Chapter

Digital System Design

Janani Rajaraman

Abstract

The main objective of this chapter is to study and design various combinational circuits like Verification of Boolean Expression, Multiplexer, Demultiplexer Circuits, Code Converters circuits using LabVIEW tools. This chapter will make the user more comfortable towards learning of Design of Digital Systems. The various types of Boolean Expressions like SOP and POS, Combinational circuits like Adder circuit (Half adder and full adder), Subtractor circuit (Half Subtractor, Full Subtractor), some code converters like Binary to Gray and Gray to Binary, BCD to Gray and Gray to BCD and also Sequential circuits with D flip flop is also being carried out using this LabVIEW.

Keywords: Combinational Circuits, Multiplexer, Code Converters, Adder and Subtractor Circuits

1. Introduction

The field of electronics are classified into two broad group namely analog electronics and digital electronics. Analog electronics deals with signals that are continuous with respect to time by nature such as any noise signal, any video streaming etc. and digital electronics deals with signals that are discontinuous or discrete with respect to time. The electronic amplifier such as op-amp circuit helps to amplify the continuous signals and such signals are termed as *analog signals* and the circuit used for such applications are called *analog circuits.* On the other hand, the discrete signals are fed as the input to the computer by electronic switches, which as two distinct values such as HIGH level and LOW level [1]. This discrete signals are further converted in to electronic signals with the help of suitable converters. Such discrete signals are called as *digital signals* and the electronic circuit used for such operation are termed as *digital circuits.*

2. Boolean Algebra

In a discrete signals the two distinct values such as HIGH and LOW has equivalent voltage levels such as 5 volts and 0 volts respectively. This two distinct levels are represented as value 1 and value 0 respectively. Any algebraic functions performed with respect to this discrete values are defined as *Boolean algebra* developed by George Boole. He also developed various suitable theorems associated with this boolean for manipulation and simplification. There are set of basic definitions which are assumed to be true which defines all the information about the system. The following are the basic definition used in boolean algebra.

- ***NOT:*** The NOT of a variable is 1 if and only if, the variable itself is 0 and vice versa
- ***AND:*** The AND of two variables is 1 if and only if *both* the variables are 1.
- ***OR:*** The OR of two variables is 1 if *either (or both)* of the variables is 1.
- ***XOR:*** The Exclusive-OR of two variables is 1 if *either* of them but *not both* is 1

3. Combinational logic circuits

There are two types of circuits exists in the digital system, Combinational Logic Circuits and Sequential Circuits. A combinational logic circuit is a circuit where the output depends on the combination of present input state. The set of operations which these combinational circuits performs logically by a set of Boolean functions. A sequential logic circuit is a circuit where the output depends on the combination of present input state and past input or previous input values. The previous output values are stored in the memory elements.

A combinational circuit consist of variables for input and output, and basic logic gates to perform the boolean function. The output signals are generated according to the inputs as well as the logic circuits employed. Here both the input and outputs are binary values either 1 or 0. **Figure 1** shows the simple block diagram of combinational logic circuits with n input variables and m output variables. If there are n number of inputs to the circuit then 2^n possible combinations of input states but each combination can produce only one output state [2]. For instance if the combinational logic circuit has 2 inputs A and B then there can be 4 possible input states.

.

3.1 Design procedure

The following are the steps considered while designing a combinational logic circuits.

- The first step is the statement of the problem for which this combinational circuit need to be designed
- Definition of the input and output variables and the variable name for inputs and outputs.
- Formation and tabulation of truth table which describes the relationship between the input and output.
- The simplified boolean expression is obtained for the output variable with the help of minimization techniques or Karanaugh map.

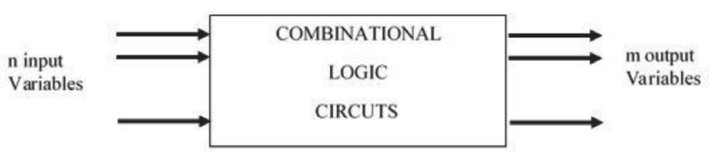

Figure 1.
Combinational Logic Circuits.

