Combinational Logic

Sekhar Mandal

September 9, 2020

Definition of Combinational Circuit

- A combinational circuit consists of logic gate. The outputs of a combinational circuit depend on the present combination of the inputs only.
- A combinational circuit is used for a specific information processing task which is logically specified by a set of Boolean functions.
- Examples of Combinational circuits are
  (i) Adder/Subtractor
  (ii) Multiplexer
  (iii)De-multiplexer/Decoder

Design Procedure

The design of a combinational circuit starts from the verbal (written) specification of the problem and ends in a logic diagram or a set of Boolean functions from which the logic diagram can be easily obtained. We have to follow the steps given below for designing a combinational circuit.

- Identify the numbers of input and output variables.
- denote each of the variable with a letter symbol.
- Obtain the truth table which defines the relationships between input and output variables.
- Simply the Boolean expression for each of the output variable.
- Draw the logic diagram.

Addition of Two binary digits

A combinational circuit that performs addition of two binary bits is called *half-adder*.

1 According to the problem, there are two inputs and they are denoted by x and y. There are two outputs, they are sum and carry. The sum is denoted by S and carry is represented by C.

2 The truth table of the half-adder is as follows:

Input		Output	
x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3 Directly from truth table, the expressions for S and C are:
$S = \overline{x}y + x\overline{y} = x \oplus y$
$C = xy$

Full Adder

- A full adder is a combinational circuit that performs addition of three input bits. This circuit has three inputs and two outputs (sum $\rightarrow$ S and carry $\rightarrow$ C).
- The truth table of the full-adder is given below.

Input			Output	
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

Full Adder

- The input-output relationships of the full adder may be express by two Boolean expression, one for each output variable.
- Directly from the truth table sum and carry can be expressed as:

$$S = \overline{x}\ \overline{y}z + \overline{x}y\ \overline{z} + x\ \overline{x}\ \overline{z} + xyz$$
$$S = x \oplus y \oplus z$$
$$C = \overline{x}yz + x\overline{y}z + xy\overline{z} + xyz$$
$$C = xy + yz + zx$$

Full Adder

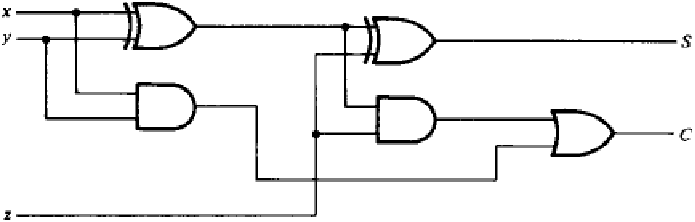

Figure: Implementation of Full-adder using two half-adder and an OR gate.

Half Subtractor

- A half subtractor is a combinational circuit that performs subtraction operation of two bits and produces their difference.
- This circuit has two outputs; (i) one is difference and, other one is borrow.
- To perform $x - y$, we have to check the relative magnitude of x and y. If $x \geq y$, there are three possibilities $0 - 0 = 0$, $1 - 0 = 1$ and, $1 - 1 = 0$. The result is called the difference.
- If $x < y$, the situation is $0 - 1$ and we have to borrow 1 from higher stage. Borrowed 1 from higher stage means add 2 with the minuend. Hence, the difference is $2 - 1 = 1$ and borrow is 1.

Half Subtractor

- The truth table of a half-subtractor can be derived as:

Input		Output	
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

- The Boolean functions for two outputs of the half-subtarctor are directly obtained from the truth table
  $D = \overline{x}y + x\overline{y}$
  $B = \overline{x}y$
  It is to be noted that the Boolean expression for D is same as the Boolean expression for S of a half-adder.

Full Subtractor

- A full subtractor is a combinational circuit that performs subtraction of two bits, taking into consideration that a 1 may have been borrowed by the lower significant stage.
- The circuit have three inputs and two outputs, difference (D) and, borrow (B). The truth table for this circuit is as follows.

Input			Output	
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

- The Boolean functions for two outputs are as follows:
  $D = \overline{x}\ \overline{y}z + \overline{x}y\ \overline{z} + x\overline{y}\ \overline{z} + xyz$
  $B = \overline{x}\ \overline{y}z + \overline{x}y\ \overline{z} + \overline{x}yz + xyz$

Code Conversion

- It is sometimes required to use the output of a digital system to the input of another system. If the two systems use different code then, it is necessary to insert a code converter between the two systems.
- Suppose we want to design a BCD to excess-3 code converter.
- The truth table of the converter is given below.

Input in BCD				Output in 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

Simplification of the Output Functions

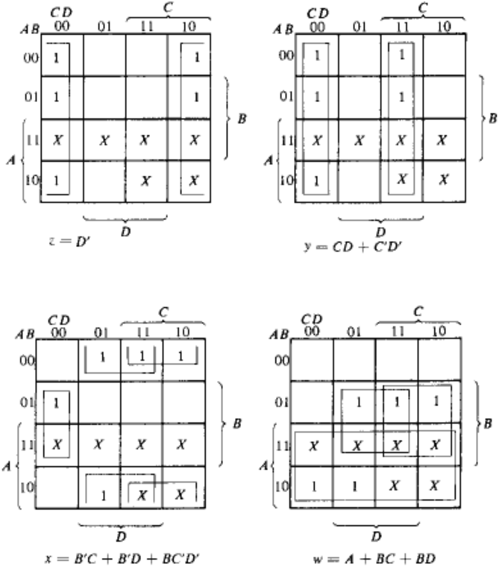

Figure: Maps for BCD to Excess-3 code converter.

Exclusive-OR Function

The following identities are applicable to XOR operation:

- $$\begin{aligned} x \oplus 0 &= x & \qquad x \oplus 1 &= \overline{x} \\ x \oplus x &= 0 & \qquad x \oplus \overline{x} &= 1 \\ x \oplus \overline{y} &= \overline{(x \oplus y)} & \qquad \overline{x} \oplus y &= \overline{(x \oplus y)} \end{aligned}$$
- Multi-input XOR gates are difficult to fabricate. Even two input XOR gate is constructed from other gates.
- The implementation of XOR function using NAND gate is shown below.

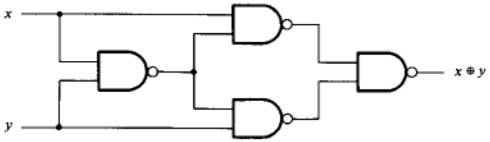

Figure: XOR implementation using NAND gates.

Exclusive-OR Function

- The first NAND produces the following term $\overline{(xy)} = (\overline{x} + \overline{y})$
- The rest of two-level NAND circuit produces the sum-of the product term.
  $(\overline{x} + \overline{y})x + (\overline{x} + \overline{y})y = x\overline{y} + \overline{x}y = x \oplus y$
- Only limited number of Boolean functions can be expressed in terms of XOR operation.
  It is useful in arithmetic operation, error detection and error correction circuits.

Parity Generation and Checking