CS303 Digital Design

Experiment 2

Logic Gates

Objectives:

After completing this experiment, you will be able to:

- Determine experimentally the truth tables for the NAND, NOR, OR, XOR and inverter gates.
- Use NAND and NOR gates to formulate other basic logic gates.
- Use OR and XOR gates to form a circuit that performs the l 's or 2's complement of a 4-bit binary number.

Theory:

Logic deals with only two normal conditions: logic "1" or logic "0." These conditions are like the yes or no answers to a question. Either a switch is closed (1) or it isn't (0); either an event has occurred (1) or it hasn't (0); and so on.

In Boolean logic, 1 and 0 represent conditions. In positive logic, 1 is represented by the term HIGH and 0 is represented by the term LOW. Thus, for positive logic, a voltage of +2.4 V = 1 and a voltage of +0.4 V = 0.

In addition to the AND, OR, and INVERT gates, two other basic gates are very important to logic designers. These are the NAND and NOR gates, in which the output of AND and OR, respectively, have been negated. These gates are important because of their "universal" property; they can be used to synthesize the other Boolean logic functions including AND, OR, and INVERT functions.

Two gates that are sometimes classified with the basic gates are the exclusive-OR (abbreviated XOR) and the exclusive-NOR (abbreviated XNOR) gates. These gates always have two inputs. The symbols are shown in Figure 1. The output of the XOR gate is logic "1" when either *A* or *B* is logic "1", but not both (inputs "disagree"). The XNOR is just the opposite; the output is logic "1" only when the inputs are the same (agree).

The logical operation of any gate can be summarized with a truth table, which shows all the possible inputs and outputs.

In this experiment, you will test the truth tables for NAND and NOR gates as well as those for several combinations of these gates. Keep in mind that if any two truth tables are identical, then the logic circuits that they represent are equivalent. In the Further Investigation, look for this idea of equivalence between a 4-gate circuit and a simpler 1-gate equivalent.

Procedure:

Step 1: Determine experimentally the truth tables (Table 1) for the AND, OR NAND, NOR, XOR, XNOR, Buffer and Inverter gates by implementing these gates and testing different inputs as shown in Figure 1 below.

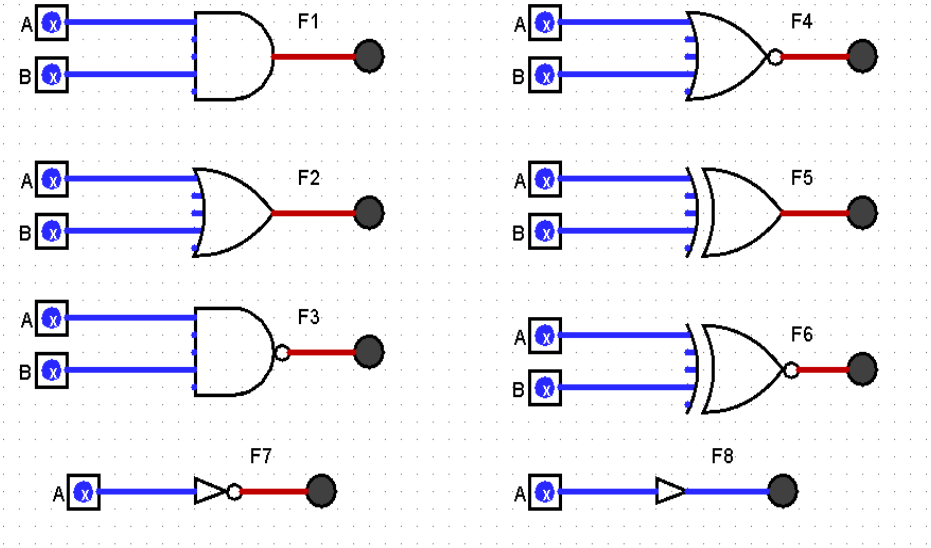


Figure 1

Table 1

A	B	F1	F2	F3	F4	F5	F6	F7	F8
0	0	0	0	1	1	0	1	-	-
0	1	0	1	1	0	1	0	1	0
1	0	0	1	1	0	1	0	0	1
1	1	1	1	0	0	0	1	-	-

Step 2: Use NAND gate only to formulate the inverter.

