Chapter-4

Combinational Logic

Introductions

The purpose of Boolean-algebra simplification is to obtain an algebraic expression that, when implemented, results in a low-cost circuit. Two circuits that perform the same function, the one that requires fewer gates is preferable because it will cost less. But this is not necessarily true when integrated circuits are used. With integrated circuits, it is not the count of gates that determines the cost, but the number and types of ICs employed and the number of interconnections needed to implement the digital circuits of varying complexities.

There are several combinational circuits that are employed extensively in the design of digital systems. These circuits are available in integrated circuits and are classified as MSI components. MSI components perform specific digital functions commonly needed in the design of digital systems.

Introductions

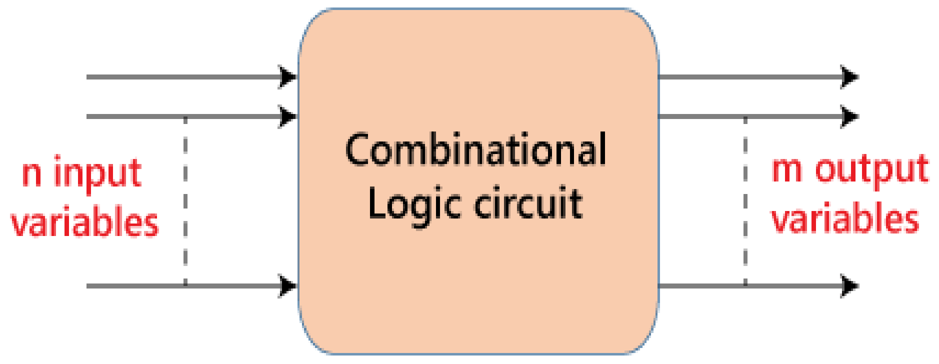
A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs and they have no memory. A sequential circuit consists of logic gates whose outputs at any time are determined from both the present combination of inputs and previous output. That means sequential circuits use memory elements to store the value of previous output.

These circuits developed using AND, OR, NOT, NAND and NOR logic gates. These logic gates are building blocks of combinational circuits. A combinational circuit consists of input variables and output variables. Since these circuits are not dependent upon previous input to generate any output, so are combinational logic circuits. A combinational circuit can have an n number of inputs and m number of outputs. In combinational circuits, the output at any time is a direct function of the applied external inputs.

Introductions

There are the following characteristics of the combinational logic circuit:

- At any instant of time, the output of the combinational circuits depends only on the present input terminals.
- The combinational circuit doesn't have any backup or previous memory. The present state of the circuit is not affected by the previous state of the input.
- The n number of inputs and m number of outputs are possible in combinational logic circuits.



Block diagram of a combinational circuit

Design procedure

I. The problem is stated.
II. The number of available input variables and required output variables is determined.
III. The input and output variables are assigned letter symbols.
IV. The truth table that defines the required relationships between inputs and outputs is derived.
V. The simplified Boolean function for each output is obtained.
VI. The logic diagram is drawn.

Adder and Subtractor

Adders:

In electronic , an adder is a digital circuit that performs addition of numbers.in modern computer adders reside in the arithmetic logic unit. Although adders can be constructed for many numerical representation, such as BCD or excess-3, most adder operate on binary numbers.

Half Adder:

A combinational logic circuit that performs the addition of two single bits is called Half Adder. it contain two inputs and produces two outputs.

Inputs are called Augend and Added bits and Outputs are called Sum and Carry. The adder is used to perform OR operation of two single bit binary numbers.

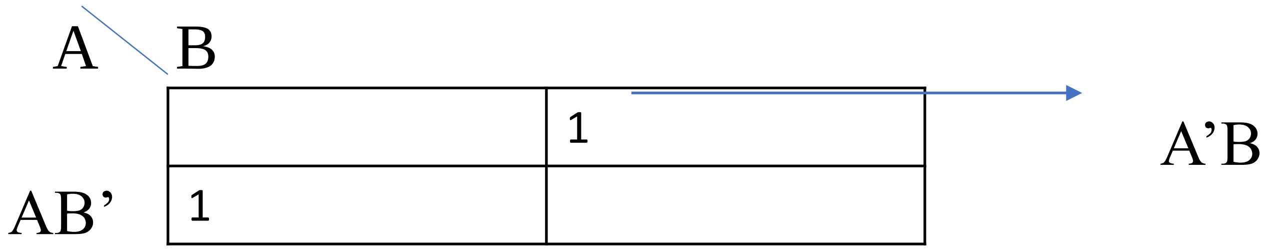
Half Adder

Truth table:

A	B	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Next Step is to draw the Logic Diagram. To draw Logic Diagram, We need Boolean Expression, which can be obtained using K-map.

So, k-map for sum



Half Adder

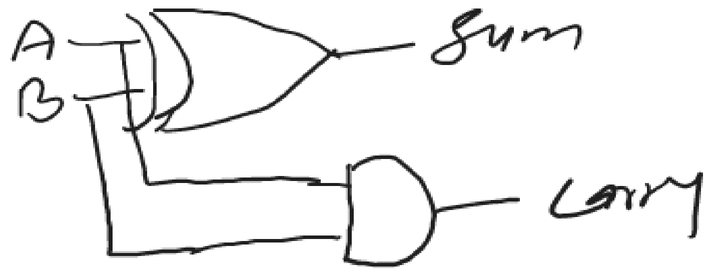
K-map for carry

$$\begin{array}{c|cc} A \backslash B & & 1 \\ \hline & & \\ 1 & & \boxed{1} \end{array}$$

carry : AB

Sum : $A'B + AB' = A \oplus B$

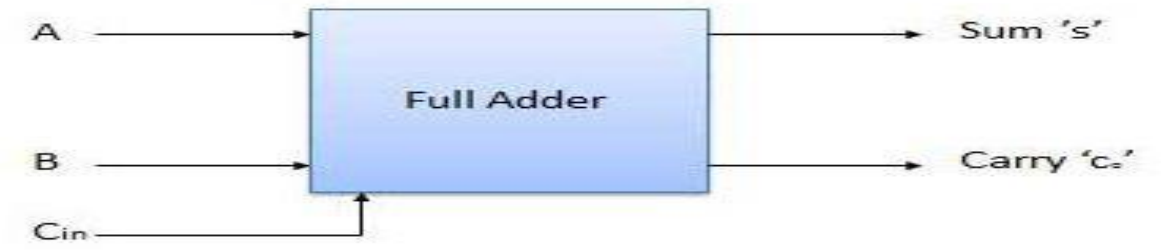
logic Diagram



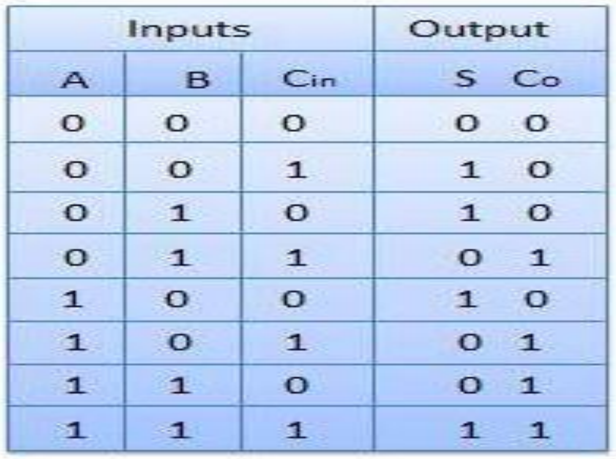
Full adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

