

ECE3281 Electronics Laboratory

Experiment #4

TITLE: EXCLUSIVE-OR FUNCTIONS and BINARY ARITHMETIC

OBJECTIVE: Synthesize exclusive-OR and the basic logic functions necessary for execution of binary arithmetic.

Commentary: The exclusive-OR (XOR) gate is a modified OR gate for which output is high only when one and only one input is high. If both inputs are high, the output is low. The arithmetic boolean form is $f = A \oplus B$. This is equivalent to $f = A\bar{B} + \bar{A}B$. The logic-gate symbol for exclusive-OR and its truth table is indicated by figure 4-1.

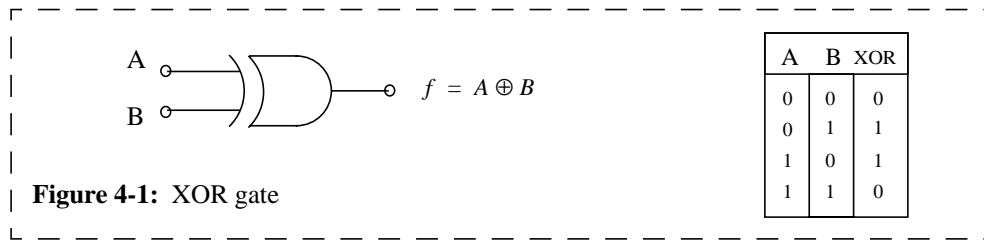


Figure 4-1: XOR gate

Among other applications, the XOR function is used for binary arithmetic. If you consider the operation of addition of two binary bits (figure 4-2), this requirement becomes self-evident

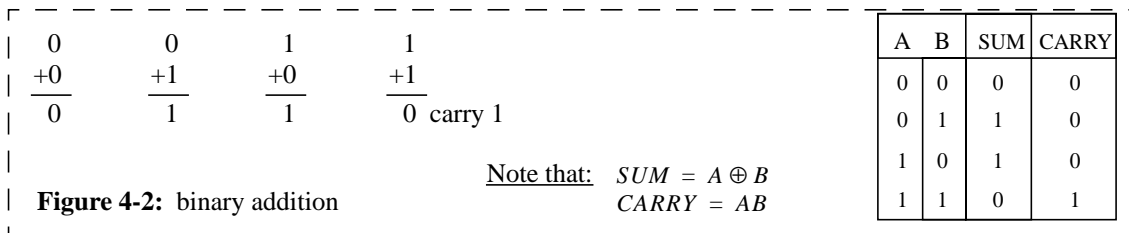


Figure 4-2: binary addition

Note that: $SUM = A \oplus B$
 $CARRY = AB$

Note that when you add two binary bits of value logic-1 each, the sum = 0, with carry = 1. This two-output function is called a *half-adder* (HA) and may be represented by the logic diagram of figure 4-3.

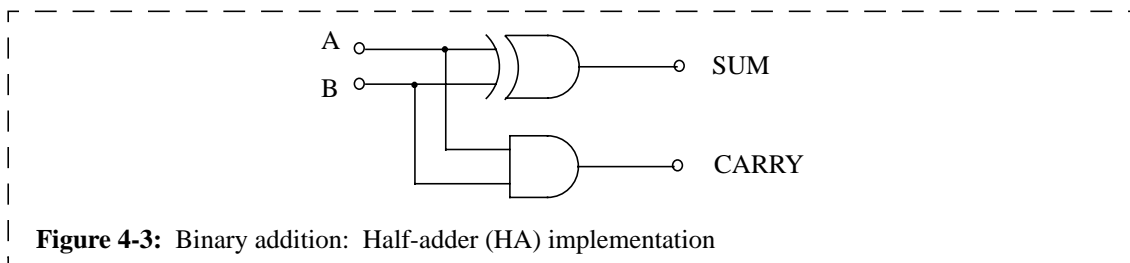
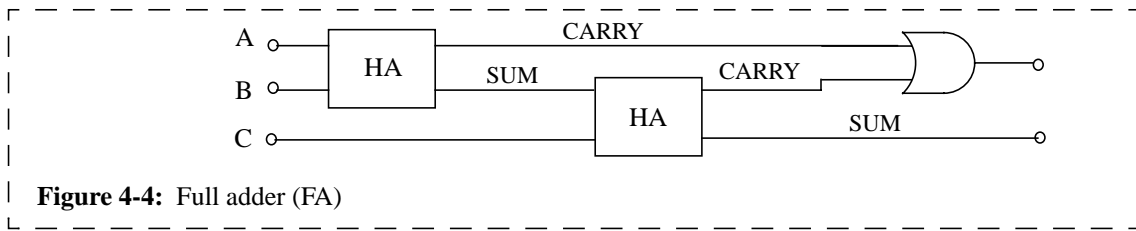


