Digital Techniques

1. Binary System

The digital computer is the best example of a digital system. A main characteristic of digital system is its ability to manipulate discrete elements of information.

Discrete elements of information are represented in a digital system by physical quantities called signals. Electrical signals such as voltages and currents are the most common. A block diagram of the digital computer is shown in Figure (1). The memory unit stores programs as well as input, output, and intermediate data. The processor unit performs arithmetic and other data processing tasks as specified by a program. The control unit supervise the flow of information between the various units. The programmed data prepared by the user are transferred into the memory unit by means of an input device such as keyboard, punch-card reader. An output device such as a printer.

The central processor unit (CPU) is enclosed in a small integrated circuit that is called a microprocessor. The memory unit, as well as the part that controls the interface between the microprocessor and the I/O devices is called a microcomputer. The digital computer manipulates discrete elements of information and that these elements are represented in the binary form.

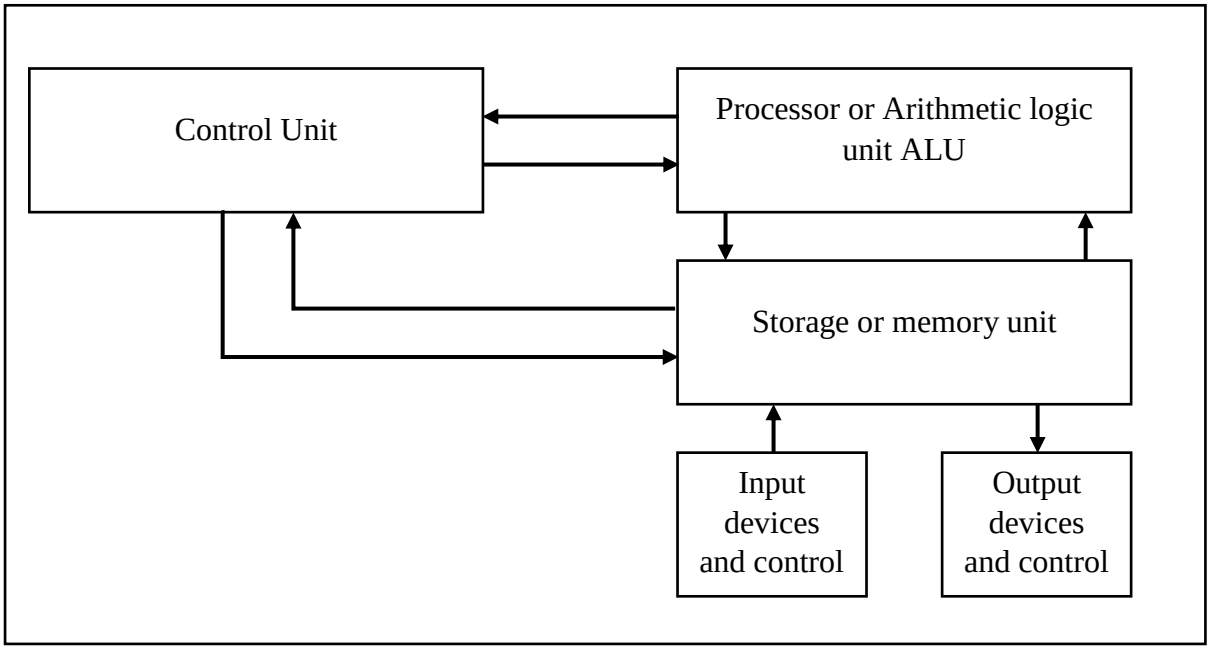


Figure 1: Block diagram of digital computer

2. Binary numbers

The binary numbers system has two possible values: "0" and "1", called a **Bit**. The physical manifestation of these binary quantities may be one of two voltages, for example, "0" volts or ground for logic "0" and "5" volts for logic "1". To convert from binary to decimal. Each coefficient a_j is multiplied by 2^j. For example, the decimal equivalent of the binary number $110)_2$ is $6)_{10}$ as shown from the multiplication of the coefficients by powers of 2:

$$2^2 \times 1 + 2^1 \times 1 + 2^0 \times 0 = 4 + 2 + 1 = 6$$

So **$110)_2$ equals to $6)_{10}$**

The conversion from decimal to binary by dividing of by "2" to give an integer quotient of no. and a remainder. The division quotient is again divided by "2" to give a new quotient and remainder. The process is continued until the integer quotient becomes "0".

Example 1: Convert decimal $6)_{10}$ to binary

Integer			Remainder	
6	2	→	0	L.S.B Least Significant Bit
3	2	→	1	
1		→	1	M.S.B Most Significant Bit

So **$6)_{10}$ equals to $110)_2$**

Example 2: Convert $41)_{10}$ to binary

Integer			Remainder	
41	2	→	1	L.S.B Least Significant Bit
20	2	→	0	
10	2	→	0	
5	2	→	1	
2	2	→	0	
1		→	1	M.S.B Most Significant Bit

So **$41)_{10}$ equals to $101001)_2$**

32	16	8	4	2	1
1	0	1	0	0	1

1+8+32=41

Example 3: Convert $0.75)_{10}$ to binary

	Integer	Fraction
o.75×2 = 1.5	1	0.5
0.5×2 = 1	1	0

So **$0.75)_{10}$ equals to $0.11)_2$**

Example 4: Convert $9.6875)_{10}$ to binary

The integer

Integer			Remainder	
9	2	→	1	L.S.B Least Significant Bit
4	2	→	0	
2	2	→	0	
1		→	1	M.S.B Most Significant Bit

The fraction

	Integer	Fraction
o.6875×2 = 1.375	1	0.375
0.375×2 = 0.75	0	0.75
0.75×2=1.5	1	0.5
0.5×2=1	1	0

So **$9.6875)_{10}$ equals to $1001.1011)_2$**

Example 4: Convert $11010.11)_2$ to decimal

$$[(2^4\times1)+(2^3\times1)+(2^2\times0)+(2^1\times1)+(2^0\times0)].[(2^{-1}\times1)+(2^{-2}\times1)]=$$

$$[16+8+0+2+0].[0.5+0.25] = 26.75$$

So **$11010.11)_2$ equals to $26.75)_{10}$**

3. Logic Gates

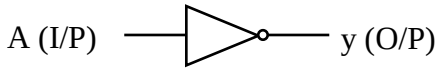
A gate is a circuit with one output that implements one of the basic functions such as inverter, OR and AND gates. Gates are available with one, two, three, four, and eight inputs.

Truth table (T.T): is a table of all possible combination of the variables showing the relation between the values that the variables may take and the result of the operation.

3.1. Inverter "Not" gate

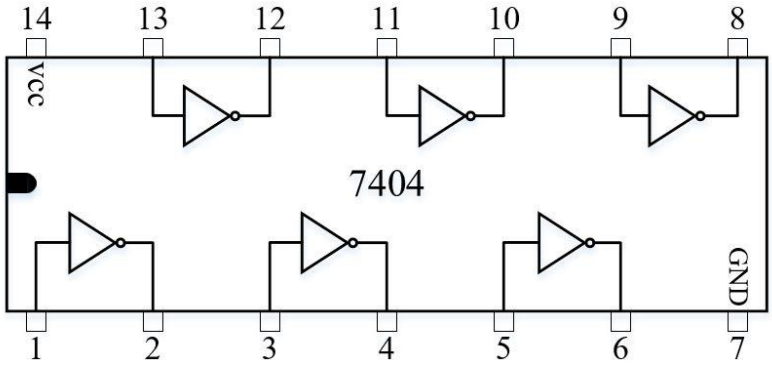
An inverter is a gate with only one input and one output. It is called an inverter because the output state is always the opposite of the input state. Specifically, when the input voltage is high the output is low. On the other hand, when the input voltage is low the output is high.