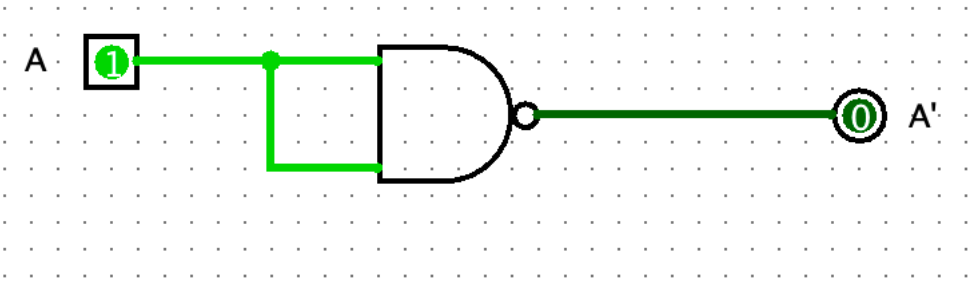


Figure 2.0 NAND gate formulated as an invertor

Step 3: Use NOR gate only to formulate the inverter.

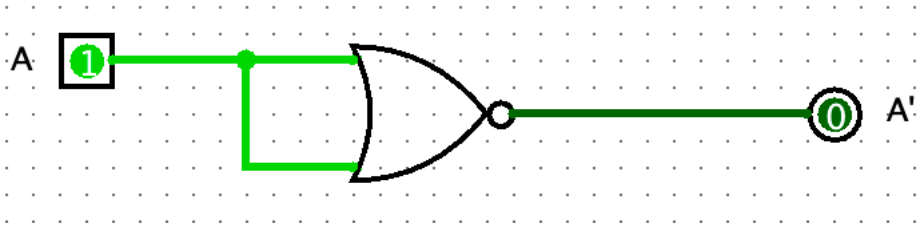


Figure 2.1 NOR gate formulated as an invertor

Step 4: Implement the circuit shown in Figure 2:

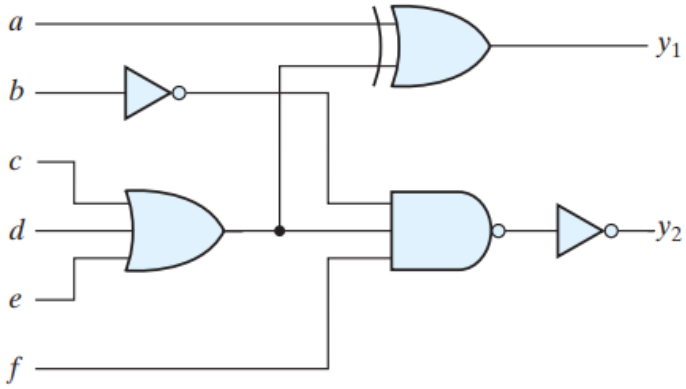


Figure 2

Step 5: Determine experimentally the values of (Table 2) based on the circuit shown in Figure 2.

Table 2

a	b	c	d	e	f	y1	y2
0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0
0	0	0	1	0	0	1	0
1	1	1	0	1	0	0	0
0	0	0	0	1	1	1	1
1	0	1	1	0	1	0	1
0	1	0	1	1	1	1	0
1	1	1	1	1	1	0	0

Step 6: Implement the circuit shown in Figure 3. Write the truth table of this circuit. Then, conclude what is the task of the circuit.

