Project Report
EET 256

James Solonika
Instructor: Walter Lara

Introduction

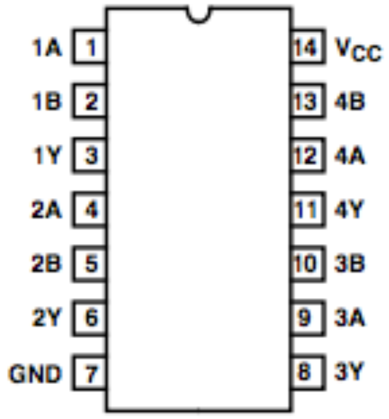
The intent of this project was to create a tester for commonly used ICs in school labs. The focus of the project is to test logic cate chips, with additional functionality to test op amps. Other ICs, such as counters and multiplexers that are less commonly used are not tested. I decided this would be very helpful in labs to test quickly if the chips being used in a large experiment are working correctly or if there is another issue present.

This is intended as a cheap alternative to other available IC testers, which can cost hundreds of dollars. The project was also created as a beginning stage for a larger project that can test more types of chips with enhanced functionality.

Experimental Procedure

Hardware:

The first step for this project was to determine how a test of a logic chip would be run. Looking at the 7400 series of logic ICs, a commonly used series, the physical layout of the pins are the same for most chips.

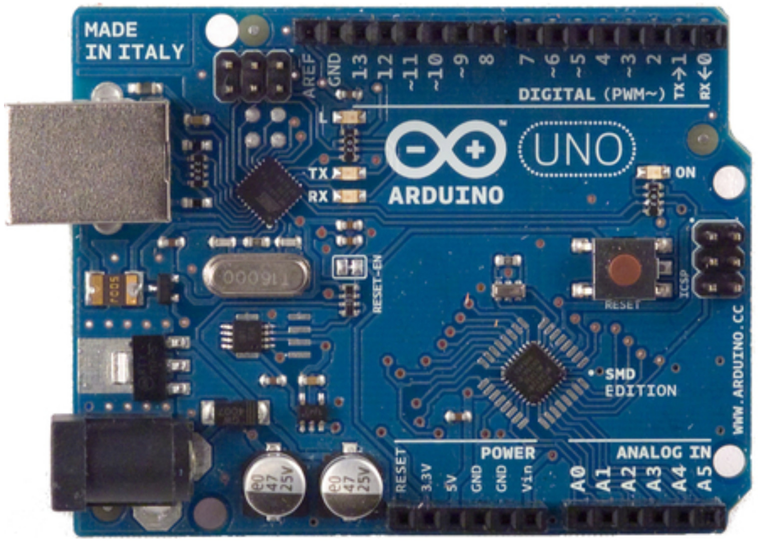


A 7400 series AND chip

Pins 1, 2, 4, 5, 9, 10, 12, and 13 are inputs and 3, 6, 8, and 11 are outputs. This means testing can be done with these pin positions assumed.

The test logic for this project is to be run on an Arduino UNO micro controller. This board has been chosen because I am familiar with it and already own one, but creates a problem right away. There are only 11 available digital I/O pins available on the board, not counting 2 I/O pins

(pins 0 and 1) used for programing the chip. those connections cannot be used without disconnecting them every time a new program is written to the chip.

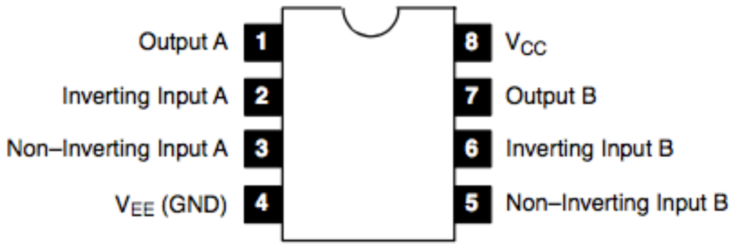


An Arduino UNO microcontroller programming board

The 7400 series logic chips have 12 pins that need to be put under test and I still want to leave room to run output LEDs and test Op Amps. To accomplish this goal, I decided to use two counters to provide the outputs to be fed to the logic inputs. Two four bit counters create an eight bit binary count that will provide all 256 unique combinations of inputs to the chips. The outputs of the logic function can also be sent to the analog in pins of the Arduino, which can be configured to act as digital inputs.

Five logic gates will be able to be tested: AND, OR, NAND, NOR and XOR. This will require five output LEDs, as well as an error or not good LED. To save pins, I used a demultiplexer to create a 3 bit to one of seven lines converter. The demultiplexer chip I have is active low output, so it must feed into an inverter in order to light one LED at a time. Six LEDs are connected to the final output, with the last line left empty so all LEDs can be off when needed.

The op amp chip has a simpler set up. It can be tested by being configured as a buffer, providing an input voltage and measuring the output voltage to see if it is the same as the input or not.

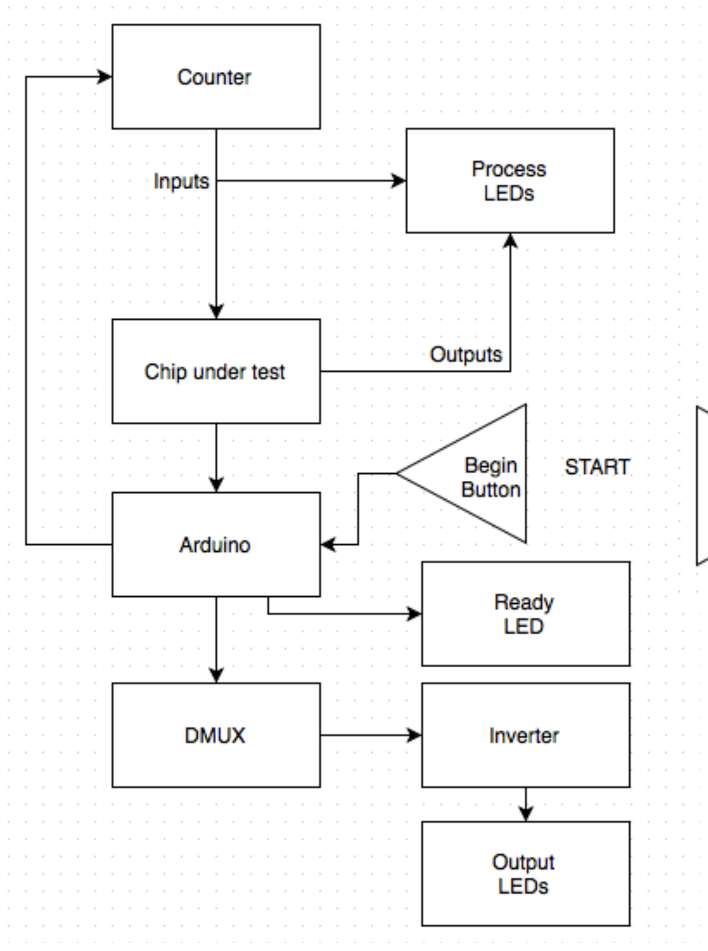


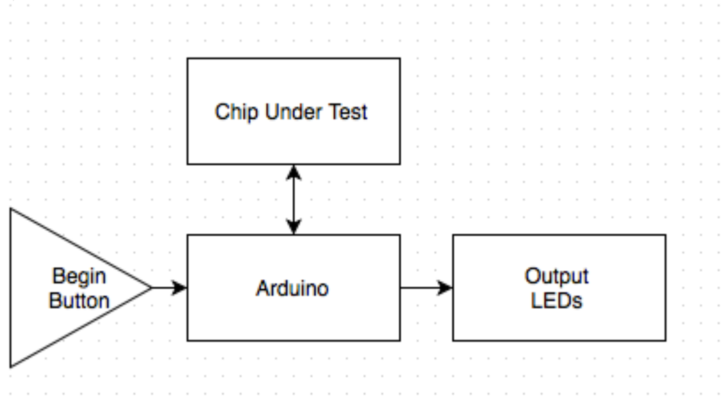
The common dual op amp chip layout that will be the testing template

By tying the outputs to the inverting inputs, the tester configuration is finished. A negative rail supply is not required, as the buffer will only be operated for a positive input. Two arduino pins are used as outputs to the non inverting op amp inputs, and two more pins are used to read the outputs, and a last pair to run indicator LEDs for good or bad.

Lastly, two pushbuttons are used to activate interrupts to run either the logic or op amp test program. One more LED is attached to pin 13, the boards built in LED pin, to indicate if the program is ready to begin or not, as well as to blink while the program is running. Safety diodes are applied between the power and ground rails and the Arduino power and ground pins to prevent accidental reverse power. A 0.1uF capacitor is placed between power and ground to stabilize power spikes and fluctuations.

For purposes of testing and demonstration, red LEDs are connected to each counter output / logic input, and green LEDs are attached to the logic outputs. 100Ω resistors are used to lower current going into them. pushbuttons are provided to simulate a such high, stuck low, and stuck adjacent errors.





Op amp test hardware block diagram

Logic test hardware block diagram

Software:

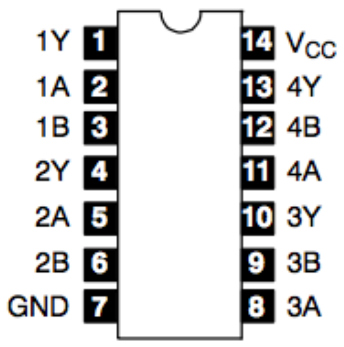
The first component of programing the tester is to create a comparison table for each logic type. The created logic tables are included in Appendix A. I have decided to put these tables as data in the Arduino EEPROM rather than as constant integer arrays in the program to save memory space and reduce boot time. The Arduino only has 1024 bytes of EEPROM available. I have five 256 value logic tables that take up four bits per value. I could fit all five tables in the EEPROM, but placing each value in one byte (twice the required space) greatly reduces the complexity of the program at a cost of being able to only store four tables. I am allowing the XOR table to be stored in the program.

The logic test function begins by initializing five boolean variables as true. If a chip has even one input combination output an incorrect value, the value will be changed to false. The test runs through 256 iterations, pulsing the counter clock each iteration and reading the values output from the chip. To read the value, I arbitrarily assigned the outputs bit positions so I could interpret the output as a number. Pin 8 is the MSB, followed by pins 11 and 6, with pin 3 being the LSB. The value is read by zeroing an integer variable, and adding 8 for a high value of the MSB, 4 for the next value, and so on. The read value is compared to a value in each logic table, and if an incorrect value for that logic function is detected, then the associated boolean is set to off. At the end of all 256 cycles, if any one boolean retains a true value, the associated LED is turned on. If all booleans are false, the red “Error” LED is turned on to indicate a malfunctioning chip.

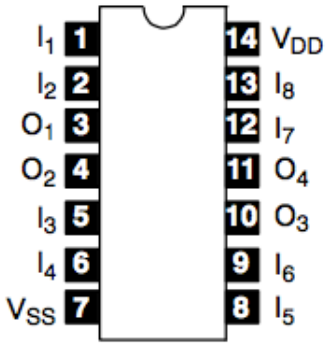
The op amp test program is much simpler. A digital high value is written to each non-inverting input, and the outputs are read. If both are outputting five volts (the Arduino digital high value) then the “op amp good” green LED is turned on, and if not the “op amp bad” red LED is turned on.

Results

The circuit works for all 7400 series logic chips I have available, as well as for my op amps. However, the NOR and XOR chips I have do not conform to the 7400 series pin layout I showed earlier, so I am not sure of the functionality of the test program for those logic types.