T.T (Truth Table)	
I/P (A)	O/P (y)
0	1
1	0



Symbol for digital logic cct.

y=Ā **Algebraic Function**

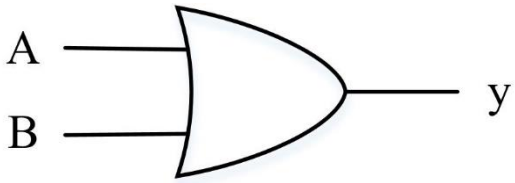


Integrated Circuit IC

3.2. OR Gate

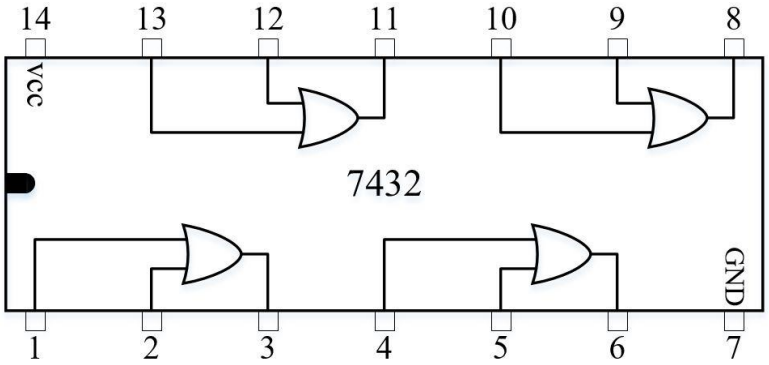
An OR gate has two or more input signals but only one output signal. It is called an OR gate because the output voltage is high if any or all of the input voltages are high.

T.T (Truth Table)		
I/Ps		O/P
A	B	y
0	0	0
0	1	1
1	0	1
1	1	1



Symbol for digital logic cct.

y=A+B **Algebraic Function**

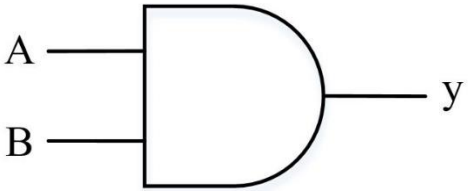


Integrated Circuit IC

3.3. AND Gate

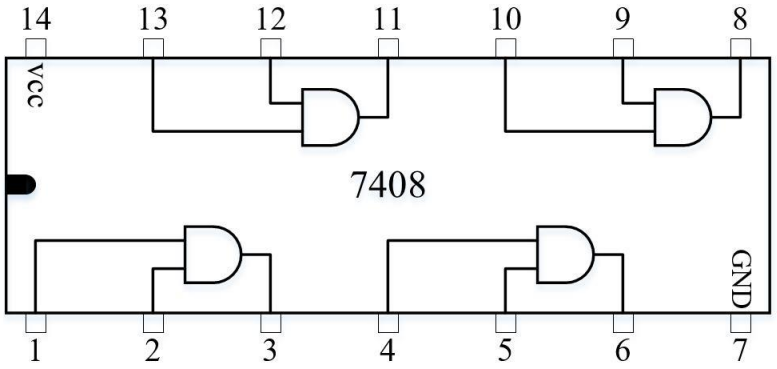
The AND gate has a high output only when all inputs are high. In other words the AND gate is an all or nothing gate, a high output occurs only when all inputs are high.

T.T (Truth Table)		
I/Ps		O/P
A	B	y
0	0	0
0	1	0
1	0	0
1	1	1



Symbol for digital logic cct.

y=A.B **Algebraic Function**

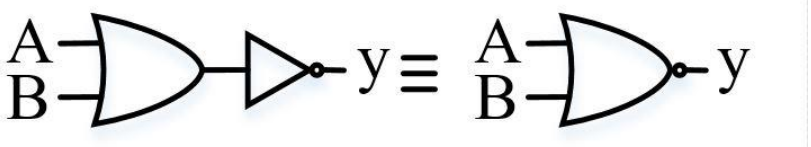


Integrated Circuit IC

3.4. NOR Gate

Read this as **y equals Not(A OR B)**. The only way to get a high output is to have both inputs low.

T.T (Truth Table)		
I/Ps		O/P
A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

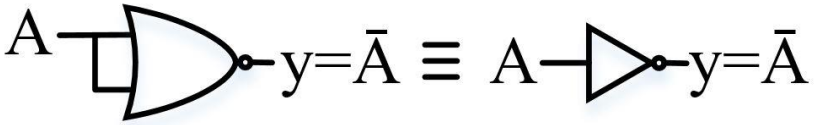


Symbol for digital logic cct.

$y=\overline{A+B}$ **Algebraic Function**

If A=B i.e. I/Ps are shorted

T.T (Truth Table)		
I/Ps		O/P
A=	B	y
0	0	1
1	1	0
0	0	1
1	1	0

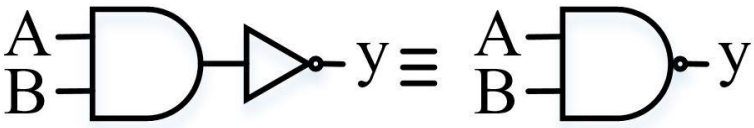


So, we get an inverter (Not gate)

3.5. NAND Gate

Read this as **y equals Not(A AND B)**. The only way to get a low output is for both inputs to be high.

T.T (Truth Table)		
I/Ps		O/P
A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

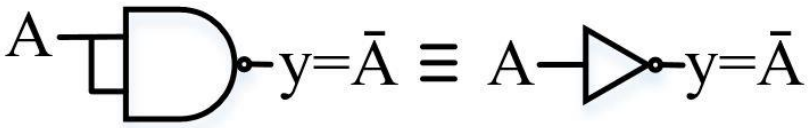


Symbol for digital logic cct.

$y=\overline{A.B}$ **Algebraic Function**

If A=B i.e. I/Ps are shorted

T.T (Truth Table)		
I/Ps		O/P
A=	B	y
0	0	1
1	1	0
0	0	1
1	1	0



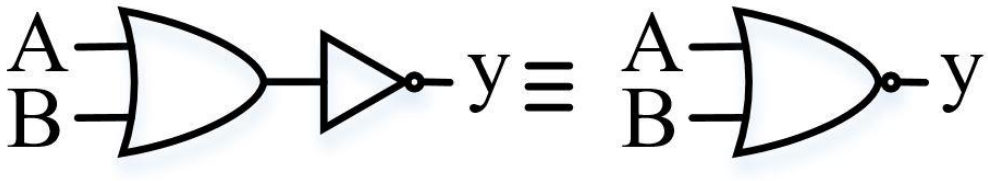
So, we get an inverter (Not gate)

4. De-Morgan's Theorems

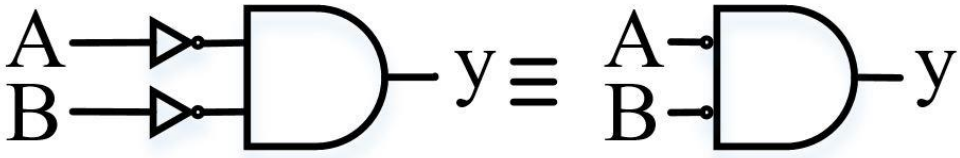
Much of today logic implementation is based upon a set of rules called De-Morgan's theorems. Basically these theorems demonstrate that, any logic function can be formed from AND and NOT gates (NAND) or from OR and NOT gates (NOR). These rules can be summarized as follows:

4.1. The complement of sum equals the product of the complements i.e.:

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



$y = \overline{A+B}$ *NOR Gate*



$y = \bar{A}.\bar{B}$ Bubbled AND Gate

Truth Table T.T.

