Combinational Logic Circuits

Lecture-16

By
Bhagyalaxmi Behera
Asst. Professor (Dept. of ECE)

Combinational Logic

Introduction

- Logic circuits for digital systems may be combinational or sequential.

- A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs.

- In contrast, sequential circuits employ storage elements in addition to logic gates. Their outputs are a function of the inputs and the state of the storage elements. The circuit behaviour must be specified by a time sequence of inputs and internal states.

Combinational circuits

- A combinational circuit consists of an interconnection of logic gates. Combinational logic gates react to the values of the signals at their inputs and produce the value of the output signal, transforming binary information from the given input data to a required output data.

- Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer.

- Some of the characteristics of combinational circuits are following:

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.

- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.

- A combinational circuit can have an n number of inputs and m number of outputs.

- The n input binary variables come from an external source; the m output variables are produced by the internal combinational logic circuit and go to an external destination.
- For n input variables, there are 2^n possible combinations of the binary inputs.
- Thus, a combinational circuit can be specified with a truth table that lists the output values for each combination of input variables.
- A combinational circuit also can be described by m Boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.

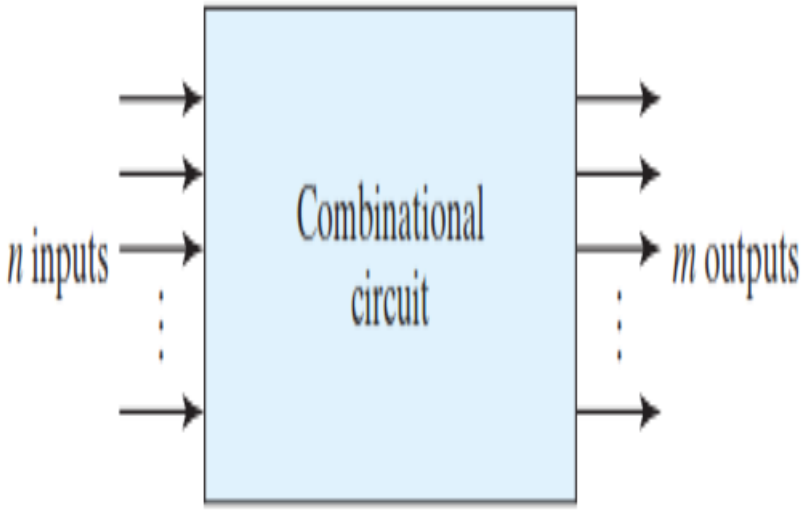


Fig 1 Block diagram of combinational circuit

Analysis Procedure

- The analysis of a combinational circuit requires that we determine the function that the circuit implements.

