Introduction to Basic Gates and Functions

Logic gates

logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels.

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

Logic Gates

- *AND gate*
- *OR gate*
- *NOT gate*
- *NAND gate*
- *NOR gate*
- *Ex-OR gate*
- *Ex-NOR gate*

Truth Tables

Truth tables are used to help show the function of a logic gate. If you are unsure about truth tables and need guidance on how go about drawning them for individual gates or logic circuits then use the truth table

Truth Tables

Logic gates

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

AND gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

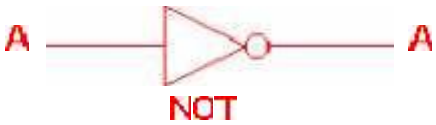
The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB

OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

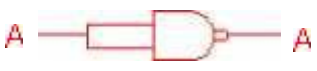
The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.

NOT gate



NOT gate	
A	$\overline{A}$
0	1
1	0

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The diagrams below show two ways that the NAND logic gate can be configured to produce a NOT gate. It can also be done using NOR logic gates in the same way.



NAND gate

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

NOR gate

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high.The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

EX-OR gate

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EOR operation.

EX-NOR gate

2 Input EXNOR gate		
A	B	$\overline{A⊕B}$
0	0	1
0	1	0
1	0	0
1	1	1

The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.

Objectives

- Get familiar with Logisim
- Use Logisim to model and simulate the behavior of digital logic circuits

Logisim Environment Layout

Logisim is an educational and user friendly tool. It mainly consists of an interactive graphical schematic editor and logic simulator. Logisim has a layout similar to most available software tools (**Figure *1***).