Figure 4-3: Binary addition: Half-adder (HA) implementation

As you might expect, if we cascade two half adders, we are then able to implement a full-adder (FA), as represented by figure 4-4. Two HAs = one full adder (FA). Makes sense. The FA is designed to handle addition of two binary bits and the carry from a preceding addition. A ripple of FA's is what is used to add a byte's worth of bits.



In your report you might identify the truth table that goes with the outputs of the FA and confirm that they perform the function of SUM(A, B, Carry), with outputs = (Sum, Carry).

Your task, should you choose to accept it, is to (1) synthesize XOR, and (2) synthesize a HA and (3) synthesize a FA.

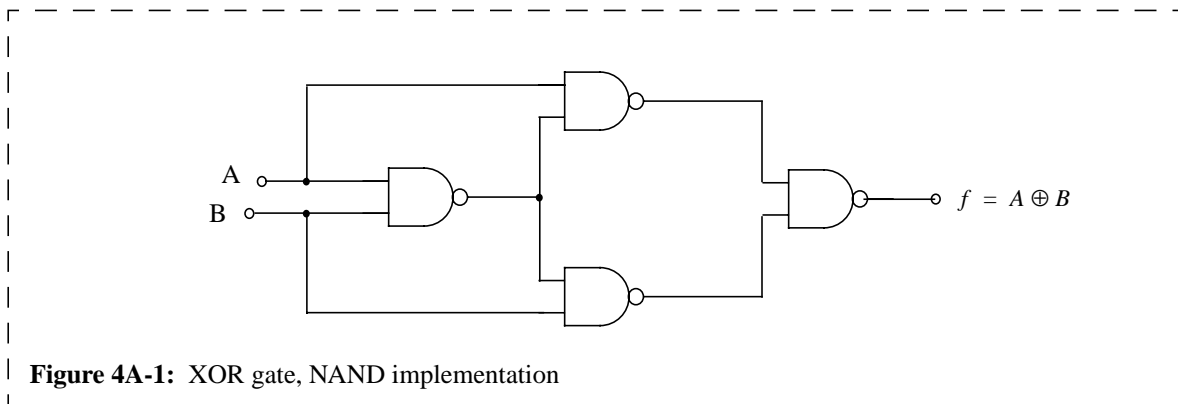
PROCEDURE:

I. XOR synthesis

A-1. Install a quad 2-input NAND (SN7400 - see experiment 3, Appendix 3A). This DIP and the other DIPs that you will need are in the same drawer as the wires, and in your parts box. If deficient, check with your instructor.

Connect the +5V and GND power supply rails to the chip. To get you started, some suggestions for placement of the DIP on the motherboard are indicated by Figure 4A-2. Have your instructor check your power connections. Do not turn on the power supply until he has made a check in your behalf.

A-2: Exercise a few brain cells and connect the circuit shown by figure 4A-1 using the SN7400 quad NAND. Use short interconnects as much as possible. Check your interconnects twice and confirm integrity before turning on the power supply.



A-3: Construct a truth table for the circuit and cycle the inputs to identify outcomes and confirm that the XOR function is implemented by this circuit.

After you have finished your measurements, turn off the power and carefully extract the SN7400 chip from the prototyping motherboard, and stow it back in the parts/wires drawer.

A-4: In your report, prove, using Boolean algebra, that the logic diagram of figure 4A-1 satisfies the XOR function $f = A\bar{B} + \bar{A}B$.

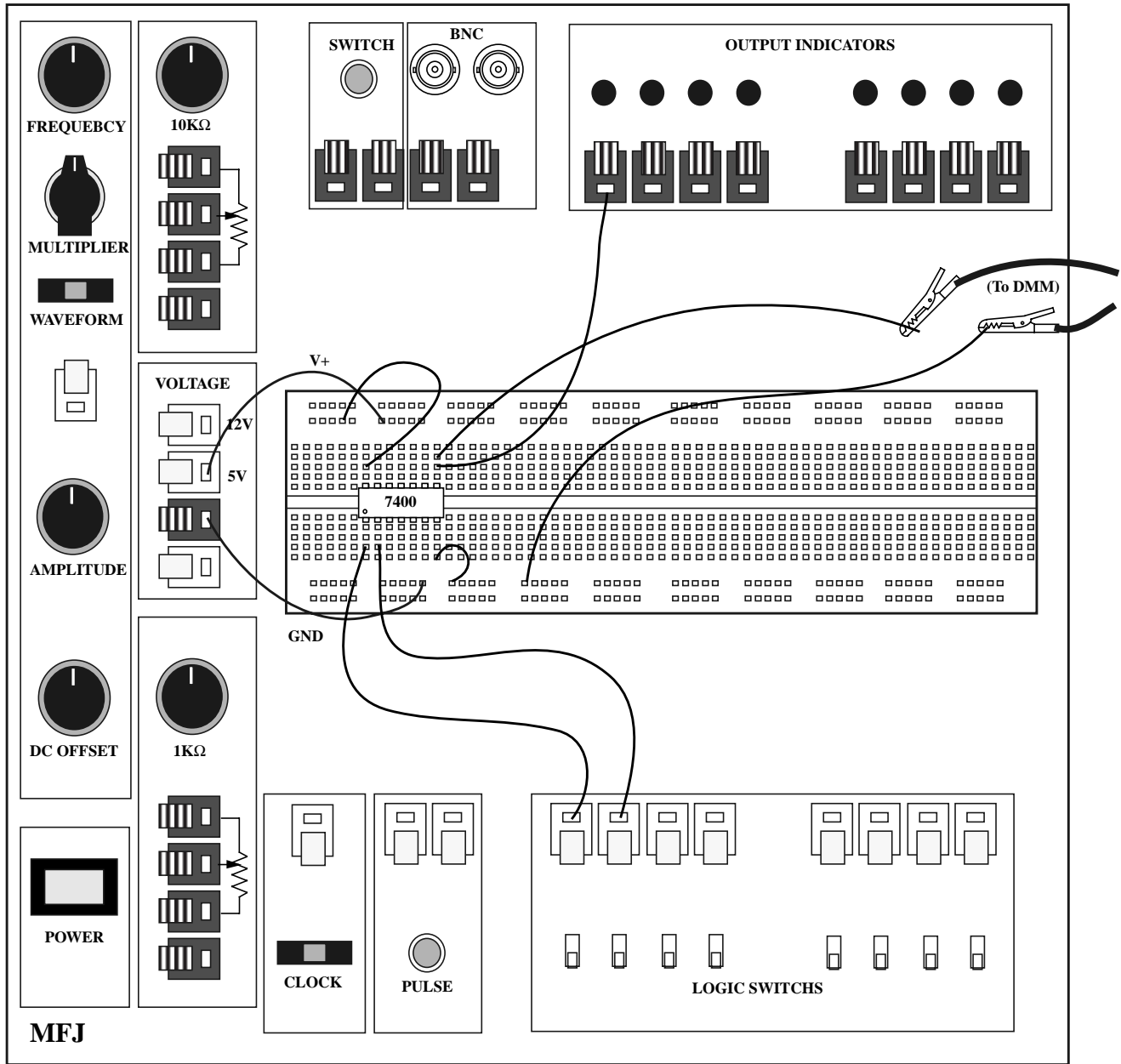
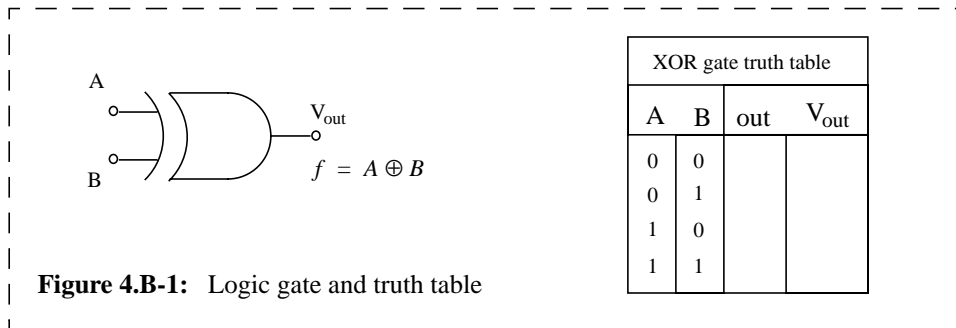


Figure 4A-2: Recommendations for set-up of power-supply connections.

