Combinational Logic Design Example

maps.

COMBINATIONAL LOGIC DESIGN

3.3.1 Evaluating the Problem and Planning Its Solution

At this point, all of the important analytical tools needed for the design of combinational logic networks have been presented. These analytical tools enable us to

1. Describe logic relationships with truth tables.
2. Express logic relationships with Boolean equations.
3. Manipulate and reduce equations using Boolean algebra.
4. Minimize logic relationships using Karnaugh map techniques.
5. Relate gate structures to Boolean equations.

In order to apply these tools effectively in designing combinational logic, it is necessary to develop an understanding of the design problem and plan a general course of action that will bring us to the solution. When such careful preparation is made, we need only execute our plan by calling to use the appropriate analytical tools. The major factors involved in evaluating a problem and planning its solution are described.

(1) Understanding the problem description: The description of a combinational logic problem may be verbal or mathematical. With a verbal description, some situation is presented in which people or events on the outside world are to cause a logic device to respond in a particular way. Often, verbal descriptions do not completely account for all possible events that may occur. Rather, the important input events are clearly described, while trivial or unlikely input events are not considered. It is the designer's responsibility to realize *all* possible input conditions, whether or not they relate to likely events. Having done so, those responses that are unspecified must either be inferred from common sense, or assumed to be unimportant and assigned "don't care" status.

If a problem is described mathematically, the task of realizing the problem is much easier. This is so because of the explicit nature of a truth table or Boolean equation. With a mathematical description, we are given a truth table or equation and must ultimately find a corresponding logic network. While the solution may not be straightfoward, the problem statement is.

(2) Analyzing the problem: Once the problem is understood, we must decide what actions are called for to bring about a solution. First, it must be determined that the problem really does involve combinational logic, and not sequential logic. If the operation of the desired circuit requires any kind of memory, a solution cannot be found using just combinational logic. Only when a fixed relationship exists between inputs and output(s) is a combinational logic solution applicable. When it

has been decided that a combinational logic solution is called for, a plan of action must be found. Whether to make a truth table, or to minimize an equation with Boolean algebra or Karnaugh maps, or to use only NAND gates in the final circuit are factors that are considered. Deciding what analytical tools to apply and when to apply them comprises the plan of action.

(3) Formulating the problem: The first concrete steps taken to obtain a solution occur at this point. To begin, the number of input and output variables that are needed must be decided. If we are given an equation or truth table as the problem description, this decision is trivial. However, a verbal problem description may require some thought before the number of variables involved becomes clear.

In the case of a verbally described problem, meaning must be assigned to a variable's two values. For instance, in our door lock example given eariler, the output variable 'L' represented the condition of the door lock. The assigned meanings were $L = 0$ for an unlocked condition and $L = 1$ for a locked condition. Only when meaning is associated with the variable values can we transform a written description into a truth table.

Perhaps the most challenging step in formulating a verbally described problem is transforming the written problem statement into a truth table. The safest way to start this procedure is to write a truth table "skeleton," showing the input variables on the upper row and each possible binary combination below. For a three-variable problem, a skeleton table would appear as follows.

INPUT VARIABLES			OUTPUT VARIABLE
A	*B*	*C*	*Z*
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

By starting in this way, we are sure not to forget about any input conditions. Now, for each input condition, an output condition is decided and entered in the table. Usually, some degree of thought will be called for to unravel the verbal description. Before proceeding to minimization, all input conditions in the table should have an assigned output value of '0', '1', or '$\emptyset$'.

Occasionally, several output variables are needed. In this case, all input conditions in the truth table should have an assigned value for *each* output variable. A truth table must be fully defined before proceeding to minimization. An example of a multiple output circuit is discussed in Section 3.3.5.

Certain problems seem simple enough so that an equation can be written directly from the verbal descritption. It is advised, however, that a truth table be formed anyhow to verify that all possible input conditions are involved correctly.

(4) Minimization: When we have a truth table, the remainder of the problem's solution is almost mechanical. By following the procedures established earlier in this chapter for Karnaugh maps, or in Chapter 2 for Boolean algebra, a reduced equation can be found that represents the function described by the truth table. Then, from the equation, a logic circuit is realized.