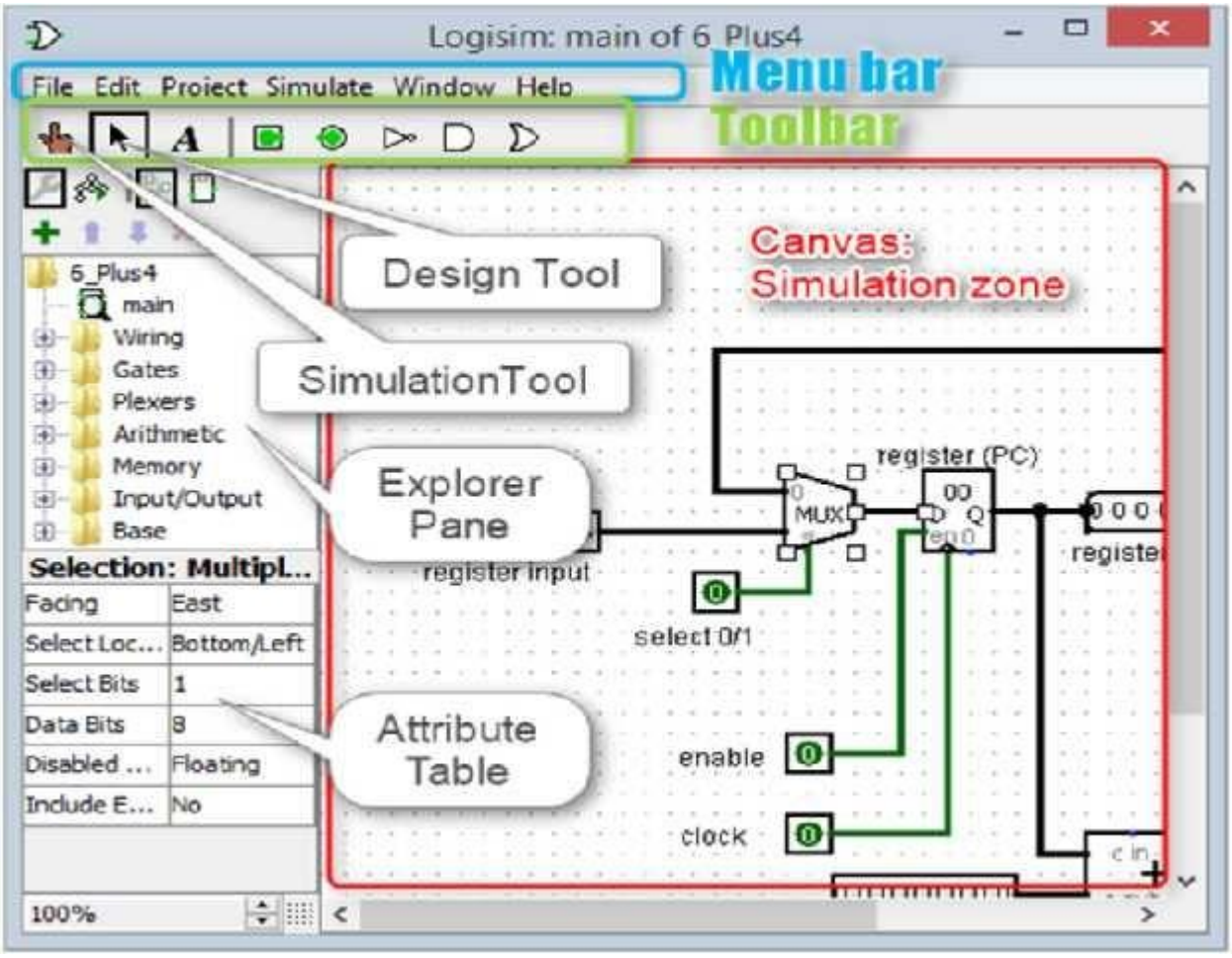


Figure 10.1: *Logisim Main Window Layout*

The main window consists of the following items:

• **Toolbar**: contains short cuts to several commonly used items
o The simulation tool: shaped like a hand, is used in simulation mode to alter input pins.
o The design tool: is used while designing the circuit.
o The input pin: green circle surrounded by a square box, is used to send a signal through a
wire. When placing the input on the canvas it initializes the input to logic 1 or 0. The number of bits can be increased in the Attribute Table.
o The output pin: green circle surrounded by a circular shape, is used to observe the output
from a gate or a block. The output pin toggles in real time as long as the simulation is enabled from the menu bar: **Simulate** > **Simulation Enabled**

• **Explorer Pane**: This pane contains the list of wiring, gates, multiplexers and other components that are available for digital design in Logisim.

• **Attribute Table**: Gives detailed attributes of digital design components (e.g., AND, OR,
XOR gates). The attribute table allows you to alter the number of inputs/outputs that a digital component may have.

• **Canvas**: The canvas is the area for you to create your digital circuits. In this area you may
simulate your circuits while designing in real time.

-

Logisim operates in two modes: Design and Simulate. Logisim is first operated in edit mode while designing a logic circuit, then in simulate mode to see how the circuit would actually work.

Edit Mode

- To use the edit mode, select the **Design** tool (Figure 10.1).
- Select then a component in the library on the left. To add it in your design, simply drag and drop the desired component in the canvas:
- To insert a logic gate, expand the folder Gates and click on the desired logic gate. Once the gate is selected, the mouse will turn into the shape of the selected gate.Place the gate in the canvas or circuit area on the right.
- Each selected component has modifiable attributes in the attribute area. For example for an AND gate, the number of input signals, the number of bits per input or output and the size of the component can all be modified.
- It is also possible to copy and paste of one or more components in the canvas. In this case, the components retain all the attributes previously defined.

- To connect the gates, select the arrow icon on top. Then drag from the output of one gate to the input of another gate. You can connect to any of the small blue dots on the gate symbol.
- To create an input to the circuit, select the "**Add Pin**" icon with an outline of a *square*.
- Similarly, add an output to the circuit by using the "**Add Pin**" icon with an outline of a *circle*.
- To assign a name to the input/output pin, click on the pin while the arrow icon is selected.
- You may then add the label for the pins.
- While components are added to the circuit, notice how the color of the wire changes (Table 10.1).