Block diagram



Truth Table



Inputs			Output	
A	B	C_{in}	S	C_o
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

Full adder

Now draw k-map for sum and carry and draw the circuit diagram.

Logical Expression for SUM:

= A’ B’ C-IN + A’ B C-IN’ + A B’ C-IN’ + A B C-IN

= C-IN (A’ B’ + A B) + C-IN’ (A’ B + A B’)

= C-IN XOR (A XOR B)

Similarly find out logical expression for Carry

??

Now draw the circuit

For sum

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

$$\therefore \quad Sum = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

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

$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C}) \qquad (\because \bar{x}y + x\bar{y} = x \oplus y)$$

$$= A \oplus B \oplus C.$$

For carry

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

Carry = AB+BC+AC

Another form of Carry out

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

$= A B C + A B + A C + A' B C$

$= A B (1 + C) + A C + A' B C$

$= A B + A C + A' B C$

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

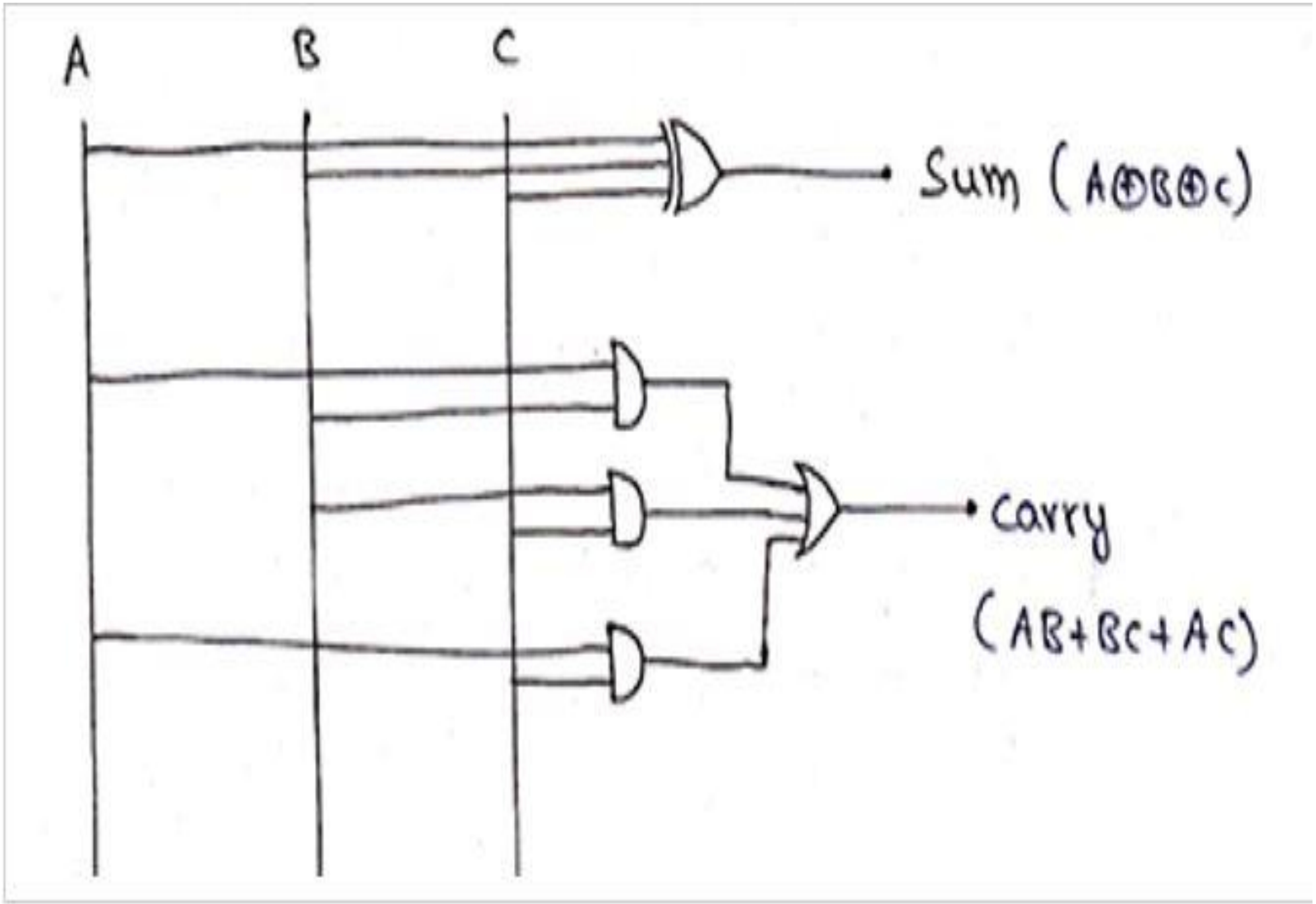
$= A B C + A B + A B' C + A' B C$

$= A B (C + 1) + A B' C + A' B C$

$= A B + A B' C + A' B C$

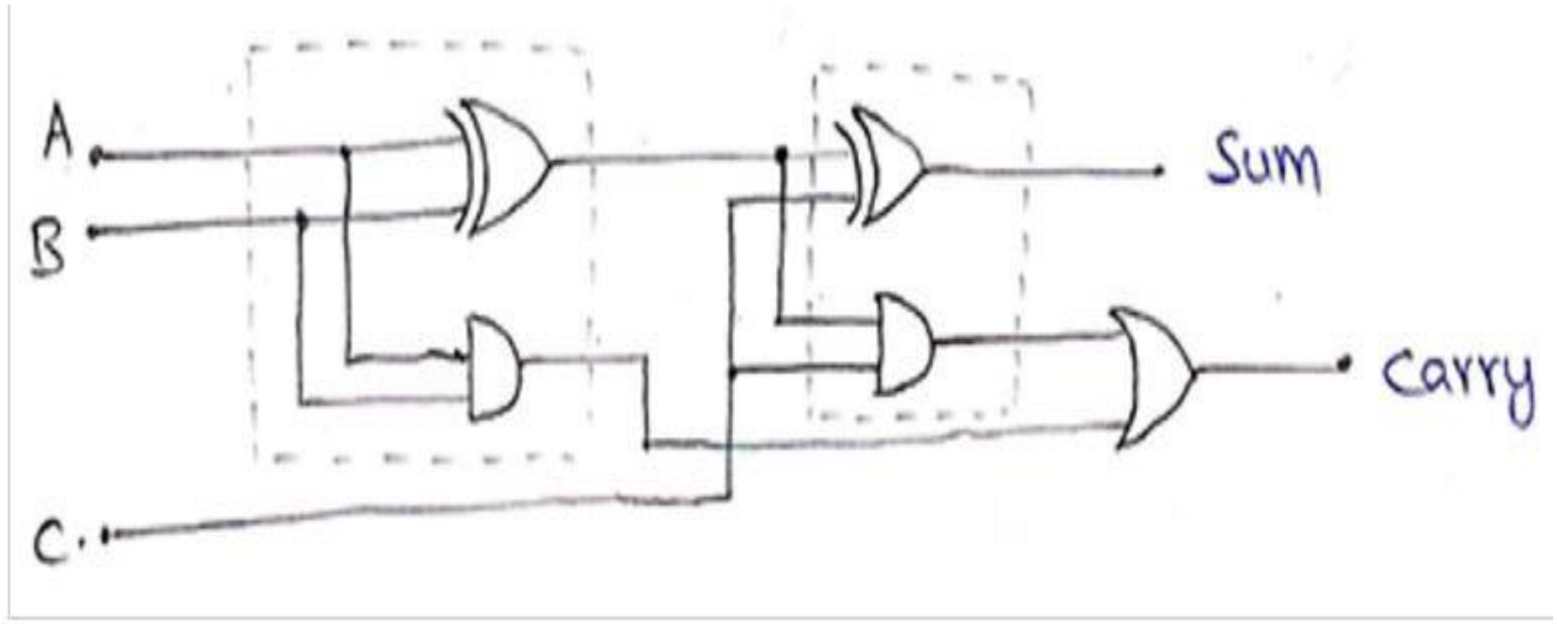
$= AB + C(A' B + A B')$

Therefore COUT = $AB + C_{in}$ (A EX – OR B)


A
B
C
Sum (A⊕B⊕C)
Carry
(AB+BC+AC)

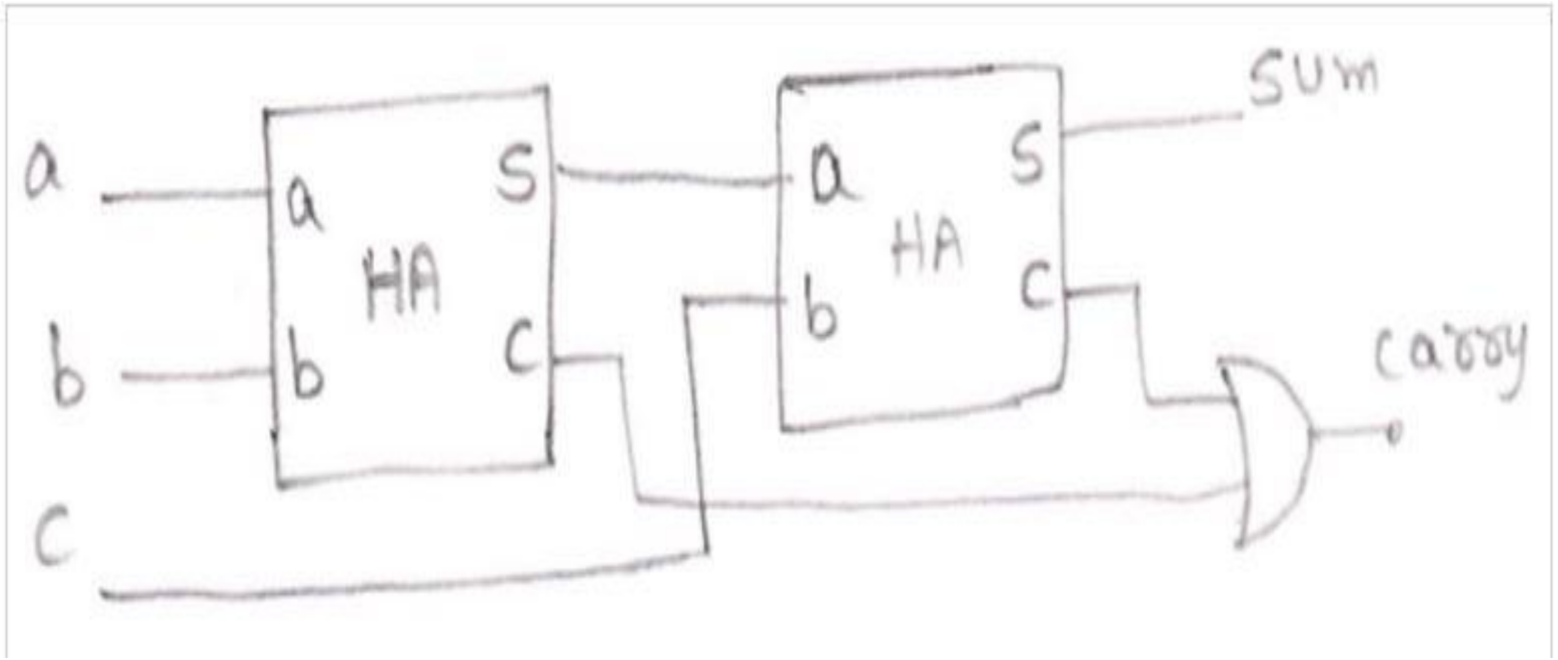
Full Adder using Half Adder

A Full Adder can also be implemented using two half adders and one OR gate.