(5) Logic gate realization: It should be straightfoward at this point to obtain a combinational logic circuit from an equation. Several points should be noted, though. First, we may not always have free access to any kind of logic gate. Thus, it may be necessary to transform a reduced equation that would be directly implemented with AND, OR, or INVERT logic into a form suitable for other kinds of gates. This usually involves using DeMorgan's law. For example, the MSP equation

$$Z = X\bar{Y} + WY$$

is easily constructed with AND, OR and INVERT gates. However, if only NAND gates were available, the following transformation would be required.

$$\begin{aligned} Z &= X\bar{Y} + WY && \text{(given)} \\ &= \overline{\overline{X\bar{Y} + WY}} && \text{(recall that } Z = \bar{\bar{Z}}\text{)} \\ &= \overline{(X\bar{Y}) \cdot (\overline{WY})} && \text{(DeMorgan's law)} \end{aligned}$$

The resulting equation is now suitable for NAND realization. Figure 3-24 compares the AND, OR, INVERT form to the NAND form.

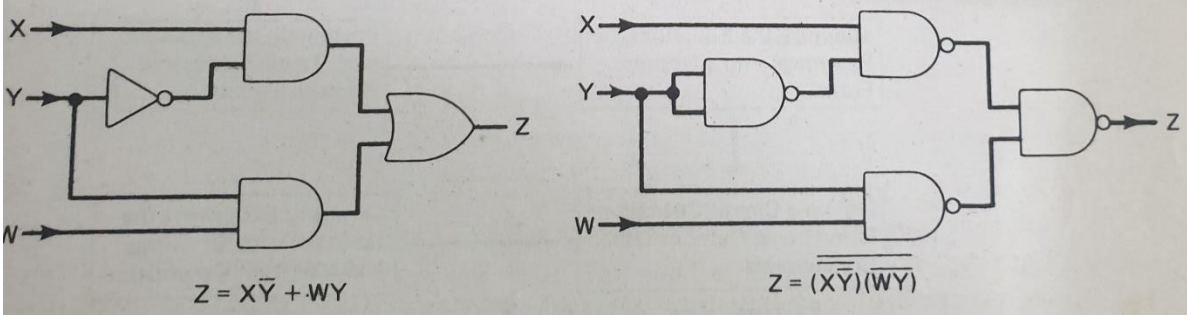
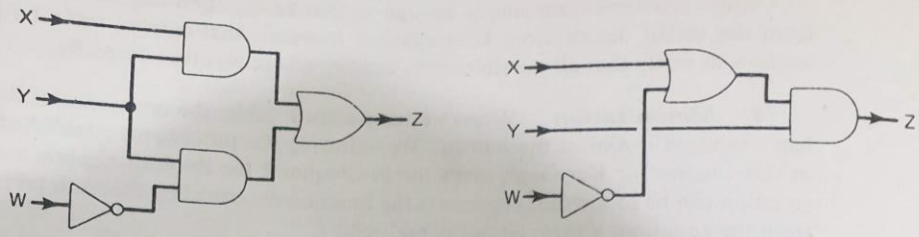


FIGURE 3-24 *Reduced equation implemented using different kinds of gates.*

A second point to be noted is that an MSP equation is not necessarily the form of the equation that should be implemented with logic gates. If possible, an MSP equation should be factored as much as possible before a gate structure is decided. For example, the MSP equation

$$Z = XY + \bar{W}Y$$



$Z = XY + \bar{W}Y$ $\qquad$ $Z = Y(X + \bar{W})$

FIGURE 3-25 *An MSP equation is factored to obtain a simpler gate structure.*

would require three two-input gates, while its factored form,

$$Z = Y(X + \bar{W})$$

requires only two two-input gates. This is illustrated in Figure 3-25.

(6) Documentation: The end results should be carefully documented in writing and schematic drawings so that your solution can be clearly understood by other people. This is a crucial step, the failure of which may render a design useless. Unless others can understand your reasoning and results, the circuitry may be misunderstood or improperly applied.

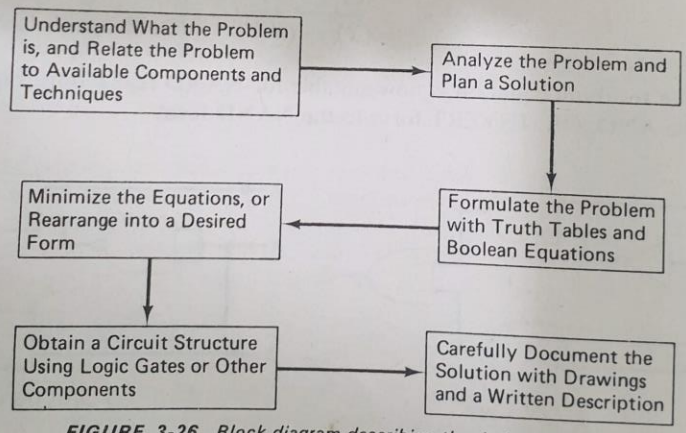


FIGURE 3-26 *Block diagram describing the design process.*

The steps that make up the design process are summarized in Figure 3-26. To demonstrate how these ideas are applied, a design example is presented.

EXAMPLE 3-5:

(1) Problem description: It is necessary to design a logic circuit that tests the operation of a traffic light. If the control circuitry for the traffic light malfunctions, it is possible that an invalid combination of signal lamps will appear. The sole purpose of

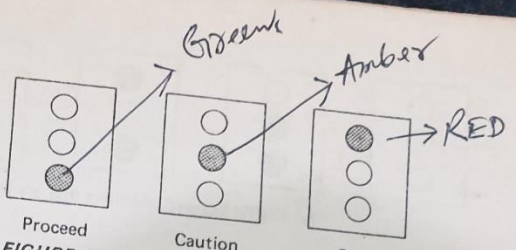


FIGURE 3-27 Valid combinations of signal lamps on a traffic light.

the test circuit is to detect any invalid combinations, and then generate an error signal that can be conveyed to the city maintenance crew. The valid combinations are shown in Figure 3-27.

(2) Analyzing the problem: First, note that memory is not required in the basic fault detection circuit. We need only detect an error condition and immediately produce an error signal. Practically, it might be desirable to have a simple latch memory involved so that a momentary error would result in a constant error signal. At this time, however, let us be concerned only with the detection of an error. For this, a combinational logic circuit is applicable. The plan of our solution will be to

(a) Deduce the possible error conditions.

(b) Write a truth table that depicts all possible signal lamp combinations, indicating in the output column whether a combination is valid or erroneous.

(c) Transfer the truth table to a Karnaugh map, decide on the essential prime implicants, and write an equation for the function.

(d) Realize the equation with a logic gate network.

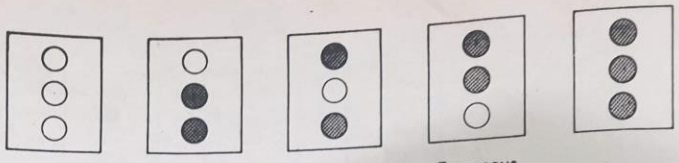
We will assume that two-input gates of any type are available. Also, we will not be concerned with the electrical coupling of the 110-V ac lamp voltage to the low-voltage DC logic gate inputs until the final logic circuit is designed.

(3) Formulating the problem: The error detection circuit will sense the on-off condition of the three signal lamps in the traffic light, and generate an error signal if an invalid combination exists. We can conclude, then, that there must be three inputs and a single output. The input variables are chosen to be 'R' (red), 'A' (amber), and 'G' (green). The single output variable is labeled 'E' since it is asserted for an error condition. The natural assignment of meaning to these variables is '0' for an "off" lamp, '1' for an "on" lamp, and at the output, '1' for an error condition.

Since there are three lamps, we know that there are eight possible combinations of "on" and "off" that may exist. The valid combination were given in the problem description, Figure 3-27. Because only three of the eight possible combinations are valid, we must conclude that the remaining five combinations are invalid and should cause an error signal to be generated. A truth table may now be compiled, as is done in Figure 3-28.

(4) Minimization: A Karnaugh map is found, based on the truth table of Figure 3-28, and the essential prime implicants denoted. This is shown in Figure 3-29. An MSP equation is now derived from the map.

$$E = \bar{R}\bar{A}\bar{G} + RG + AG + RA \qquad \text{(MSP)} \qquad \textbf{(3-18)}$$



(a) Traffic Signal Conditions that are Erroneous.

R	A	G	E
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1-	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b) A Truth Table which Defines those Input Conditions that are to Result in the Generation of an Error Signal. For the Inputs, '1' Indicates that a Lamp is On. For the Output, '1' Indicates that an Error Signal Should be Produced.

FIGURE 3-28

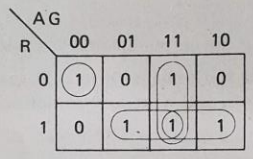


FIGURE 3-29 *The Karnaugh map related to the truth table of Figure 3-31.*

(5) Logic gate realization: Using only two-input gates, we see that three OR gates, five AND gates, and three INVERT gates (a total of eleven gates) would be required to realize the MSP equation. Assuming that two-input NOR gates are available, the MSP equation is factored and rearranged as follows:

$$E = \bar{R}A\bar{G} + RG + AG + RA \qquad \text{(MSP, given)}$$
$$= \bar{R}A\bar{G} + R(A + G) + AG$$
$$= \overline{\overline{R + A + G} + R(A + G) + AG}$$

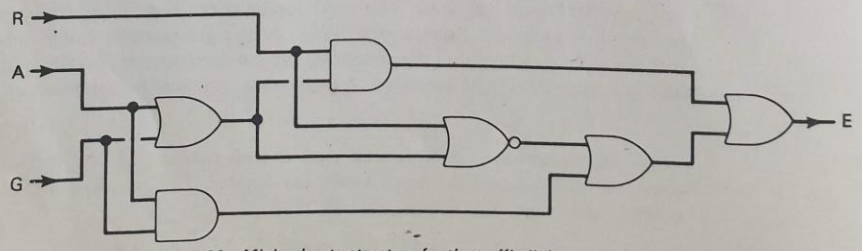


FIGURE 3-30 *Minimal gate structure for the traffic light error detection circuit.*

With this equation, a minimal gate structure, requiring only six gates, is obtained, as shown in Figure 3-30. The complete circuit, including coupling from the AC lamp power, is shown in Figure 3-31.