B-1: Implementation of XOR function using NAND gates is fine, but the XOR function does exist as an integrated-circuit package, as represented by appendix 4A. Install a quad 2-input XOR gate (SN7486) on your motherboard for test and evaluation.

Connect up the power rails, just like before, making a check to confirm correct hook-up. This will be the last time we will lay down this step as part of the sequence. Henceforth we will assume that you realize that it must be done as an initial step before implementing a circuit, and it is the first checkpoint.

B-2: Choose any one of the four logic gates in the package and cycle through the four input logic states as indicated by the truth table and record the output states, both as binary values and as voltages.



B-3. Identify a circuit using 2-input XOR gates that will implement the logic function $f = A \oplus B \oplus C$. Wire up the circuit on the motherboard connections for this circuit and cycle through the inputs to confirm that your circuit does implement this function.

C-1: Install a quad 2-input AND gate, a quad 2-input XOR gate, and a quad 2-input OR gate on the motherboard. Some suggestions for placement of the DIPs on the motherboard are indicated by Figure 4C-1.

C-2: Formulate appropriate connections with these ICs to implement a HA function (figure 4-3). Cycle the data switches through the input options and record outputs levels. Fill these in as a truth table to confirm that they perform the HA function.

C-3. With the connections of C-2 still intact, construct a second HA implementation, and test its functionality (cycle through its inputs) and check outputs.

C-4. Connect the two HA's to form a FA, as represented by figure 4.4. Cycle through its (three) inputs to test functionality and construct a truth table that confirms this function. All of this information is part of your report.

After you are through with these tests and measurements, carefully extract parts and wires from the prototyping board and stow them back in the parts/wires drawer.

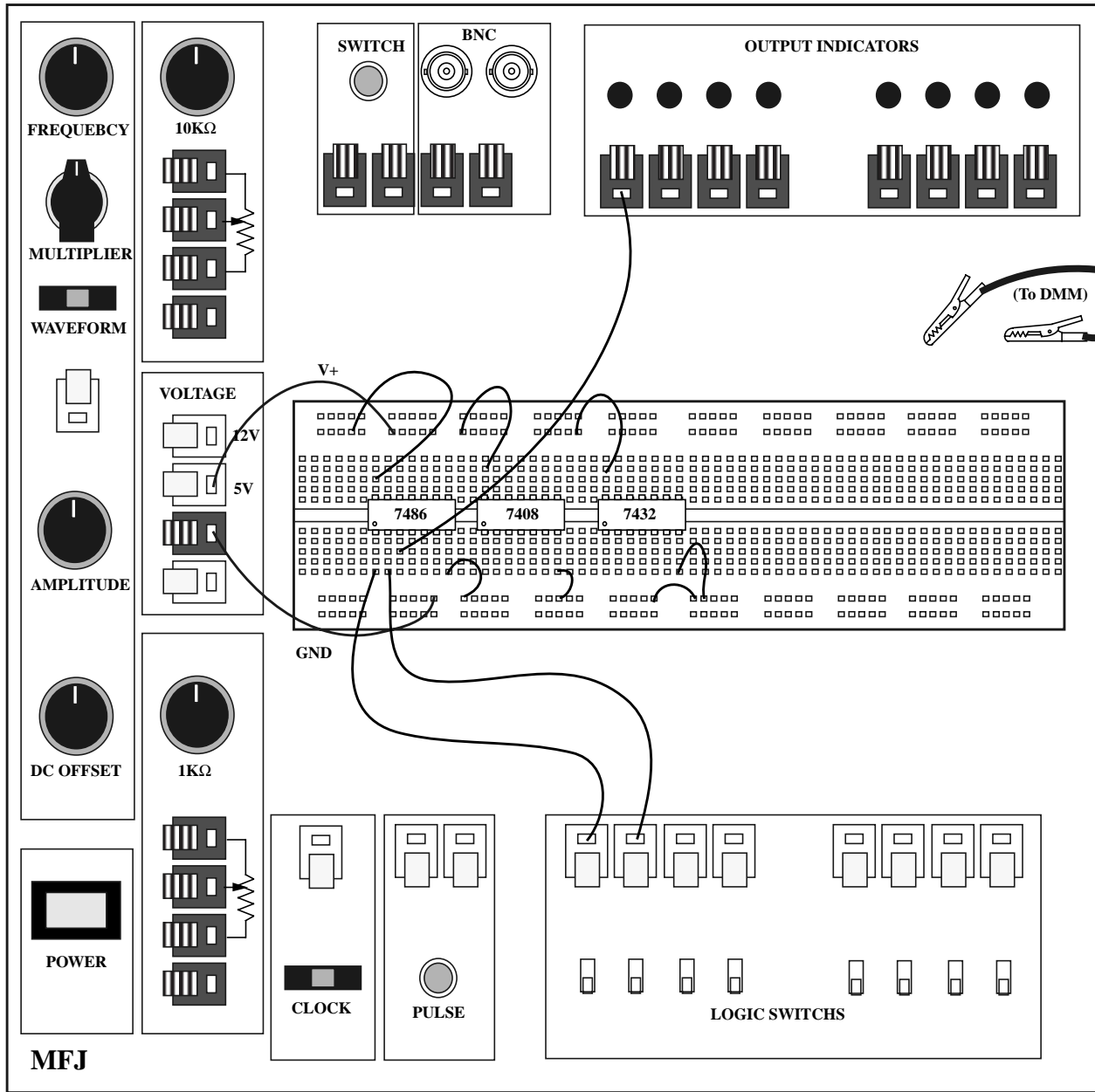
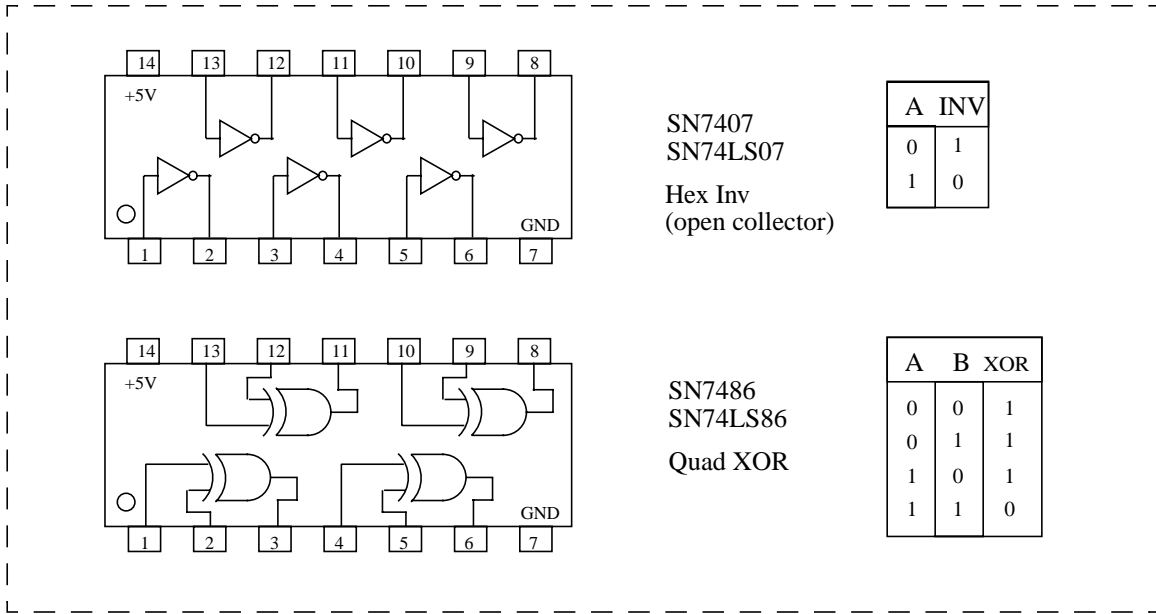


Figure 4C-1: Recommendations for set-up of initial connections for HA/FA tests..

REPORT: This is a formal report. Write it up and return as a report next class period. Be sure to include all of your truth tables and results arranged in neat form, along with logic analysis.

APPENDIX 4A: Pinout diagrams for more logic gates:



SN7407
SN74LS07
Hex Inv
(open collector)

A	INV
0	1
1	0

SN7486
SN74LS86
Quad XOR

A	B	XOR
0	0	1
0	1	1
1	0	1
1	1	0