My NOR chip

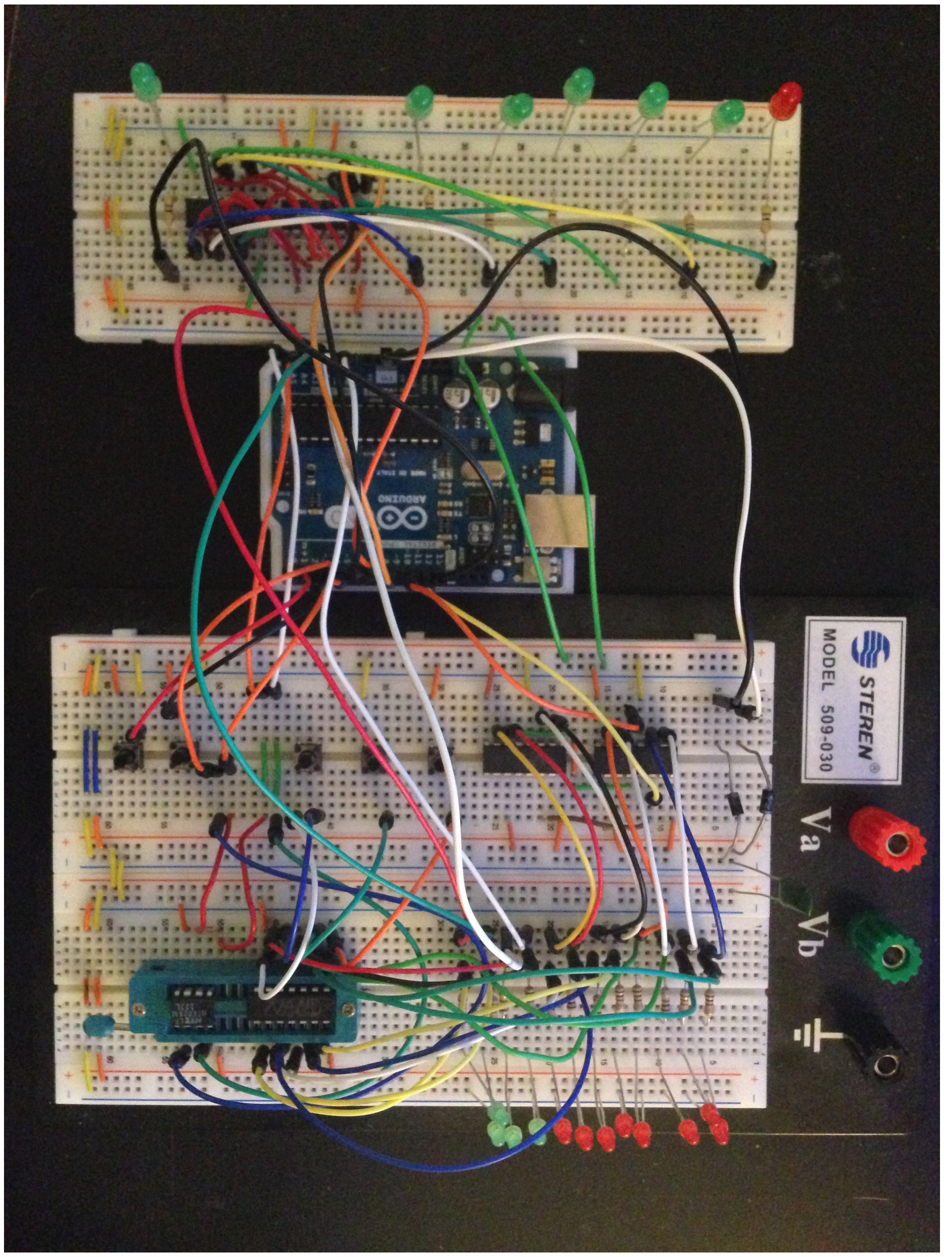


My XOR chip

Discussion of Results

I have achieved the results that I wanted to with this project. My initial assumption that the logic chips would all have the same layout proved to be incorrect. I did attempt to correct this mistake and allow testing of a second layout by using two high impedance bus chips. I hooked the counter output up to both busses and attached their outputs the the logic chip in two different configurations. This allowed me to use the Arduino to select which bus was allowing data flow and which was in the high impedance state. I then hooked eight pins up to a quad two line selector in the two different output configurations and used the Arduino to select which set of four inputs were sent to the Arduino. This created an extremely messy wiring setup and messed up the logic tests that were working. Given more time I could probably have fixed this, but I decided the best course of action for this class and project would be to go back to the configuration that worked.

Conclusions



The final project

I found great value in this project. From the initial concept, I thought this would be a project I could use in future schooling, as well as one I could expand and improve for the future. My plans are to purchase a advanced Arduino board, the ATmega, which has far more pins available than the UNO. By using this board, I will be able to dedicate a board pin to each pin of the chip under test, allowing tests of different configurations, three and greater inputs, as well as other chips.

For now, I have learned a great deal about planning and executing a project. In the beginning, I thought this project would be much simpler than it was. I assumed all chips were set up in the same way and I would have more than enough pins to complete my objectives. Understanding datasheets and doing more research than you think is necessary are necessary when undertaking a project that has no pre made instructions. I also taught myself to keep logs of my changes to both hardware and software. and to save a new file for each additional change to my project so I could go back in versions if needed.

