جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني

Department of Cyber Security

Subject:

COMPUTER ORGANIZATION & LOGIC DESIGN

Class:

First

Lecture: (6)

Number systems

(excess-3, gray code, conversions, operations, complements) BCD:

Lecturer:

Msc :Muntather AL-mussawee

Number systems

(excess-3, gray code, conversions, operations, complements) BCD:

BCD or **Binary Coded Decimal**

is a coding scheme used to represent decimal number (0 to 9) in the form of binary digits of a group of 4-bits. Binary coded decimal is the simplest form to convert decimal numbers into their equivalent binary format. Although, binary coded decimal or BCD is not the same as the normal binary representation. In binary coded decimal (BCD) coding scheme, each decimal digit is represented as a group of 4-bit binary number. For a multi-digit decimal number, each digit of the decimal number is encoded separately in the BCD.

As we know, a 4-bit binary number can represent 16 decimal digits, but in binary coded decimal, BCD codes 1010, 1011, 1100, 1101, 1110, and 1111 equivalent to decimal 10, 11, 12, 13, 14, and 15 are considered illegal combinations.

كما نعلم، يمكن أن يمثل الرقم الثنائي المكون من 4 بتات 16 رقمًا عشرياً، ولكن في النظام العشري المشفر ثنائياً، تكون رموز

BCD codes 1010, 1011, 1100, 1101, 1110, and 1111 مكافئة الى النظام الرقمي العشري 10, 11,

12,13,14 and 15 ((تعتبر مجموعات غير قانونية)).

The following is the truth table representing binary coded decimal (BCD) equivalent of decimal digits from 0 to 9 –

Decimal Digit	Binary Coded Decimal
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example: Decimal to Binary Coded Decimal Conversion

The following example shows how a decimal number is converted to a BCD code – Convert $(125)_{10}$ into its equivalent binary coded decimal (BCD) code.

Decimal number	1				2				5			
BCD Weights	8	4	2	1	8	4	2	1	8	4	2	1
BCD Code	0	0	0	1	0	0	1	0	0	1	0	1

Hence, the binary coded decimal for $(125)_{10}$ is $(0001\ 0010\ 0101)_2$.

Excess-3 code:

The excess-3 code (or XS3) is a non-weighted code used to express code used to express decimal numbers. It is a self-complementary binary coded decimal (BCD) code and numerical system which has biased representation. It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.

Excess-3 arithmetic uses different algorithm than normal non-biased BCD or binary

positional number system.

Representation of Excess-3 Code

Excess-3 codes are unweighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit. An Excess-3 equivalent of a given binary binary number is obtained using the following steps:

- Find the decimal equivalent of the given binary number.
- Add +3 to each digit of decimal number.
- Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.

You can add 0011 to each four-bit group in binary coded decimal number (BCD) to get desired excess-3 equivalent.

These are following excess-3 codes for decimal digits −

Decimal Digit	BCD Code	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

The codes 0000 and 1111 are not used for any digit.

Example-1 −Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the **decimal number** then convert into 4-bit binary number for result of each digit. Therefore,

= 23+33=56 =0101 0110 which is required excess-3 code for given decimal number 23.

Example-2 −Convert decimal number 15.46 into Excess-3 code.

According to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

= 15.46+33.33=48.79 =0100 1000.0111 1001 which is required excess-3 code for given decimal number 15.46.

Advantages of Excess-3 Codes

These are following advantages of Excess-3 codes,

- These are unweighted binary decimal codes.
- These are self-complementary codes.
- These use biased representation.
- The codes 0000 and 1111 are not used for any digit which is an advantage for memory organization as these codes can cause fault in transmission line.
- It has no limitation, and it considerably simplifies arithmetic operations.
- It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.

Gray Codes:

Gray code is a form of binary and the most popular absolute encoder output type. This lecture will explain Gray Code, discuss converting Gray Code to Binary, explain how to use software to convert to gray code, and converting Gray Code to Binary using logic (XOR). Gray Code uses a different method of incrementing from one number to the next.