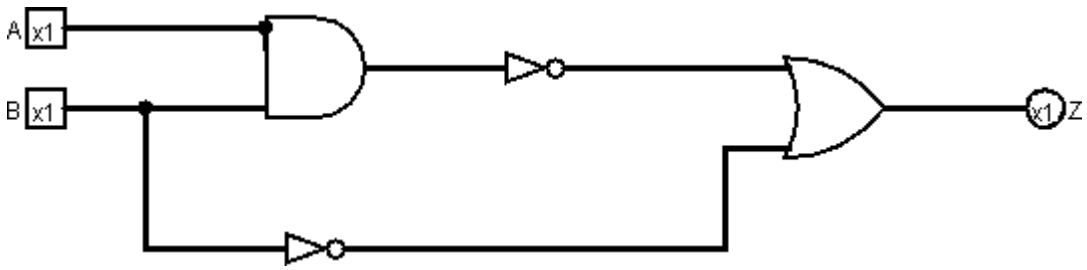
Table 10.1: Logisim Wire Color Conventions

Wire Color	Meaning
dark green	logical `0'
light green	logical `1'
blue	unknown value
gray	unconnected wire
black	wire has more than one bit (bus)
red	conflicting values, error
orange	incompatible bus width

1) For the following functions, construct a truth table and draw a circuit diagram.

1. y(A,B) = (AB)' + B'

Circuit Diagram :

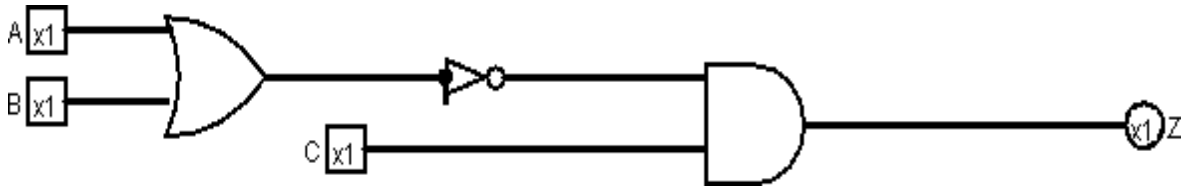


Truth Table :

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

2. y(A,B,C) = (A + B)' C

Circuit Diagram :

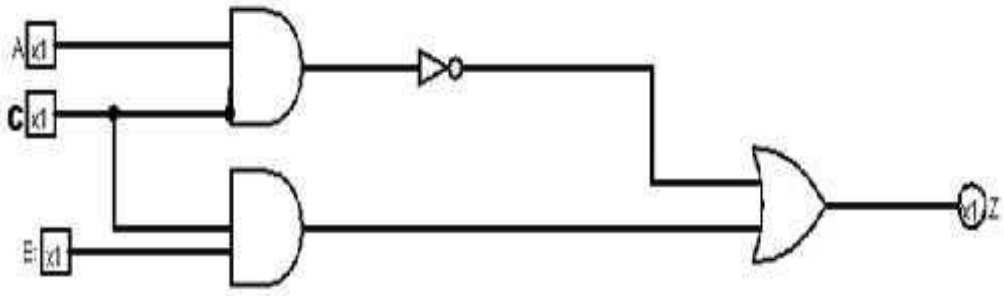


Truth Table :

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

3. y(A,B,C) = (AC)' + BC

Circuit Diagram :

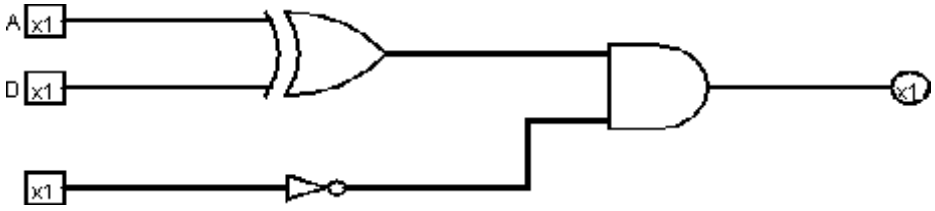


Truth Table :

A	B	C	Z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

4. y(A,B,C) = (A Å B)C'

Circuit Diagram :

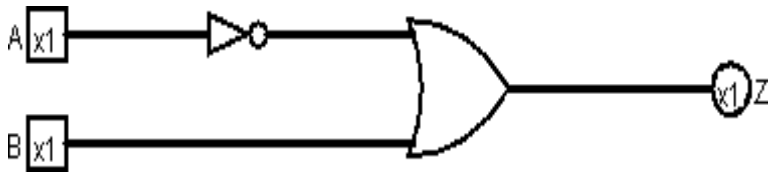


Truth Table :

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

5. y(A,B) = A' + B

Circuit Diagram :

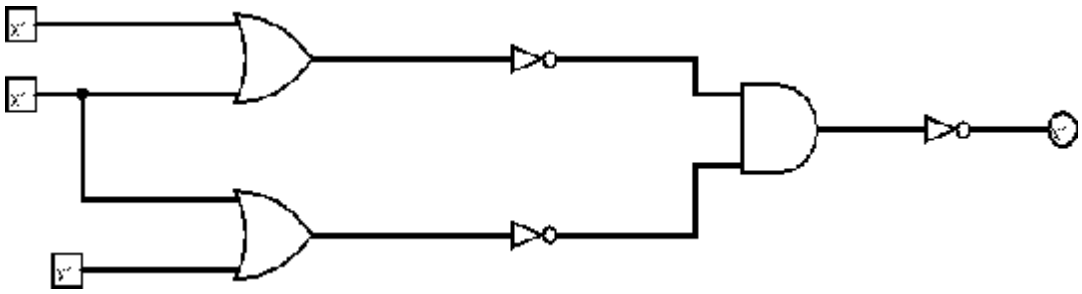


Truth Table :

A	B	Z
0	0	1
0	1	1
1	0	0
1	1	1

6. y(A,B,C) = ((A+B)'(B+C)')'

Circuit Diagram :



Truth Table :

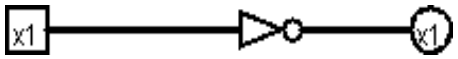
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2. Study and verify the truth table of various logic gates

· **NOT, AND, OR, NAND, NOR, EX-OR, and EX-NOR**

1.NOT GATE

Circuit Diagram :

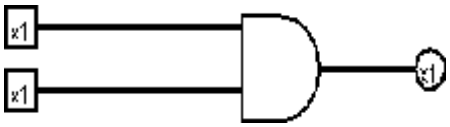


Truth table

A	A`
0	1
1	0

2. AND GATE

Circuit Diagram :



Truth table

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

3. OR GATE

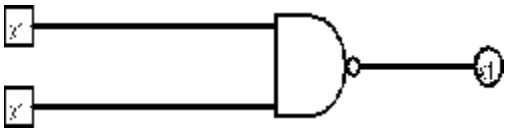
Circuit Diagram :

Truth table

A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

4. NAND GATE

Circuit Diagram :



Truth table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR GATE

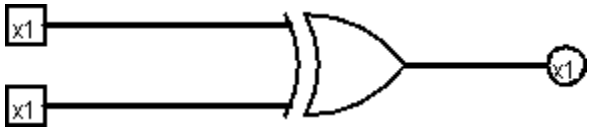
Circuit Diagram :

Truth table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

6. XOR GATE

Circuit Diagram :

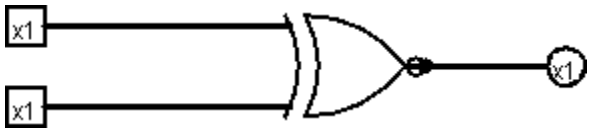


Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

7.EXNOR GATE

Circuit Diagram :



