

Supplement to

Logic and Computer
Design Fundamentals
4th Edition¹

LOGIC ANALYSIS

S elected topics not covered in the fourth edition of *Logic and Computer Design Fundamentals* are provided here for optional coverage and for self-study. This material fits well with the desired coverage in some programs but not may not fit within others due to time constraints or local preferences. This supplement, referenced in Section 3-4 on Verification, provides more detailed information on logic analysis of combinational circuits. It covers a systematic, but very detailed, analysis approach for finding truth tables and Boolean equations taken from a prior edition. Additionally, it gives a method for analyzing circuits containing NAND gates, NOR gates, or both, while avoiding most of the applications of DeMorgan's theorem in simplifying the Boolean equations. The coverage of these two topics is independent.

The first step in the analysis is to make sure that the given circuit is combinational and not sequential. The diagram of a combinational circuit has logic gates with no feedback or storage elements. A feedback path exists if, from the output of a gate, an input of the same gate can be reached via a path of interconnections or gates. Feedback paths or storage elements in a digital circuit may result in a sequential circuit and must be analyzed by using procedures outlined in Chapter 6.

Once the logic diagram is verified to be that of a combinational circuit, one can proceed to obtain the output Boolean functions or the truth table. If the function of the circuit is to be investigated, then it is necessary to interpret how the circuit operates from the derived Boolean functions or truth table. The success of such an investigation is enhanced if one has previous experience and familiarity with a wide variety of digital circuits.

Derivation of Boolean Functions

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

1. Label all gate outputs that are a function only of input variables or their complements with distinct, arbitrary variables. Determine the Boolean function for each gate output.

¹© Pearson Education 2008. All rights reserved.

2. Label the gates that are a function of input variables and previously labeled gate output variables with distinct, arbitrary variables. Find the Boolean functions for the output variables of these gates.
3. Repeat the process in step 2 until the outputs of the circuit are obtained in terms of the inputs.

• **EXAMPLE 1 Derivation of Boolean Function from a Logic Diagram**

The analysis of the combinational circuit of Figure 1 illustrates this procedure. Note that the circuit has four binary input variables A , B , C , and D and two binary output variables F_1 and F_2 . The outputs of the various gates are labeled with intermediate variables. The outputs of gates that are a function of input variables only are T_1 and T_2 . The Boolean functions for these two outputs are

$$T_1 = \overline{B}C$$

$$T_2 = \overline{A}B$$

Next, we consider the output variables of gates that depend on gate output variables with Boolean functions defined:

$$T_3 = A + T_1 = A + \overline{B}C$$

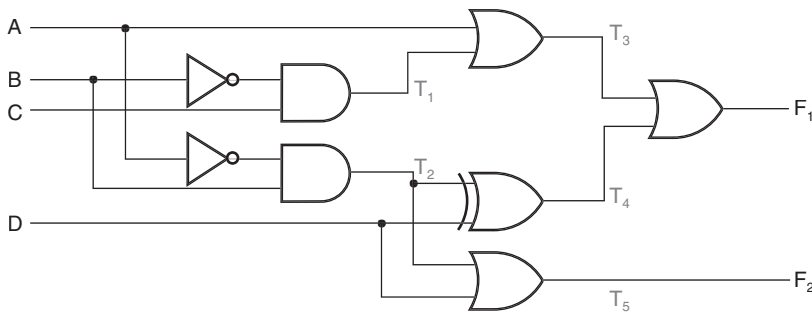
$$T_4 = T_2 \oplus D = (\overline{A}B) \oplus D = \overline{A}B\overline{D} + AD + \overline{B}D$$

