Logic Gates and Combinational Logic Report

Michael Visser
300652084

Objective:

The primary objective of this experiment is to review the characteristics of basic logic gates, acquire hands-on experience in constructing circuits using a breadboard, and utilize logic gates as combinatorial building blocks to synthesize complex logic functions, and learning the implementation of logic functions using multiplexers.

Materials:

- Breadboard
- K&H ETS-5000
- Wires
- 74HCT00 (Quad 2-input NAND gate)
- 74HCT02 (Quad 2-input NOR gate)
- 74HCT04 (Hex inverter)
- 74HCT153 (4:1 multiplexer)

Making a one-bit half adder using logic gates (74HCT00, 74HCT02, 74HCT04):

74HCT00 truth table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

74HCT02 truth table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

74HCT00 truth table

A	X
0	1
1	0

Using these 3 logic gates, we needed to create a half adder. The half adder uses a XOR gate for the sum output, and an AND gate for the carry output. To achieve the XOR gate, I used a NAND gate and an XOR gate going into an inverter. Then the two outputs from those

gates I put into a NAND gate then another inverter. The Output for this gate looked like this afterwards.

Makeshift XOR gate truth table:

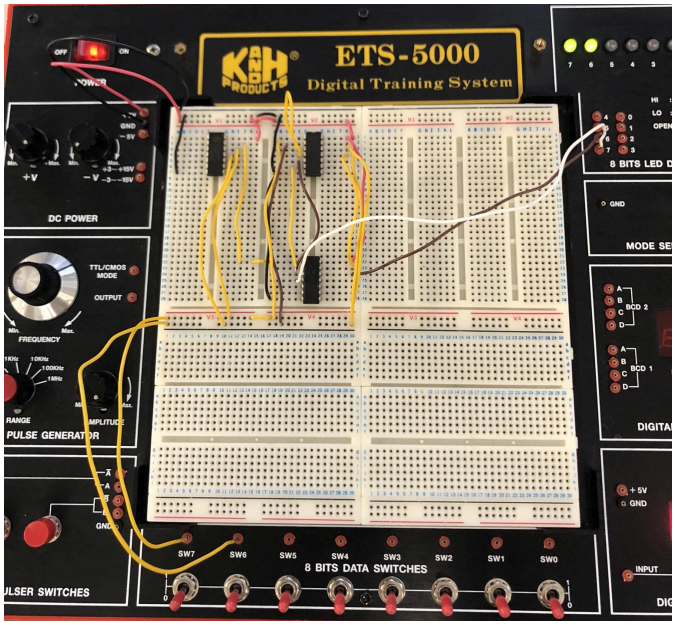
A	B	NAND	NOR	INV	NAND	INV
0	0	1	1	0	1	0
0	1	1	0	1	0	1
1	0	1	0	1	0	1
1	1	0	0	1	1	0

After I created the XOR gate, I used a NAND gate and an inverter to create the AND gate for the carry output.

This is what the truth table for the half adder looked like:

A	B	SUM (XOR)	CARRY (AND)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Photo of the completed circuit:

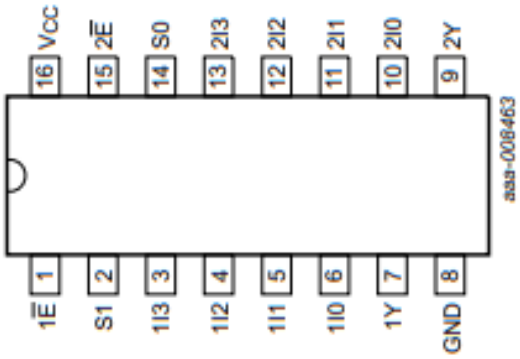


The gate on the top left is the 74HCT02, top right is the 74HCT00, and bottom right is the 74HCT04. The two switches wired into the circuit are used to control bit 1 and 2 so that we can easily test the half adder.

Making a one-bit half adder using a multiplexer (74HCT153):

The 74HCT153 is a multiplexer that can simulate 2 different logic gates by changing the signal of the 4 input pins. This makes it easier and more compact to create a half adder using one. We can create both an xor and an and gate on a single chip instead of needing 3 and a mess of wires.