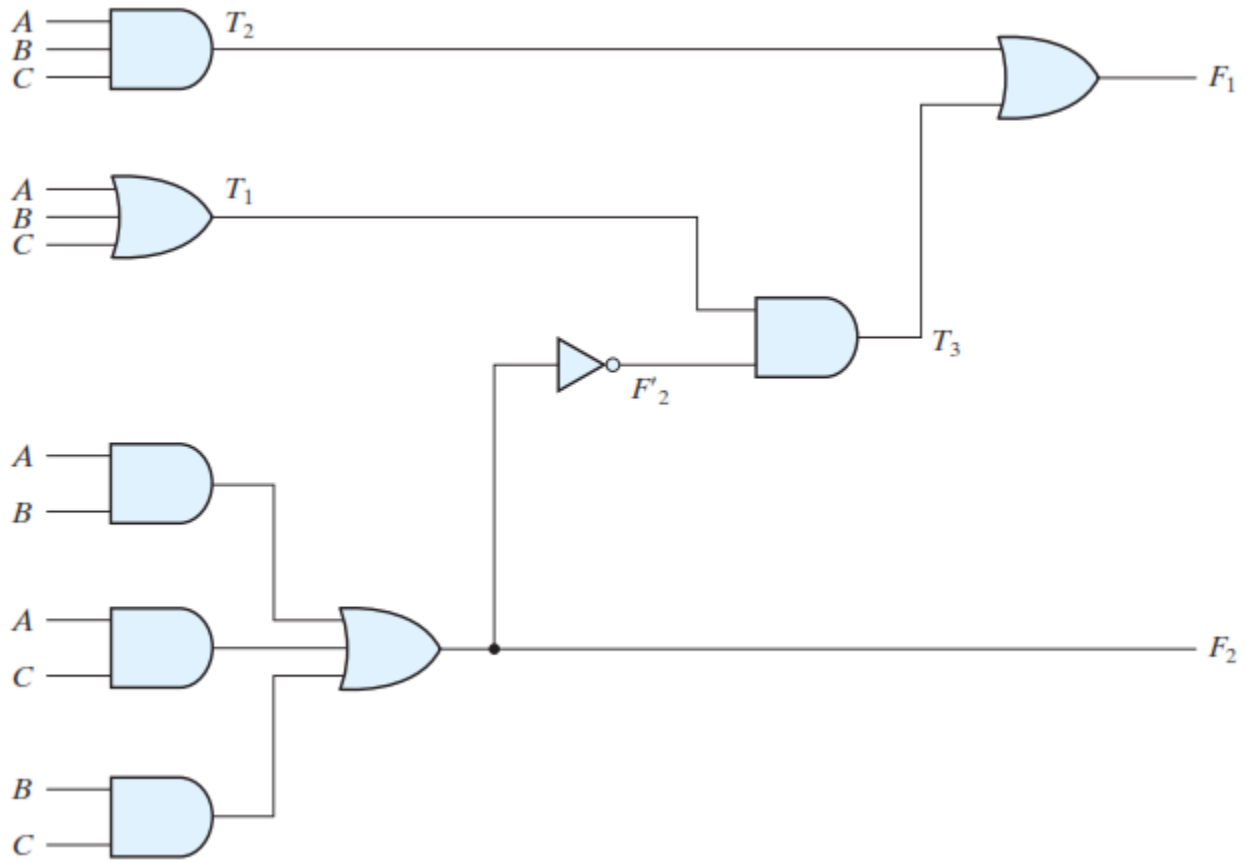
- To obtain the output Boolean functions from a logic diagram, we proceed as follows:

1. Label all gate outputs that are a function of input variables with arbitrary symbols but with meaningful names. 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 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.



$$F_2 = AB + AC + BC$$
$$T_1 = A + B + C$$
$$T_2 = ABC$$

Next, we consider outputs of gates that are a function of already defined symbols:

$$T_3 = F_2' T_1$$
$$F_1 = T_3 + T_2$$

To obtain F_1 as a function of A, B, and C, we form a series of substitutions as follows:

$$\begin{aligned} F_1 = T_3 + T_2 &= F_2' T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\ &= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\ &= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\ &= A'BC' + A'B'C + AB'C' + ABC \end{aligned}$$

- To obtain the truth table directly from the logic diagram without going through the derivations of the Boolean functions, we proceed as follows:

1. Determine the number of input variables in the circuit. For *n inputs, form the* 2^n possible input combinations and list the binary numbers from 0 to $(2^n - 1)$ in a table.
2. Label the outputs of selected gates with arbitrary symbols.
3. Obtain the truth table for the outputs of those gates which are a function of the input variables only.
4. Proceed to obtain the truth table for the outputs of those gates which are a function of previously defined values until the columns for all outputs are determined.

Truth Table

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

Design Procedure

- The design of combinational circuits starts from the specification of the design objective and culminates in a logic circuit diagram or a set of Boolean functions from which the logic diagram can be obtained. The procedure involves the following steps:

- From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.

- Derive the truth table that defines the required relationship between inputs and outputs.

- Obtain the simplified Boolean functions for each output as a function of the input variables.

- Draw the logic diagram and verify the correctness of the design (manually or by simulation).