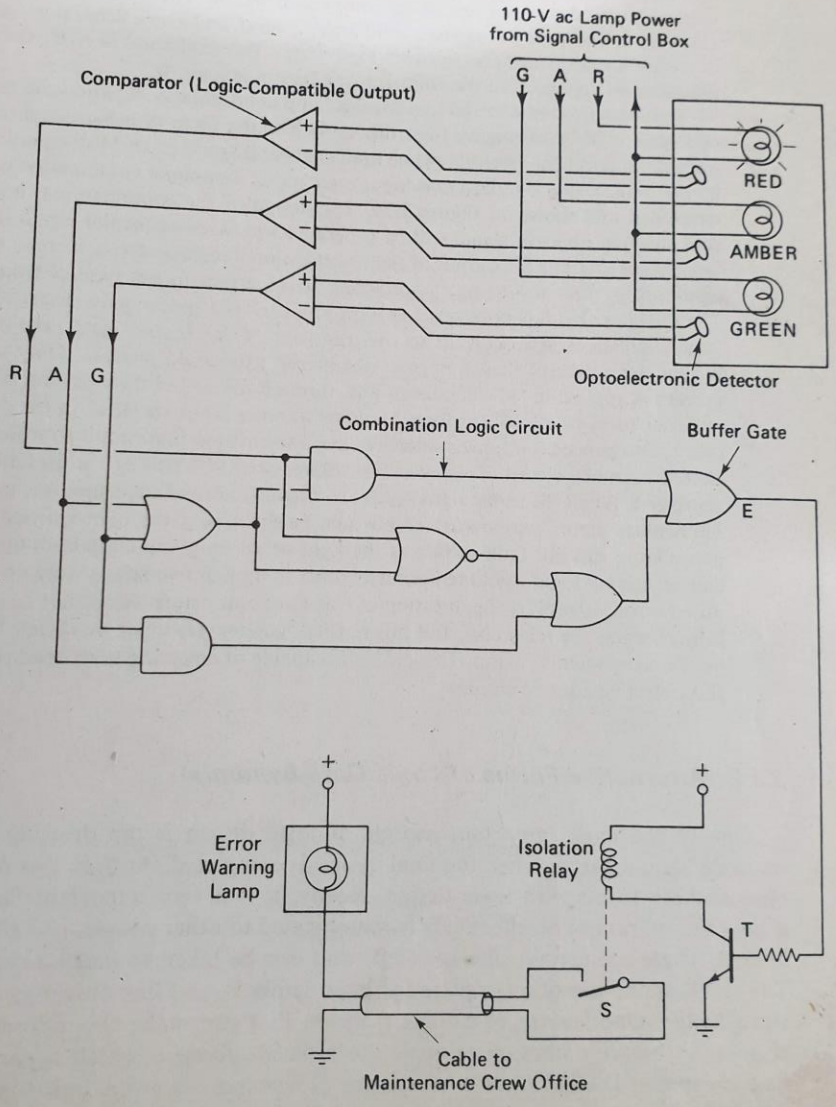


FIGURE 3-31 *Complete circuit for stoplight fault detector, shown in the valid red light state.*

(6) Documentation: A three-lamp traffic signal is represented in the upper right corner of Figure 3-31. The lamps must be supplied with 110-V AC power to glow, and will remain dark otherwise. A control box, not shown in the figure, causes power to be selectively applied to the lamps at periodic intervals so that proper operation of the traffic signal is achieved.

When a lamp is energized, a small percentage of the emitted light is captured by an optoelectronic detector, which responds to light by producing a small voltage. This voltage is also developed across the comparator inputs. The comparator contains an amplifier to boost the voltage to a valid logic '1' level, and also a Schmitt trigger circuit to ensure a sharp transition between logic states. In the absence of light, the detector produces no voltage, and the comparator's output is logic '0'.

The identification of valid and invalid lamp conditions is shown in the truth table of Figure 3-28. Rearranging the truth table into the form of a Karnaugh map, it is possible to select the essential prime implicants and arrive at an MSP equation, stated by Eq. 3-18. Using standard two-input logic gates, a minimal combinational circuit is developed and shown in Figure 3-30. The design of the combinational logic circuit provides that an error signal will be generated whenever an invalid signal lamp combination exists. The '*E*' output of the combinational logic asserts a '1' only if an error condition occurs, and being '1', provides drive current to the base of transistor '*T*'. The OR gate that has been selected to produce '*E*' is a special gate electrically able to supply sufficient drive current to the transistor. This "buffer" gate provides greater output drive current than a normal unbuffered gate could provide. Only when drive current is applied to '*T*' will current flow through the coil of the isolation relay, which in turn causes switch '*S*' to close. An error warning lamp, installed in the office of the city maintenance crew, glows whenever the transmission line circuit to which it is connected becomes closed. This condition exists when the coil of the isolation relay is energized. When the traffic signal cycles through its normal sequence, it is possible that momentary errors may occur. This would be the case if the light emitted by an off-going lamp did not fade as fast as the light of an on-going lamp built up. Thus, the combinational logic would temporarily react as though two lamps were on—an undisputed error. However, the duration of this temporary state would not be sufficient to fully energize the relay coil, and this normal, momentary error *would not* be indicated on the error warning lamp. This circuit is capable of detecting both dead signal lamps, and defective control circuitry.

3.3.2 Alternative Forms of Logic Gate Symbols

One of the most important aspects of logic design is the drawing of a clear, readable logic diagram when the final design is completed. At first, this may seem to have nothing to do with logic design. Really, it is a very important factor in that if your design cannot be effectively communicated to other people, its value is greatly limited. There are certain obvious steps that can be taken to make a drawing clear. They include the use of a template for logic symbols, and line drawings made with a straightedge. One feature of a logic diagram that can make the difference between clear and obscure communication is the *symbolic form* in which a particular logic gate is expressed. DeMorgan's law allows us to express a given logic function in one of two equivalent forms. It is the choice of the most readable form with which we are concerned.