The multiplexer has a pinout which looks like this:



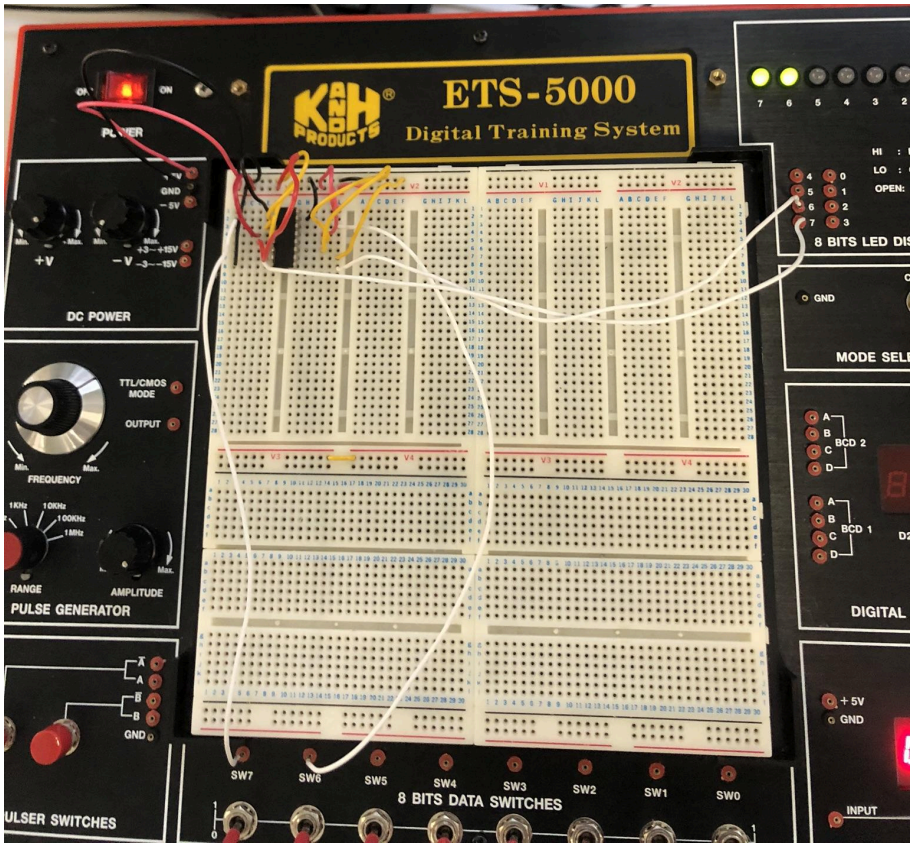
I used the left side of the chip (notch facing up) for the SUM (XOR gate) and the right for the CARRY (AND gate). To create the XOR gate, I connected 1I3 and 1I0 to low, then connected 1I2 and 1I1 to high. This made the output from 1Y become and XOR gate. The same went for the right side, I connected 2I0, 2I1, and 2I2 to low, then connected 2I3 to high to create and and gate.

The truth table for the multiplexer looked like this:

S0	S1	Y1	Y2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

This shows that the multiplexer has been turned into a half adder and the correct gates have been made.

The circuit looked like this:



Operations of the multiplexer (74HCT153):

The multiplexer specifically functions as a data selector/multiplexer and has two 4-input multiplexer sections. The inputs for these sections are labeled through from I0-I3. These are used to input the data that you want to select or multiplex.

The multiplexer also has two select inputs, S0 and S1. These inputs determine which input line is selected and routed to the output. The combination of binary values on these select inputs selects one of the four input lines:

S0	S1	selector
0	0	I0
0	1	I1
1	0	I2
1	1	I3

The data from the input lines is directed out of output Y for each section. The output is the result of the selected input based on the values of the selected inputs.

The multiplexer also has two active-low enable pins, G1 and G2. when either are low, the corresponding multiplexer is enabled allowing the selection of input lines based on the values of S0 and S1. when G1 and G2 are both high, the outputs are in a high-impedance state (disconnected).

The device can be used to simulate basic logic gates by activating the corresponding inputs on I0-I3 that you need to create a gate. If you want an XOR gate, you activate I1 and I2. This is because if we look at the table above, we will see that I1 and I2 will be high and I0 and I3 will be low. So when we have S0 on high and S1 on low, (or vise versa), the multiplexer will simulate an XOR gate.

Internal Structure Sketch of a 2:1 MUX: