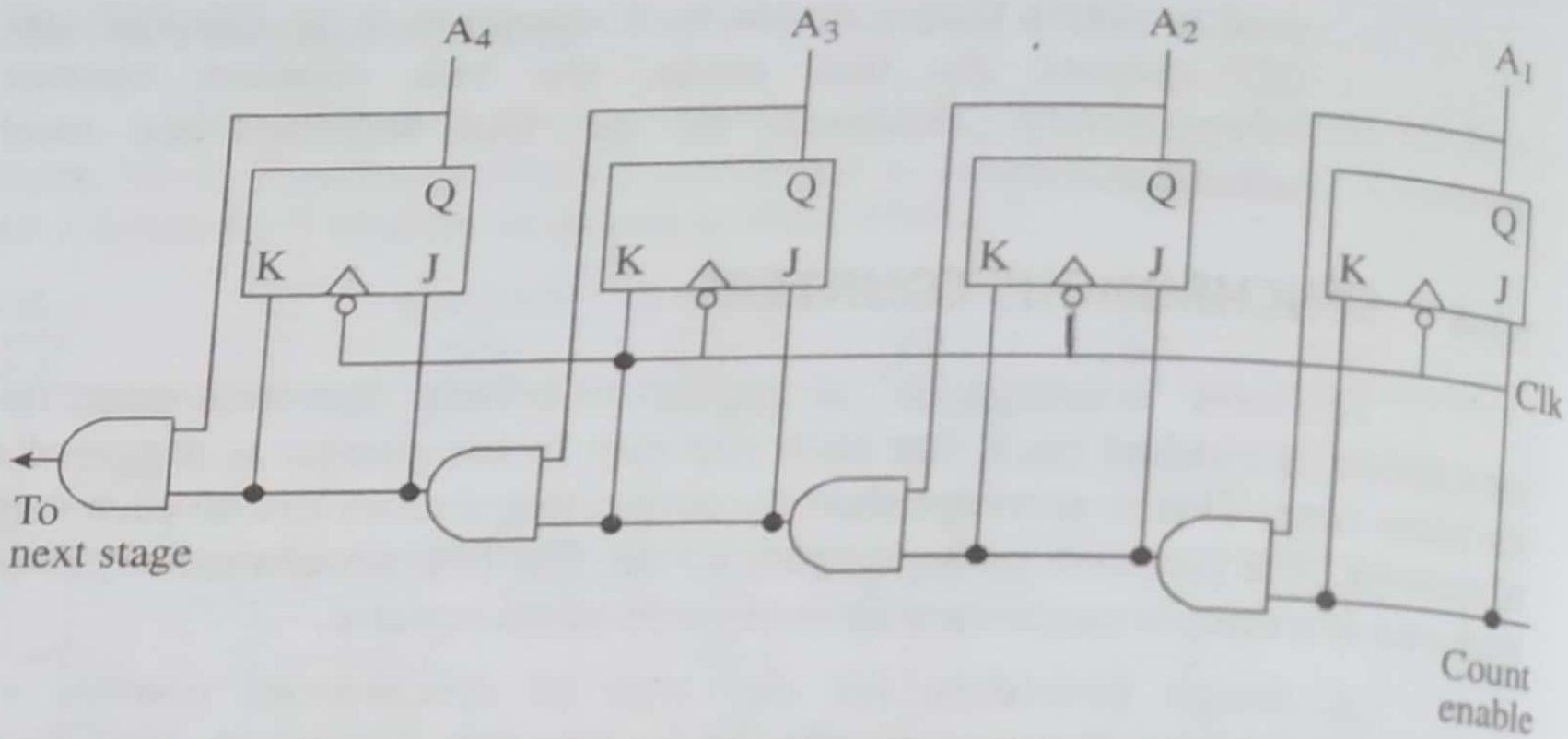
Counter

Condition for each flip-flop state transition are as accordingly:

1. On negative edge of every count pulse,Q1 is complemented.
2. When Q8 = 0 and Q1 goes from 1 to 0 then Q2 is complemented. Similarly when Q8 = 1 and Q1 goes from 1 to 0 then Q2 becomes cleared.
3. When Q2 goes from 1 to 0 then Q4 is complemented.
4. When Q4Q2 = 11 and Q1 goes from 1 to 0 then Q8 is complemented. Similarly, when if either Q4 or Q2 is 0 and Q1 goes from 1 to 0 then Q8 becomes cleared.

