Analyses and Synthesis of Combinational Logic Circuits

The important terms we are discussing in this section are

a. Logic Expression - a mathematical formula consisting of logical operators and variables.

b. Logic Operator - a function that gives a well-defined output according to switching algebra.

c. Logic Variable - a symbol representing the two possible switching algebra values of 0 and 1.

d. Logic Literal - the values 0 and 1 or a logic variable or it's complement.

The analysis means a digital circuit is given and we are asked to determine its input-output relationship (its purpose, operation, what it does). One studies the circuit and then states the input-output relationship of the circuit in text or on a truth table or on an operation table or on an operation diagram. The synthesis means an input-output relationship is given and we are asked to design the digital circuit. The input-output relationship is a very crucial component of digital circuit study. Complex digital circuits are blocks are analysed/designed individually and finally the whole circuit is analysed/designed.

Combinational circuit analysis starts with a schematic and answers the following questions:

What is the truth table(s) for the circuit output function(s)

What is the logic expression(s) for the circuit output function(s)

Two types of analyses are possible literal as well as symbolic analysis. Literal analysis is process of manually assigning a set of values to the inputs, tracing the results, and recording the output values. For 'n'' inputs there are 2n possible input combinations. From input values, gate outputs are evaluated to form next set of gate inputs and evaluation continues until gate outputs are circuit outputs. The literal analysis only gives us the truth table.

Symbolic analysis also starts with the circuit diagram like literal analysis. But instead of assigning values, gate output expressions are determined. Intermediate expressions are combined in following gates to form complex expressions. Symbolic analysis is more work but gives us complete information of both the truth table and logic expression.

Now we will consider an example for the analysis of combinational logic circuit shown in Fig. 3.

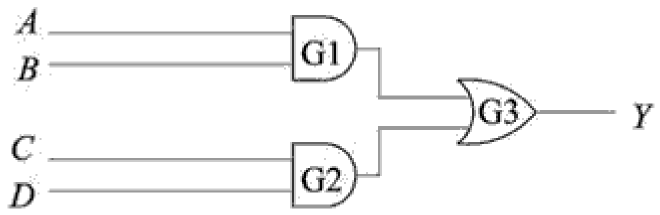


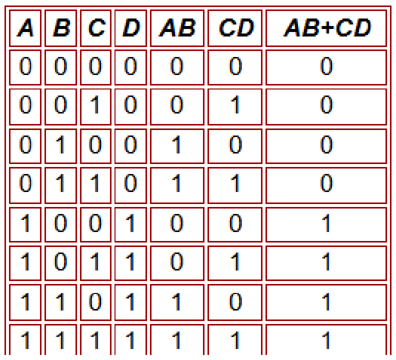
Figure 3

Analyzing this circuit, it can be seen that

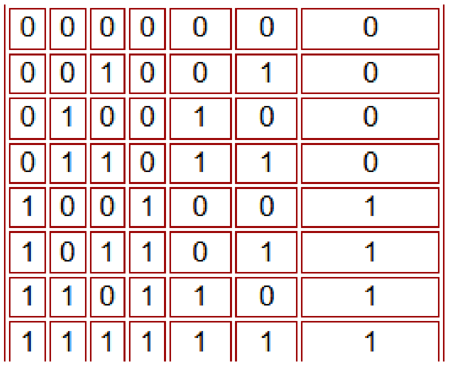
- Output of Gate G1 = *AB*
- Output of Gate G2 = *CD*
- Output of Gate G3 = *AB* + *CD*

From this we could then construct a truth table (Table 3) to calculate the output of the circuit. The truth table is constructed by considering the output of each gate in turn and then building up towards the complete output.

Table 3. Truth Table



A	*B*	*C*	*D*	*AB*	*CD*	*AB+CD*
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1



0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

Alternatively, the output of the circuit can be evaluated by substituting values directly into the logic equation.

For example, when *A = 1, B = 1, C = 1, D = 0*
then *Y = AB + CD = 1 . 1 + 1. 0 = 1 + 0 = 1*

This can then be repeated for all other input combinations.