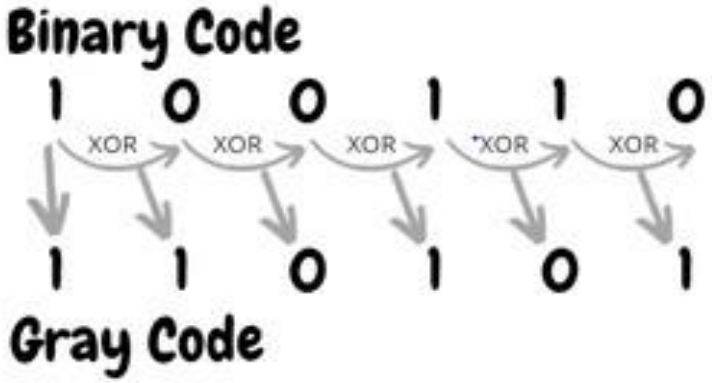
With Gray Code, only one bit changes state from one position to another. This feature allows a system designer to perform some error checking (i.e., if more than one bit changes, the data must be incorrect). In table below, explain the difference between Binary and Gray Code. Gray Code is the most popular absolute encoder output type because its use prevents certain data errors that can occur with Binary during state changes. For example, in a highly capacitive circuit (or sluggish system response), a Binary state change from 0011 to 0100 could cause the counter/PLC to see 0111. This sort of error is not possible with Gray Code, so the data is more reliable.

Gray Code					Binary			
2^3	2^2	2^1	2^0	Position	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1
0	0	1	1	2	0	0	1	0
0	0	1	0	3	0	0	1	1
0	1	1	0	4	0	1	0	0
0	1	1	1	5	0	1	0	1
0	1	0	1	6	0	1	1	0
0	1	0	0	► 7 ◄	0	1	1	1
1	1	0	0	► 8 ◄	1	0	0	0
1	1	0	1	9	1	0	0	1
1	1	1	1	10	1	0	1	0
1	1	1	0	11	1	0	1	1
1	0	1	0	12	1	1	0	0
1	0	1	1	13	1	1	0	1
1	0	0	1	14	1	1	1	0
1	0	0	0	15	1	1	1	1

Note that even from position 7 to 8, Gray Code only changes one bit state.

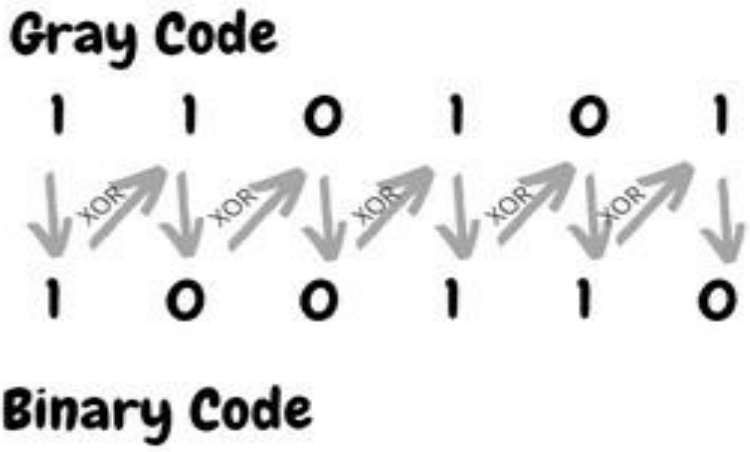
Binary to Gray conversion :

1. The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
2. Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.
3.



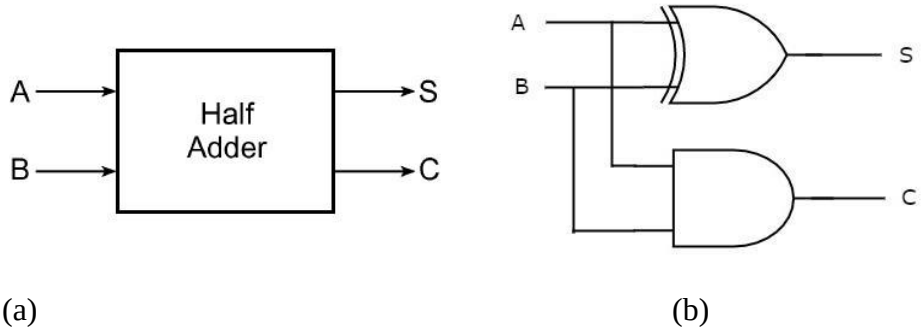
Gray to binary conversion :

1. The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.
2. Other bits of the output binary code can be obtained by checking the gray code bit at that index. If the current gray code bit is 0, then copy the previous binary code bit, else copy the invert of the previous binary code bit.