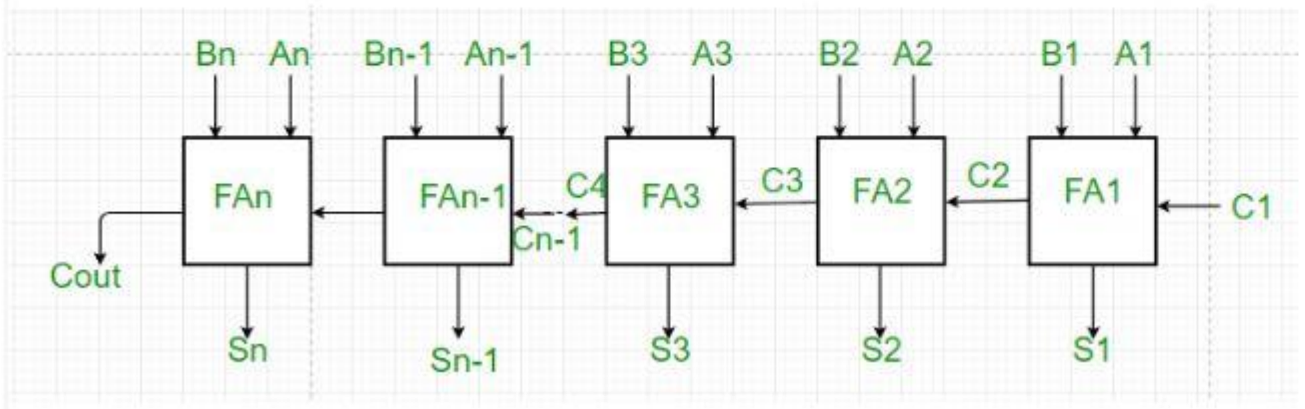
Full Adder using Half Adder

representation in block diagram as:



Parallel Adder

A single full adder performs the addition of two one bit numbers and an input carry. But a Parallel Adder is a digital circuit capable of finding the arithmetic sum of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel. It consists of full adders connected in a chain where the output carry from each full adder is connected to the carry input of the next higher order full adder in the chain. A n bit parallel adder requires n full adders to perform the operation. So for the two-bit number, two adders are needed while for four bit number, four adders are needed and so on. Parallel adders normally incorporate carry lookahead logic to ensure that carry propagation between subsequent stages of addition does not limit addition speed.

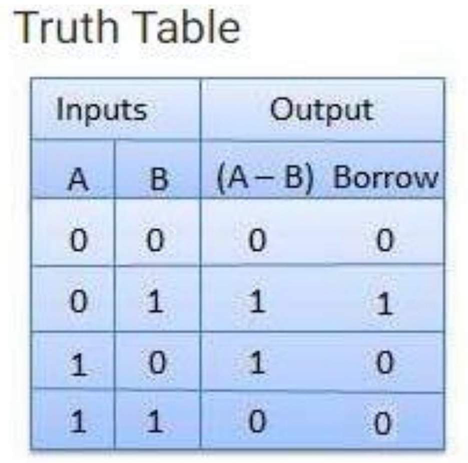


Working of parallel Adder

- As shown in the figure, firstly the full adder FA1 adds A1 and B1 along with the carry C1 to generate the sum S1 (the first bit of the output sum) and the carry C2 which is connected to the next adder in chain.
- Next, the full adder FA2 uses this carry bit C2 to add with the input bits A2 and B2 to generate the sum S2(the second bit of the output sum) and the carry C3 which is again further connected to the next adder in chain and so on.
- The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and Bn to generate the last bit of the output along last carry bit Cout.

