# ELEC 2210 - EXPERIMENT 1
# Basic Digital Logic Circuits

## Objectives

The experiments in this laboratory exercise will provide an introduction to digital electronic circuits. You will learn how to use the IDL-800 "Bit Bucket" breadboarding system to build circuits using common logic gates. The objectives of this experiment include:

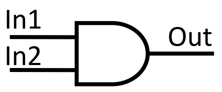
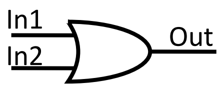
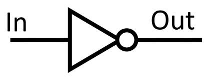
- Review basic principles of digital logic from ELEC 2200
- Learn how to use the Bit Bucket breadboarding system
- Develop professional lab skills and written communication skills.

## Introduction

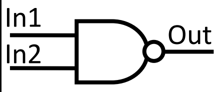
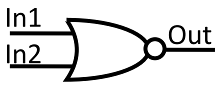
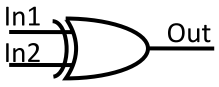
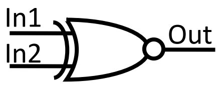
Almost all computers today use binary digital logic circuits. These circuits have just two possible output voltages, which can be called by any contrasting terms; the most common are "HIGH/LOW", or "TRUE/FALSE", or "ONE/ZERO." Such an output is called a "binary digit," or *bit*. The decimal numbers and alphabet characters we are familiar with are converted to binary bits before they are fed into a computer's arithmetic logic unit (ALU). Inside the ALU, the computer executes a program and generates binary results. The binary results are converted back to decimal numbers, alphabet letters, graphics, or sound so we can understand them. Most of the fundamental data processing inside computers is done using *logic gates*. Logic gates combine individual bits according to certain rules. These rules, taken together, form the basis of *Boolean algebra*, which you studied in depth in ELEC 2200 Digital Logic Circuits.

### Logic Gates

We will introduce the most common logic gates in this section, including the AND, OR, XOR, NOT, NOR, and NAND. For each gate, we will show the circuit symbol, the Boolean algebra logic function, and the truth table. The truth table lists all possible combinations of inputs, and the resulting output for each. The three most common gates are the AND, OR and NOT (inverter) gates from which any digital logic circuit can be constructed. These gates are summarized below in terms of their logic symbol, logic equation, and truth table, where it should be noted that the NOT gate has only one input while the AND and OR gates can have two or more inputs.

| **Logic Gate** | **AND** | **OR** | **NOT (inverter)** |
|---|---|---|---|
| Symbol |  |  |  |
| Logic Equation | $Out = In1 \bullet In2$ | $Out = In1 + In2$ | $Out = \overline{In}$ |
| Truth Table | In1 In2 \| Out<br>0 0 \| 0<br>0 1 \| 0<br>1 0 \| 0<br>1 1 \| 1 | In1 In2 \| Out<br>0 0 \| 0<br>0 1 \| 1<br>1 0 \| 1<br>1 1 \| 1 | In \| Out<br>0 \| 1<br>1 \| 0 |

Other common gates include the NAND, NOR, exclusive-OR (XOR), and exclusive-NOR (XNOR) gates. While the NAND and NOR gates are functionally complete (meaning that any digital logic circuit can be constructed from either one of these gates), the XOR and XNOR gates are not functionally complete and, therefore, are not considered to an elementary logic gate by most designers. These gates are summarized below in terms of their logic symbol, logic equation, and truth table where it should be noted that the NAND and NOR gates (like their AND and OR counterparts) can have two or more inputs, while the XOR and XNOR gates generally have only two inputs. The NAND and NOR gates are a combination of an AND gate and NOT gate and a combination of an OR gate and NOT gate, respectively, as can be observed by comparing their truth tables. Similarly, the exclusive-NOR (XNOR) gate has the inverse output of the XOR gate truth table.