- The logical diagram using logic gates is realized for the simplified expression obtained in the previous step

- In practical design and real time implementation one should consider to use minimum number of gates.

3.2 Arithmetic circuits

3.2.1 Adder circuits

A combinational circuit that performs the addition of two bits is called *half-adder*. When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits. The combinational circuit that performs the addition of three bits is called a *full-adder*. The full adder can also be obtained by using two half adder circuits.

(i) Design of half-adders

A half adder is a combination logic circuit that uses two inputs (A and B) and two outputs (Sum S and Carry C). **Table 1** shows the truth table the various combinations of inputs and its corresponding outputs. The output Sum S and Carry C is obtained and the k-map is used to get the logical equation. The Boolean expressions are

$$Sum\ S = A \bigoplus B \tag{1}$$

$$Carry\ C = AB \tag{2}$$

Figure 2 shows the design and implementation of half adder circuit in LabVIEW environment, where the front panel that two inputs Input A and Input B, the outputs are Sum and Carry [3]. The block diagram in LabVIEW environment shows the logic gate implementation for the above obtained expression.

(ii) Design of full-adders

A full adder is a combination logic circuit that uses three inputs (A, B and C_{in}) and two outputs (Sum S and Carry C). **Table 2** shows the truth table the various combinations of inputs and its corresponding outputs. The output Sum S and Carry C is obtained and the k-map is used to get the logical equation.

$$Sum\ S = A \bigoplus B \bigoplus C_{in} \tag{3}$$

Input Variables		***Output Variables***	
A	**B**	**Sum S**	**Carry C**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 1.
Truth Table – Half Adder.

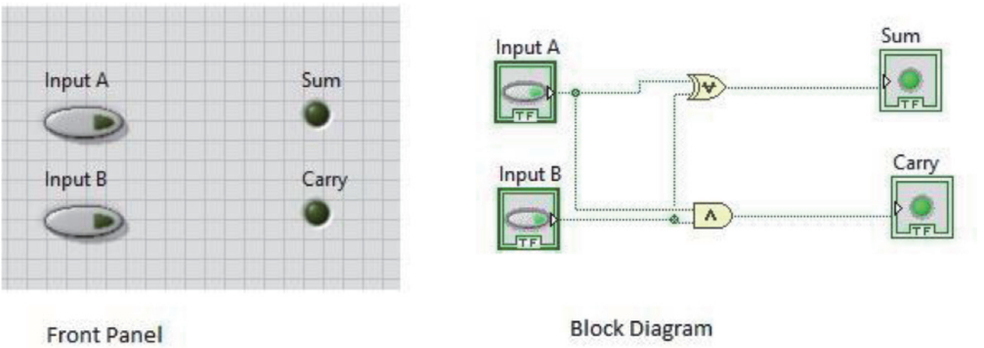

Figure 2.
Logic Diagram – Half Adder.

Input Variables			*Output Variables*	
A	**B**	**C_{in}**	**Sum S**	**Carry C**
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

Table 2.
Truth Table – Full Adder.