I have also learned several things about the chips I was using, such as the counter chips output a low when a pin is not a high. This was an issue when I was attempting to setup the circuits to allow me to test two separate setups as I assumed they wouldn't output a value at all, but they drove a pin low when I was trying to send a different value to it. I also taught myself how to compress a lot of data into a smaller space when necessary, which I feel is an important consideration and skill for many industries.

Appendix A
Logic tables

AND

	Lower Nibble	0,0,0,0	0,0,0,1	0,0,1,0	0,0,1,1	0,1,0,0	0,1,0,1	0,1,1,0	0,1,1,1	1,0,0,0	1,0,0,1	1,0,1,0	1,0,1,1	1,1,0,0	1,1,0,1	1,1,1,0	1,1,1,1
Upper Nibble		0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,0,0,0	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,0,0,1	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,0,1,0	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,0,1,1	4	4	4	4	5	4	4	4	5	4	4	4	5	6	6	6	7
0,1,0,0	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,1,0,1	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,1,1,0	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
0,1,1,1	4	4	4	4	5	4	4	4	5	4	4	4	5	6	6	6	7
1,0,0,0	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
1,0,0,1	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
1,0,1,0	0	0	0	0	1	0	0	0	1	0	0	0	1	2	2	2	3
1,0,1,1	4	4	4	4	5	4	4	4	5	4	4	4	5	6	6	6	7
1,1,0,0	8	8	8	8	9	8	8	8	9	8	8	8	9	10	10	10	11
1,1,0,1	8	8	8	8	9	8	8	8	9	8	8	8	9	10	10	10	11
1,1,1,0	8	8	8	8	9	8	8	8	9	8	8	8	9	10	10	10	11
1,1,1,1	12	12	12	12	13	12	12	12	13	12	12	12	13	14	14	14	15

OR

	Lower Nibble	0,0,0,0	0,0,0,1	0,0,1,0	0,0,1,1	0,1,0,0	0,1,0,1	0,1,1,0	0,1,1,1	1,0,0,0	1,0,0,1	1,0,1,0	1,0,1,1	1,1,0,0	1,1,0,1	1,1,1,0	1,1,1,1
Upper Nibble		0	1	1	1	2	3	3	3	2	3	3	3	3	3	3	3
0,0,0,0	0	0	1	1	1	2	3	3	3	2	3	3	3	3	3	3	3
0,0,0,1	4	4	5	5	5	6	7	7	7	6	7	7	7	7	7	7	7
0,0,1,0	4	4	5	5	5	6	7	7	7	6	7	7	7	7	7	7	7
0,0,1,1	4	4	5	5	5	6	7	7	7	6	7	7	7	7	7	7	7
0,1,0,0	8	8	9	9	9	10	11	11	11	10	11	11	11	11	11	11	11
0,1,0,1	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
0,1,1,0	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
0,1,1,1	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
1,0,0,0	8	8	9	9	9	10	11	11	11	10	11	11	11	11	11	11	11
1,0,0,1	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
1,0,1,0	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
1,0,1,1	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
1,1,0,0	8	8	9	9	9	10	11	11	11	10	11	11	11	11	11	11	11
1,1,0,1	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
1,1,1,0	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15
1,1,1,1	12	12	13	13	13	14	15	15	15	14	15	15	15	15	15	15	15

NAND

	Lower Nibble	0,0,0,0	0,0,0,1	0,0,1,0	0,0,1,1	0,1,0,0	0,1,0,1	0,1,1,0	0,1,1,1	1,0,0,0	1,0,0,1	1,0,1,0	1,0,1,1	1,1,0,0	1,1,0,1	1,1,1,0	1,1,1,1
Upper Nibble		3	3	3	2	3	3	3	2	3	3	3	2	1	1	1	0
0,0,0,0	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
0,0,0,1	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
0,0,1,0	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
0,0,1,1	8	11	11	11	10	11	11	11	10	11	11	11	10	9	9	9	8
0,1,0,0	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
0,1,0,1	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
0,1,1,0	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
0,1,1,1	8	11	11	11	10	11	11	11	10	11	11	11	10	9	9	9	8
1,0,0,0	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
1,0,0,1	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
1,0,1,0	12	15	15	15	14	15	15	15	14	15	15	15	14	13	13	13	12
1,0,1,1	8	11	11	11	10	11	11	11	10	11	11	11	10	9	9	9	8
1,1,0,0	4	7	7	7	6	7	7	7	6	7	7	7	6	5	5	5	4
1,1,0,1	4	7	7	7	6	7	7	7	6	7	7	7	6	5	5	5	4
1,1,1,0	4	7	7	7	6	7	7	7	6	7	7	7	6	5	5	5	4
1,1,1,1	0	3	3	3	2	3	3	3	2	3	3	3	2	1	1	1	0