Truth table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

3. Simplify Boolean expressions and realize it.

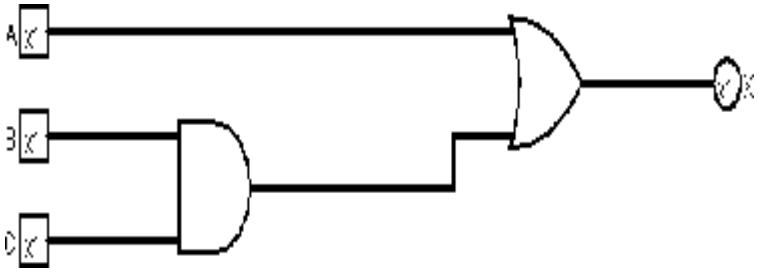
A+B(A+C)+AC

Soln: A+AB+BC+AC

∴ Apply the rule (A+AB=A)

=A+BC+AC
=A+BC

Circuit Diagram :



Truth Table :

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

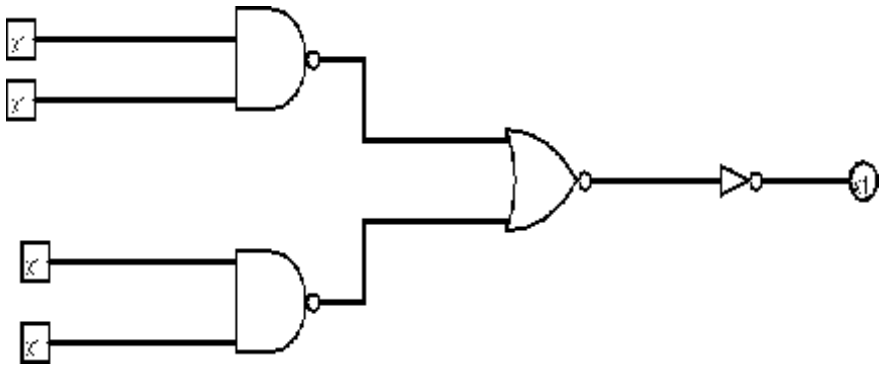
4. Verification of Boolean Theorems using basic gates.

Circuit Diagram :

Truth Table :

5. Design a 4-input NAND gate using two 2-input NAND gates and one 2-input NOR gate.
Hint: Use DeMorgan's law

Circuit Diagram :



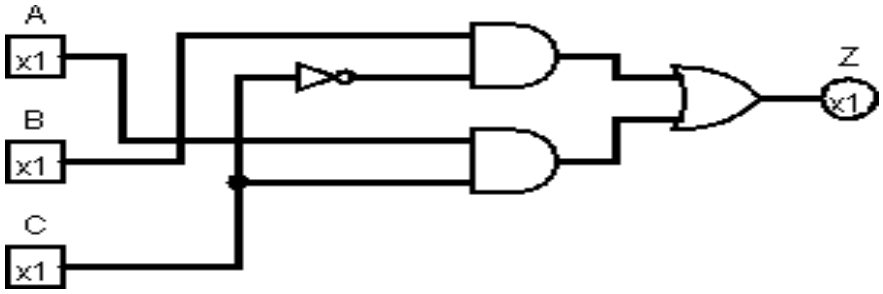
Truth Table :

A	B	C	D	X
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

6. Construct the K-map for each of the following functions

(a) f(A,B,C) = AB + A'BC' + AB'C

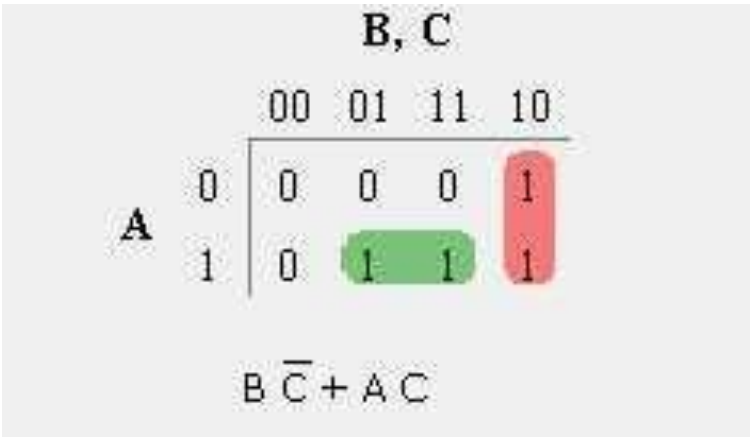
Circuit Diagram :



Truth Table :

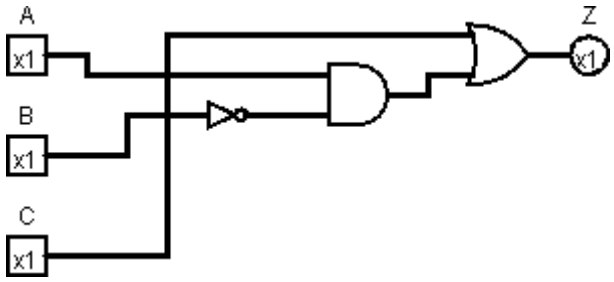
A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

K-Map



(b) g(A,B,C) = A'C + ABC + AB'

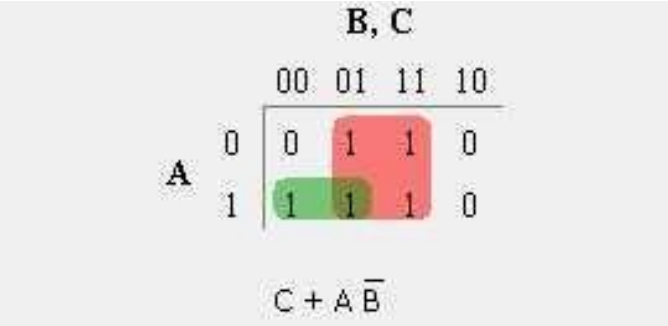
Circuit Diagram :



Truth Table :

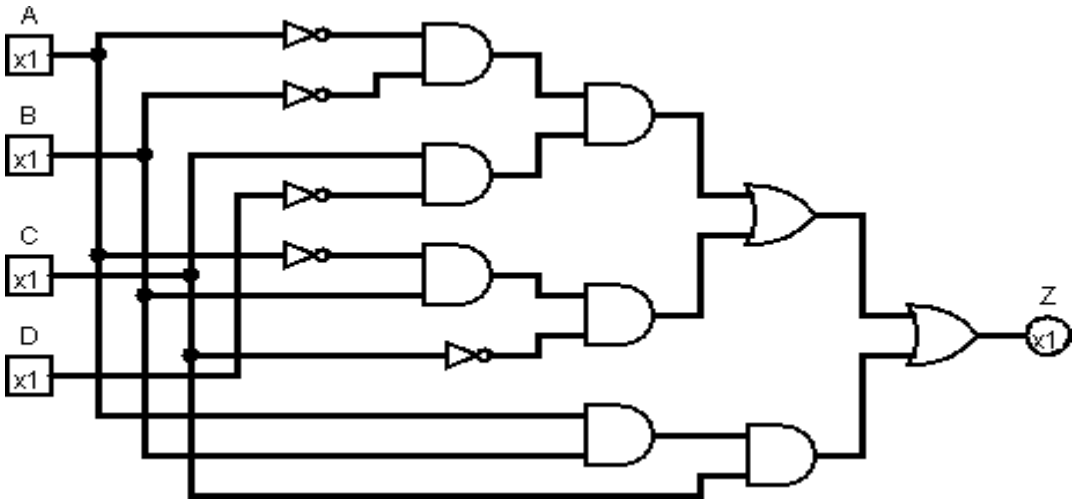
A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

K-Map



(c) h(A,B,C,D) = A'BC' + (A Å B)C + A'B'CD' + ABC

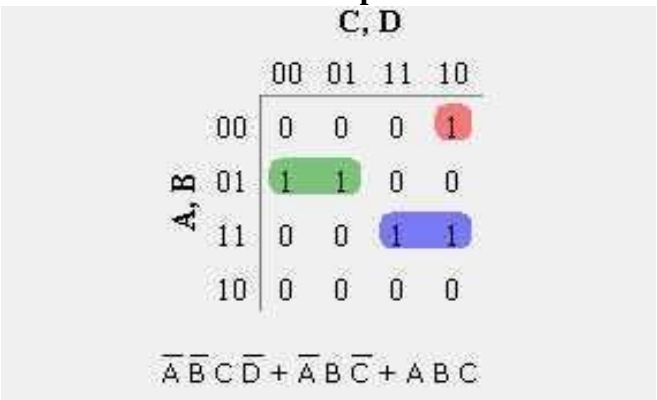
Circuit Diagram :



Truth Table :

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

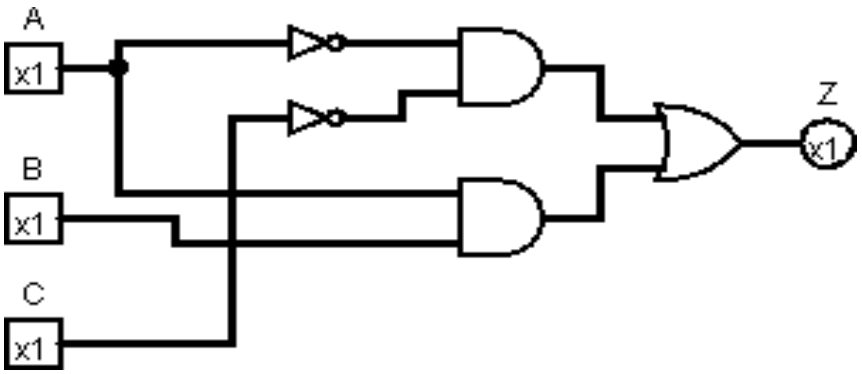
K-Map



8. For the functions listed below, construct a K-map and determine the minimal SOP expression.

a) f(a,b,c) = a'b'c' + a'bc' + abc' + abc

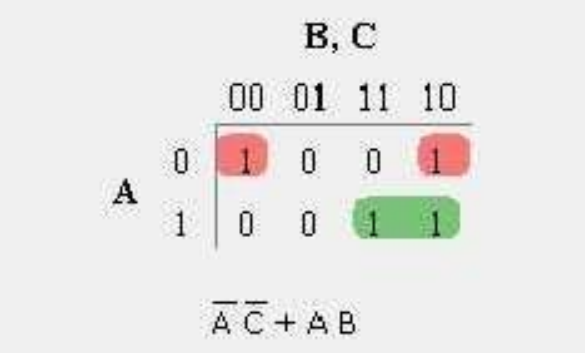
Circuit Diagram:



Truth Table:

A	B	C	Z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

K-Map



b. g(a,b,c) = ab'c' + abc' + abc + don't cares(a'bc + ab'c)

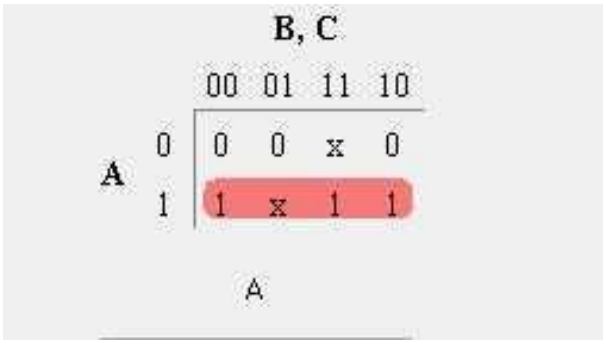
Build the circuit required for (b).

Circuit Diagram:

Truth Table:

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	x
1	0	0	1
1	0	1	x
1	1	0	1
1	1	1	1

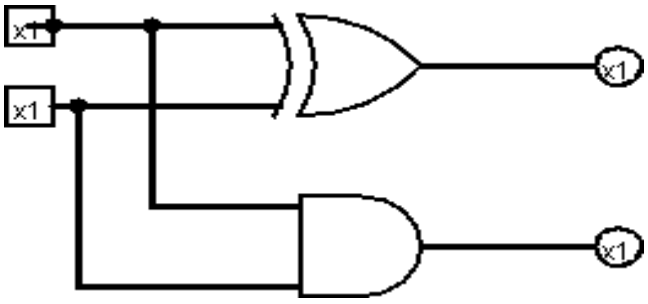
K- Map:



9. Design and verify a half/full adder

- Half adder

Circuit Diagram:

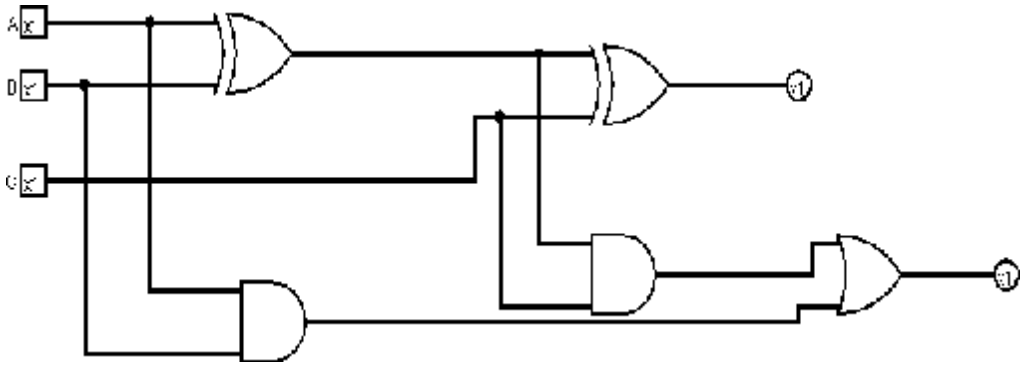


Truth Table:

A	B	X	Y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Full adder

Circuit Diagram:



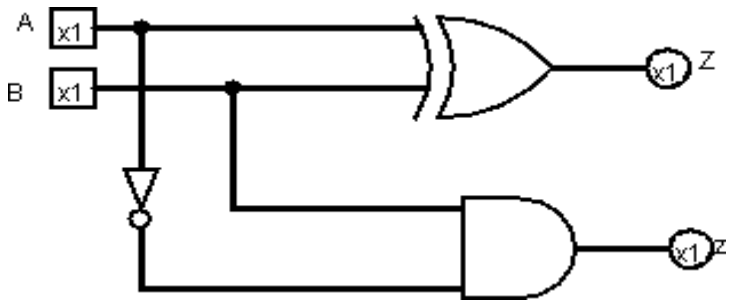
Truth Table:

A	B	C	X	Y
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

10. Design and verify half/full subtractor

- half subtractor

Circuit Diagram:

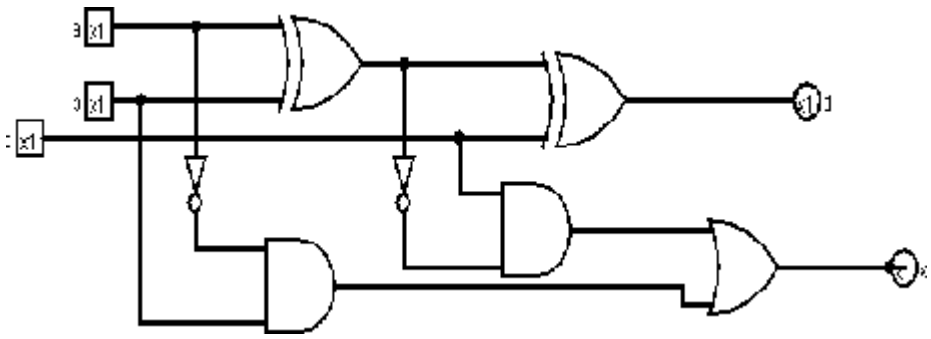


Truth Table :

A	B	X	Y
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- full subtractor

Circuit Diagram:



Truth Table :

A	B	C	D	X
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1