$$F_2 = T_2 + D = \overline{A}B + D$$

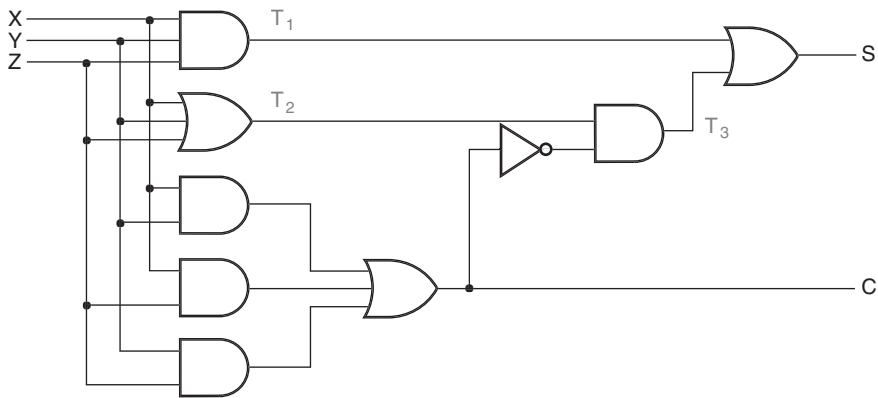
Finally, the Boolean functions for the outputs are

$$F_2 = \overline{A}B + D$$

$$\begin{aligned} F_1 &= T_3 + T_4 = A + \overline{B}C + \overline{A}B\overline{D} + AD + \overline{B}D \\ &= A + \overline{B}C + \overline{B}D + \overline{B}D \end{aligned}$$



□ **FIGURE 1**
Logic Diagram for Analysis Example



□ **FIGURE 2**
Logic Diagram for Binary Adder

The last simplification for F_1 can be performed by algebraic manipulation or by means of a map. ●

If the circuit to be analyzed is designed hierarchically, the diagrams for the lowest level block instances are analyzed first, those for block instances in the next level up are analyzed next, and so on, until Boolean equations are obtained for all outputs.

Derivation of the Truth Table

If the output Boolean functions are available in sum-of-products form, the derivation of the truth table is straightforward:

1. Determine the number of input variables in the circuit. For n inputs, list the binary numbers from 0 to $2^n - 1$ in the input part of the table. Label the top of the input columns with the input variables and the top of the output columns with the output variables.
2. For a given output function, place 1's in the truth table in the rows corresponding to each of the product terms. For example, for a function $F(A, B, C, D)$ with product term AC , enter 1's in all rows of the column for output F for which the column for input A contains 1 and the column for input C contains a 1.
3. Repeat for each product term in all of the output Boolean functions.
4. Fill in the remaining vacant truth table entries with 0's.

• EXAMPLE 2 Derivation of Truth Table from a Logic Diagram

The foregoing procedure can be illustrated by using the binary adder in Figure 2. The problem here is to verify that the circuit forms the arithmetic sum of the three bits at inputs X , Y , and Z . The output pair (C, S) ranges in value from binary 00 to 11 (decimal 3), depending on the number of 1's in the inputs. For example, when $(X, Y, Z) = 101$, (C, S) must be equal to binary 10 to indicate that there are two 1's on the inputs.

Finding the output Boolean functions,

$$C(X, Y, Z) = XY + XZ + YZ$$

$$\begin{aligned} S(X, Y, Z) &= XYZ + (X + Y + Z)\bar{C} \\ &= XYZ + (X + Y + Z)(\bar{X} + \bar{Y})(\bar{X} + \bar{Z})(\bar{Y} + \bar{Z}) \\ &= XYZ + (X + Y + Z)(\bar{X}\bar{Y} + \bar{X}\bar{Z} + \bar{Y}\bar{Z}) \\ &= XYZ + X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z \end{aligned}$$

Note that these equations were developed without using intermediate variables. The truth table for this circuit with three inputs X , Y , and Z and two outputs S and C appears as Table 1 with the eight input combinations for X , Y , and Z shown. The column for output C can be formed by entering 1's for each of the product terms of C . For example, for XY , 1's are entered in rows 110 and 111. Likewise, the column for output S can be formed by entering 1's in rows 111, 100, 010, and 001. Filling of the remaining table cells with 0's completes the truth table.

Inspection of the truth table reveals that $(C, S) = 00, 01, 10, \text{ or } 11$ when the total number of 1's on the three inputs X , Y , and Z is either zero, one, two, or three, respectively. This verifies the operation of the circuit as a binary adder. Note that the process of finding the output Boolean function for S was quite complex. Some of this complexity can be avoided by finding equations for parts of the circuit and then combining these equations in forming the truth table. For the binary adder example, the equation for C can be found as before and then the equation for S , using C as a variable, can be found as well. The resulting sum-of-product equations are:

$$C(X, Y, Z) = XY + XZ + YZ$$

$$\begin{aligned} S(X, Y, Z, C) &= XYZ + (X + Y + Z)\bar{C} \\ &= XYZ + X\bar{C} + Y\bar{C} + Z\bar{C} \end{aligned}$$

Returning to Table 1, the equation for C can be evaluated as before. Instead of evaluating S based on X , Y , and Z , S is evaluated using X , Y , Z , and C . This makes

□ **TABLE 1**
Truth Table for Binary Adder

X	Y	Z	C	$\bar{\mathbf{C}}$	T₁	T₂	T₃	S
0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0	0
1	1	1	1	0	1	1	0	1

effective use of the truth table values already developed for C and avoids the complex algebraic manipulation involved in finding the equation for S. ●

In the previous example, C was an output and so the intermediate variable for the equations used had already been defined. In general, in finding equations and truth tables, intermediate variables can be placed at any point that simplifies the analysis required. Overuse of these variables, such as placing one on the output of every gate, while systematic, complicates the analysis effort.

Next, we illustrate a procedure that permits us to minimize the number of intermediate variables and to avoid numerous applications of DeMorgan's theorems when NAND and/or NOR gates are present.

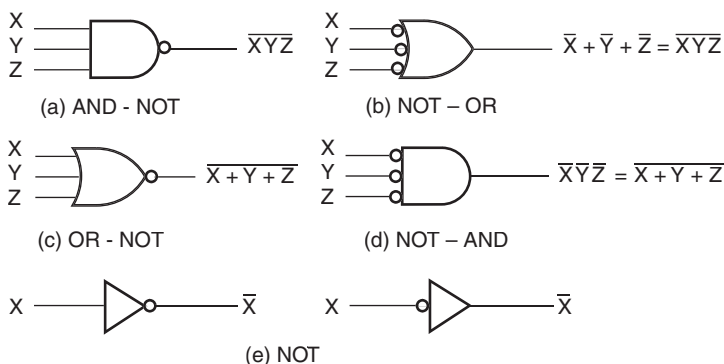
Boolean Equations for NAND and NOR Circuits

We consider analyzing circuits containing all types of gates including NAND and NOR gates. The analysis could be done in a very straightforward way by writing the NAND and NOR functions for n inputs in the equation you are developing as follows

$$\text{NAND}(x_1, x_2, \dots, x_n) = \overline{x_1 \cdot x_2 \cdot \dots \cdot x_n}$$

$$\text{NOR}(x_1, x_2, \dots, x_n) = \overline{x_1 + x_2 + \dots + x_n}$$

After doing this, a very complex equation often results that requires many tedious, successive applications of DeMorgan's theorem. To avoid this, we introduce alternate symbols for NAND, NOR and NOT gates shown in Figure 3. The alternate NOT-OR symbol for the NAND gate in Figure 3(b) is obtained by applying DeMorgan's theorem to the Boolean expression corresponding to the AND-NOT symbol for the NAND gate in Figure 3(a) and then converting the result back to symbolic form. The same procedure is applied to the OR-NOT symbol for the NOR in Figure 3(c) to obtain the NOT-AND symbol in Figure 3(d). Likewise, by treating the buffer in the NOT gate as $\text{Out} = \text{In}$ and the bubble as $\text{Out} = \overline{\text{In}}$, the bubble can be moved from the output of the buffer to the input of the buffer in Figure 3(e) without changing the overall function.



