

Lab # 3

Implementing Combinational Logic Circuits

A. Objective

The objective of this laboratory is to investigate various methods of implementing combinational logic circuits using basic logic gates, including the use of Sum Of Products (SOP) and Product Of Sums (POS) expressions and the use of NAND-only circuits, NOR-only circuits, and circuits using Exclusive-OR gates.

B. Materials

Breadboard	Two 7400 Quad 2-input NAND IC's
5V Power Supply	Two 7402 Quad 2-input NOR IC's
Wire, switches, LEDs, etc.	One 7486 Quad 2-input XOR IC's

C. Introduction

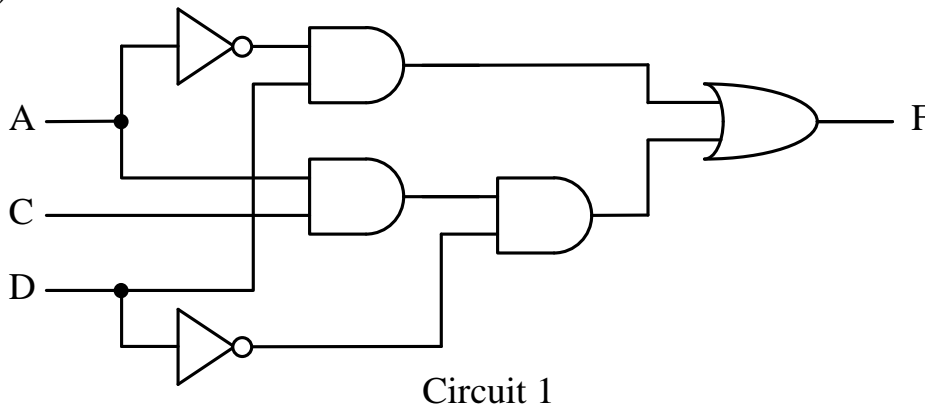
There are numerous ways to implement combinational logic circuits using simple gates. These methods should be familiar and will only be reviewed briefly.

Sum of Products Implementations

Sum of Products (SOP) expressions can be formed by grouping 1's in a Karnaugh map. An example is illustrated below for the function $f(A, B, C, D) = \Sigma(1, 3, 5, 7, 10, 14)$.

		CD			
	AB	00	01	11	10
00	0	1	1	0	0
01	0	1	1	0	0
11	0	0	0	1	0
10	0	0	0	1	0

The groupings shown yield $f(A, B, C, D) = \bar{A} \cdot D + A \cdot C \cdot \bar{D}$. This can be implemented using AND - OR - NOT gates as shown in Circuit 1 (assuming that only 2-input AND's and OR's are available).



Product Of Sums Implementations

Product Of Sums (POS) expressions can be formed by grouping 0's in a Karnaugh map to form an expression for \bar{f} . An expression for f can then be determined from \bar{f} using DeMorgan's theorem. Using the same example as before for the function $f(A, B, C, D) = \Sigma(1, 3, 5, 7, 10, 14)$:

		CD			
		00	01	11	10
AB	00	0	1	1	0
	01	0	1	1	0
	11	0	0	0	1
	10	0	0	0	1

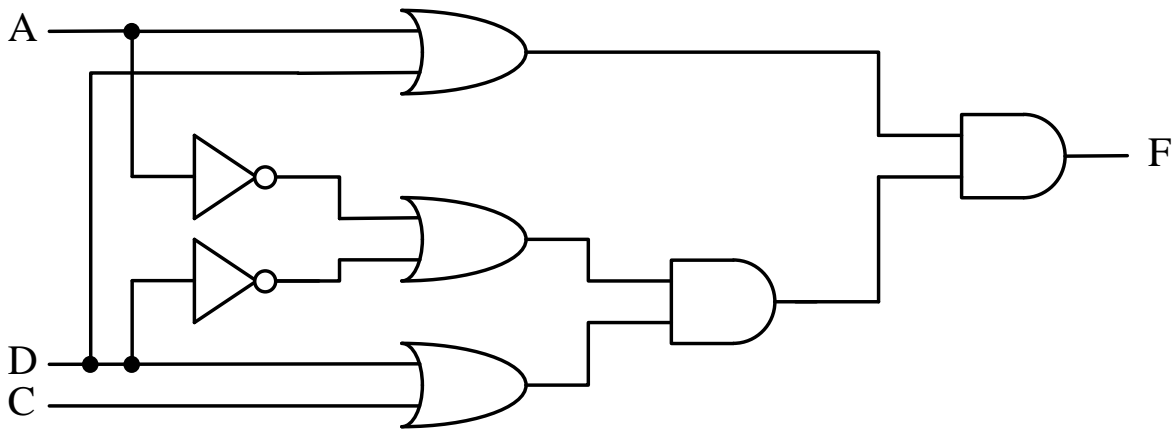
The groupings shown yield $\bar{f}(A, B, C, D) = \bar{A} \cdot \bar{D} + A \cdot D + \bar{C} \cdot \bar{D}$. Simplifying this expression with DeMorgan's theorem yields:

$$f(A, B, C, D) = \overline{\bar{A} \cdot \bar{D} + A \cdot D + \bar{C} \cdot \bar{D}}$$

$$f(A, B, C, D) = \overline{(\bar{A} \cdot \bar{D})} \cdot \overline{(A \cdot D)} \cdot \overline{(\bar{C} \cdot \bar{D})}$$

$$f(A, B, C, D) = (A + D) \cdot (\bar{A} + \bar{D}) \cdot (C + D)$$

This can be implemented using AND - OR - NOT gates as shown in Circuit 2 (assuming that only 2-input AND's and OR's are available).



Circuit 2

Implementing Circuits using only NAND gates

Recall that NAND gates are referred to as “universal” gates, meaning that any other type of logic gate can be constructed using only NAND gates. Therefore, any combinational logic circuit could be constructed using only NAND gates. Figure 3.6a below illustrates how AND and OR gates can be replaced by combinations of NAND gates.

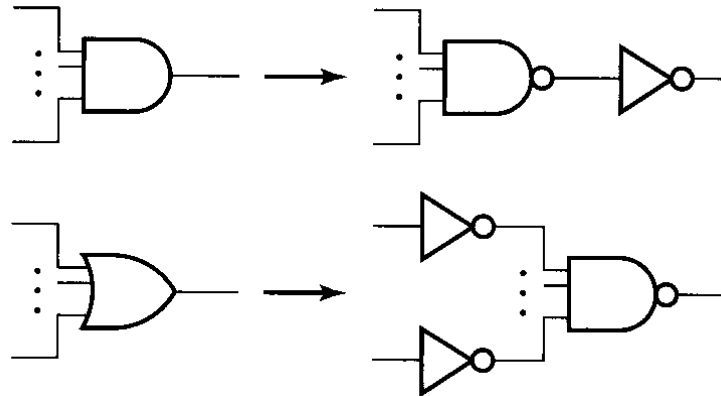
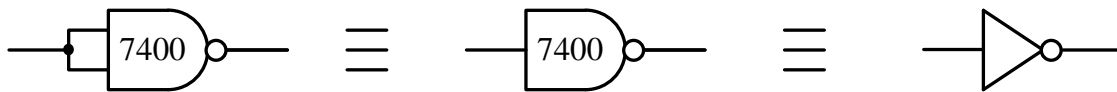


Figure 3-6a: Mapping to NAND gates (ref: Logic and Computer Design Fundamentals, 4E, by Mano)

Also recall that a NAND gate with its inputs tied together acts as an inverter. Since this applies to a NAND with any number of inputs, a NAND tied as an inverter is typically shown with only a single input, as illustrated below.



Implementing Circuits using only NOR gates

Recall that NOR gates are also “universal” gates. Therefore, any combinational logic circuit could be constructed using only NOR gates. Figure 3.6b below illustrates how AND and OR gates can be replaced by combinations of NOR gates.

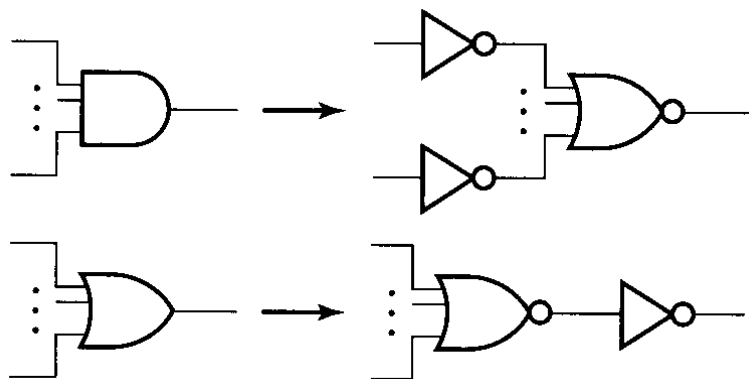
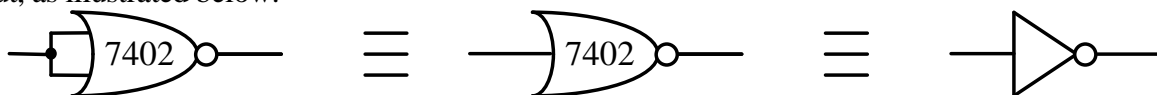


Figure 3-6b: Mapping to NOR gates (ref: Logic and Computer Design Fundamentals, 4E, by Mano)

Also recall that a NOR gate with its inputs tied together acts as an inverter. Since this applies to a NOR with any number of inputs, a NOR tied as an inverter is typically shown with only a single input, as illustrated below.



Procedure for Converting a Combinational Logic Circuit to only NAND (or only NOR) gates

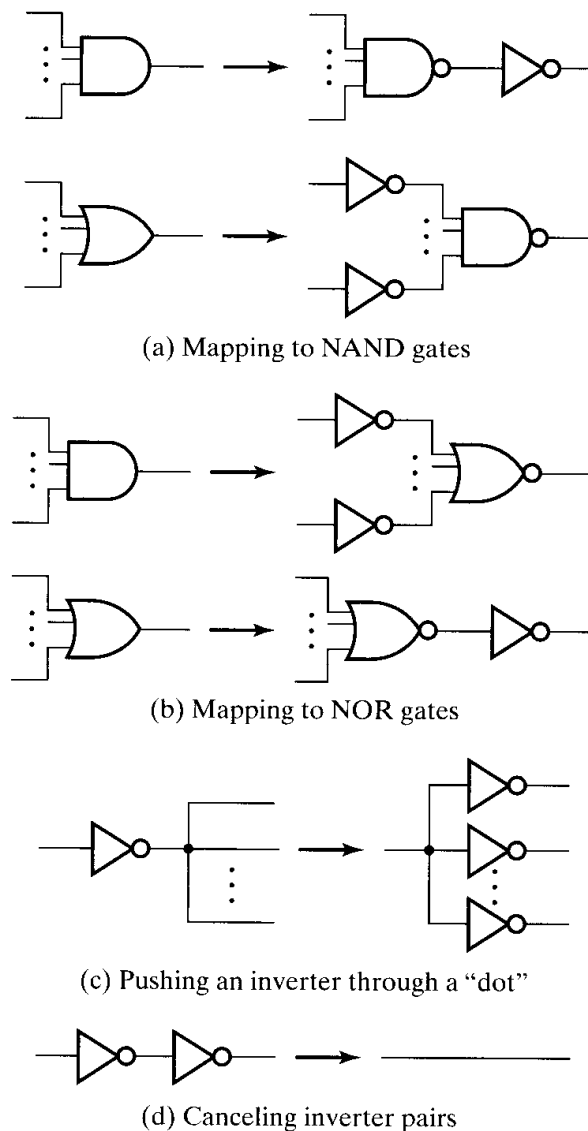
1. Draw the logic circuit using AND, OR, and NOT gates.
2. Replace each AND and OR gate with the NAND (or NOR) equivalent circuit.
3. Cancel all back-to-back NOT gates.
4. Replace any remaining NOT gates with the NAND (or NOR) equivalent.

Figure 3-6 below describes some of the operations referred to in the procedure above.

Example 3-4 illustrates the procedure using NAND gates.

Example 3-5 illustrates the procedure using NOR gates.