Counter

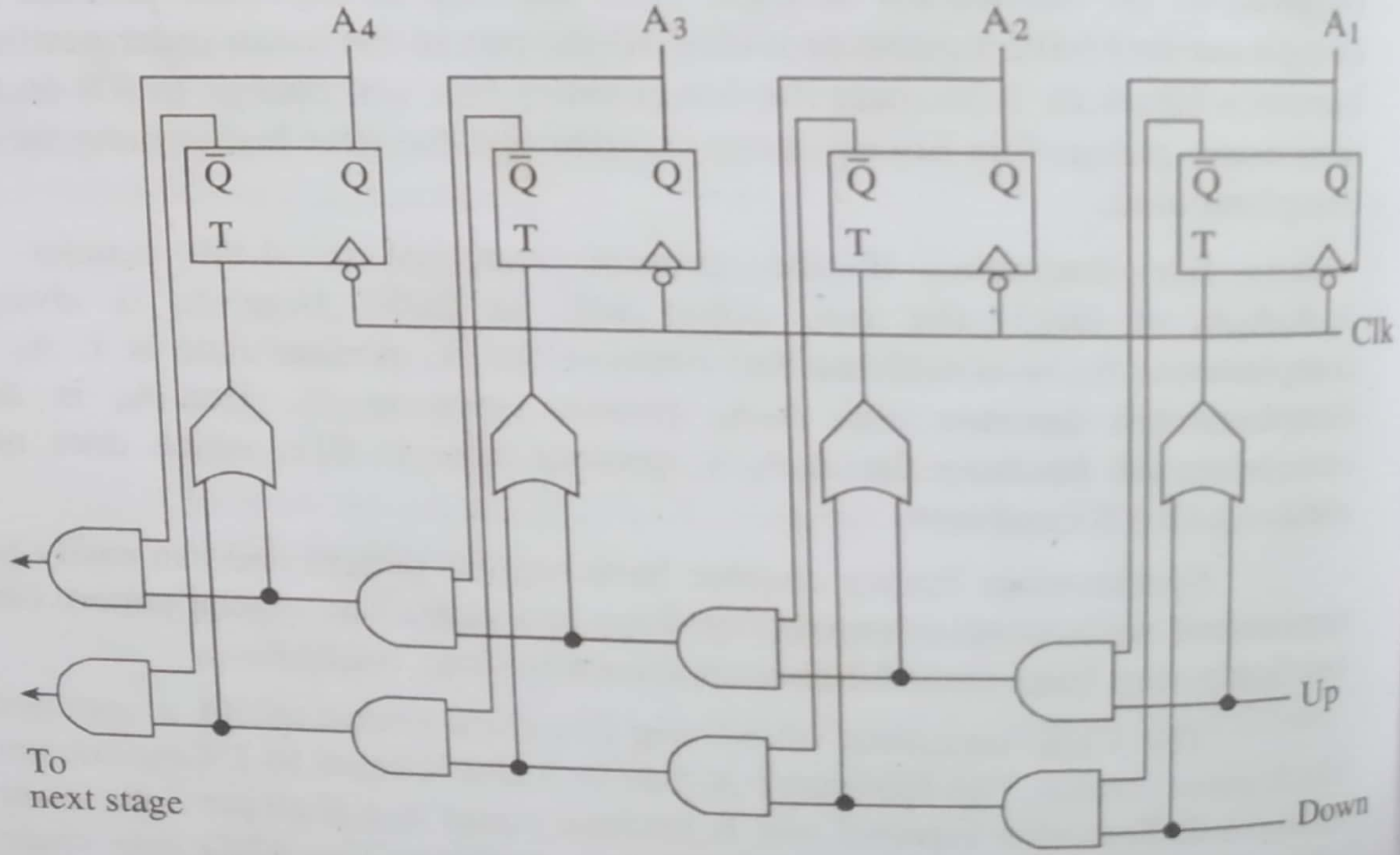
A 4-bit synchronous counter:



Diag. 10010.8: 4-bit synchronous binary counter

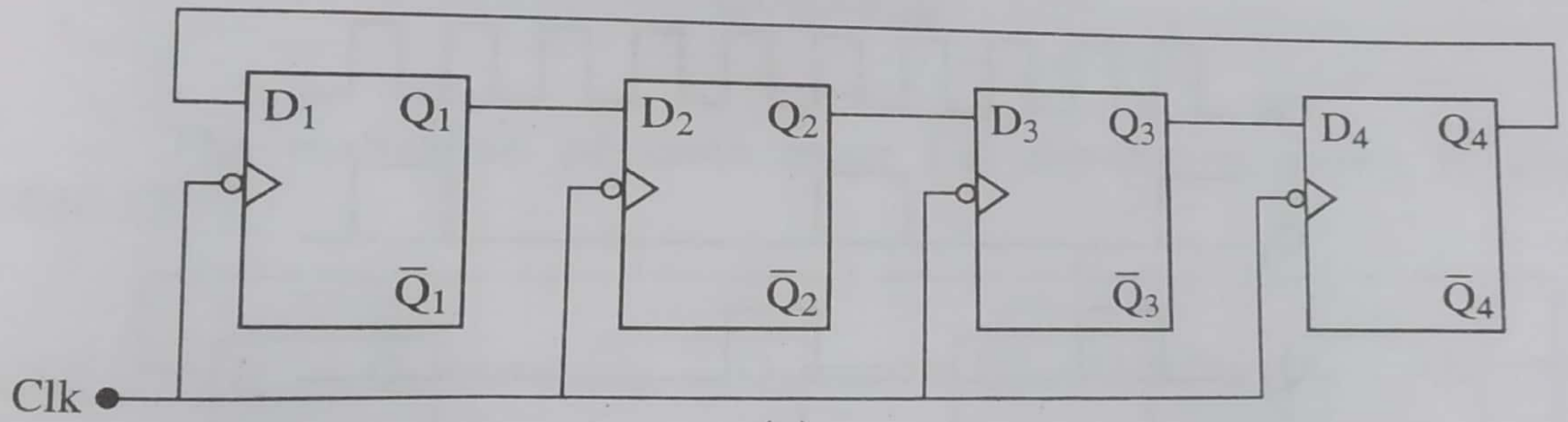
Once can extend the counter for any number of stages, with each stage having an additional flip-flop and an AND gate that gives an output of 1 if all previous flip-flops outputs are 1'S.

Counter

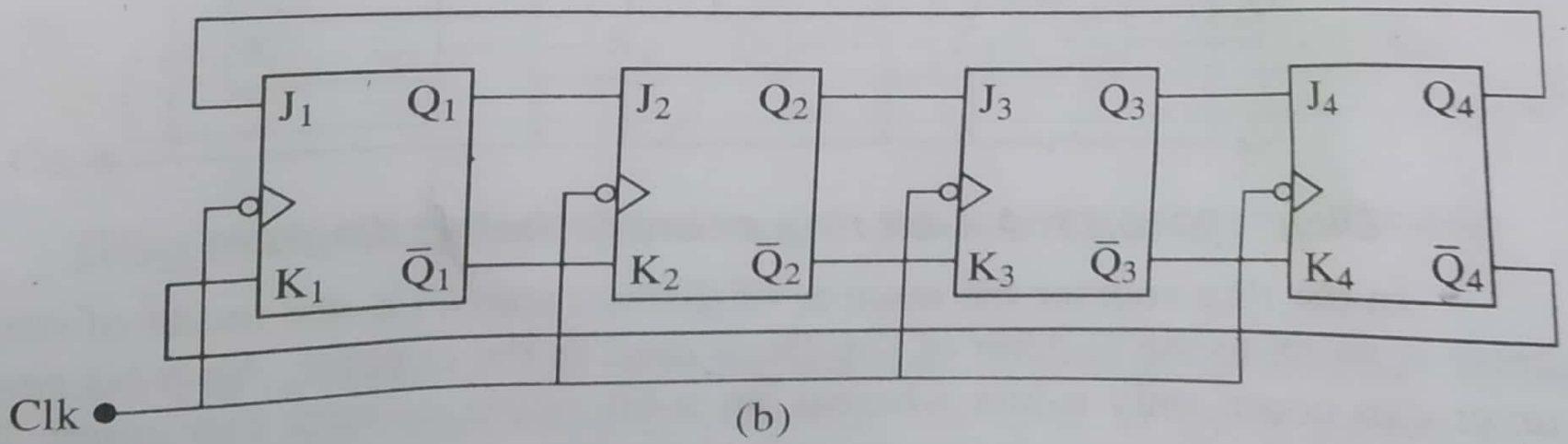


Diag. 10010.9: A 4-bit up/down synchronous counter

Counter



(a)