□ **FIGURE 3**

Alternative Symbols for NAND, NOR and NOT gates

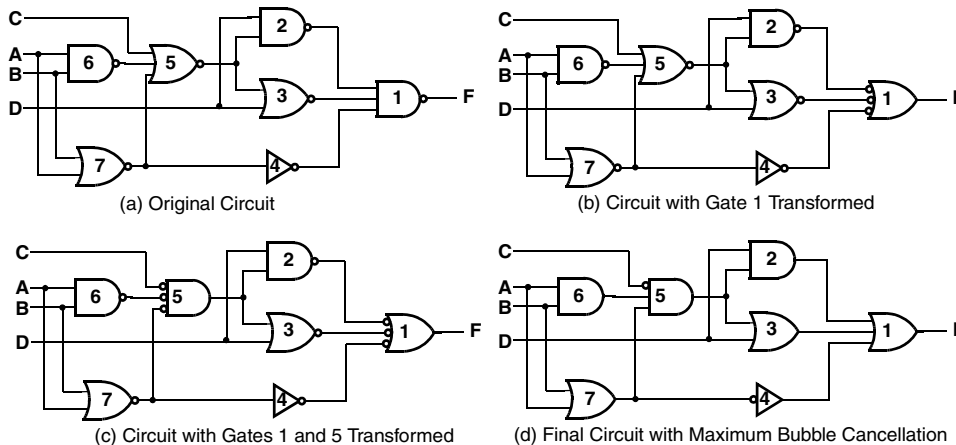
By selectively replacing the original symbols with their alternates, we can manipulate the logic diagram to eliminate as many of the NOT operations, i.e., bubbles and inverters, as possible. Once this is done, most of the tedious applications of DeMorgan's theorem can be avoided, reducing greatly the effort required to simplify the equation. Elimination of bubbles is achieved by using alternate symbols in such a way that bubbles and inverters cancel by application of involution,

$$\overline{\overline{X}} = X$$

Bubble elimination proceeds from the circuit outputs to the circuit inputs. The output gate (or gates) are level 1 and symbol replacement is used to eliminate output inversion. After all level 1 gates are processed, we proceed to processing successively higher levels using symbol replacement to match bubbles and inverters. A given gate is not processed until all gates connected to its output have been processed. After all gates have been processed, we attempt to eliminate as many bubbles as possible. If a bubble is removed from an output, then the bubbling of all inputs it drives must be changed, i.e. if an input has a bubble, the bubble is removed, and if an input does not have a bubble, a bubble is added. This is essential to keep the circuit function unchanged. This may produce some buffers without bubbles or with two bubbles that can be removed, completing the transformation. After the transformation, we treat any inverters or bubbles remaining as NOT operations in forming the equation.

• **EXAMPLE 3 Derivation of Boolean Equation from a Logic Diagram with Mixed Gate Types**

Figure 4 gives an example that illustrates the logic diagram transformation. Figure 4 (a) contains the original logic diagram. Beginning at output F , to eliminate the inversion on the output, gate 1 is replaced with its alternate symbol in Figure 4(b). Since the bubble between gate 1 and gates 2, 3, and 4 will cancel, we move on to



□ **FIGURE 4**
Gate Transformation and Bubble Cancellation in Logic Circuit Analysis

gate 5. Again, to eliminate the inversion on its output, we replace gate 5 with its alternate symbol in Figure 4(c). We now can cancel out bubble pairs between gates 2 and 1, 3 and 1, 4 and 1, and 7 and 5. Since the output of gate 7 and the input of one of the two gates it drives have bubbles, we remove the bubble from the output of 7 and from the input to 5 and add a bubble to the input of gate 4. This completes the transformations giving Figure 4(d). From the resulting circuit, we can write and simplify the equation:

$$\begin{aligned}
 F &= \overline{C}(AB)(A+B)D + \overline{C}(AB)(A+B) + D + \overline{A+B} \\
 &= ABC\overline{C} + D + \overline{A+B}
 \end{aligned}$$

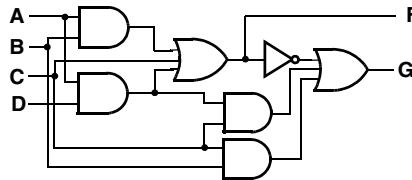
This same procedure could be applied to a circuit containing other gate types with the transformations confined to the areas with NAND and/or NOR gates. ●

REFERENCES

1. MANO, M. M. AND C. R. KIME. *Logic and Computer Design Fundamentals*, 4th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2008.

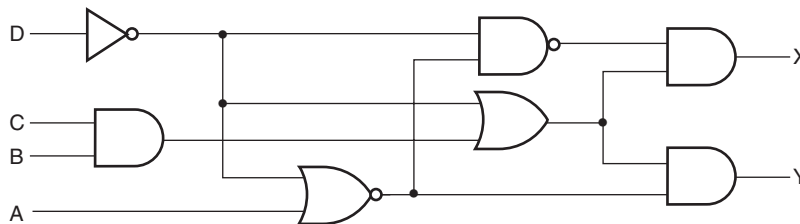
PROBLEMS

1. Determine the Boolean functions for outputs F and G as a function of the four inputs in the circuit in Figure 5. Place variables on the outputs of all gates.



□ **FIGURE 5**
Circuit for Problem 1

2. Determine the truth table for the circuit in Figure 6. Place variables on the outputs of all gates.



□ **FIGURE 6**
Circuit for Problem 2

- Repeat problem 1 using F and variable T_1 on the output of the inverter as the only intermediate variables.
- Obtain the truth table for output F for the NAND circuit in Figure 7 by using the procedure illustrated in Figure 4.

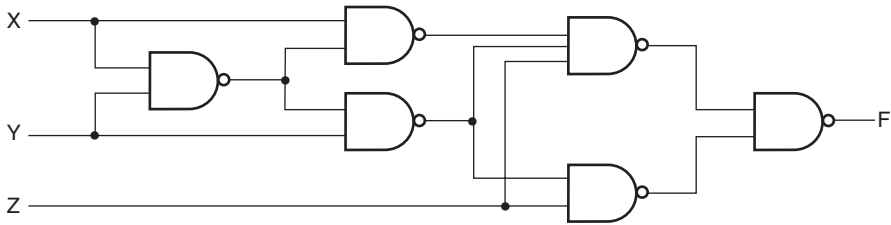


FIGURE 7
Circuit for Problem 4

- Obtain the Boolean equations for F and G for the NAND circuit in Figure 8 by using the procedure illustrated in Figure 4.

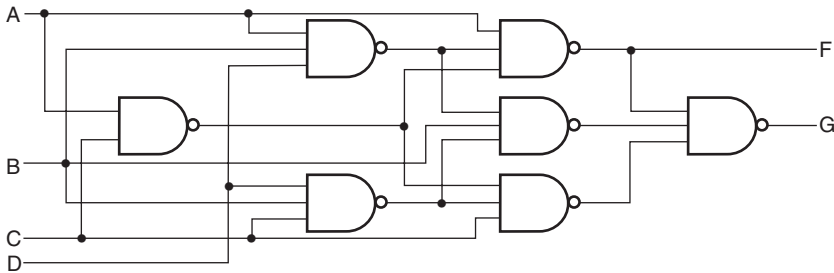


FIGURE 8
Circuit for Problem 5

- Obtain the Boolean equations for P and Q for the NAND/NOR circuit in Figure 9 by using the procedure illustrated in Figure 4.

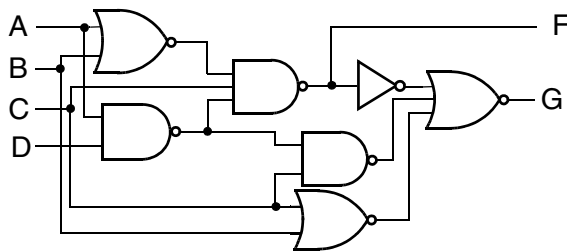


FIGURE 9
Circuit for Problem 5