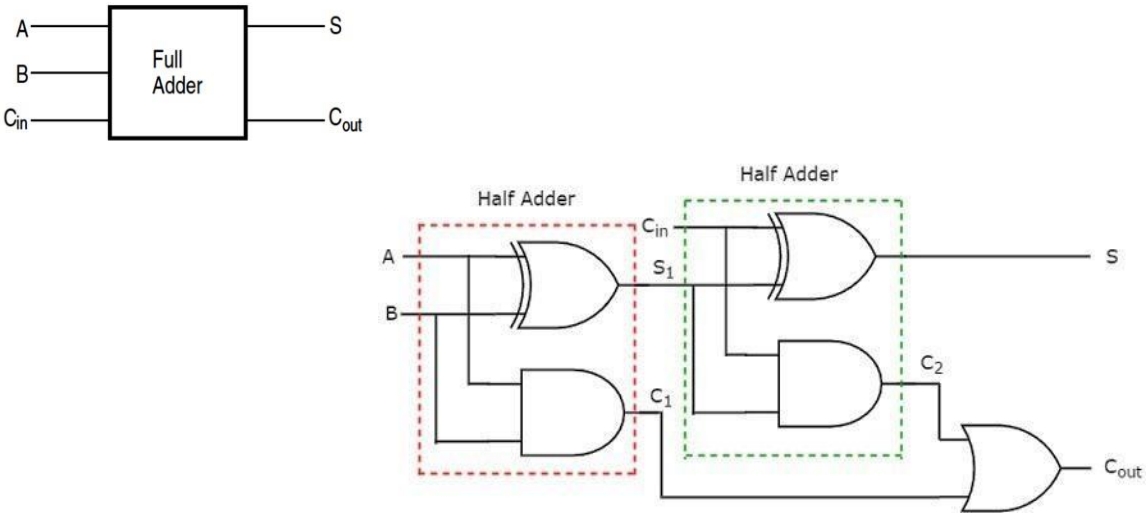
Arithmetic operations (half adder, full adder circuit)

Half Adder and Full Adder



(a) (b)

Half Adder. لدائرة (Block diagram) مخطط الكتلة b(Half Adder(الدائرة المنطقية التي تمثل a)



(b) (a)

a(الدائرة المنطقية التي تمثل Full Adder)b(مخطط الكتلة (Block diagram) لدائرة Full Adder.

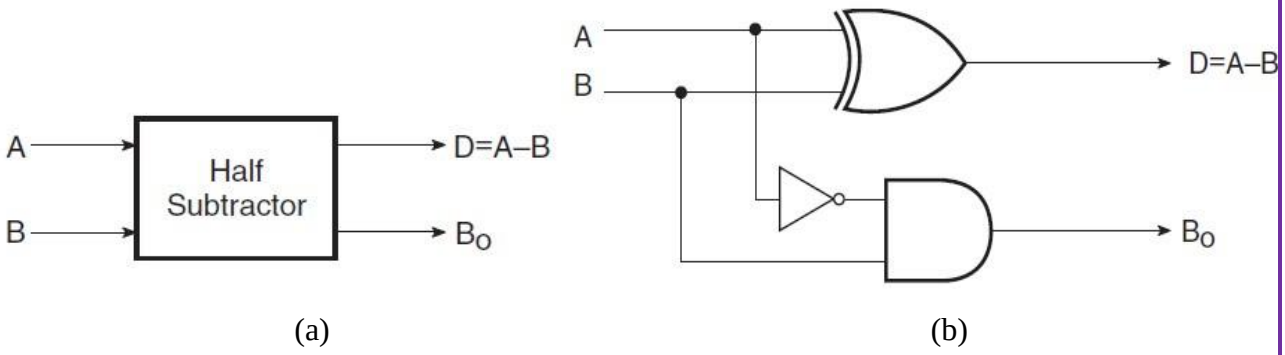
جدولHalf Adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