NOR

	Lower Nibble	0,0,0,0	0,0,0,1	0,0,1,0	0,0,1,1	0,1,0,0	0,1,0,1	0,1,1,0	0,1,1,1	1,0,0,0	1,0,0,1	1,0,1,0	1,0,1,1	1,1,0,0	1,1,0,1	1,1,1,0	1,1,1,1
Upper Nibble		3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
0,0,0,0	12	15	14	14	14	13	12	12	12	13	12	12	12	13	12	12	12
0,0,0,1	8	11	10	10	10	9	8	8	8	9	8	8	8	9	8	8	8
0,0,1,0	8	11	10	10	10	9	8	8	8	9	8	8	8	9	8	8	8
0,0,1,1	8	11	10	10	10	9	8	8	8	9	8	8	8	9	8	8	8
0,1,0,0	4	7	6	6	6	5	4	4	4	5	4	4	4	5	4	4	4
0,1,0,1	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
0,1,1,0	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
0,1,1,1	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
1,0,0,0	4	7	6	6	6	5	4	4	4	5	4	4	4	5	4	4	4
1,0,0,1	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
1,0,1,0	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
1,0,1,1	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
1,1,0,0	4	7	6	6	6	5	4	4	4	5	4	4	4	5	4	4	4
1,1,0,1	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
1,1,1,0	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0
1,1,1,1	0	3	2	2	2	1	0	0	0	1	0	0	0	1	0	0	0

XOR

	Lower Nibble	0,0,0,0	0,0,0,1	0,0,1,0	0,0,1,1	0,1,0,0	0,1,0,1	0,1,1,0	0,1,1,1	1,0,0,0	1,0,0,1	1,0,1,0	1,0,1,1	1,1,0,0	1,1,0,1	1,1,1,0	1,1,1,1
Upper Nibble		0	1	1	0	2	3	3	2	2	3	3	2	0	1	1	0
0,0,0,0	0	0	1	1	0	2	3	3	2	2	3	3	2	0	3	1	0
0,0,0,1	4	4	5	5	4	6	7	7	6	6	7	7	6	4	7	5	4
0,0,1,0	4	4	5	5	4	6	7	7	6	6	7	7	6	4	7	5	4
0,0,1,1	0	0	1	1	0	2	3	3	2	2	3	3	2	0	3	1	0
0,1,0,0	8	8	9	9	8	10	11	11	10	10	11	11	10	8	11	9	8
0,1,0,1	12	12	13	13	12	14	15	15	14	14	15	15	14	12	15	13	12
0,1,1,0	12	12	13	13	12	14	15	15	14	14	15	15	14	12	15	13	12
0,1,1,1	8	8	9	9	8	10	11	11	10	10	11	11	10	8	11	9	8
1,0,0,0	8	8	9	9	8	10	11	11	10	10	11	11	10	8	11	9	8
1,0,0,1	12	12	13	13	12	14	15	15	14	14	15	15	14	12	15	13	12
1,0,1,0	12	12	13	13	12	14	15	15	14	14	15	15	14	12	15	13	12
1,0,1,1	8	8	9	9	8	10	11	11	10	10	11	11	10	8	11	9	8
1,1,0,0	0	0	1	1	0	2	3	3	2	2	3	3	2	0	3	1	0
1,1,0,1	4	4	5	5	4	6	7	7	6	6	7	7	6	4	7	5	4
1,1,1,0	4	4	5	5	4	6	7	7	6	6	7	7	6	4	7	5	4
1,1,1,1	0	0	1	1	0	2	3	3	2	2	3	3	2	0	3	1	0