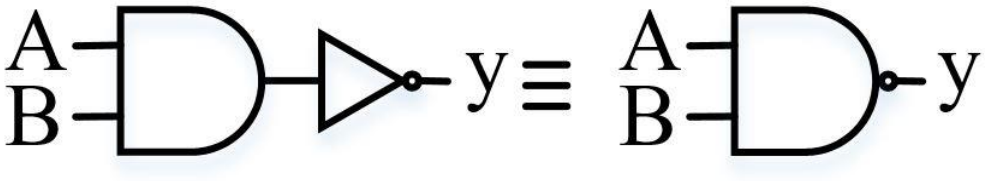
A	B	$y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

$\bar{A}$	$\bar{B}$	$y = \bar{A}.\bar{B}$
1	1	1
1	0	0
0	1	0
0	0	0

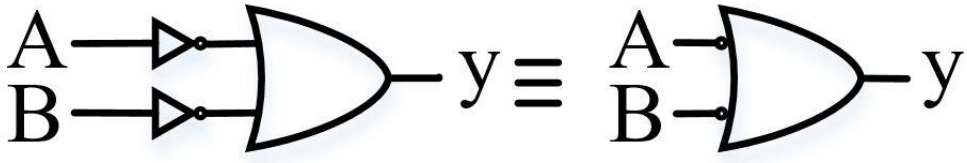
A NOR gate and bubbled AND gate are equivalent

4.2. The complement of a product equals the sum of the complement

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



$y = \overline{A.B}$ *NAND Gate*



$y = \bar{A}+\bar{B}$ Bubbled OR Gate

Truth Table T.T.

A	B	$y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

$\bar{A}$	$\bar{B}$	$y = \bar{A}+\bar{B}$
1	1	1
1	0	1
0	1	1
0	0	0

A NAND gate and bubbled OR gate are equivalent

5. Laws and theorems of Boolean Algebra

Properties of Boolean Algebra

When we simplify a complicated Boolean expression we change a complicated digital circuit into simpler one.

5.1. The first group of laws are backbone of Boolean algebra

- x=0 if x≠1
  x=1 if x≠0
- x.y=1 if x=1 and y=1
  =0 otherwise
  x+y=0 if x=0 and y=0
  =1 otherwise
- if x=0 then $\bar{x} = 1$
  if x=1 then $\bar{x} = 0$
- x+0=x
  x.0=0
- x+1=x
  x.1=x
- $x+\bar{x} = 1$
  $x.\bar{x} = 0$
- x+x=x
  x.x=x

5.2. The second group of Boolean algebra

- **Distributive laws**

$x(y + z) = xy + xz$ OR distributive law

$x + yz = (x + y)(x + z)$ AND distributive law

- **Commutative laws**

$x + y = y + x$

$xy = yx$

- **Associative laws**

$(x + y) + z = x + (y + z) = x + y + z$

$(x.y).z = x.(y.z) = x.y.z$

5.3. The third group of laws are the double complement theorem an De-Morgan's theorems

$$\overline{\overline{X}} = X$$

$\overline{x + y + z + w} = \bar{x}.\bar{y}.\bar{z}.\bar{w}$

$\overline{x.y.z.w} = \bar{x}+\bar{y}+\bar{z}+\bar{w}$

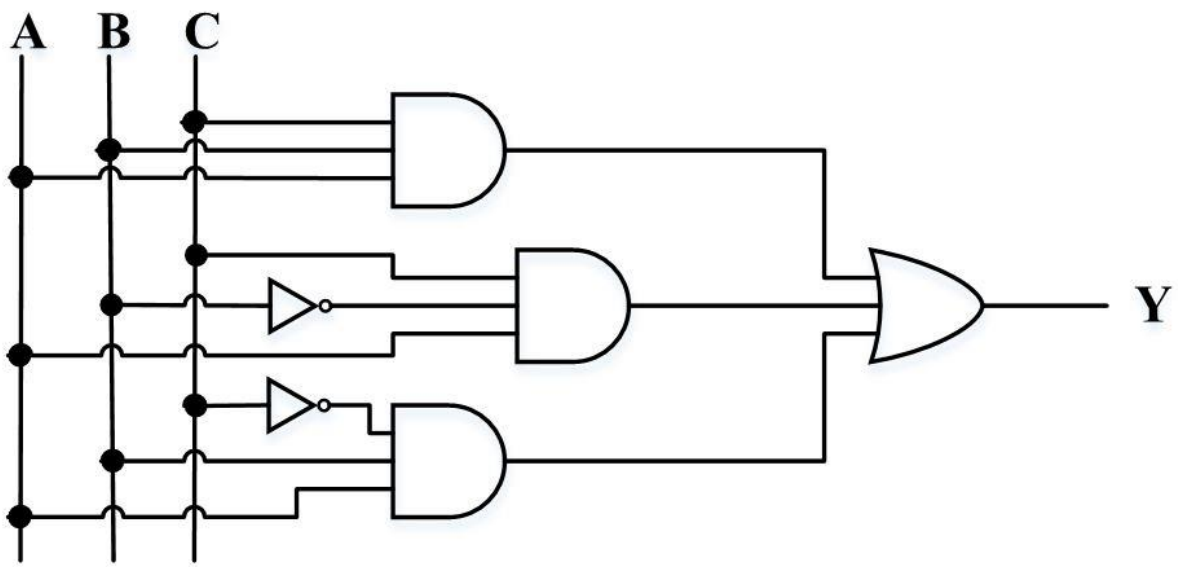
Ex: $\overline{\overline{x + y + z + w}} = \overline{\bar{x}.\bar{y}.\bar{z}.\bar{w}} = \bar{\bar{x}} + \bar{\bar{y}} + \bar{\bar{z}} + \bar{\bar{w}} = x + y + z + w$

5.4. Other theorems

- A+AB=A
- A(A+B)=A
- (A+B)(A+C)=A+BC
- A+$\bar{A}$B=A+B
  L.H.S=A(1+B)+ $\bar{A}$B=A+AB+$\bar{A}$B=A+B(A+$\bar{A}$)=A+B=R.H.S
- A($\bar{A}$+B)=AB
- (A+B)($\bar{A}$+C)=AC+$\bar{A}$B
- AB+$\bar{A}$C=(A+C)($\bar{A}$+B)

Example 5:

To build $y = ABC + A\bar{B}C + AB\bar{C}$



So we need 3-input OR gate, three 3-input AND gates, and two Not gates to build above cct. Using laws and theorems of Boolean algebra to simplify the following expression.