Note:

- A truth table for a combinational circuit consists of input columns and output columns. The input columns are obtained from the 2^n binary numbers for the n *input* variables. The binary values for the outputs are determined from the stated specifications.
- The output functions specified in the truth table give the exact definition of the combinational circuit. The output binary functions listed in the truth table are simplified by any available method, such as algebraic manipulation, the map method, or a computer-based simplification program.
- A practical design must consider such constraints as the number of gates, number of inputs to a gate, propagation time of the signal through the gates, number of interconnections, limitations of the driving capability of each gate (i.e., the number of gates to which the output of the circuit may be connected), and various other criteria that must be taken into consideration when designing integrated circuits.

- Since the importance of each constraint is dictated by the particular application, it is difficult to make a general statement about what constitutes an acceptable implementation.

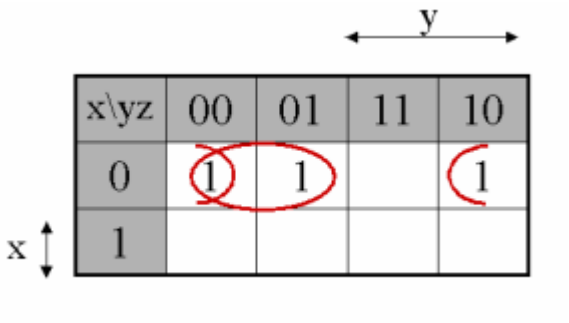
- In most cases, the simplification begins by satisfying an elementary objective, such as producing the simplified Boolean functions in a standard form.

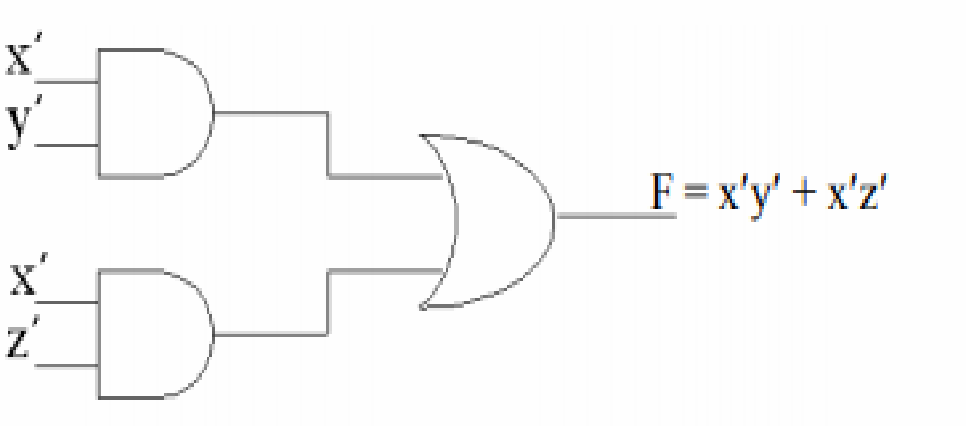
- Then the simplification proceeds with further steps to meet other performance criteria.

Some Examples:

1. Design a combinational circuit with three inputs and one output. The output is 1 when the binary value of the inputs is less than 3. The output is 0 otherwise.

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0





HDL for the above circuit:

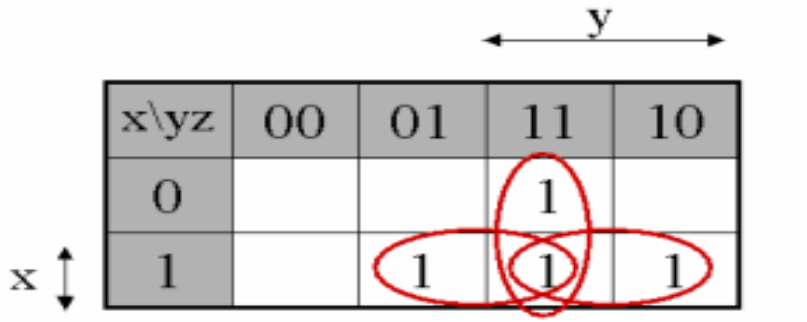
// Verilog model for combinational circuit

```
module circuit1-df (F, x, y, z);
output F;
input x, y, z;
assign F = ((!x) && (!y)) || ((!x) && (!z));
endmodule
```

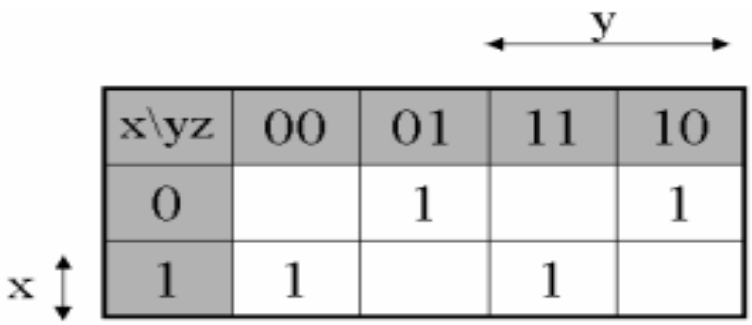
2) Design a combinational circuit with three inputs, x, y, and z, and three outputs, A, B, and C.

- **When the binary input is 0, 1, 2, or 3, the binary output is one greater than the input.**
- **When the binary input is 4, 5, 6, or 7, the binary output is one less than the input.**

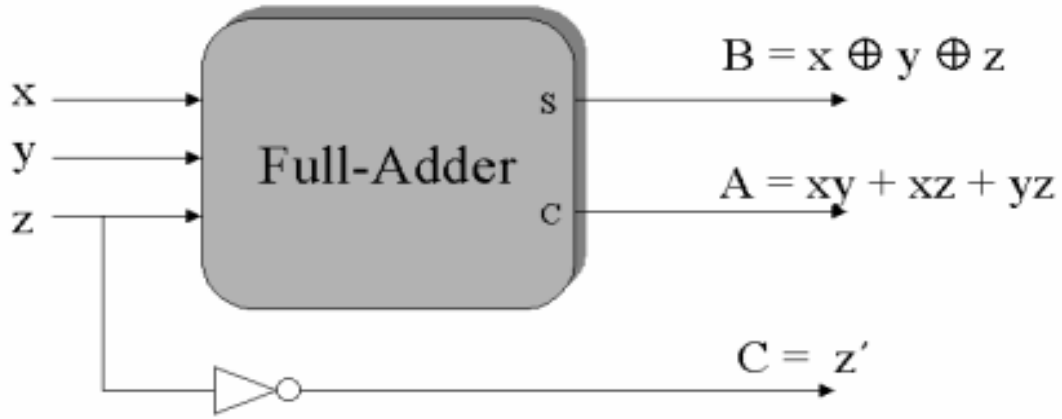
x	y	z	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0



$A = xy + xz + yz$



$B = x \oplus y \oplus z$ $\quad C = z'$



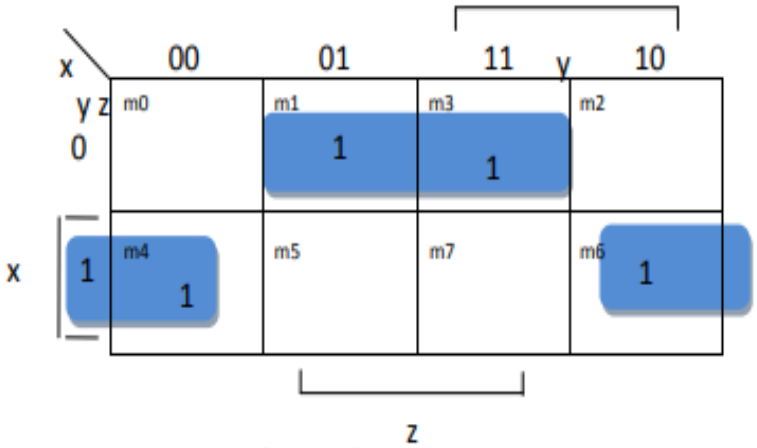
HDL for the above circuit:

// Verilog model for combinational circuit

```
module circuit2-df (A, B, C, x, y, z);
output A, B, C;
input x, y, z;
assign A =  (x && y) || (x && z) || (y && z);
assign  B = ((!x) && (!y) && (z)) || ((!x) && (y) && (!z)) || ((x) && (!y) && (!z)) ||
(x &&  y  &&  z);
Assign C = (!z);
endmodule
```

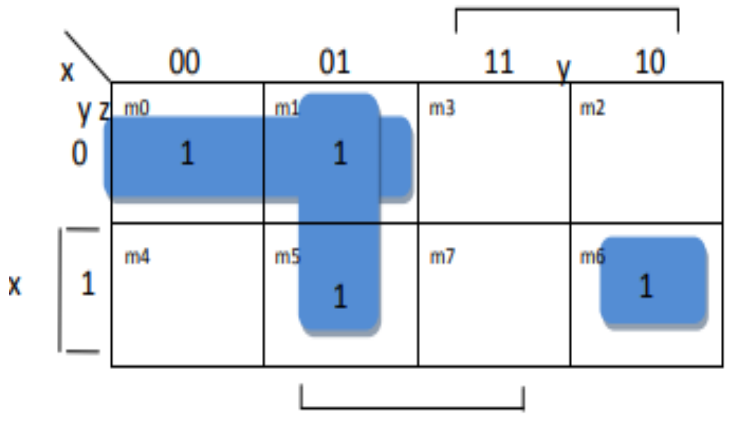
- **Design a combinational circuit with three inputs, x, y and z and three outputs A,B, and C. When Binary inputs is 0,1,2 or 3, the binary outputs is two greater than the input. When the binary input is 4,5,6 or 7, the binary output is three less than the input.**

x	Y	z		A	B	C
0	0	0		0	1	0
0	0	1		0	1	1
0	1	0		1	0	0
0	1	1		1	0	1
1	0	0		0	0	1
1	0	1		0	1	0
1	1	0		0	1	1
1	1	1		1	0	0

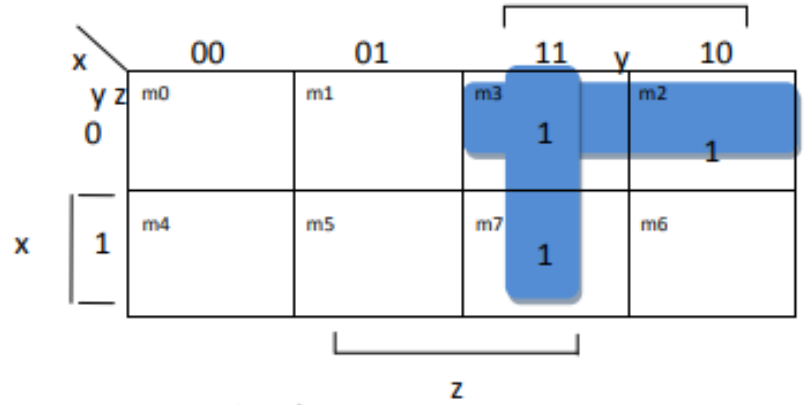


$C = x' z + x z'$

$C = x \oplus z$

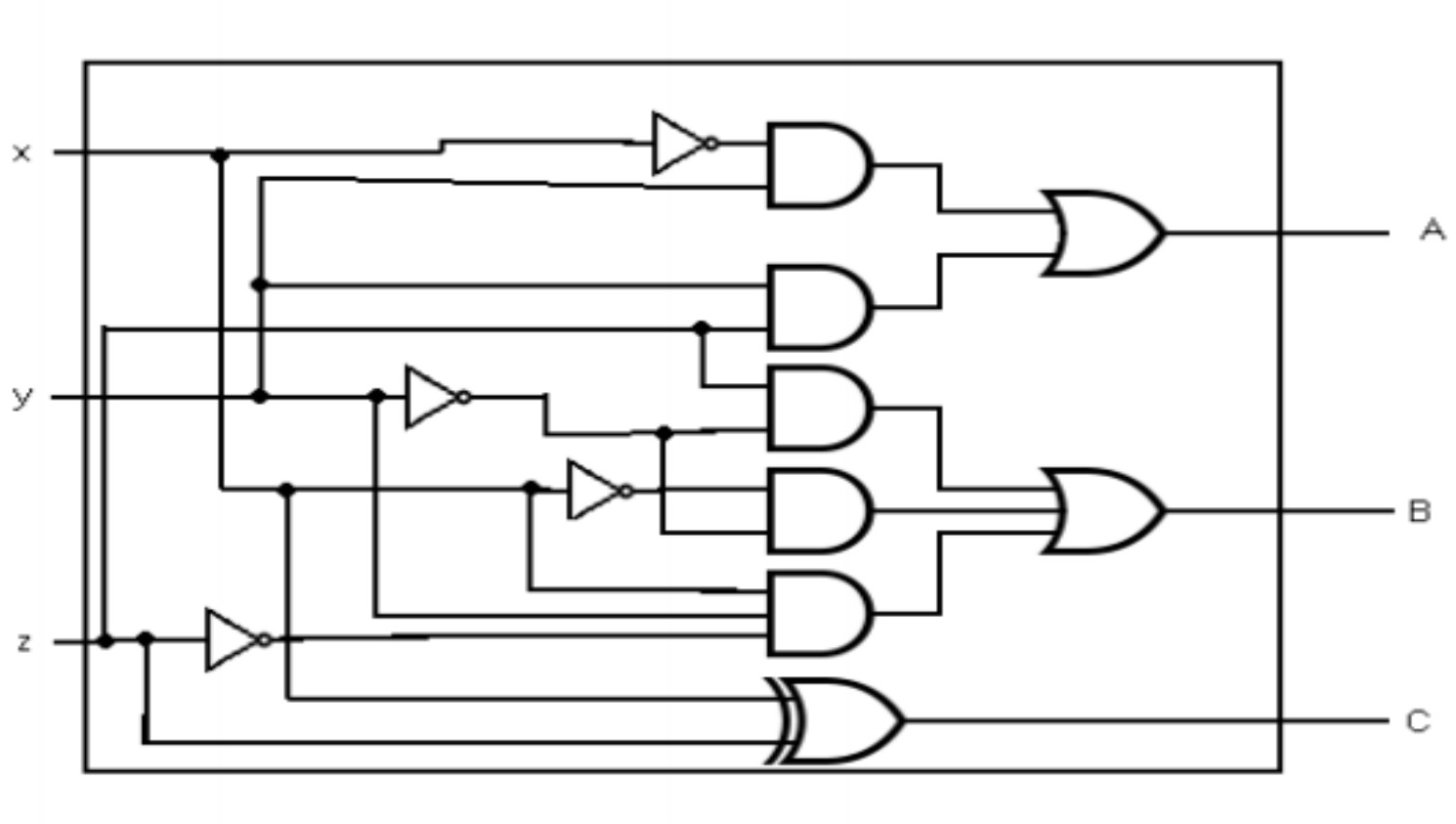


$B = y' z + x' y' + x y z'$



$A = x' y + y z$

The final circuit



HDL for the above circuit:

```
// Verilog model for combinational circuit
module circuit3-df (A, B, C, x, y, z);
output A, B, C;
input x, y, z;
assign A = ((!x) && y) || (y && z);
assign B = ((!y) && (z)) || ((!x) && (!y) || ((x) && (y) && (!z));
Assign C = ((!x) && z) || (x && (!z));
endmodule
```

THANK YOU