Half Subtractors

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

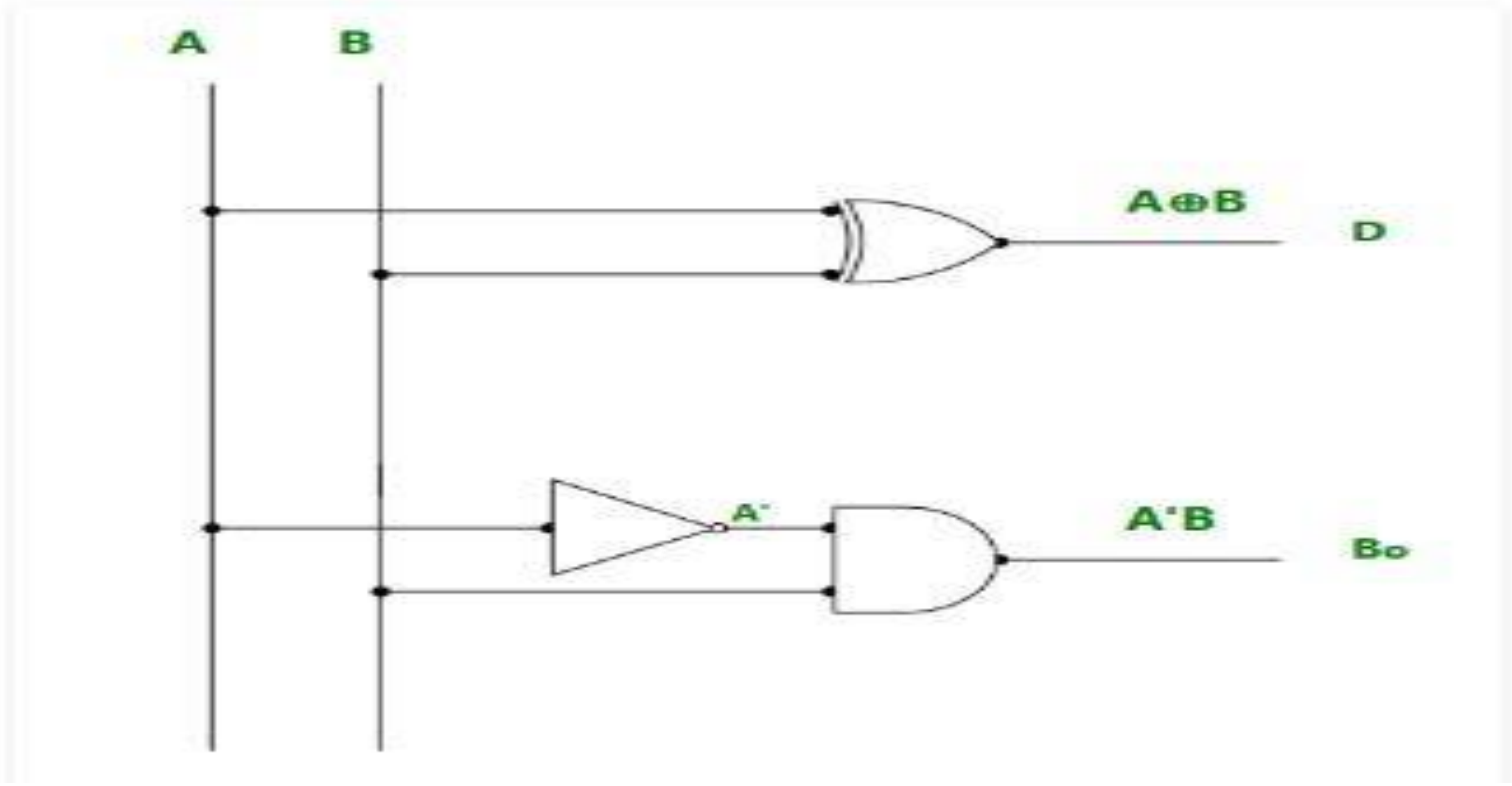


Truth Table

Inputs		Output	
A	B	(A – B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

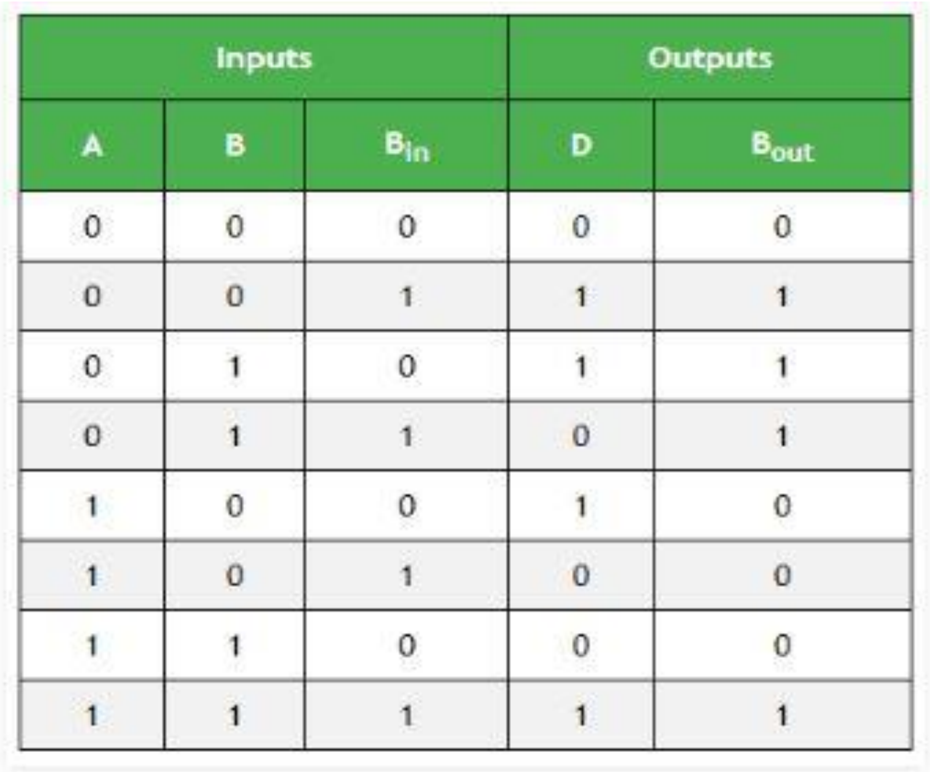
Now, simplify above expression using k-map and draw circuit.

Half Subtractors



Full Subtractor

The Half Subtractor is used to subtract only two numbers. To overcome this problem, a full subtractor was designed. The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., diff and borrow.



Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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

Full Subtractor

Now , simplify the expression using k-map and draw the circuit diagram.

The equation obtained from above K-map is

For difference,

D = A'B'Bin + AB'Bin' + ABBin + A'BBin’

D = B'(A'Bin + ABin') + B(ABin + A'Bin')

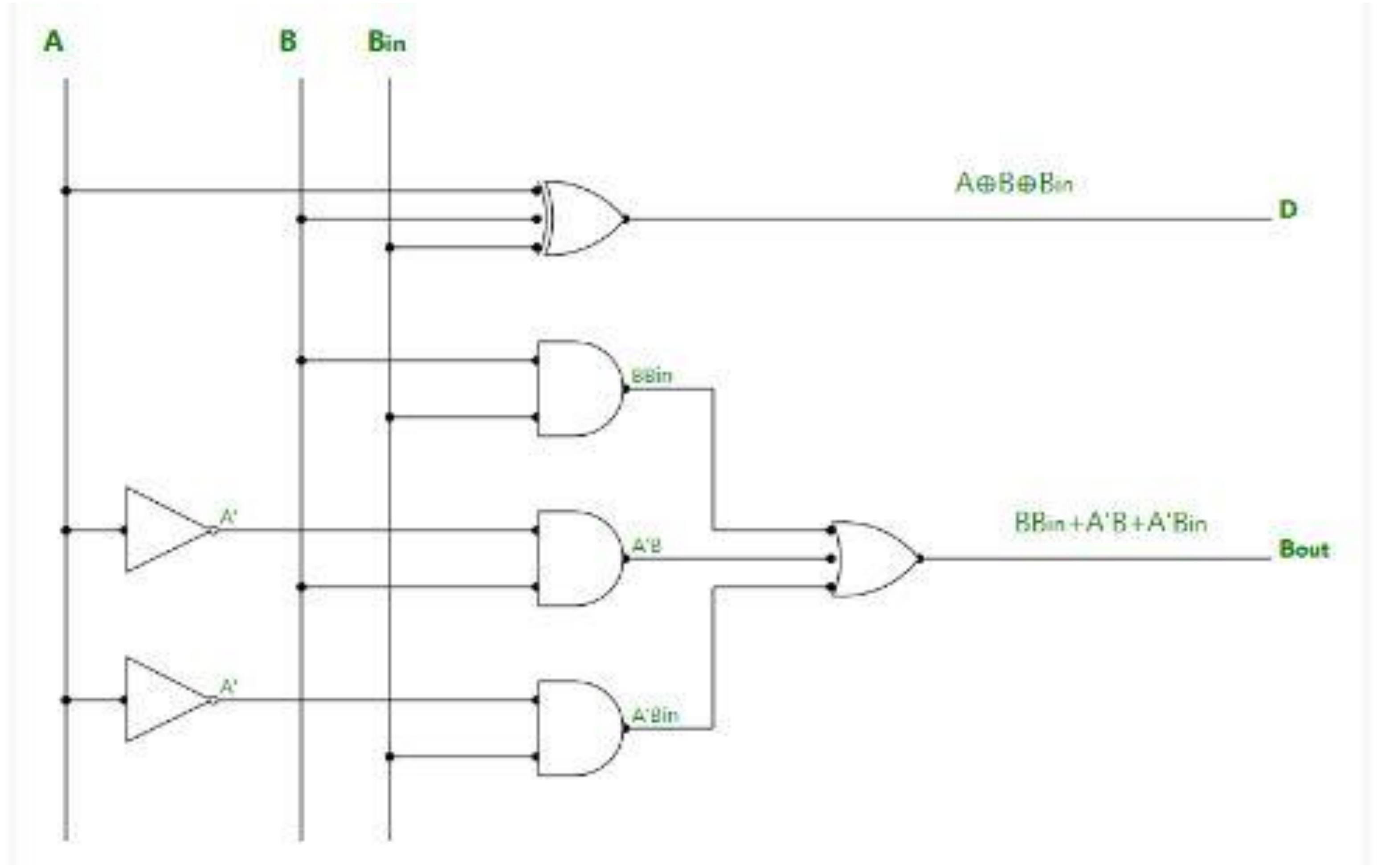
D = B'(A xor Bin) + B(A xor Bin)'

D = A xor B xor Bin

For Borrow,

Bout = BBin + A'B + A'Bin

Full Subtractor



Exercise

1. Design a full adder circuit using only NOR gates
2. Design a full subtractor circuit using only NAND gates.
3. Design full adder circuit using Half adder.
4. Design full subtractor circuit using half subtractor.