| **Logic Gate** | **NAND** | | **NOR** | | **XOR** | | **XNOR** | |
|---|---|---|---|---|---|---|---|---|
| Symbol |  | |  | |  | |  | |
| Logic Equation | Out = In1•In2 | | Out = In1+In2 | | Out = In1⊕In2 | | $\text{Out} = (\overline{\text{In1} \oplus \text{In2}})$ | |
| | In1 In2 | Out | In1 In2 | Out | In1 In2 | Out | In1 In2 | Out |
| Truth Table | 0 0 | 1 | 0 0 | 1 | 0 0 | 0 | 0 0 | 1 |
| | 0 1 | 1 | 0 1 | 0 | 0 1 | 1 | 0 1 | 0 |
| | 1 0 | 1 | 1 0 | 0 | 1 0 | 1 | 1 0 | 0 |
| | 1 1 | 0 | 1 1 | 0 | 1 1 | 0 | 1 1 | 1 |

DeMorgan's Theorems also relate the operation of the NAND and NOR gates to AND and OR gates as follows:

$$(\overline{A \cdot B}) = \overline{A} + \overline{B}$$
$$(\overline{A + B}) = \overline{A} \cdot \overline{B}$$

DeMorgan's Theorems in conjunction with the Involution Theorem, which states that $(\overline{\overline{A}}) = A$, can be used to convert any 2-level AND-OR implementation of a sum-of-products (SOP) expression to an all-NAND gate implementation of the circuit. Similarly, any 2-level implementation of a product-of-sums (POS) expression can be converted to an all-NOR gate implementation of the circuit.

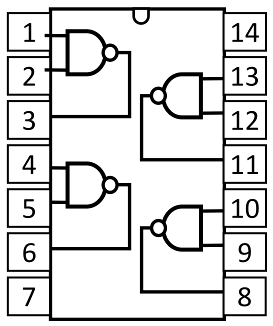
## Logic Families

A logic family is a complete set of logic gates that are manufactured using a particular type of electronic circuitry. There are numerous commercially available logic families to suit different design requirements. The most common logic families are listed in the table below, together with their relative advantages and disadvantages.

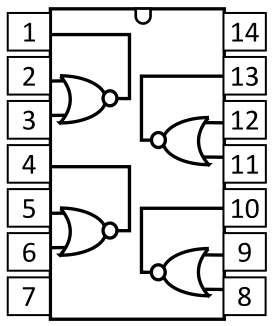
| **Acronym** | **Full name** | **Advantages** | **Disadvantages** |
|---|---|---|---|
| CMOS | Complementary metal-oxide semiconductor | Lowest power consumption. Most common logic family- used in all microcomputer chips today. | Easily damaged by static discharge and voltage spikes. |

| | | | |
|---|---|---|---|
| TTL | Transistor-transistor logic | Earliest developed. Most rugged – least susceptible to electrical damage. | Consumes more power than CMOS – not suitable for battery operated devices. |
| ECL | Emitter-coupled logic | Fastest available logic family | Consumes more power than CMOS. Requires extreme care in wiring. |

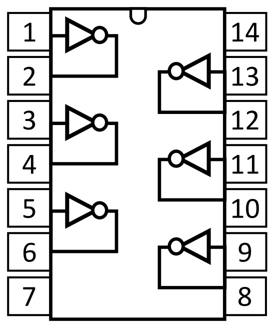
The standard part number for the TTL NAND gate is 7400. However, most manufacturers have their own designation which includes these numbers, but adds some extra characters. The following are examples of valid part numbers you might find on a "7400" chip: SN7400N, MM74C00N, SN74LS00N, SN74H00N, etc. In addition to these, there will often be another part code stamped on the chip by the manufacturer, and there might be a code stamped underneath the chip as well. In general, as long as you can find the digits 7400 somewhere on the chip, you have the right one. Below is a top view of a 7400, which is a quad NAND gate integrated circuit (IC). The term quad means there are four separate NAND gates on a single chip (they all share the same power supply, but their inputs and outputs are independent). Note the pin numbering. The chip has an indentation at one end to distinguish the ends (shown in the middle of the top of the package in the figure below). For this chip to operate, you must connect 5V to pin 14 (VCC), and ground to pin 7 (GND). Some standard TTL ICs for basic digital logic functions are illustrated below at the package level for dual-inline packages (DIPSs). The pin numbers are shown along with the internal connections to the gates. **Note that in all three cases Pin14 = Vcc and Pin7 = Gnd for all three of these devices.**



7400 Quad 2-input NAND gates



7402 Quad 2-input NOR gates



7406 Hex Inverters

**TTL Logic**

In this experiment, we will be using TTL logic. All ICs in the TTL logic family have the following specifications:

POWER SUPPLIES: +5 V and Ground (0 V).
LOGIC HIGH: 2.0 to 5 V
LOGIC LOW: 0 to 0.8 V
FAN-OUT (Number of inputs that each output can be connected to): 10

More details can be found in most textbooks on digital electronics, and also on the internet. Here is a useful website for TTL information: http://en.wikipedia.org/wiki/Transistor-transistor_logic

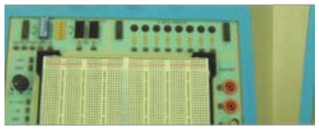
## Pre-Lab

Design an all NAND gate implementation of a 2-to-1 multiplexer which has inputs A, B, and S and output Z, where Z=A when S=0 and Z=B when S=1. Construct the truth table for the multiplexer and use a Kmap to obtain a minimized SOP expression for the multiplexer. Draw the logic diagram for the 2-level AND-OR implementation for the minimized SOP expression, labeling all inputs and output. Use the Involution Theorem and DeMorgan's Theorems to obtain an all-NAND gate implementation of the circuit and draw the logic diagram labeling all inputs and output. Using the package-level drawing of a 7400 Quad NAND gate integrated circuit, draw the connections of the various individual gates in the package to construct the multiplexer.

## Lab Exercise

Your lab instructor will show you how to insert DIP IC chips into the breadboard, and also how to remove (extract) chips from the breadboard. Each time you obtain a chip, inspect it to make sure it is the right part number (see the next paragraph), and to make sure all the pins are intact. Very often, people put chips back in the wrong bin. It is also common for pins to break off due to careless handling. If a pin breaks, inform your instructor and discard the chip.

**IDL-800 "Bit-Bucket" Digital Logic Breadboard System**
In this experiment, we will use the IDL-800, a breadboard system especially designed for TTL digital logic. In the lab ad in subsequent writeups, we commonly refer to this as the Bit Bucket. It provides regulated 5 V power, switches for manual input, and an adjustable clock. It also provides LED's and 7-segment displays for monitoring outputs.



**Inputs and Outputs**
For studying digital logic circuits we will use several types of inputs and outputs, including switches, clock signals, LED's, and 7-segment displays.
***Switches***: A switch is used to manually connect the input of a gate to a HIGH or LOW voltage. We will use pushbutton (PB) and toggle (T) switches.
***Clock***: A Clock is a circuit that produces a periodic output that alternates between HIGH and LOW voltage. We will use a clock signal to test some of our logic circuits.
***LED's and 7-segment displays***: Light-Emitting Diodes (LED's) can be used to display the output of a logic gate. Usually, they are connected so that if the output is HIGH, the LED is on. Seven-segment displays are packaged arrays of LED's that are used to display numbers and letters.

### PART 1 – NAND Gate Truth Table

Connect a 7400 quad NAND gate and verify the truth table (record the truth table for your lab report). Connect two toggle switches to the inputs of one of the NAND gates, and connect the output to the Logic Probe. Verify the truth table for this gate by stepping through each of the four possible combinations of switch settings.

Draw the circuit, labeling which pin numbers you use for each column of the truth table. Record your truth table and verify that this gate implements the NAND function. Repeat this process for each of the other three NAND gates on the chip (the main purpose of this is to help you learn how to count pins).

*Before proceeding, have your GTA check off Part 1 on your checklist.*

## **PART 2 - Multiplexer**

Connect the four NAND gates of the 7400 IC to construct your multiplexer design from the Prelab. Connect three toggle switches to the inputs of the multiplexer and connect the output to the Logic Probe. Record and verify the multiplexer truth table for all possible input combinations - debug your implementation as needed. Record each truth table and describe the problems and corrections you made to the circuit as you proceed to a working multiplexer.

*Have your GTA check off Part 2 on your checklist.*

## **PART 3 - Cleanup**

Return all components, put away all wires, and turn off all equipment.

*Have your GTA inspect your workstation and check off Part 3 on your checklist if acceptable.*

---

*If your circuit is not operating correctly at the end of the lab period, you may lose some or all of your in-lab points. You will not be allowed to work "overtime."*

*Submit your report to your GTA's mailbox prior to the start time of your lab period one week from today. Attach the required cover sheet and your initialed checklist to your report.*