$y = ABC + A\bar{B}C + AB\bar{C} = AB(C + \bar{C}) + A\bar{B}C$ since $C + \bar{C} = 1$

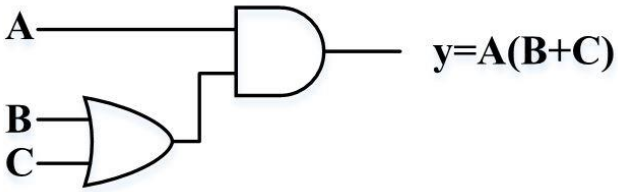
$= AB + A\bar{B}C$

$= AB(1 + C) + A\bar{B}C$

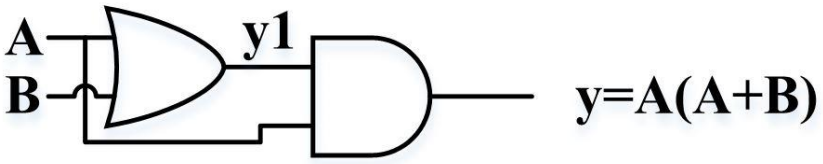
$= AB + ABC + A\bar{B}C$

$= AB + AC(B + \bar{B})$

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

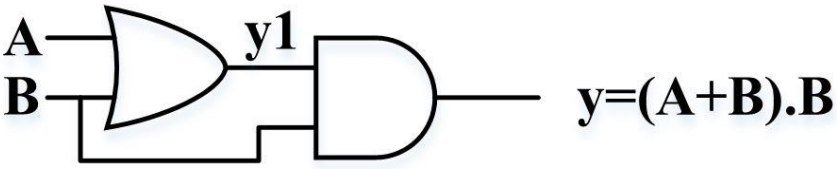


Example 6: write the Boolean equation for the output of the circuit then find the T.T.



T.T (Truth Table)			
B	A	y1	y
0	0	0	0
0	1	1	1
1	0	1	0
1	1	1	1

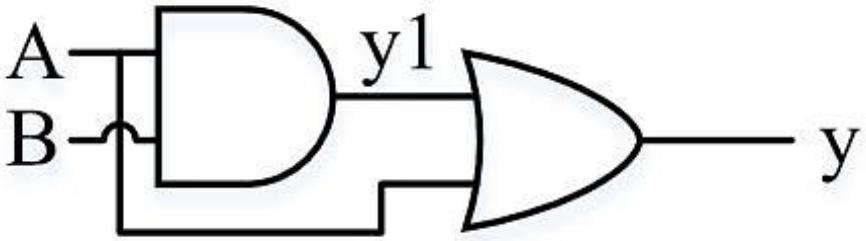
Example 7: write the Boolean equation for the output of the circuit then find the T.T.



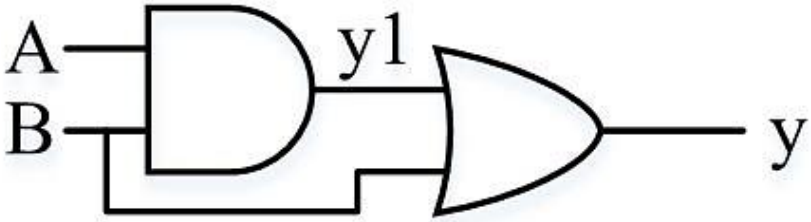
T.T (Truth Table)			
B	A	y1	y
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Homework 1: Write the Boolean equation for the O/P of the circuits then find the truth table T.T.

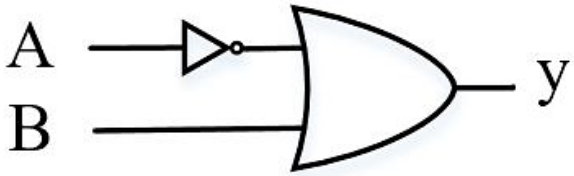
1)



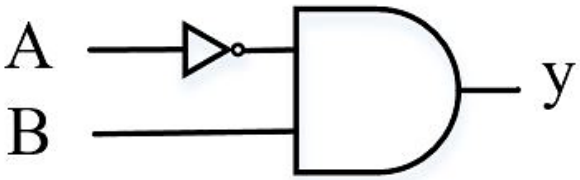
2)



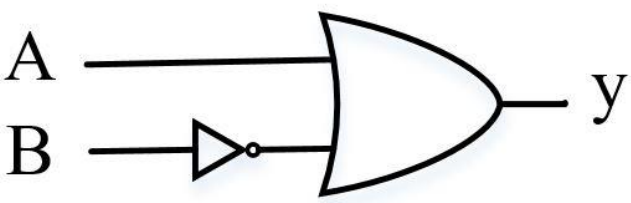
3)



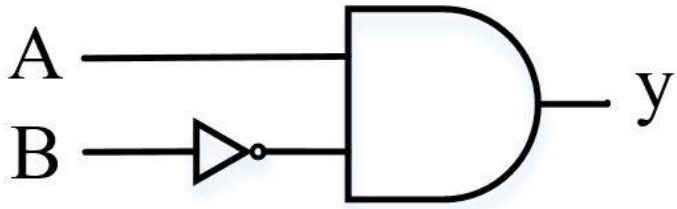
4)



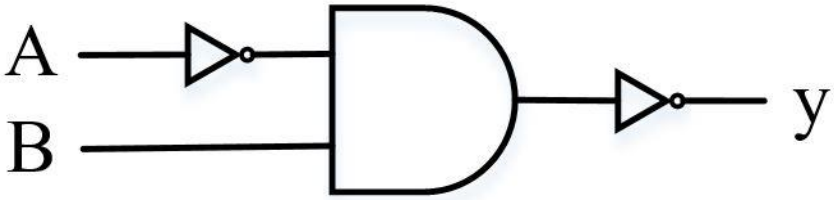
5)



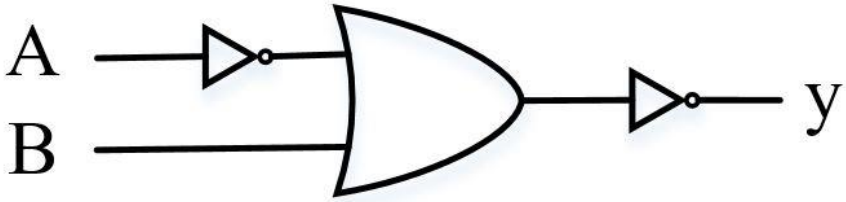
6)



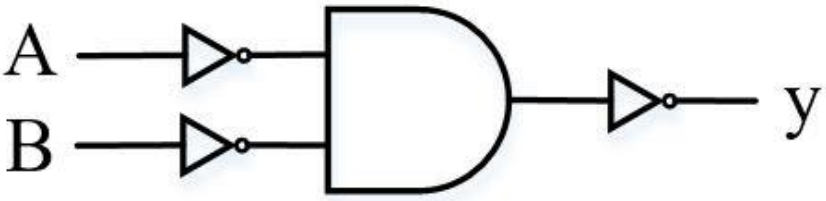
7)



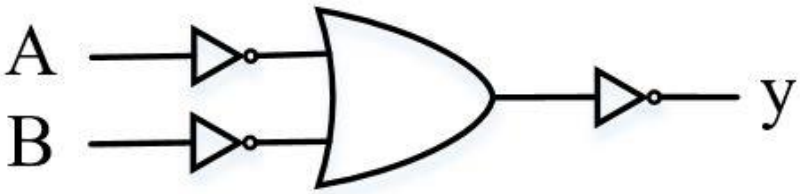
8)



9)



10)



6. Arithmetic Circuits

Digital electronics enables us to build circuits that duplicate some of processes for our minds. By combining AND, OR and NOT circuits in the right way, we can build circuits that add and subtract.