جدولFull adder

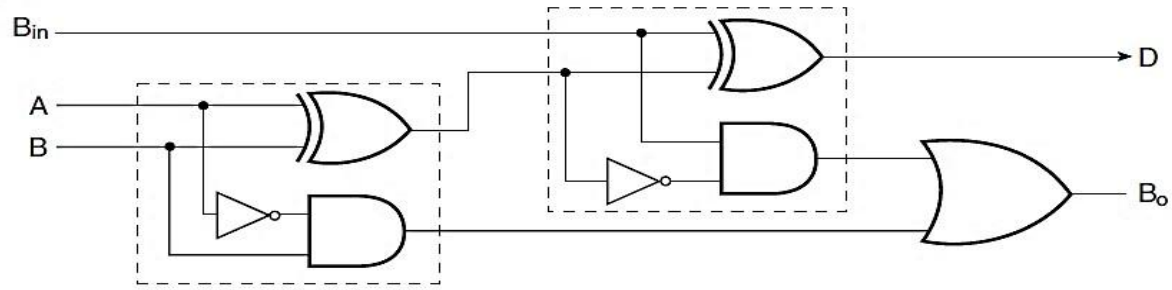
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

Arithmetic operations (Half subtractor, Full subtractor circuit) Half Subtractor and Full Subtractor



(a) (b)

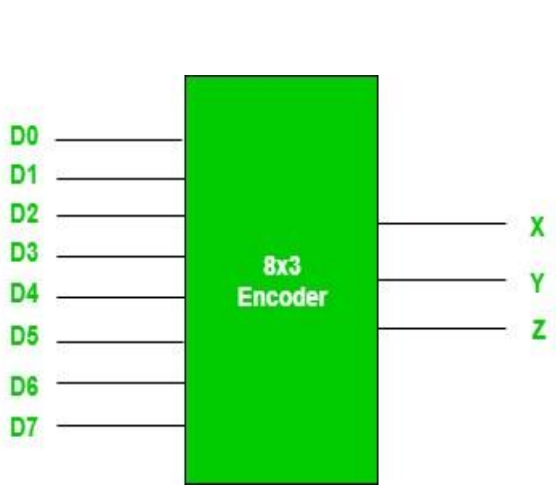
الشكل 1 a) الدائرة المنطقية التي تمثل Half Subtractor(b مخطط الكتلة)(Block diagram) لدائرة Half Subtractor.

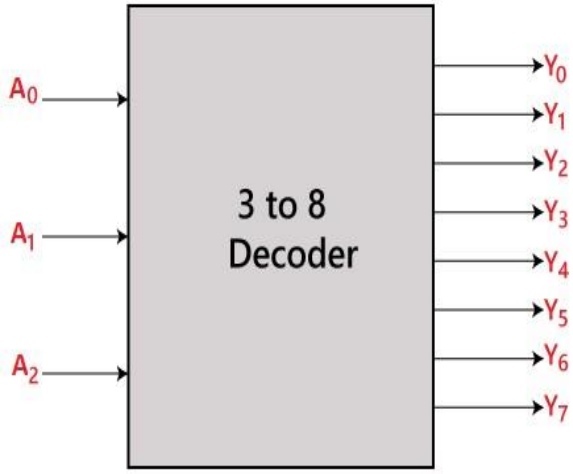


(a)

(b)

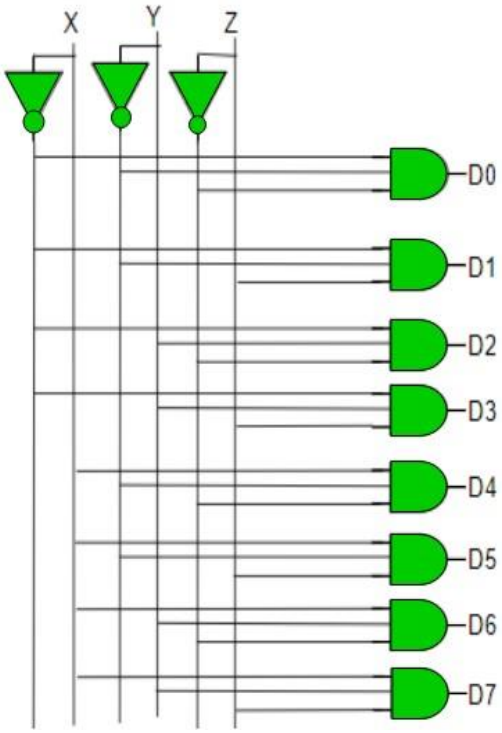
(decoder, encoder)