Reference: Logic and Computer Design Fundamentals, 4E, by Mano

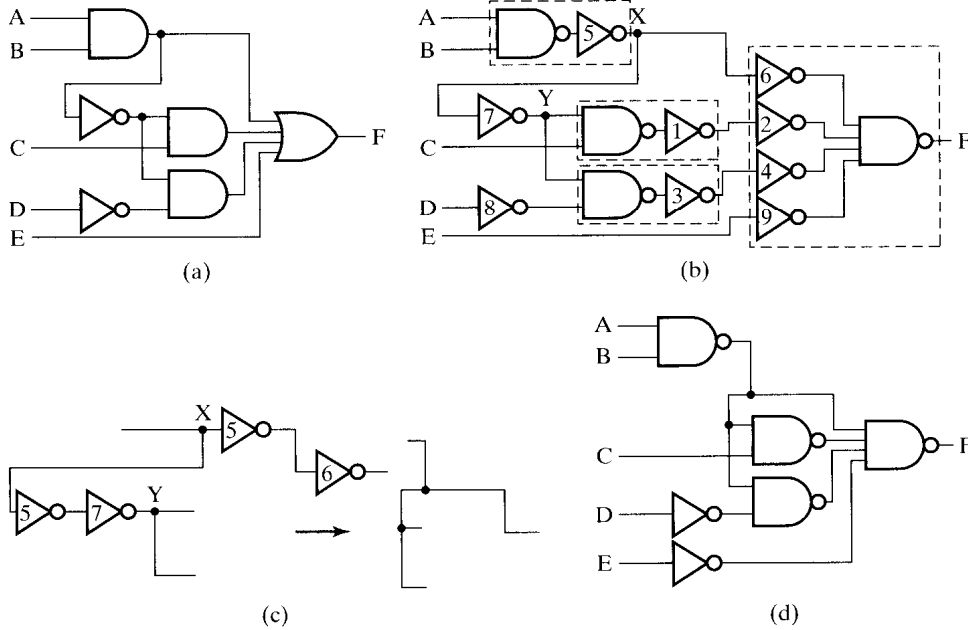


□ FIGURE 3-6
 Mapping of AND Gates, OR Gates and Inverters to
 NAND Gates, NOR Gates, and Inverters

EXAMPLE 3-4 Implementation with NAND Gates

Implement the following optimized function with NAND gates:

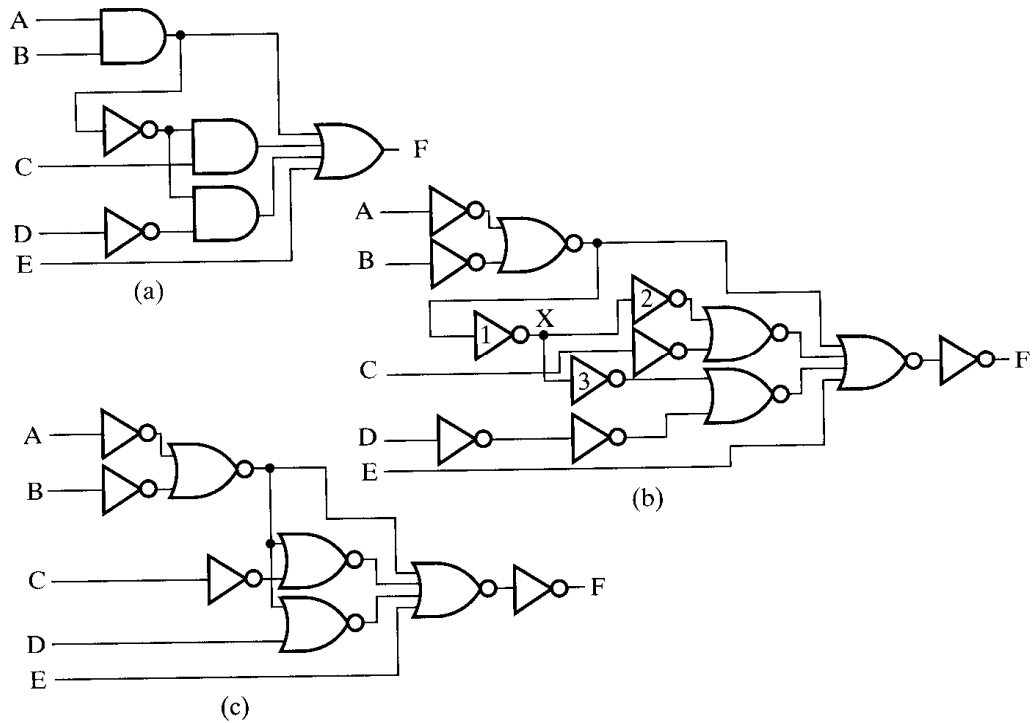
$$F = AB + (\overline{AB})C + (\overline{AB})\overline{D} + E$$



EXAMPLE 3-5 Implementation with NOR Gates

Implement the same optimized Boolean function used in Example 3-4 with NOR gates:

$$F = AB + (\overline{AB})C + (\overline{AB})\overline{D} + E$$



Exclusive-OR Implementations

SOP and POS are only two possible implementations. Additional savings in gates or IC's can sometimes be achieved by considering other implementations, including implementations which use Exclusive-OR (XOR) gates. Diagonal or staggered patterns of groupings on a Karnaugh Map often indicate that XOR gates should be considered in order to produce a minimal implementation.