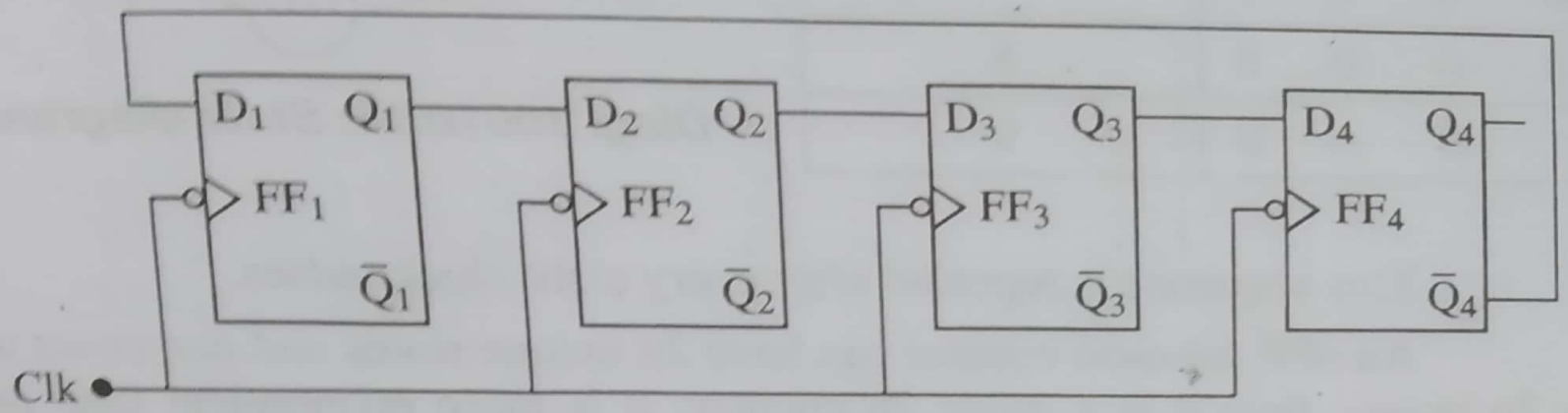


(b)

Diag. 10010.10: (a) A 4-bit ring counter using D-flip-flop (b) A 4-bit ring counter using JK-flip-flop