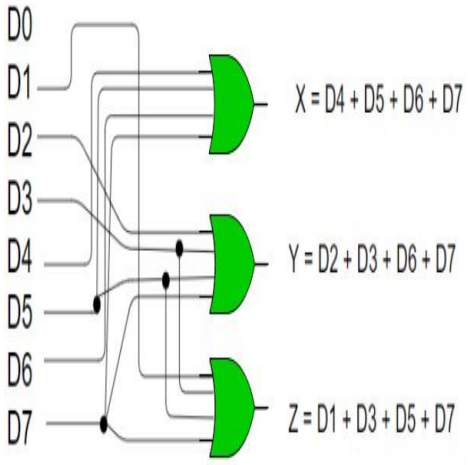
Truth Table Decoder:

X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

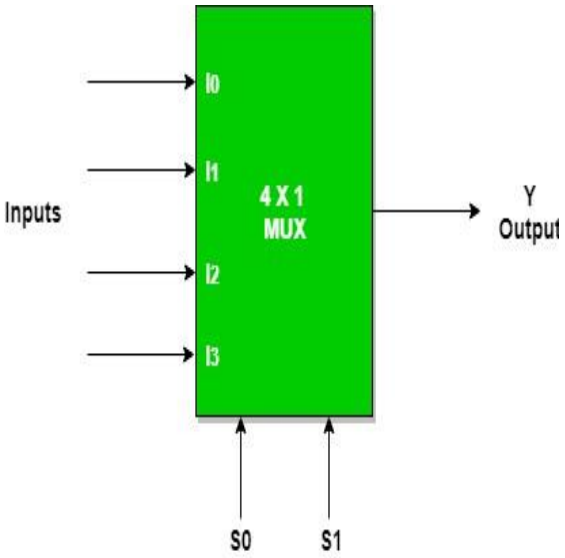


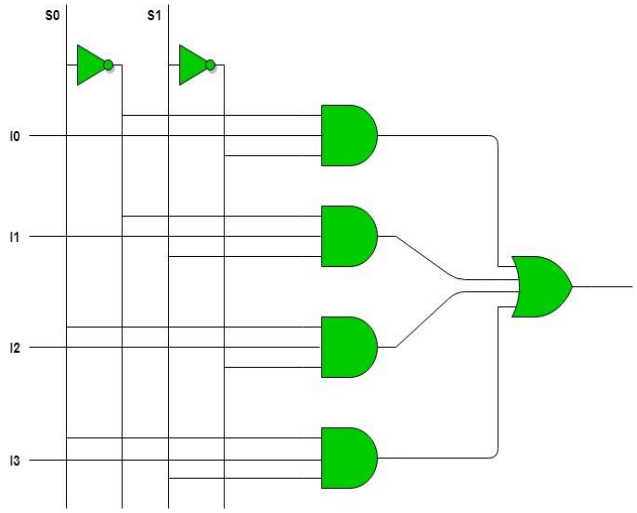
Truth Table Encoder:

D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



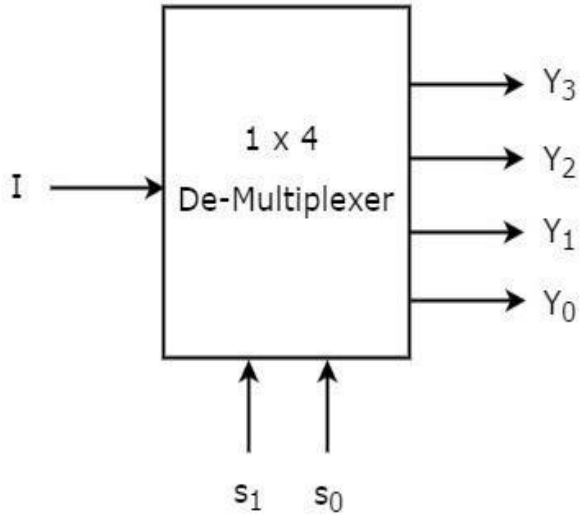
Multiplexer

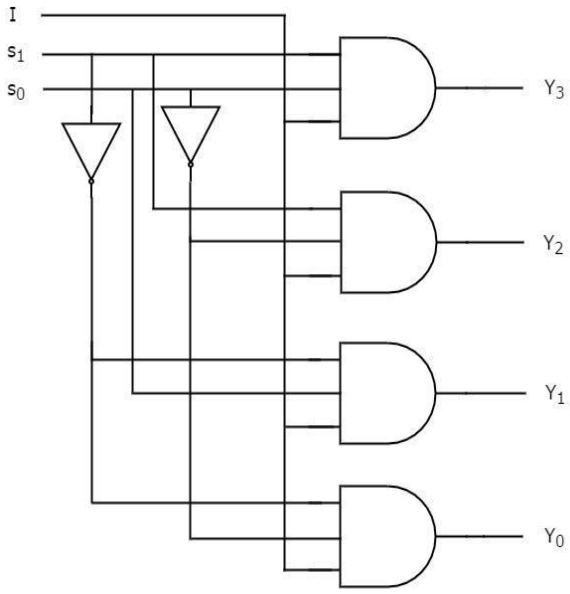




Inputs		Outputs
S_1	S_0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

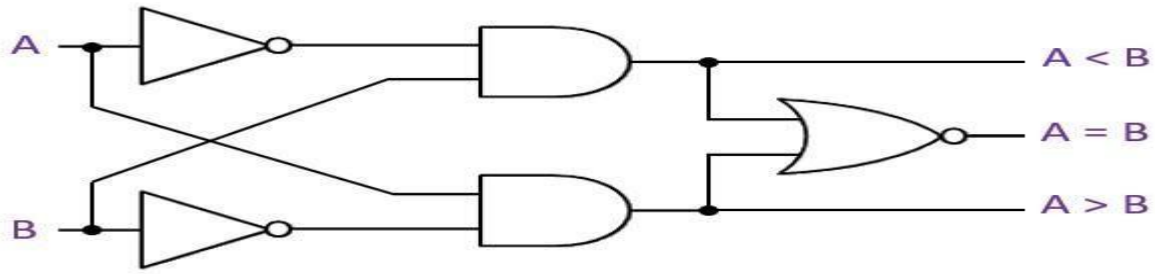
Demultiplexer

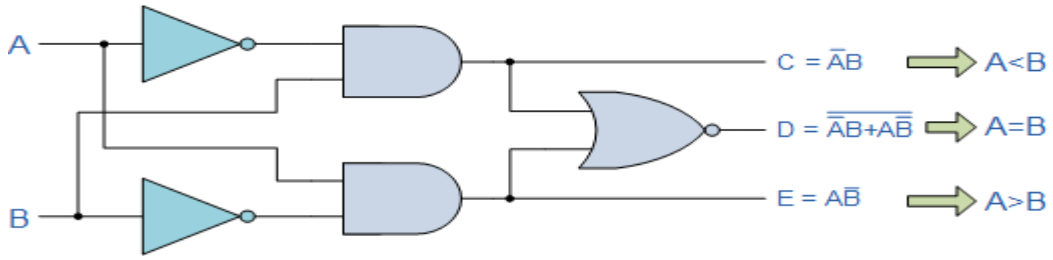




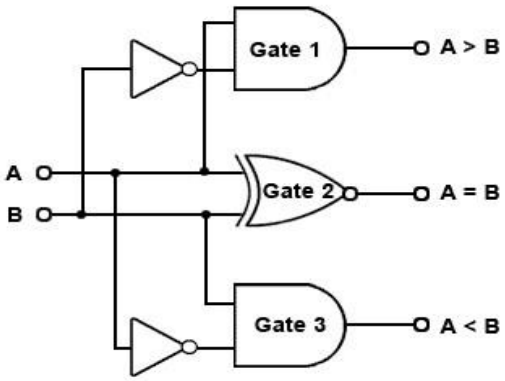
Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

Comparators





OR



Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0