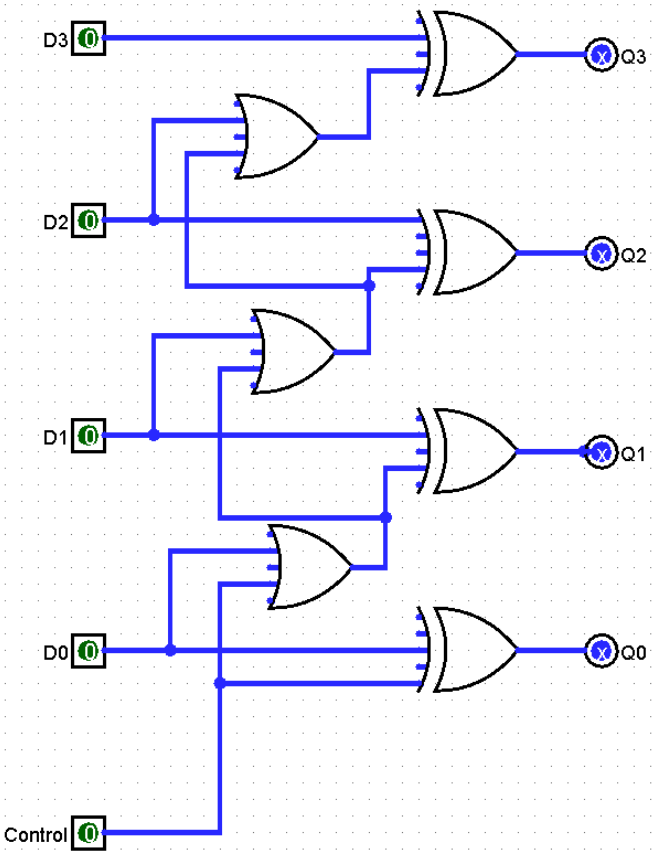


Figure 3

Report:

1. Write the Truth Table for the circuit shown in Figure 4. What do you note from the truth table?

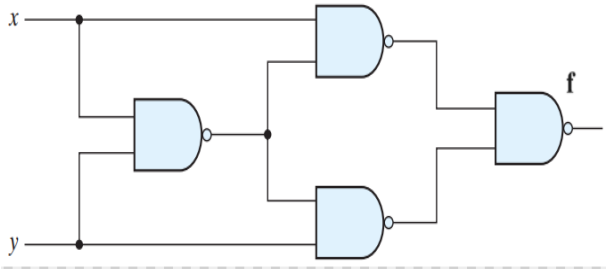


Figure 4

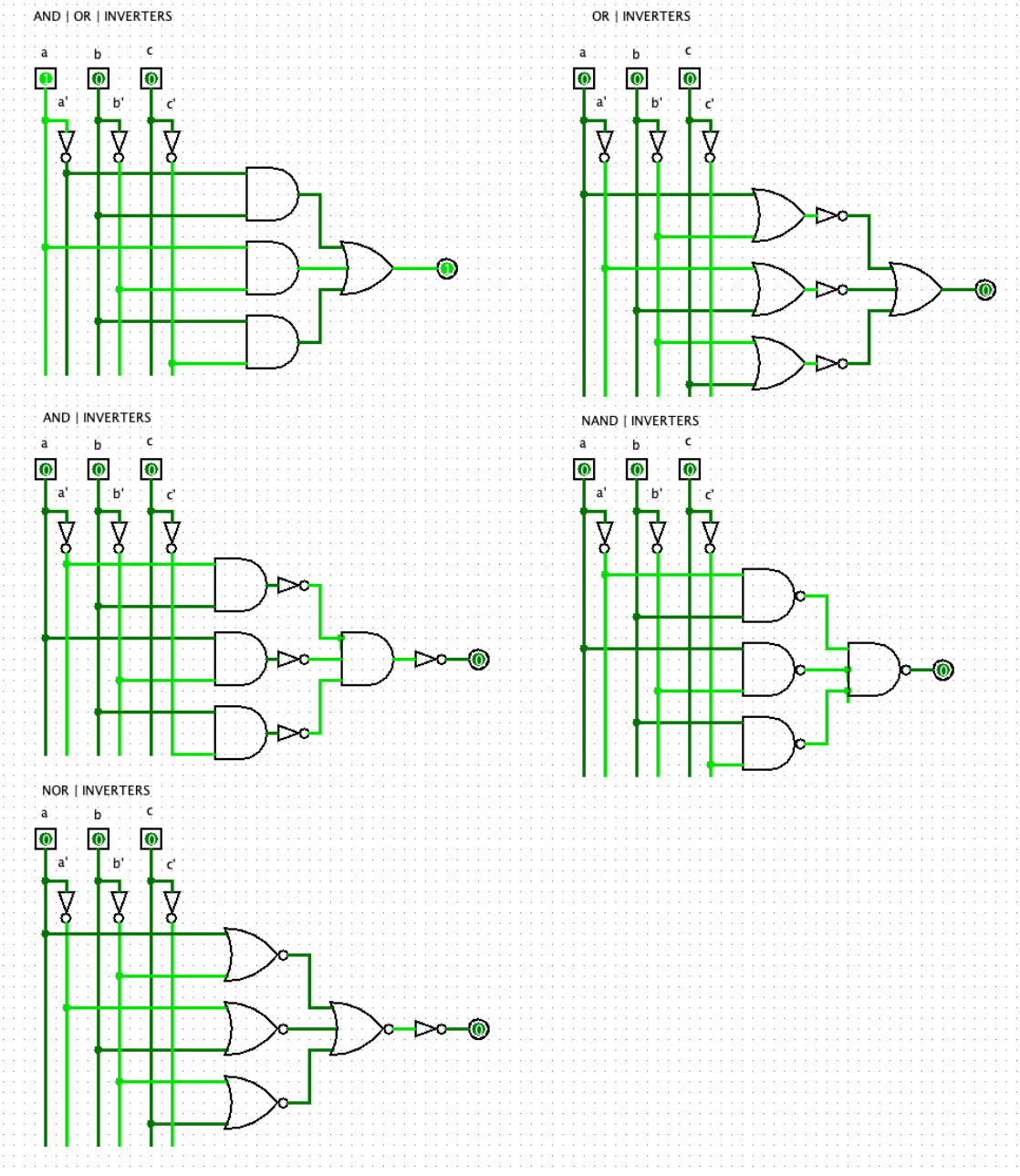
X	Y	f
0	0	0
0	1	1
1	0	1
1	1	0

From the truth table we can see that the following circuit acts like the EXOR (Exclusive Or) gate.

2. Implement the Boolean function:

$$F = a'b + ab' + bc'$$

a) With AND, OR, and inverter gates only; b) With OR and inverter gates only;
c) With AND and inverter gates only; d)With NAND and inverter gates only
e) With NOR and inverter gates only



3. Implement the two circuits shown in Figure 5. Complete Table 3. Compare the values of x and y. Can you prove (or disprove) that the circuits perform equivalent logic.

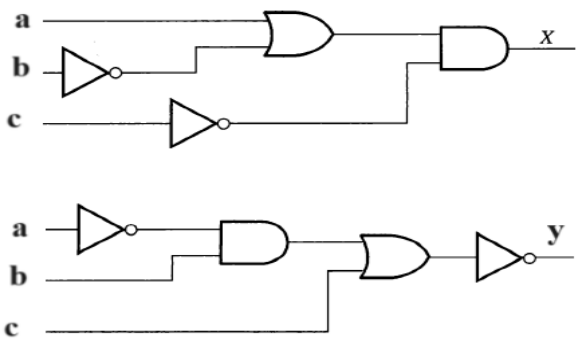


Figure 5

Table 3

a	b	c	x	y
0	0	0	1	1
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

The two circuits perform equivalent logic because their final table columns stay the same at the end. Meaning for the exact same outputs they provide the exact same input