Code Conversion

- The availability of a large variety of codes for the same discrete elements of information results in the use of different codes by different digital systems. It is sometimes necessary to use the output of one system as the input to another. A conversion circuit must be inserted between the two systems if each uses different codes for the same information. Thus, a code converter is a circuit that makes the two systems compatible even though each uses a different binary code.
- To convert from binary code A to binary code B, code converter has input lines supplying the bit combination of elements as specified by code A and the output lines of the converter generating the corresponding bit combination of code B. A Code converter (combinational circuit) performs this transformation by means of logic gates.
- The design procedure of code converters will be illustrated by means of a specific example of conversion from the BCD to the excess-3 code. I will describe 5-step design procedure of this code converter so that you guys will be able to understand how practical combinational circuits are designed.

BCD to Excess-3 Code Converter

Specification

Transforms BCD code for the decimal digits to Excess-3 code for the decimal digits

BCD code words for digits 0 through 9: 4-bit patterns 0000 to 1001, respectively.

Excess-3 code words for digits 0 through 9: 4-bit patterns consisting of 3 (binary 0011) added to each BCD code word

Implementation:

multiple-level circuit

2. **Formulation**

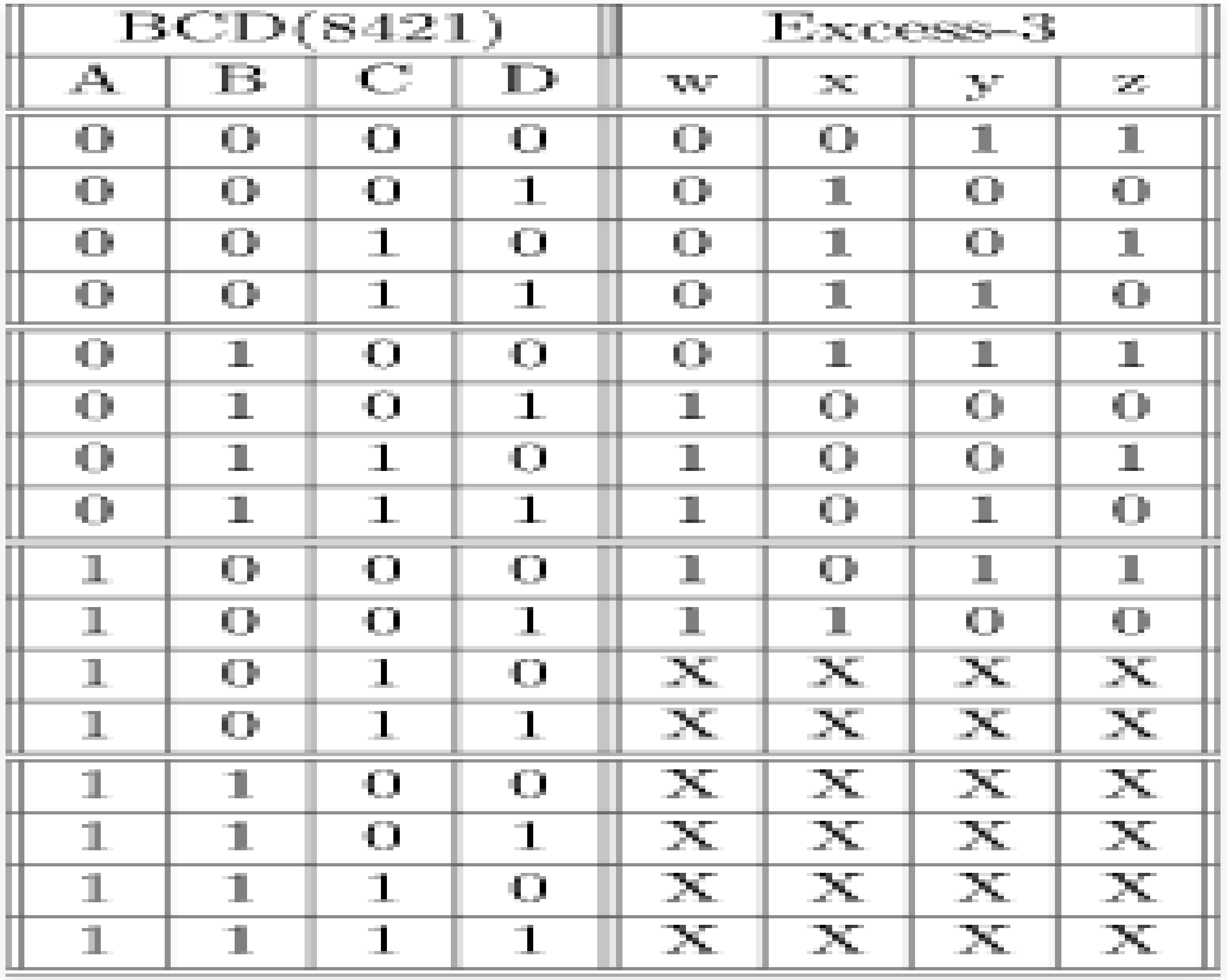
Conversion of 4-bit codes can be most easily formulated by a truth table

Variables- BCD: A, B, C, D

Variables- Excess-3: W, X, Y, Z

Don’t Cares: BCD 1010 to 1111

BCD to Excess-3 Code Converter



BCD(8421)				Excess-3			
A	B	C	D	w	x	y	z
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
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

BCD to Excess-3 Code Converter

Now, draw K-map for BCD to excess -3 code convertor

………………………………………………………………

The Boolean functions for the outputs lines of the circuit are derived from k-maps which are:

$W = A + BC + BD = A + B(C + D)$

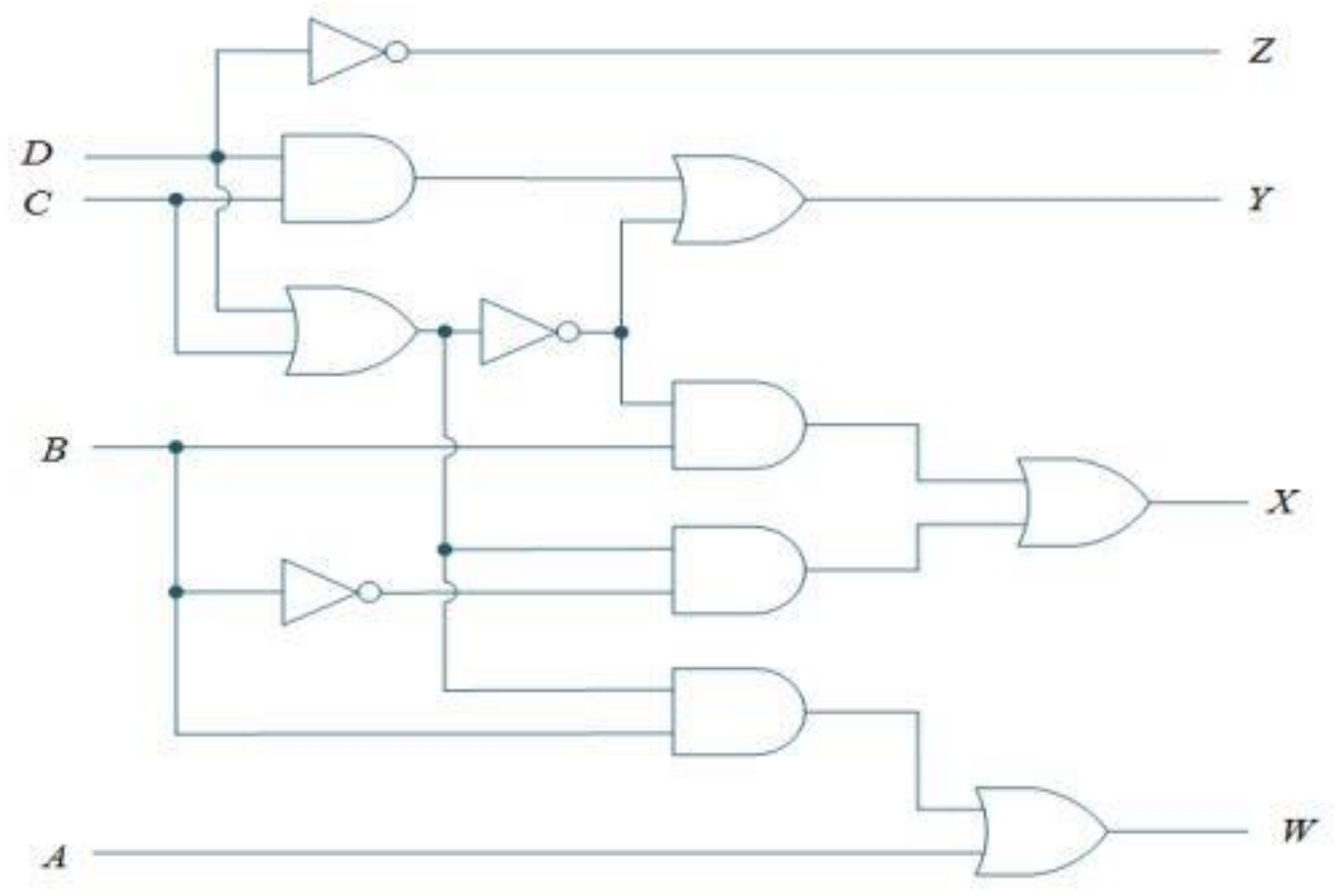
$X = B'C + B'D + BC'D' = B'(C + D) + B(C + D)'$

$Y = CD + C'D' = CD + (C + D)'$

$Z = D'$

Draw the logical diagram from above expression..

BCD to Excess-3 Code Converter



Exercise

1. **Design a circuit to convert Excess-3 code to BCD code.**

Analysis Procedure

To obtain the Boolean expressions and truth tables from the combinational logic circuit, we need to analyze the circuit. First ensure that the circuit is combinational - that is there is no feedback of an output to an input that the output depends on.

1st step: make sure that circuit is combinational.

2nd step: obtain the output Boolean functions or the truth table.

Obtaining Boolean functions from logic diagram:

Steps:

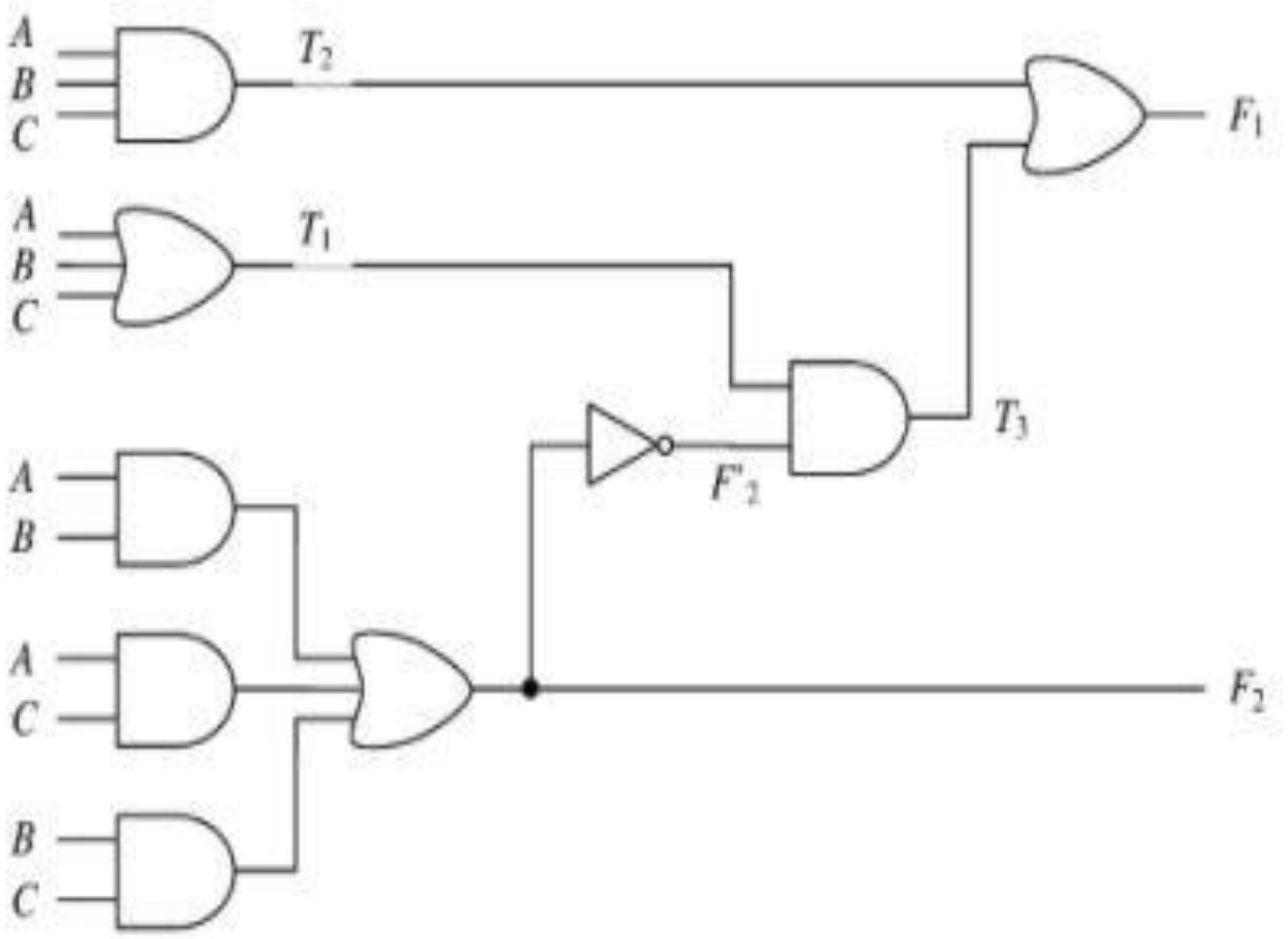
1. Label all gate outputs that are a function only of input variables or their complements with arbitrary symbols. Determine the Boolean functions for each gate output.

2. Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for the outputs of these gates.

3. Repeat the process outlined in step 2 until the outputs of the circuit are obtained.

4. By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables only.

Analysis Procedure



Analysis Procedure

Step 1:

$F2 = AB + AC + BC$

$T1 = A + B + C$

$T2 = ABC$

Step 2 & 3:

$T3 = F2'T1$

$F1 = T3 + T2$

Step 4:

$F1 = T3 + T1 = F2'T1 + ABC$

$=(AB + AC + BC)'(A + B + C) + ABC$

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

$=(A' + B'C')(AB' + AC' + BC' + B'C) + ABC$

$=A'BC' + A'B'C + AB'C' + ABC$

Analysis Procedure

Obtaining truth table from logic diagram:

1. Determine the number of input variables in the circuit. For n inputs, list the binary numbers from 0 to $2^n - 1$ in a table.

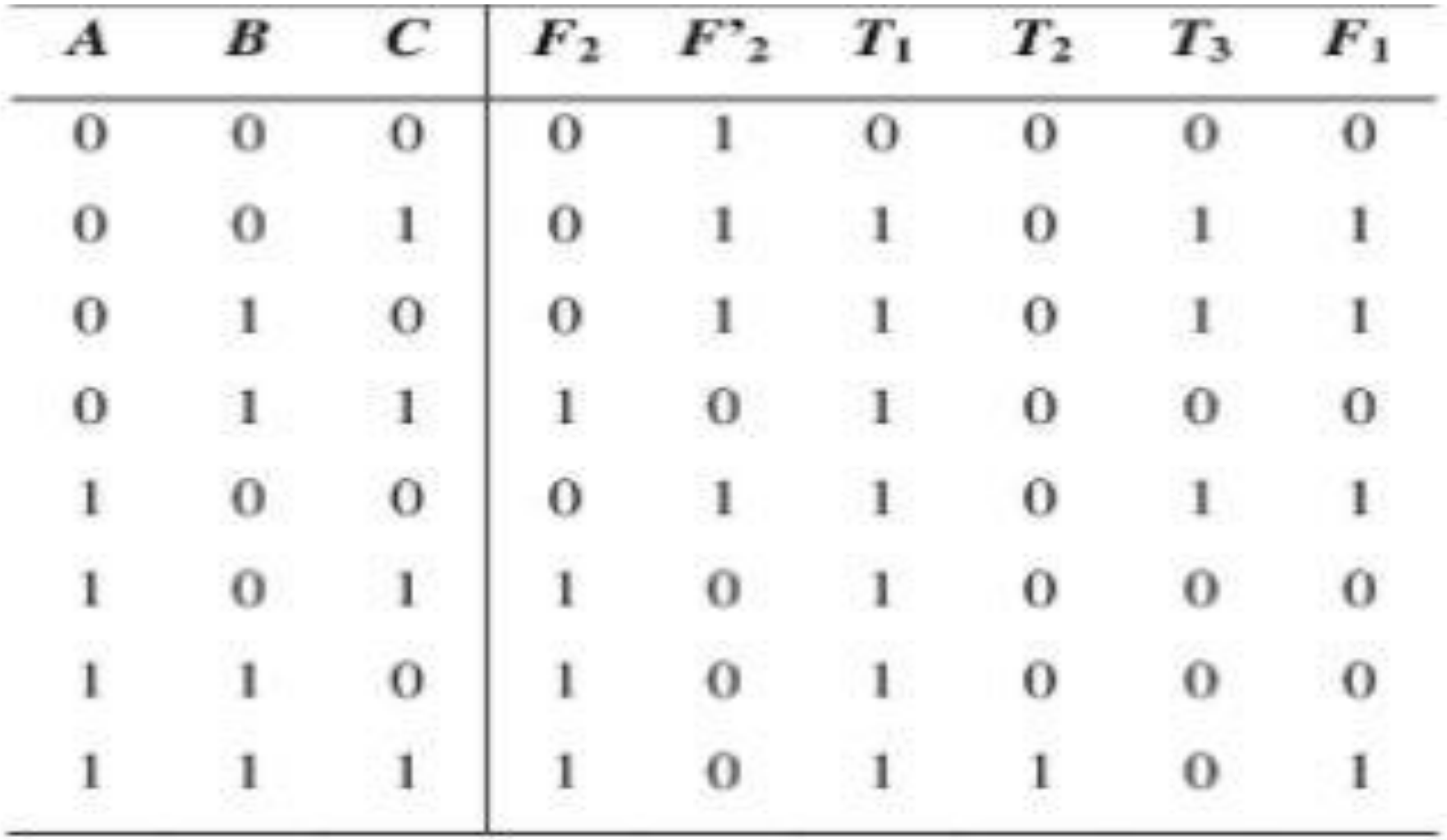
2. Label the output of selected gates.

3. Obtain the truth table for the output of those gates that are a function of the input variables only.

4. Obtain the truth table for those gates that are a function of previously defined variables at step 3, until all outputs are determined.

Truth table for the above logic diagram:

Analysis Procedure



A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1