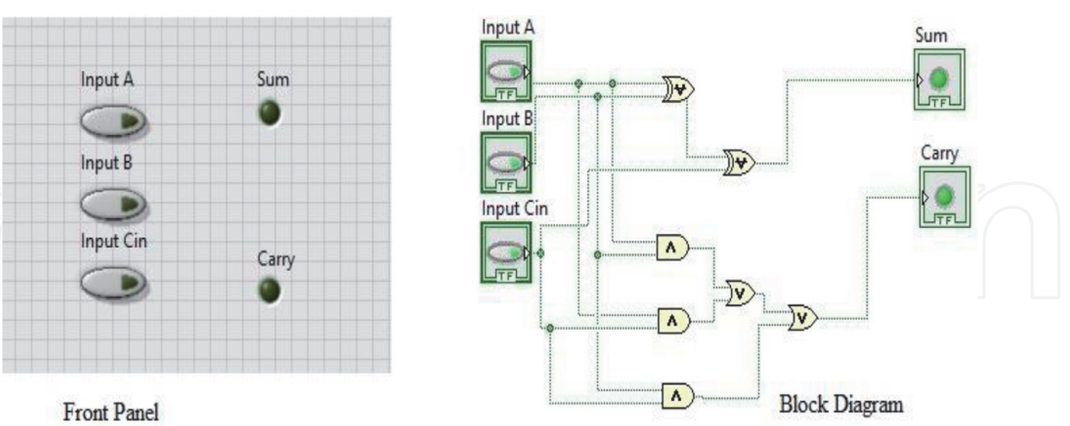

Figure 3.
Logic Diagram – Full Adder.

$$Carry\ C = AB + BC_{in} + C_{in}A \tag{4}$$

Figure 3 shows the design and implementation of full adder circuit in LabVIEW environment, where the front panel that two inputs Input A, Input B, and Input C_{in}, the outputs are Sum and Carry. The block diagram in LabVIEW environment shows the logic gate implementation for the above obtained expression.

3.2.2 Subtractor circuits

A combinational circuit that performs the difference of two bits is called *half-subtractor*. When the first input (minuend) is 0 and the second input(subtrahend) is 1 then there exists a output variable as *Borrow*. The combinational circuit that determines the difference of three bits is called a *full-subtractor*.

(i) Design of half-subtractor

A half subtractor is a combination logic circuit that uses two inputs (A and B) and two outputs (Difference D and Borrow B). **Table 3** shows the truth table the various combinations of inputs and its corresponding outputs. The output Difference D and Borrow B is obtained and the k-map is used to get the logical equation. **Figure 4** shows the design and implementation of half subtractor circuit in LabVIEW environment, where the front panel that two inputs Input A and Input B, the outputs are Difference and Borrow

$$Difference = A \bigoplus B \tag{5}$$

$$Borrow = \overline{A}B \tag{6}$$

(ii) Design of Full-subtractor

A full subtractor is a combination logic circuit that uses three inputs (A, B and B_{in}) and two outputs (Difference D and Borrow B). **Table 4** shows the truth table the various combinations of inputs and its corresponding outputs. The output difference D and Borrow B is obtained and the k-map is used to get the logical equation

$$Difference\ D = A \bigoplus B \bigoplus B_{in} \tag{7}$$

$$Borrow\ B = \overline{A}B + A \bigoplus B \tag{8}$$

Input Variables		***Output Variables***	
A	**B**	**Difference D**	**Borrow B**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 3.
Truth Table – Half Subtractor.

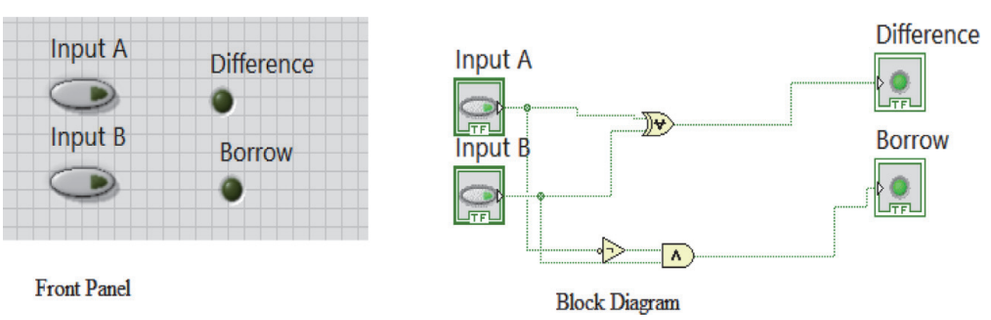

Figure 4.
Logic Diagram – Half Subtractor.

Input Variables			*Output Variables*	
A	**B**	B_{in}	**Difference D**	**Borrow B**
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

Table 4.
Truth Table – Full Subtractor.

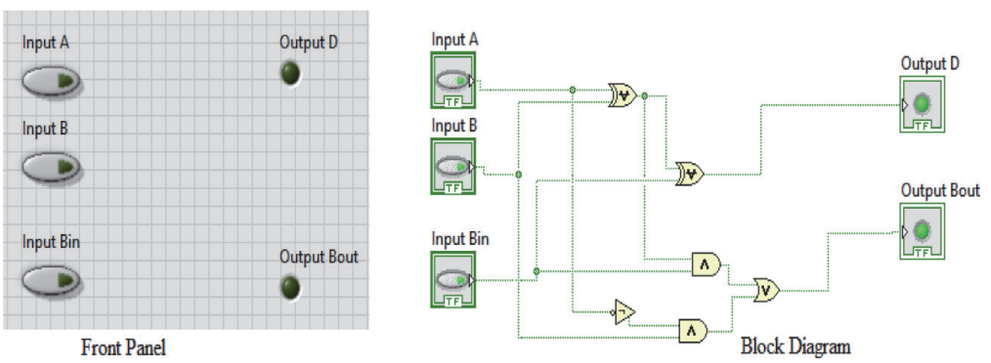

Figure 5.
Logic Diagram – Full Subtractor.

Figure 5 shows the design and implementation of full subtractor circuit in LabVIEW environment, where the front panel that two inputs Input A, Input B, and Input B_{in}, the outputs are Difference D and Borrow Ouptut. The block diagram in LabVIEW environment shows the logic gate implementation for the above obtained expression.

3.3 Multiplexer and demultiplexer

The most important form of a combinational circuit and which is widely used in the field of communication is *Multiplexer and Demultiplexer Circuits.* The multiplexer circuit is used to transfer large number of channels carrying information to a smaller number of channels. Such circuit used to transmit digital data or binary information is called as *data selector or digital multiplexer.* In this data selector, the input line is selected according to the combination of select lines, suppose if there exists 2^n input line then the number of select line is *n* and there will be *only one* output line. For example in a 4x1 multiplexer, the number of input lines is 4 (2^2) which shows there exists of *2* select lines. In these multiplexer circuits the inputs are named as *I0, I1, I2 and I3* and the two select lines are named as *S0* and *S1*. **Table 5** shows the various combinations of select line and corresponding input line is selected and obtained as output Y [4]. The boolean expression for the output Y is given below. **Figure 6** shows the Front panel and Block Diagram of 4x1 multiplexer. This design can be extended for higher versions like 8x1 and 16x1 types of multiplexer.

Selection Inputs			*Input Channels*			*Output*
S1	*S0*	*I0*	*I1*	*I2*	*I3*	*Y*
0	0	0	X	X	X	0
0	0	1	X	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	0	X	0	X	0
1	0	1	X	1	X	1
1	1	0	X	X	0	0
1	1	1	X	X	1	1

Table 5.
*Truth Table – 4*1 Multiplexer.*

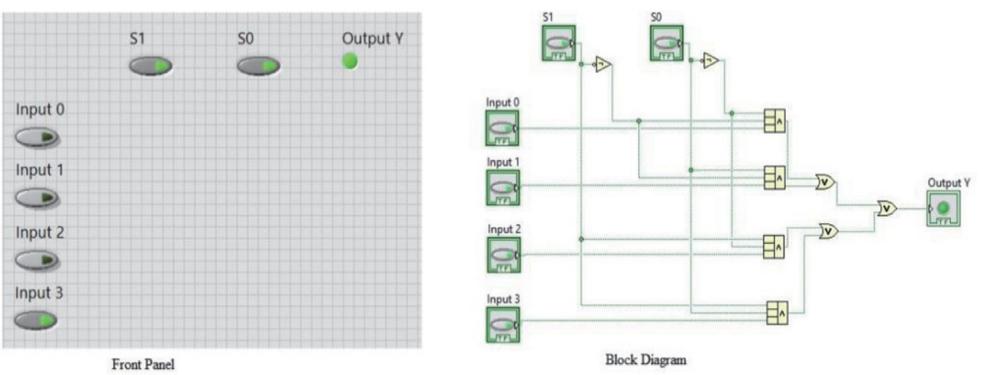

Figure 6.
*Logic Diagram – 4*1 Multiplexer.*

The Boolean expressions are

$$Output\ Y = \overline{S_0}\overline{S_1}I_0 + \overline{S_0}S_1I_1 + S_0\overline{S_1}I_2 + S_0S_1I_3 \tag{9}$$

The term demultiplex is just a opposite way of multiplexer, here in this combinational circuit there are one input channel and distributes the data over several channels. Therefore if the number of input channel is *1* then the number of output will be 2^n output channels. The combination of select lines control the output channel through which the input data must be transmitted. **Table 6** gives the truth table for 1-to-8 demultiplexer, the front panel and block diagram for 1-to-8 demultiplexer is shown in **Figure 7**.

The selection input line S0, S1, S2 are activated according to the bit combination for each output as given in Eq. (10) to Eq. (17). For instance, if the selection input combination is 010, the input *I* is transmitted to Y2. The Boolean expressions for each output line is given below

$$Y0 = \overline{S_2}\overline{S_1}\overline{S_0}I \tag{10}$$

$$Y1 = \overline{S_2}\overline{S_1}S_0I \tag{11}$$

$$Y2 = \overline{S_2}S_1\overline{S_0}I \tag{12}$$

$$Y3 = \overline{S_2}S_1S_1I \tag{13}$$

$$Y4 = S_2\overline{S_1}\overline{S_0}I \tag{14}$$

Selection Inputs			*Output Channels*							
S2	*S1*	*S0*	*Y0*	*Y1*	*Y2*	*Y3*	*Y4*	*Y5*	*Y6*	*Y7*
0	0	0	Y0 = I	0	0	0	0	0	0	0
0	0	1	0	Y1 = I	0	0	0	0	0	0
0	1	0	0	0	Y2 = I	0	0	0	0	0
0	1	1	0	0	0	Y3 = I	0	0	0	0
1	0	0	0	0	0	0	Y4 = I	0	0	0
1	0	1	0	0	0	0	0	Y5 = I	0	0
1	1	0	0	0	0	0	0	0	Y6 = I	0
1	1	1	0	0	0	0	0	0	0	Y7 = I

Table 6.
*Truth Table – 1*8 Demultiplexer.*

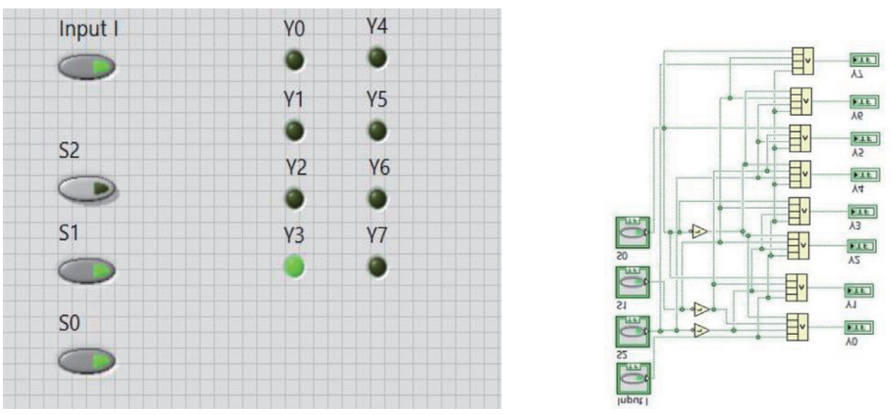

Figure 7.
*Logic Diagram – 1*8 Demultiplexer.*

$$Y5 = S_2\overline{S_1}S_0I \tag{15}$$

$$Y6 = S_2S_1\overline{S_0}I \tag{16}$$

$$Y7 = S_2S_1S_0I \tag{17}$$

3.4 Code converters

For the same discrete elements of information, there are several different codes available, resulting in the use of different codes for different digital systems. It's sometimes necessary to connect two digital blocks that use different coding systems. Hence a conversion digital circuit is designed and implemented between two digital systems to use information of one digital system to another. The input lines must provide the bit combinations of elements as designed by binary code A and the output is generated by the bit combinations of code B. This code converters circuit consisting of logic gates to perform this transformation operations. Some of the few code conversion techniques are discussed below.