The analysis is followed by synthesis i.e. we will consider how to design and implement a logic circuit to enable it to perform the desired specified operation. In this instance, we start with the equation and determine circuit to implement. For example consider the logic function

X = AB + CDE

This is composed of two terms, *AB* and *CDE* . The first term is formed by ANDing *A* and *B* and the second term is formed by ANDing together *C* , *D* and *E* . These two terms are then ORed together. This can then be implemented using the AND and OR gates, as shown in Fig. 4. Generally, as the number of levels are increased, the overall delay is increased due to the contribution of propagation delays at each gate.

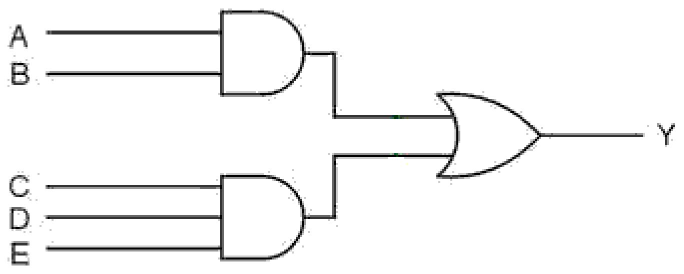


Figure 4: Implementation of X=AB+ CDE

Analysis also can be categorized into the functional analysis (determine what is computed) and the timing analysis (determine how long it takes to compute it). The logic expression is manipulated using Boolean (or switching) algebra and optimized to minimize the number of gates needed, or to use specific type of gates.

Canonical and Standard forms

A binary variable may be either in its true form A or its complement $\overline{A}$. For *n* variables, the maximum number of input variable combinations is given by $N = 2^n$. Then considering the AND gate, each of the N logic expressions formed is called a *standard product* or *minterm.* As indicated in Table 1-13, binary digits '1' and '0' are taken to represent a given variable for example A or its complement $\overline{A}$ respectively. Also from Table 1-13 note that each minterm is assigned a symbol (P_j) each where j is the decimal equivalent to the binary number of the minterm designated.

Similarly, if we consider an OR gate, each of the *N* logic expressions formed is called a *standard sum* or *maxterm.* In this case binary digits '1' and '0' are taken to represent a given complemented variable $\overline{A}$ and its true form A respectively. As shown in Table 1-13, a symbol (S_j) is assigned to each maxterm where j is the decimal equivalent to the binary number of the maxterm designated. Also observe that each maxterm is the complement of its corresponding minterm, and vice versa.

Input			Minterms		Maxterms	
A	B	C	Terms	Designation	Terms	Designation
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	P_0	$A+B+C$	S_0
0	0	1	$\overline{A}\,\overline{B}C$	P_1	$A+B+\overline{C}$	S_1
0	1	0	$\overline{A}B\overline{C}$	P_2	$A+\overline{B}+C$	S_2
0	1	1	$\overline{A}BC$	P_3	$A+\overline{B}+\overline{C}$	S_3
1	0	0	$A\overline{B}\,\overline{C}$	P_4	$\overline{A}+B+C$	S_4
1	0	1	$A\overline{B}C$	P_5	$\overline{A}+B+\overline{C}$	S_5
1	1	0	$AB\overline{C}$	P_6	$\overline{A}+\overline{B}+C$	S_6
1	1	1	ABC	P_7	$\overline{A}+\overline{B}+\overline{C}$	S_7

The minterms and maxterms may be used to define the two standard forms for logic expressions, namely the **sum of products (SOP)**, or sum of minterms, and the **product of sums (POS)**, or product of maxterms. These standard forms of expression aid the logic circuit designer by simplifying the derivation of the function to be implemented. Boolean functions expressed as a sum of products or a product of sums are said to be in canonical form. Note the POS is not the complement of the SOP expression.

SUM OF PRODUCTS (OR of AND terms)

The SOP expression is the equation of the logic function as read off the truth table to specify the input combinations when the output is a logical 1. To illustrate, let us consider Table 6.