Chapter 4
Combinational Logic

Combinational Logic

- 4.1 Introduction
- 4.2 Design Procedure
- 4.3 Adders
- 4.4 Subtractors
- 4.5 Code Conversion
- 4.6 Analysis Procedure
- 4.7 Multilevel Nand Circuits
- 4.8 Multilevel NOR Circuits
- 4.9 Exclusive-OR and Equivalence Functions

4.1 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 directly from the present combination of inputs without regard to previous inputs.
- A combinational circuit performs a specific information-processing operation fully specified logically by a set of Boolean functions.
- Sequential circuits employ memory elements (binary cells) in addition to logic gates.

4.1 Introduction

- A combinational circuit consists of input variables, logic gates, and output variables.
- The logic gates accept signals from the inputs and generate signals to the outputs.
- This process transforms binary information from the given input data to the required output data.
- Obviously, both input and output data are represented by binary signals, i.e., they exist in two possible values, one representing logic-1 and the other logic-0.

4.1 Introduction

- A block diagram of a combinational circuit is shown in Fig. 4-1.

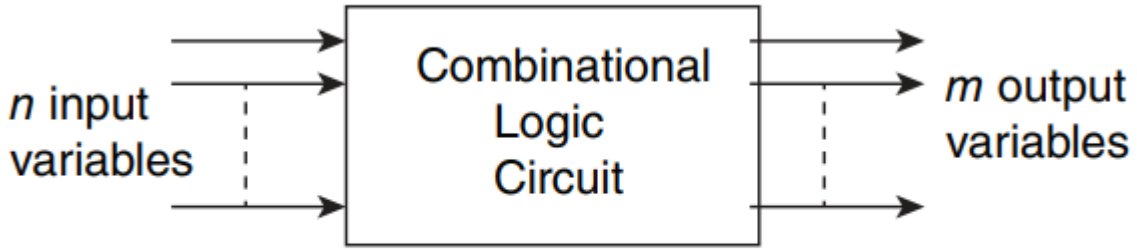

Figure 4.1 Block diagram of a combinational circuit

4.2 Design Procedure

- The design of combinational circuits starts from the verbal outline of the problem and ends in a logic circuit diagram, or a set of Boolean functions from which the logic diagram can be easily obtained.
- The procedure involves the following steps:

1. The problem is stated.
2. The number of available input variables and required output variables is determined.

4.2 Design Procedure

3. The input and output variables are assigned letter symbols.
4. The truth table that defines the required relationships between inputs and outputs is derived.
5. The simplified Boolean function for each output is obtained.
6. The logic diagram is drawn.

4.3 Adders

4.3.1 Half-Adder

4.3.2 Full-Adder

4.3.1 Half-Adder

- From the verbal explanation of a half-adder, we find that this circuit needs two binary inputs and two binary outputs.
- The input variables designate the augend and addend bits; the output variables produce the sum and carry.
- It is necessary to specify two output variables because the result may consist of two binary digits.
- We arbitrarily assign symbols x and y to the two inputs and S (for sum) and C (for carry) to the outputs.

4.3.1 Half-Adder

- The Truth table is shown below:

$S = x'y + xy'$

$C = xy$

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

4.3.2 Full-Adder

A full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y represent the two significant bits to be added. The third input, z represents the carry from the previous lower significant position.