The basics of the arithmetic circuits are:

1. Exclusive OR Gate.
2. Half and Full Adder.
3. Half and Full Subtractor.
4. 8421 adders.

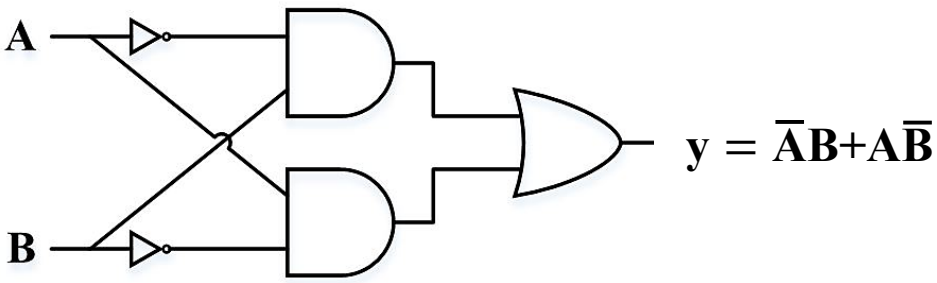
6.1. Exclusive OR (XOR) Gate

XOR gate has two or more input signals but only one output signal. A 2-input XOR has an output of "1" if one of the inputs is "1" and an output of "0" if both inputs are equivalents, i.e. the inputs must be different to get a high O/P.

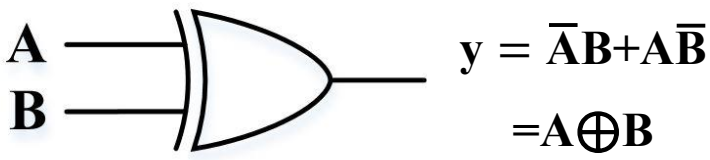
T.T.			
A	B	y	
0	0	0	
0	1	1	$\rightarrow \ \bar{A}B$
1	0	1	$\rightarrow \ A\bar{B}$
1	1	0	

$y=\bar{A}B+A\bar{B}$ Algebraic function

$y=A\oplus B$



|||



Symbol for digital logic circuit

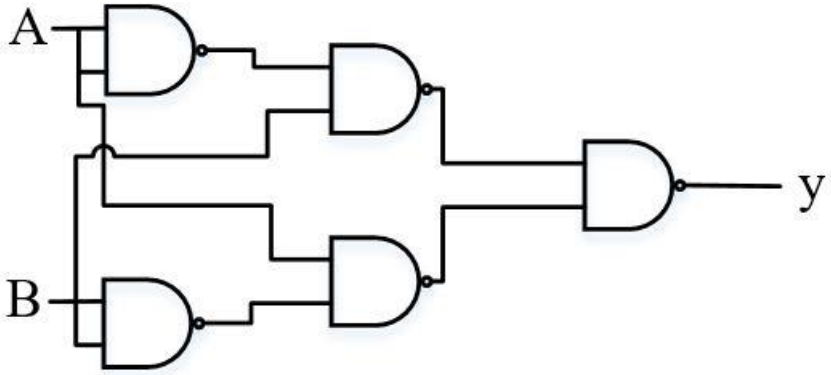
Example 8: Implement the XOR gate using NAND gates only.

Sol:

$y=\bar{A}B+A\bar{B}$ take double complement since $\bar{\bar{X}} = X$

$\bar{\bar{y}} = \overline{\overline{A\bar{B}+\bar{A}B}}$ using De Morgan's theorem

$y=\overline{\overline{A\bar{B}}.\overline{\bar{A}B}}$

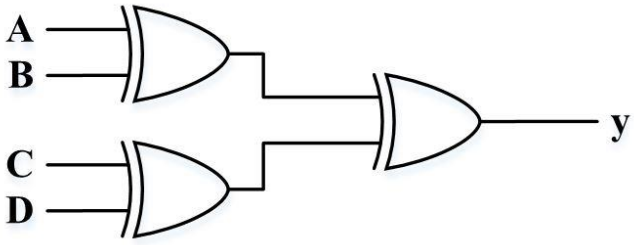


6.2. Applications of XOR gate

6.2.1. Parity checking bit:

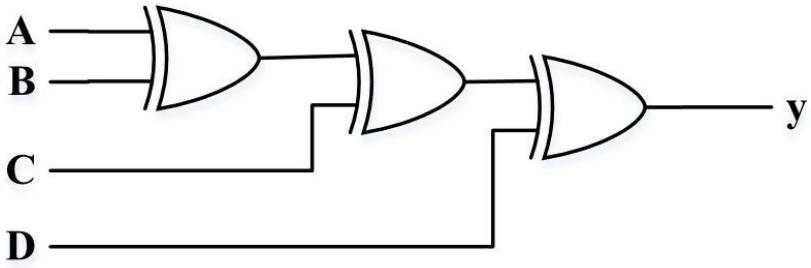
The basic of most error protection in the odd or even parity check. A single bit will be added to each N-bit word so that the number of ones is always odd or even. The XOR gate is the most suitable circuit to provide parity checker.

How to build a circuit of four-bit parity generator "even parity"



Check bit is a bit added to the end of string of binary code that indicates whether the no. of bits in the string with value "1" is even or odd.

Another way of building a 4-bit parity checker



Four-bit parity checker

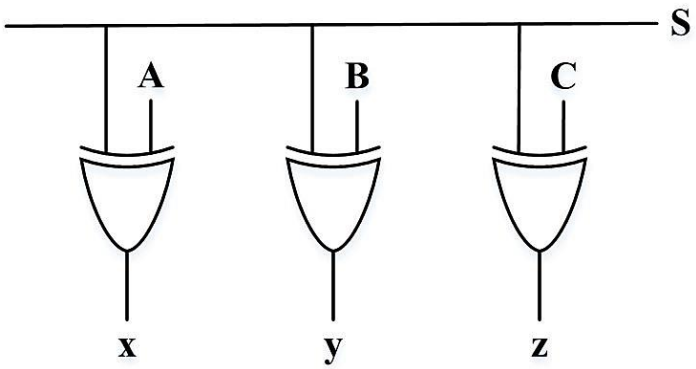
The T.T. for the 4-I/P variable				
D	C	B	A	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Hint: In odd parity generator we need to complement the O/P.

Hint: The error will be accepted as true. The check is only reliable as a single error detecting code.

6.2.2. Controlled inverter (programmed inverter)

The controlled inverter is used in the complement for subtraction.



The control signal (S)

if S=0 → x=A, y=B, z=C

To transmit the register word to the o/p.

if s=1 → x=$\overline{A}$, y=$\overline{B}$, z=$\overline{C}$

To transmit the 1's complement

Value obtained by inverting all the bits in the binary number.

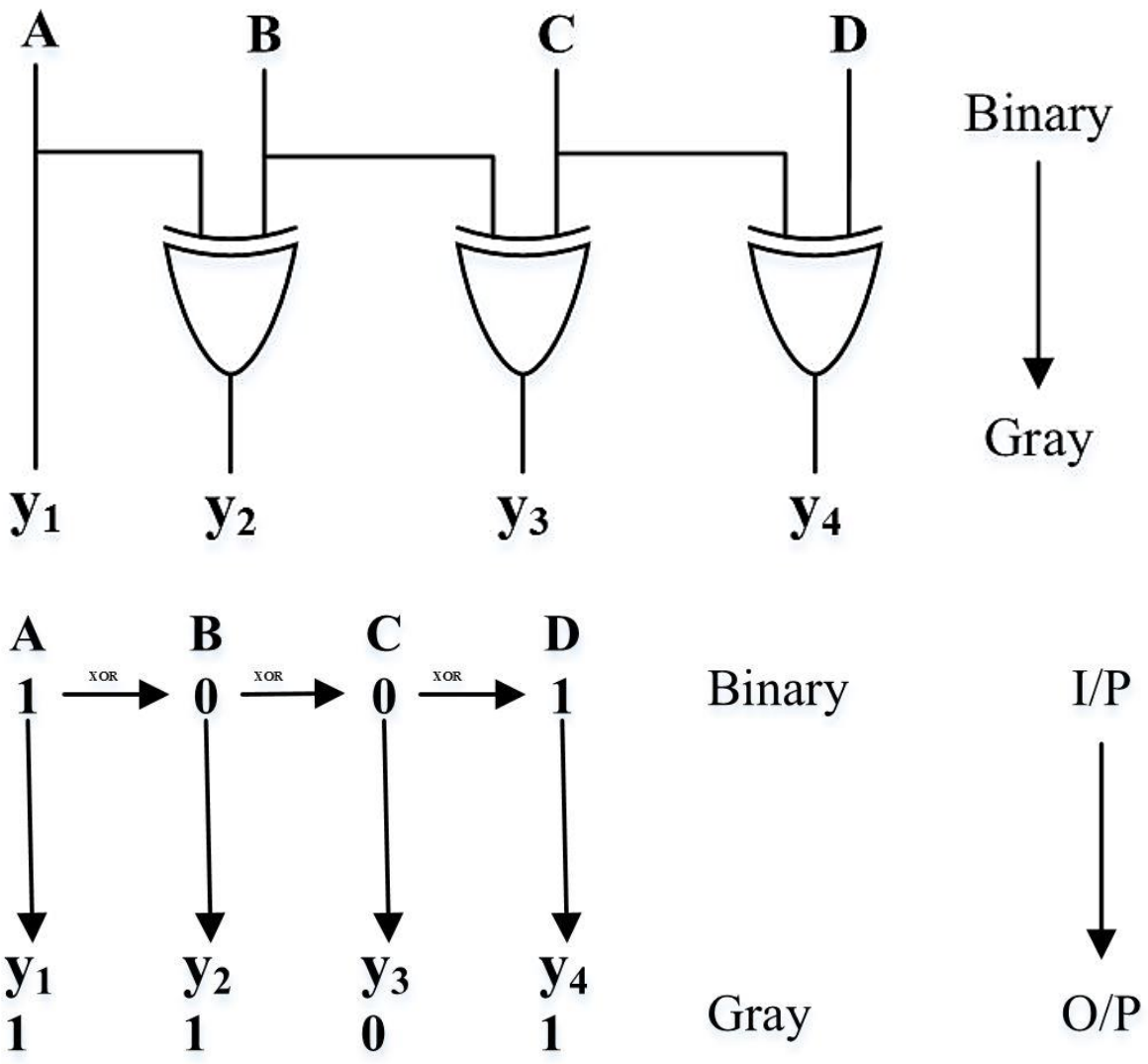
6.2.3. Binary to Gray conversion

Gray code is unweighted code, not suitable for arithmetic usage. It has a lot of applications in I/P-O/P devices such as encoder, analog-to-digital converters (ADC), etc. y_1

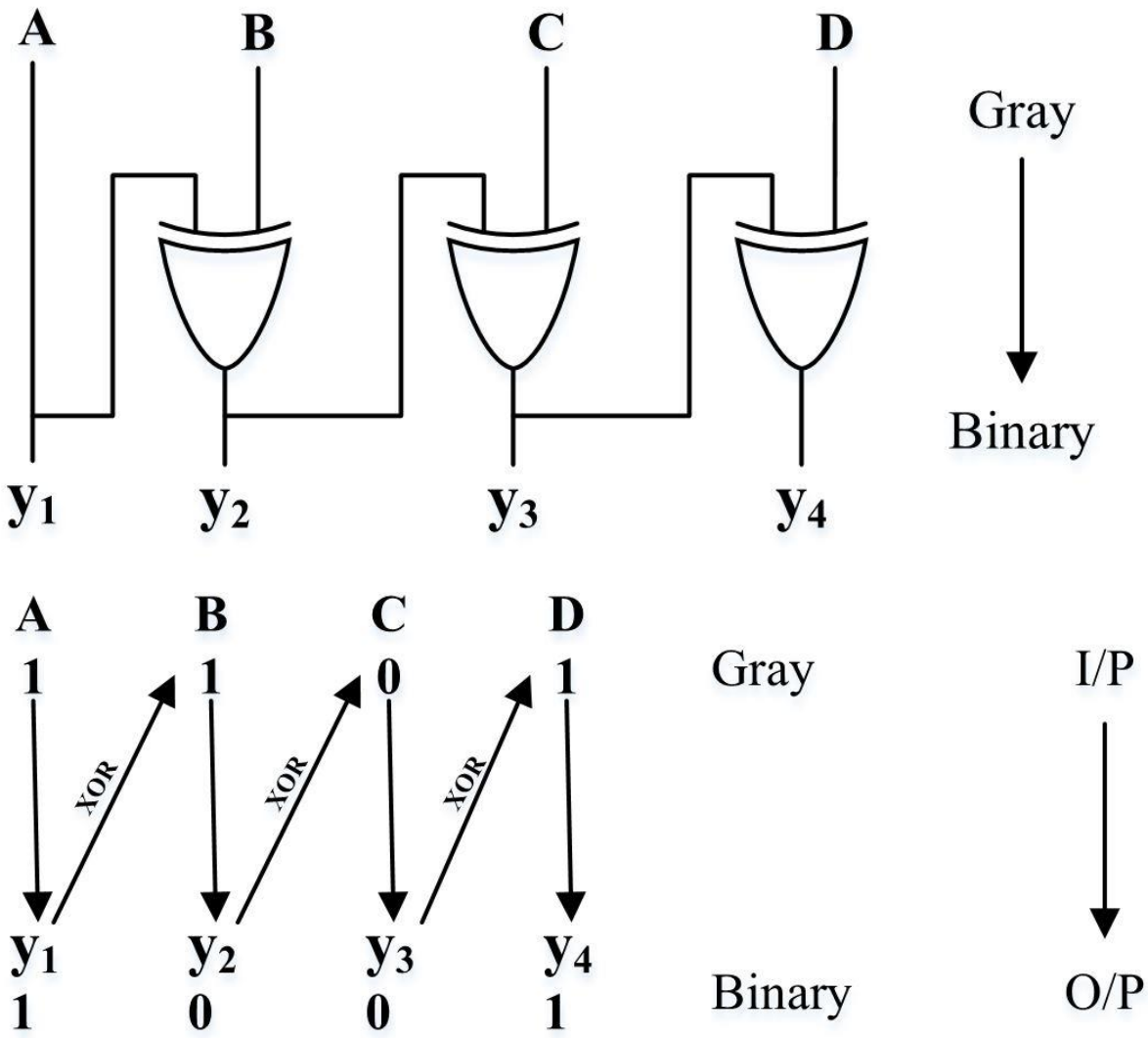
One of the main problems in a binary number system when going from one number to the next is that more than one position may change at the same time. For example, when 0111_2 advances to 1000_2. Four digits must change simultaneously, so we can use gray code instead of binary in which every number differs from the preceding number by a single bit.

How to convert from binary to gray?