3.4.1 Binary to gray code converters

Table 7 shoes the conversion of 4-bit binary code to its equivalent gray code values. The 4 bit binary code input is defined as B_0, B_1, B_2, B_3 and corresponding

B_0	B_1	B_2	B_3	G_0	G_1	G_2	G_3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Table 7.
Truth Table – Binary to Gray Converter.

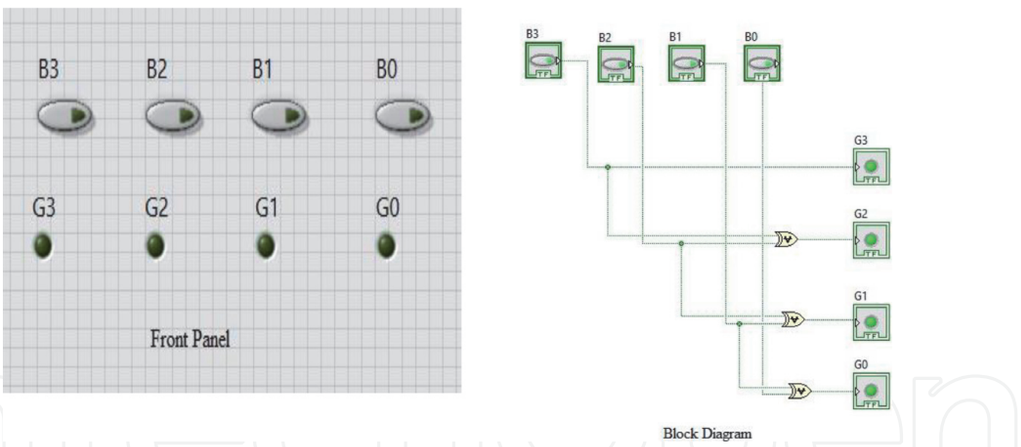

Figure 8.
Logic Diagram – Binary to Gray Code Converter.

output 4-bit gray is defined as G_0, G_2, G_3, and G_4 as shown in **Figure 8**. The corresponding boolean expression for binary to gray code conversion is given below

$$G_0 = B_0 \tag{18}$$

$$G_1 = B_0 \bigoplus B_1 \tag{19}$$

$$G_2 = B_2 \bigoplus B_1 \tag{20}$$

$$G_3 = B_3 \bigoplus B_2 \tag{21}$$

3.4.2 Gray to binary code converters

Gray code is also called as Reflected Binary Code (RBC), Reflected Binary (RB) or Gray code, Cyclic Code, is defined as an ordering of the binary number system

such that each incremental value can only differ by one bit. The main objective in this code converter is that while traversing from one step to another step, one bit in the code group changes as in **Figure 9**. This gray code is not applicable for arithmetic operations, but it is applicable in analog to digital converters, as well as error correction techniques in digital communications (**Table 8**).

$$B_3 = G_3 \tag{22}$$

$$B_2 = G_2 \bigoplus G_3 \tag{23}$$

$$B_1 = G_1 \bigoplus G_2 \bigoplus G_3 \tag{24}$$

$$B_0 = G_1 \bigoplus G_2 \bigoplus G_3 \bigoplus G_0 \tag{25}$$

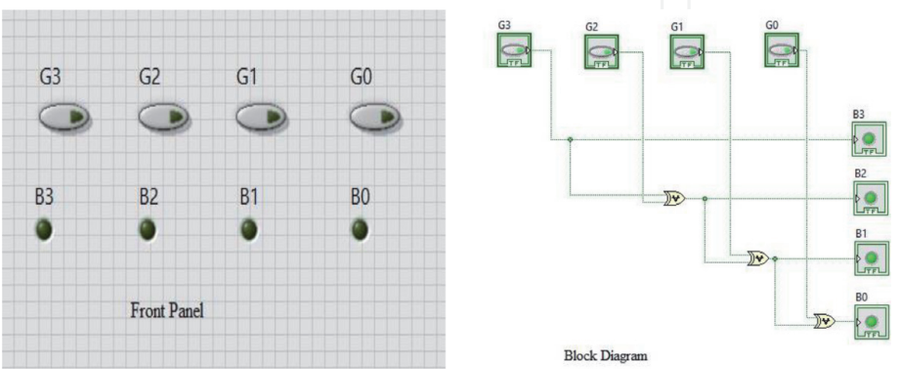

Figure 9.
Logic Diagram – Gray to Binary Code Converter.

G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Table 8.
Truth Table – Gray to Binary Code Converter.

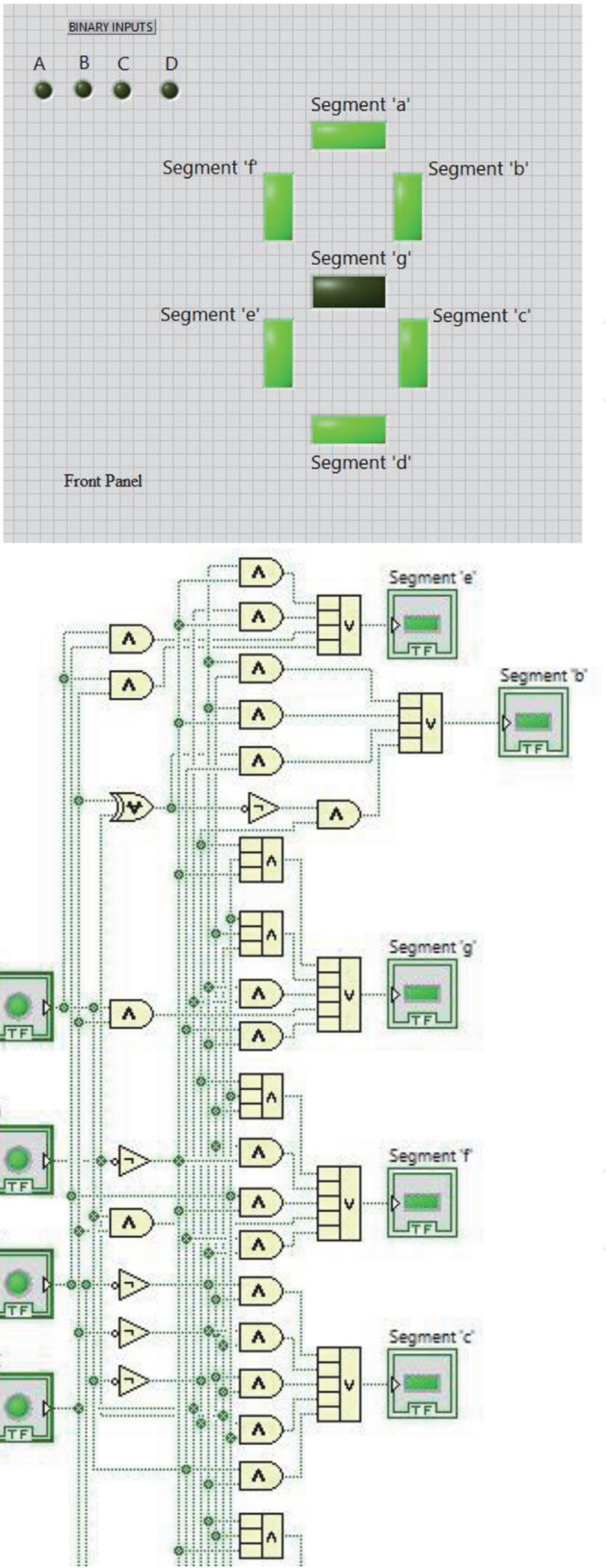

Figure 10.
Logic Diagram – Seven Segment Decoder.

3.4.3 Seven segment decoder

A digital decoder IC is a device that converts one digital format into another, and one of the most commonly-used device for doing this is the binary-coded decimal (BCD) to 7-segment display decoder. The 7-segment light emitting diode (LED) provides a convenient way of displaying information or digital data in the form of numbers, letters and alphanumeric characters. Typically, 7-segment displays consist of seven same coloured LEDs (called segments) within a single display package. In order to display the correct character or number, the correct combination of LED segments has to be illuminated. This LabVIEW program demonstrates the illumination of each segment by displaying hex values (0000 through FFFF) in decimal form from 0 through 9 and A through F. The standard 7-segment LED display has eight input connections, one for each LED segment and one that acts as a common terminal or connection for all internal display segments. Some displays also have an additional input pin for displaying a decimal point.

3.4.4 Types of digital display

There are two important types of 7-segment LED displays, namely, common cathode and common anode. In Common cathode display (CCD) display, all cathode connections of the LEDs are joined together to a low logic or ground or 0 [5]. The individual segment is illuminated by the application of high logic or + Vcc or 1 to the individual anode terminal. In Common anode display (CAD) In a CAD, all anode connections of the LEDs are joined together to a high logic or + Vcc and individual segments are illuminated by connecting individual cathode terminals to low logic or ground as shown in **Figure 10**. The boolean expressions of the outputs.

$$a = \overline{A}BD + A\overline{BC} + \overline{BD} + AC + A\overline{D} + BC \tag{26}$$

$$b = \overline{A}C \bigoplus D + A(C \bigoplus D) + \overline{BC} + \overline{BD} \tag{27}$$

$$c = \overline{BC} + \overline{B}D + \overline{C}D + \overline{A}B + A\overline{B} \tag{28}$$

$$\mathrm{d} = \overline{BCD} + \overline{B}DC + \overline{A}C\overline{D} + B\overline{C}D + A\overline{C} + AB\overline{D} \tag{29}$$

$$e = \overline{BD} + C\overline{D} + AB + AC \tag{30}$$

$$f = \overline{A}B\overline{C} + \overline{CD} + B\overline{D} + A\overline{B} + AC \tag{31}$$

$$g = \overline{A}B\overline{D} + B\overline{C}D + A\overline{B} + AC + C\overline{B} \tag{32}$$

4. Conclusion

This chapter brings an overview of design of combinational logic circuits in LabVIEW. This LabVIEW programming tool is a graphical representation tool which helps the designer to simplify the design work. This tool can be further extended for designing sequential circuits as well as PLA and PAL logic design.

Acknowledgements

The author would like to thank the Department of Electronics and Instrumentation Engineering of Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya, Enathur, Kanchipuram, India, for providing online library facility as well as for permitting to use the lab environment.

DOI: http://dx.doi.org/10.5772/intechopen.97611

Author details

Janani Rajaraman
Department of Electronics and Instrumentation Engineering,
Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya, Enathur,
Kanchipuram, Tamilnadu, India

*Address all correspondence to: janani.rajaraman@kanchiuniv.ac.in

References

[1] Holdsworth, Brian and Clive Woods, *Digital Logic Design*, Elsevier, 2002.

[2] Mano, M. Morris and Charles R. Kime, *Logic and Computer design fundamentals,* Prentice-Hall, Inc, 1997

[3] Dietmeyer, Donald Leo, *Logic Design of Digital Systems*, 1978.

[4] Friedman, Arthur D, *Logical Design of Digital Systems*, 1977.

[5] Rajaraman Janani and K. Saraswathi, *A LabVIEW based Monitoring and Controlling of Various Process Variables",* International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol.3, 2014.