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

4.3.2 Full-Adder

$S = x'y'z + x'yz' + xy'z' + xyz$

$C = xy + xz + yz$

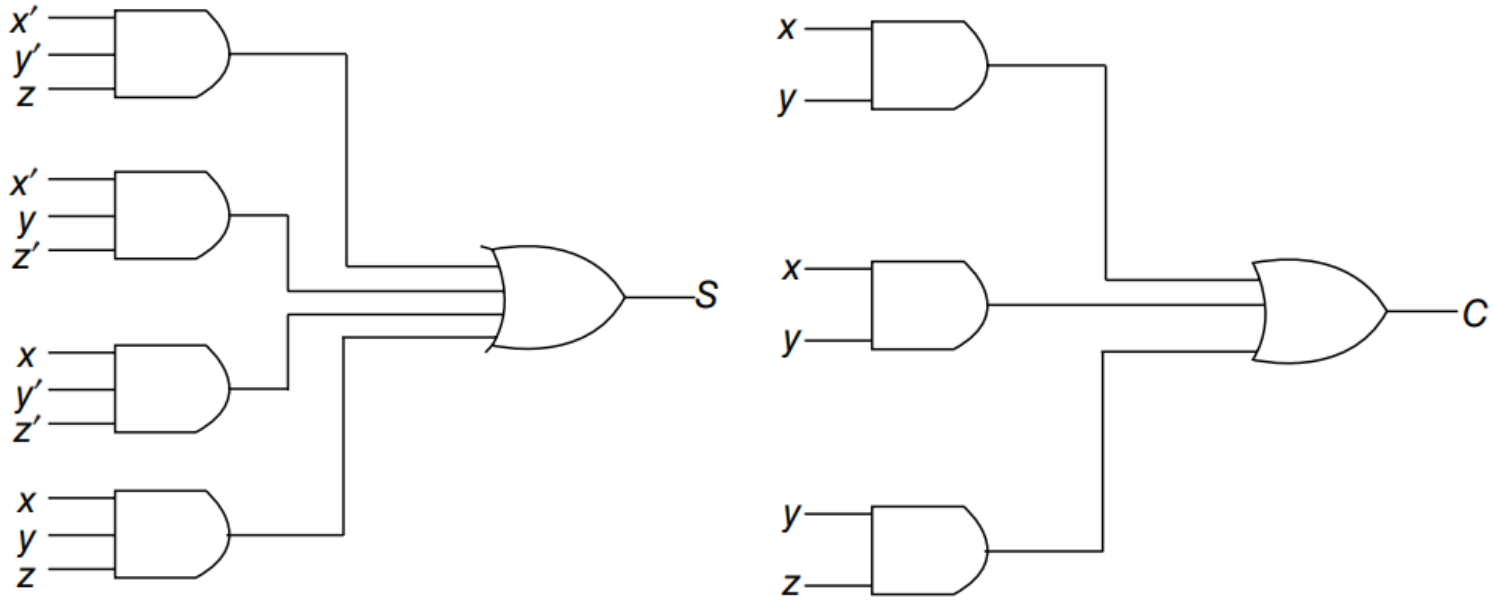

Figure 4.4 Implementation of full-adder in sum of products

4.4 Subtractors

- 4.4.1 Half Subtractor
- 4.4.2 Full Subtractor

4.4.1 Half Subtractor

- A half-subtractor is a combinational circuit that subtracts two bits and produces their difference.
- It also has an output to specify if a 1 has been borrowed.
- Designate the minuend bit by x and the subtrahend bit by y.
- To perform x - y, we have to check the relative magnitudes of x and y.
- If x > y, we have three possibilities: 0 - 0 = 0, 1 - 0 = 1, and 1 - 1 = 0.

4.4.1 Half Subtractor

$D = x'y + x\,y'$

$B = x'y$

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

4.4.2 Full Subtractor

- A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage.
- This circuit has three inputs and two outputs.
- The three inputs, x, y, and z, denote the minuend, subtrahend, and previous borrow, respectively.
- The two outputs, D and B, represent the difference and output borrow, respectively.
- The truth table for the circuit is as follows:

4.4.2 Full Subtractor

D = x′y′z + x′yz + x y′z′ + x yz

B = x′y + x′z + yz

x	*y*	*z*	*B*	*D*
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

4.5 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.

4.5 Code Conversion

Table 4-1 Truth table for code-conversion example

Input BCD				Output Excess-3 code			
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

$z = D'$

$y = CD + C'D'$

$= CD + (C + D)'$

$x = B'C + B'D + BC'D'$

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

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

$w = A + BC + BD$

$= A + B(C + D)$

BCD to Excess-3 Code Converter

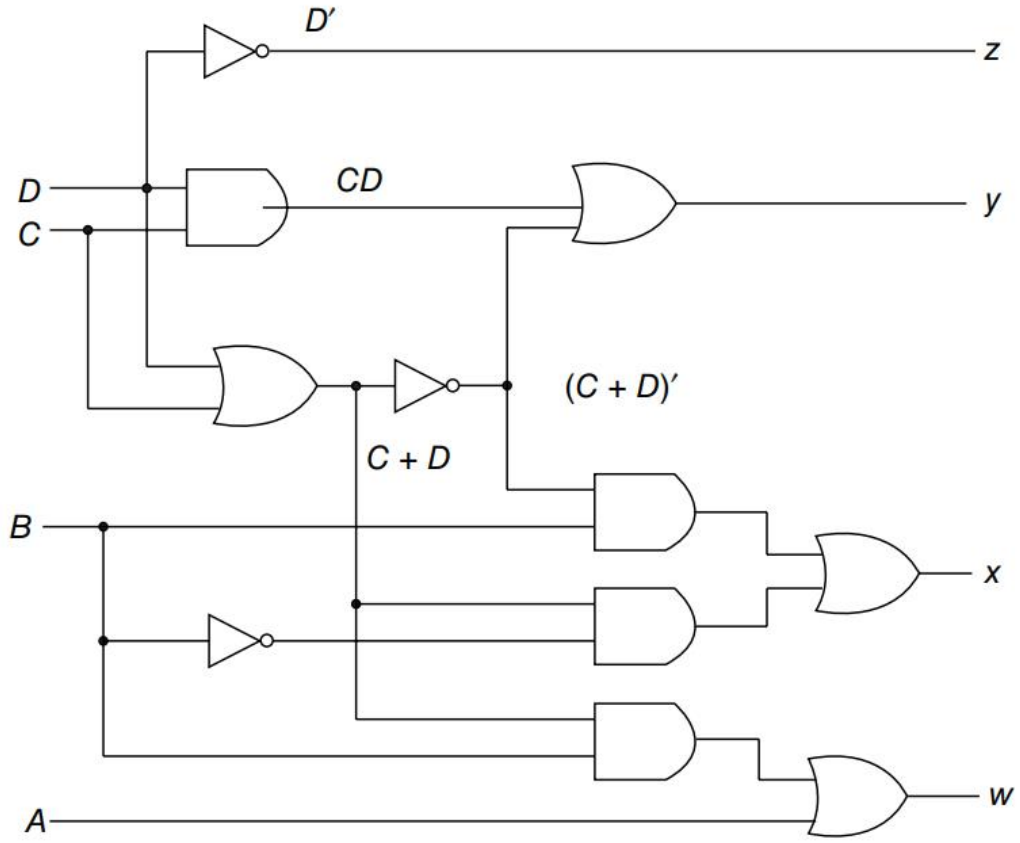

Figure 4.8 Logic diagram for BCD-to-excess-3 code coverter

4.6 Analysis Procedure

- The design of a combinational circuit starts from the verbal specifications of a required function and culminates with a set of output Boolean functions or a logic diagram.
- The analysis of a combinational circuit is somewhat the reverse process.
- It starts with a given logic diagram and culminates with a set of Boolean functions, a truth table, or a verbal explanation of the circuit operation. If the logic diagram to be analyzed is accompanied by a function name or an explanation of what it is assumed to accomplish, then the analysis problem reduces to a verification of the stated function.

4.6 Analysis Procedure

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

1. Label with arbitrary symbols all gate outputs that are a function of the input variables. Obtain the Boolean functions for each gate.
2. Label with other arbitrary symbols those gates which are a function of input variables and/or previously labeled gates. 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 only.

4.6 Analysis Procedure

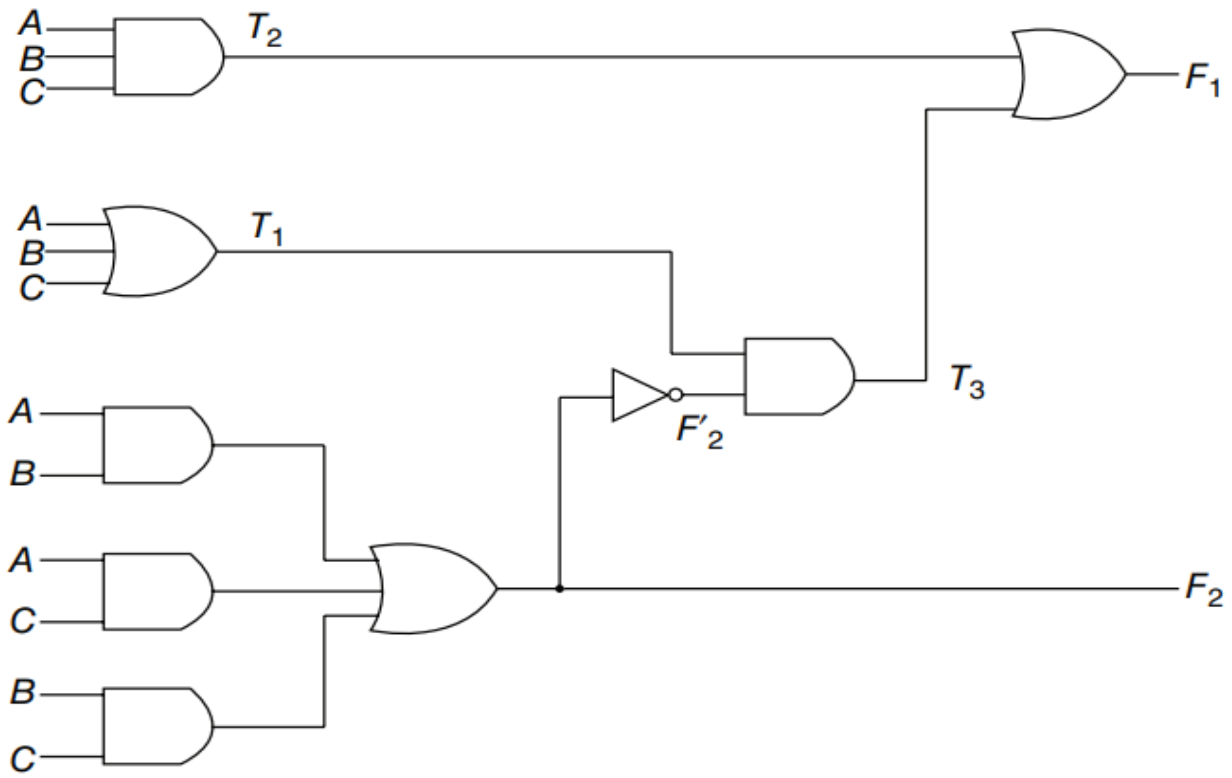

Figure 4.9 Logic diagram for analysis example

4.6 Analysis Procedure

$F2 = AB + AC + BC$

$T1 = A + B + C$

$T2 = ABC$

Now

$T3 = F' \; 2 \; T1$

$F1 = T3 + T2$

$F1 = T3 + T2 = F' \; 2 \; T1 + ABC$

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

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

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

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

4.6 Analysis Procedure

If we want to pursue the investigation and determine the information-transformation task achieved by this circuit, we can derive the truth table directly from the Boolean functions and try to recognize a familiar operation.

Table 4-2 Truth table for logic diagram of Fig. 4-9

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	I	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

4.6 Analysis Procedure

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

1. Determine the number of input variables to the circuit. For n inputs, form the 2n possible input combinations of 1’s and 0’s by listing the binary numbers from 0 to 2n - 1.
2. Label the outputs of selected gates with arbitrary symbols.
3. Obtain the truth table for the outputs of those gates that are a function of the input variables only.
4. Proceed to obtain the truth table for the outputs of those gates that are a function of previously defined values until the columns for all outputs are determined.

4.7 Multilevel Nand Circuits

Combinational circuits are more frequently constructed with NAND or NOR gates rather than AND and OR gates. NAND and NOR gates are more common from the hardware point of view because they are readily available in integrated-circuit form. Because of the prominence of NAND and NOR gates in the design of combinational circuits, it is important to be able to recognize the relationships that exist between circuits constructed with AND-OR gates and their equivalent NAND or NOR diagrams.

4.7.1 Universal Gate

4.7.2 Boolean Function Implementation — Block Diagram Method

4.7.3 Analysis Procedure

4.7.4 Derivation of the Boolean Function by Algebraic Manipulation

4.7.5 Derivation of the Truth Table

4.7.6 Block Diagram Transformation

4.7.1 Universal Gate

4.7.2 Boolean Function Implementation — Block Diagram Method

4.7.3 Analysis Procedure

4.7.4 Derivation of the Boolean Function by Algebraic Manipulation

4.7.5 Derivation of the Truth Table

4.7.6 Block Diagram Transformation