The Exclusive-OR operation is defined as: $A \oplus B = \overline{A} \oplus \overline{B} = A \cdot \overline{B} + \overline{A} \cdot B$

The Equivalence (or Exclusive-NOR) operation is defined as: $A \odot B = A \cdot B + \overline{A} \cdot \overline{B}$

The TTL 7486 is a 2-input XOR gate. Since there is no TTL Equivalence gate manufactured, the Equivalence operation is generally implemented using a 7486 by inverting either input or the output according to the following relationships:

$$A \odot B = \overline{A} \oplus B = A \oplus \overline{B} = \overline{A \oplus B}$$

Using a new example with the function $f(A, B, C, D) = \Sigma(1, 3, 5, 7, 10, 14)$:

	CD			
AB	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	0	1	0	1
10	0	1	0	1

The diagonal groupings indicated that the use of XOR gates might be beneficial. First expressing f in SOP form yields:

$$f(A, B, C, D) = \overline{A} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{C} \cdot D + \overline{A} \cdot C \cdot D + A \cdot C \cdot \overline{D}$$

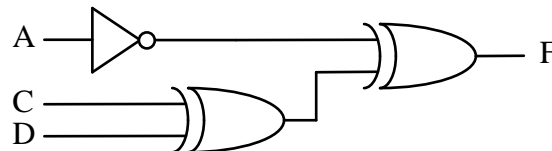
This expression can be regrouped as follows:

$$f(A, B, C, D) = \overline{A} \cdot (\overline{C} \cdot \overline{D} + C \cdot D) + A \cdot (\overline{C} \cdot D + C \cdot \overline{D})$$

$$f(A, B, C, D) = \overline{A} \cdot (\overline{C \oplus D}) + A \cdot (C \oplus D)$$

$$f(A, B, C, D) = \overline{A} \oplus C \oplus D = A \oplus \overline{C} \oplus D = A \oplus C \oplus \overline{D}$$

Now the function f can be implemented using XOR's as shown below.



Circuit 3

So the function f could be implemented using:

- Two XOR's and one inverter (total: 3 gates, 2 IC's) - shown in Circuit 5 above.
- Four 3-input AND's, one 4-input OR, and three inverters (total: 8 gates, 4 IC's)
- Eight 2-input AND's, three 2-input OR's, and three inverters (total: 14 gates, 4 IC's)
- Twenty 2-input NAND's (total: 20 gates, 5 IC's)

The implementation shown in Circuit 3 clearly produced a savings in both gates and IC's.

D. Preliminary Work

1. Determine minimal SOP and minimal POS expressions for the function $f(A, B, C, D) = \Sigma(1, 3, 4, 5, 6, 7, 14, 15)$.
2. Draw a circuit to implement the function of step 1 above:
 - a) using the SOP expression for f implemented using only 2-input AND's, 2-input OR's, and inverters
 - b) using the SOP expression for f implemented using only 2-input NAND's
 - c) using the POS expression for f implemented using only 2-input AND's, 2-input OR's, and inverters
 - d) using the POS expression for f implemented using only 2-input NOR's
3. Form a table comparing the number of gates and number of IC's required for each circuit in step 2 above.
4. Determine minimal SOP and minimal POS expressions for the function $f(A, B, C, D) = \Sigma(2, 7, 8, 13)$.
5. Draw a circuit to implement the function of step 4 above:
 - a) using the SOP expression for f implemented using only 2-input AND's, 2-input OR's, and inverters
 - b) using the POS expression for f implemented using only 2-input AND's, 2-input OR's, and inverters
6. Simplify the SOP expression generated in step 4 to make use of XOR gates such that the total number of gates required to implement the circuit is reduced. Hint: You should be able to implement the circuit using two XOR's, one 2-input AND, and one inverter (only two IC's if you use a NAND to form the AND and the inverter). Draw the final circuit.
7. Form a table comparing the number of gates and number of IC's required for each circuit in steps 5a, 5b, and 6 above.
8. Generate **full circuit documentation** for the circuits of steps 2b, 2d, and 6. These three circuits will be constructed and tested in lab. Recall that full circuit documentation consists of:
 - A) Pinouts
 - B) IC Diagram (breadboard layout with IC's identified and also numbered U1, U2, etc)
 - C) Logic Diagram (with all IC's and pin numbers identified).

E. Laboratory Work

1. Construct the circuit of step 2b in the Preliminary Work according to the logic diagram generated in step 8. Note any changes. Test the circuit for all possible input switch combinations and record the results in a truth table.
2. Construct the circuit of step 2d in the Preliminary Work according to the logic diagram generated in step 8. Note any changes. Test the circuit for all possible input switch combinations and record the results in a truth table.
3. Construct the circuit of step 6 in the Preliminary Work according to the logic diagram generated in step 8. Note any changes. Test the circuit for all possible input switch combinations and record the results in a truth table.

F. Report

In addition to the normal requirements for any lab report, discuss the following in the Discussion/Conclusion section:

- Discuss the different implementations for logic circuits introduced in this lab.
- What are the primary issues that should be considered when comparing different implementations?