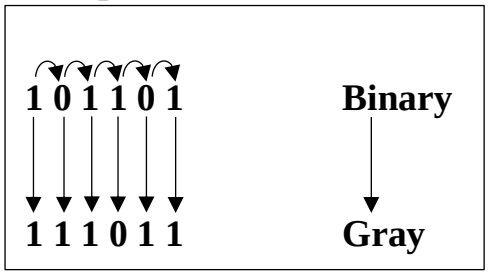
1. **Gray digit is the same as the first binary digit.**
2. **Add the 1st binary to the 2nd binary.**



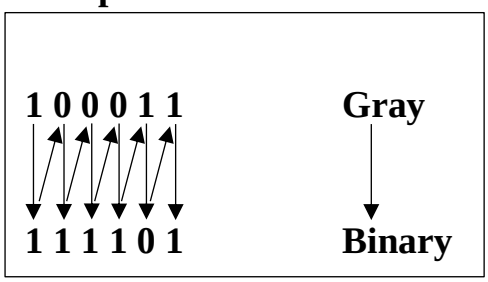
Gray to binary conversion



Example 9: Convert the following binary code to gray code 1101101_2



Example 9: Convert the following gray code to binary code 100011_2



6.3. Half-Adder (H.A)

It adds two binary digits at a time.

0+0=0

0+1=1

1+0=1

1+1=0 → with a carry c=1

Example 10: Add the two binary numbers A (=1010_2) and B (=0010_2).

carry		**1**		
A	1	0	1	0
B	0	0	1	0
	1	1	0	0

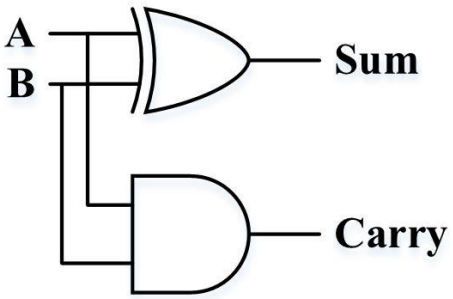
Example 11: Add the two floating point numbers A (=1101.1010_2) and B (=0110.1100_2).

carry	**1**	**1**	**1**	**1**	**1**				
A	1	1	0	1	.	1	0	1	0
B	0	1	1	0	.	1	1	0	0
	1	0	1	0	0	.	0	1	1	0

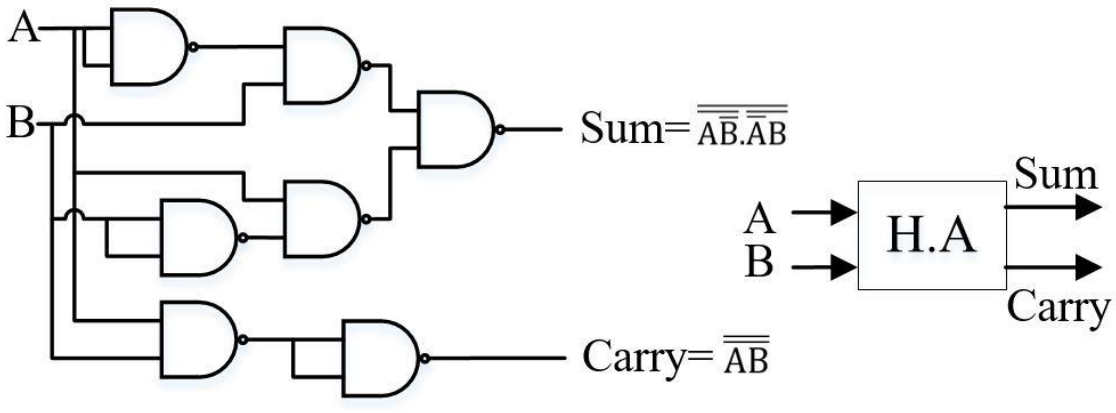
B	A	Sum	Carry	
0	0	0	0	
0	1	1	0	$A\overline{B}$
1	0	1	0	$\overline{A}B$
1	1	0	1	AB

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

Carry=AB



Half-Adder H.A

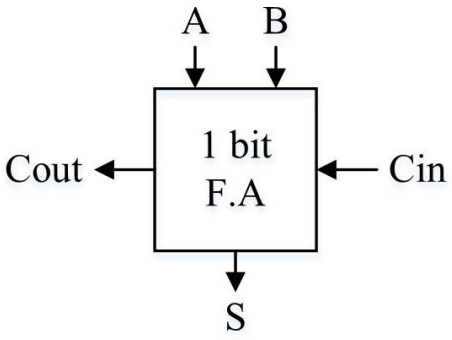


Half-Adder implementation using NAND gates only

6.4. Full-Adder (F.A)

When adding two binary numbers, you may have a carry one column to the next. Practically, we need a logic circuit that adds (or handle) binary digits at a time. This next circuit is named Full-Adder.

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



To use in multi-bit adder

Sum= $A\bar{B}\bar{C}+\bar{A}B\bar{C}+\bar{A}\bar{B}C+ABC = \bar{C}(A\bar{B}+\bar{A}B) + C(\overline{A\bar{B}+AB}) = \bar{C}(A\oplus B) + C(\overline{A\oplus B})$

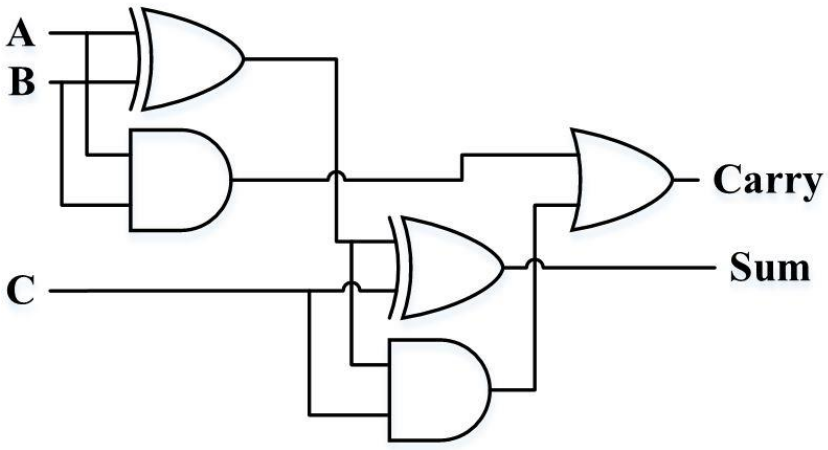
Let $(A\oplus B) = X$

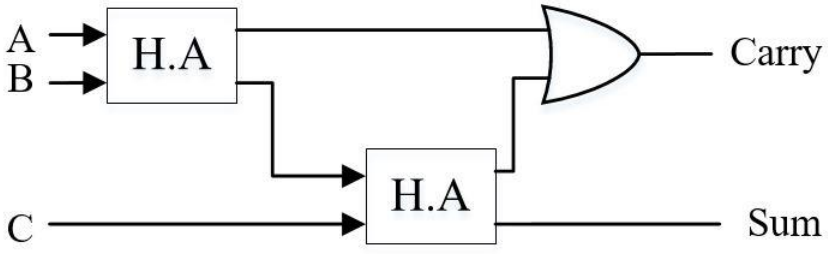
Sum=$\bar{C}X + C\bar{X}=C\oplus X=C\oplus A\oplus B$

Carry=$AB\bar{C} + A\bar{B}C + \bar{A}BC + ABC = AB\bar{C} + C(A\bar{B}+\bar{A}B) + ABC = AB(\bar{C}+C)+C(A\oplus B)$

Since $C+\bar{C}=1$

Carry=$AB + C(A\oplus B)$

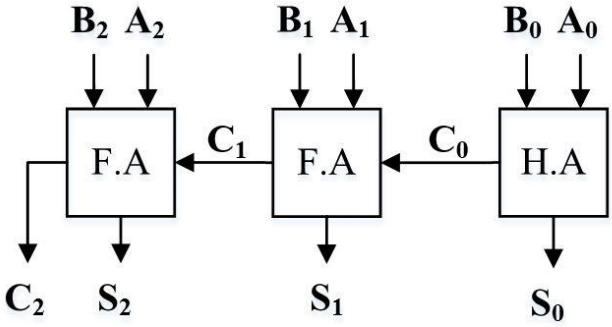




Full-Adder

6.5. Parallel Adder

Several full adders can be connected as shown below to add two binary numbers.
$A=A_0A_1A_2A_3\ldots A_n$
$B=B_0B_1B_2B_3\ldots B_n$
Only the 1st column requires a half-adder while other columns uses full adders since there may be a carry from the preceding columns.



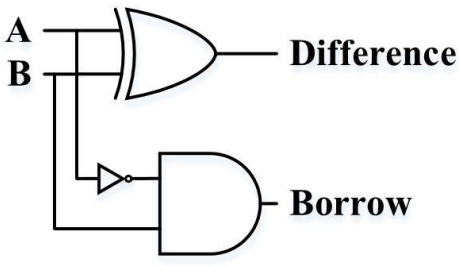
6.6. Half-Subtractor (H.S)

In binary subtraction:
0 – 0 = 0 with borrow = 0
0 – 1 = 1 with borrow = 1
1 – 0 = 1 with borrow = 0
1 – 1 = 0 with borrow = 0

B	A	Diff	Borrow
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

$A\overline{B}$
$\overline{A}B$
$\overline{A}B$

$Diff=\overline{A}B+A\overline{B}=A\oplus B$
$Carry=\overline{A}B$



Half-Subtractor H.S

6.7. Full-Subtractor (F.S)

C	B	A	Diff	Borrow
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Diff= $A\bar{B}\bar{C}+\bar{A}B\bar{C}+\bar{A}\bar{B}C+ABC = \bar{C}(A\bar{B}+\bar{A}B) + C(\bar{A}\bar{B}+AB) = \bar{C}(A\oplus B) + C(\overline{A\oplus B})$

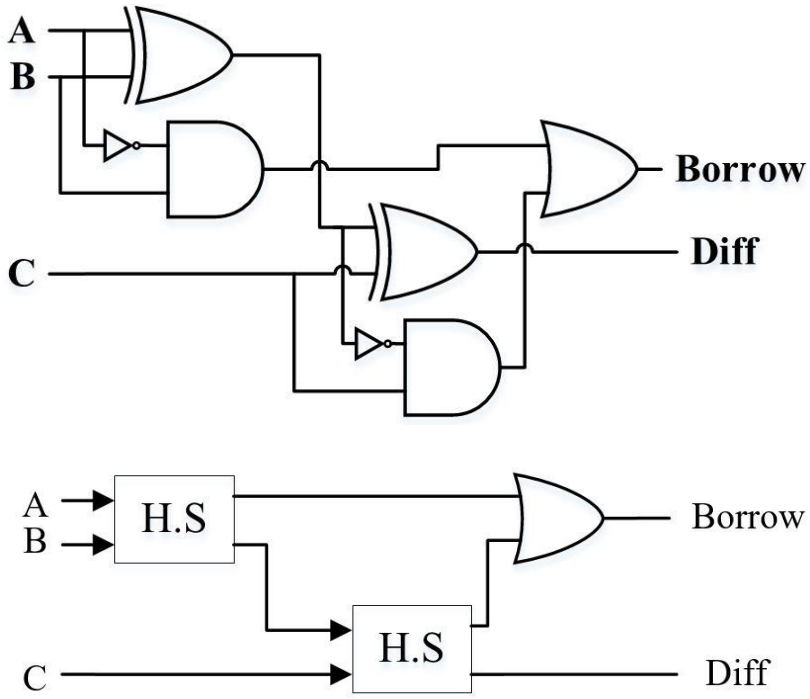
Let $(A\oplus B) = X$

Diff=$\bar{C}X + C\bar{X}=C\oplus X=C\oplus A\oplus B$

Borrow=$\bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + ABC = \bar{A}B\,(\bar{C}+C) + C(\bar{A}\bar{B}+AB)$

Since $C+\bar{C}=1$

Borrow=$\bar{A}B + C(A\oplus B)$



Full-Subtractor

6.8. Subtraction using 1's and 2's complement

6.8.1. 1's complement subtraction

Instead of subtracting a number, we can add the 1's complement of the number. The last carry is then added to get the final answer. If there is no carry the final answer will have a negative sign and it is in the 1's complement form. To get the true answer take the 1's complement and put a negative sign.

Example 12:

							1's			
	7	1	1	1		1	1	1		
−	6	1	1	0	→	0	0	1	+	
	1	0	0	1		(1)	0	0	0	
								1	+	
						0	0	1		

Example 13:

	14		1	1	1	0	1's		1	1	1	0	
–	10		1	0	1	0	→		0	1	0	1	+
	4							(1)	0	0	1	1	
												1	+
									0	1	0	0	

Example 14:

	6		1	1	0	1's	1	1	0	
–	7		1	1	1	→	0	0	0	+
	-1						1	1	0	

No carry which means that the no. is negative.

1	1	0					
0	0	0	1's				
1	1	0	→	-	0	0	1

Example 15:

	10		1	0	1	0	1's	1	0	1	0				1's	
–	14		1	1	1	0	→	0	0	0	1	+	No carry	→	-0	1	0	0
	-4							1	0	1	1							

6.8.2. 1's complement subtraction

We can add the 2's complement of the number instead of subtracting it.

Hint: if there is carry neglect it to get the true answer. If there is no carry take 2's complement of the result and put a negative sign to get the true answer.

Example 15:

	14		1	1	1	0	1's					2's			1					
–	10		1	0	1	0	→	0	1	0	1	→	0	1	0	1				
	4															1	+			
													0	1	1	0				

Then

	1	1			
	1	1	1	0	
	0	1	1	0	+
(1)	0	1	0	0	

Note: If the final carry over the sum is 1. It is dropped and the result is positive.

Example 16:

$$\begin{array}{rr} & 5 \\ - & 14 \\ \hline & -9 \end{array} \qquad \begin{array}{cccc} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array} \quad 1\text{'s} \rightarrow \begin{array}{cccc} 0 & 0 & 0 & 1 \end{array} \quad 2\text{'s} \rightarrow \begin{array}{cccc} 0 & 0 & 0 & 1 \\ & & & 1 \\ \hline 0 & 0 & 1 & 0 \end{array} +$$

Then

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 \end{array} + \qquad \text{No carry} \rightarrow$$

Note: The two's complement of the sum will be the result and it is negative.

$$\downarrow \text{ 1's}$$

$$\begin{array}{cccc} 1 & 0 & 0 & 0 \\ & & & 1 \\ \hline - & 1 & 0 & 0 & 1 \end{array} + \quad \downarrow \text{ 2's}$$

6.9. Parallel four-bit Adder-Subtractor

A four-bit Adder-Subtractor is shown below. The I/P Z controls the operation of the circuit. When Z=0 the circuit functions as an adder, and when Z=1 the circuit functions as a Subtractor. Each XOR gate receives input Z and one of the inputs of B. when Z=0, we have $B \oplus 0 = B$. the full adders receive the value of B, the input carry is '0', and the circuit performs A+B.

When Z=1, we have $B \oplus 1 = \overline{B}$ and $C_0 = 1$. The B inputs are all complemented and a '1' is added through the input carry. The circuit performs the operation A plus the 2's complement of B. the XOR with output y is for detecting an over flow.