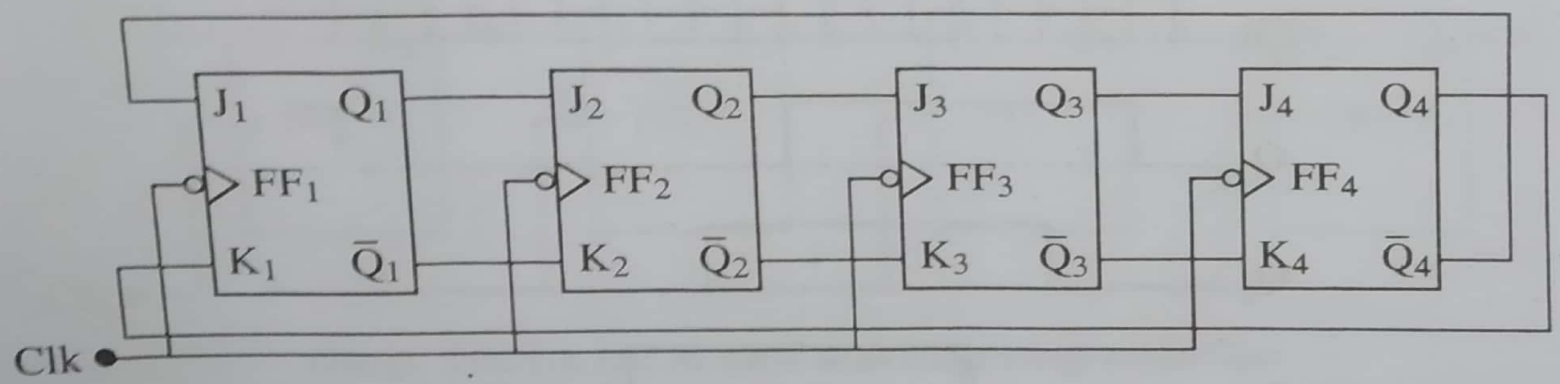
Table No. 10010.11: Sequence table

Counter



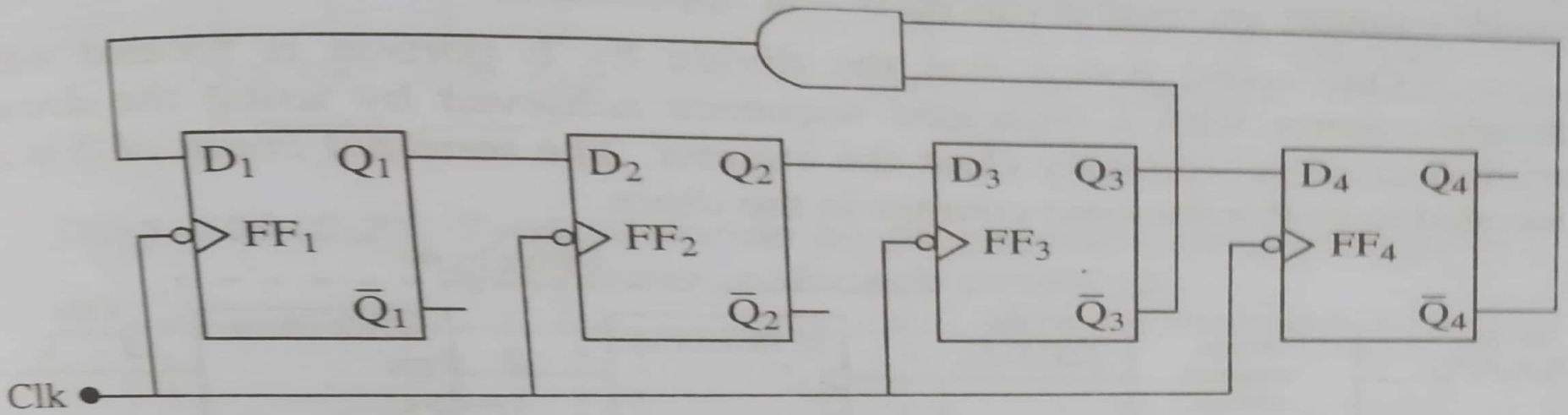
Diag. 10010.14: A 4-bit twisted ring (Johnson) counter using D-flip-flop

The realization of same using J-K flip-flop is shown in diag. 10010.13.

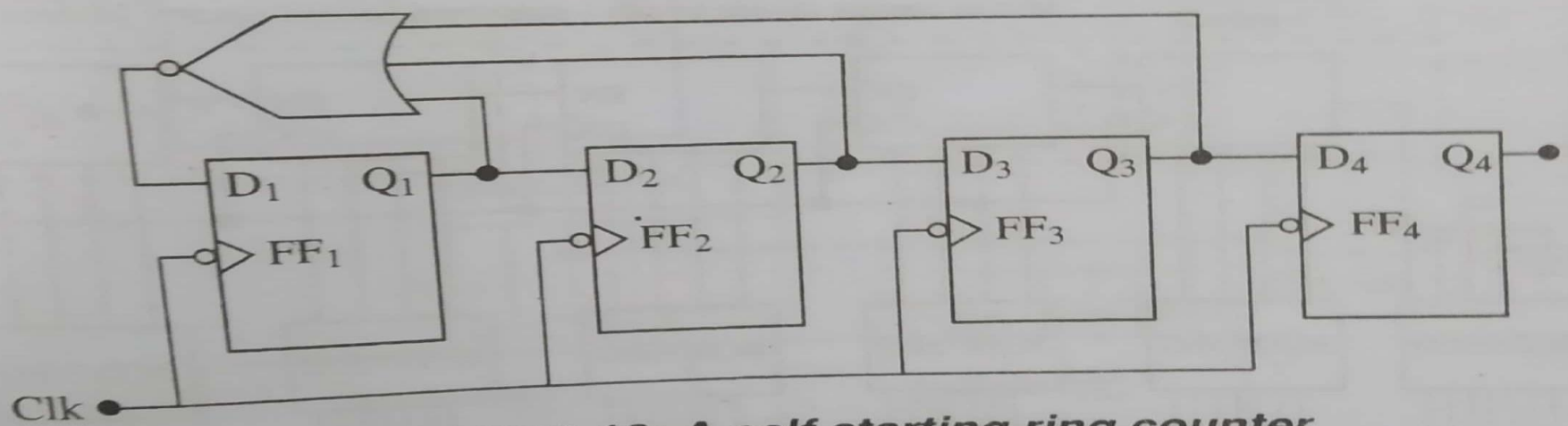


Diag. 10010.15: A 4-bit twisted ring counter using J-K flip-flop

Counter



Diag. 10010.18: Johnson counter design to prevent lock-out



Diag. 10010.19: A self starting ring counter

Counter Application

Counter Application:

- Digital clock
- Auto parking control
- Parallel to serial data conversion
- Frequency counter