LAB REPORT : 2

NAME: Romica Raisinghani

ROLL NO: 2021101053

GROUP NO: 1

AIM OF THE EXPERIMENT:

PART A:

To find the approximate threshold value where one LED stops glowing and the other one starts to with the help of NOT gate.This is to assign digital numbers HIGH(1) for values above the threshold value and LOW(0) to values below the threshold value.

PART B:

To take input from the serial monitor and verify truth table for the logic gates namely AND,OR,NOT,NAND and XOR using TTL 74XX family of ICs.

PART C:

To verify De Morgan's Law for the equation (A.B)' = A' + B' with the help of NOT,OR and NAND gates using TTL 74XX family of ICs.

PART D:

To justify the working of binary full adder with the help of a circuit that adds two bits A and B along with a carry in C to generate SUM and CARRY bits as output by using the XOR and AND gates.

ELECTRONIC COMPONENTS USED:

PART A:

Arduino Uno R3, two multimeters(for measuring voltage), two LEDs, two resistances(1 kohm each), Hex Inverter 74HC04(NOT Gate), potentiometer(250 ohms) and connecting wires.

PART B:

Arduino Uno R3, seven LEDs, two resistances(200 ohms each), five resistances(1 kohm each), Hex Inverter 74HC04, QUAD AND 74HC08, QUAD OR 74HC32, QUAD NAND 74HC132, QUAD XOR 74HC86 and connecting wires.

PART C:

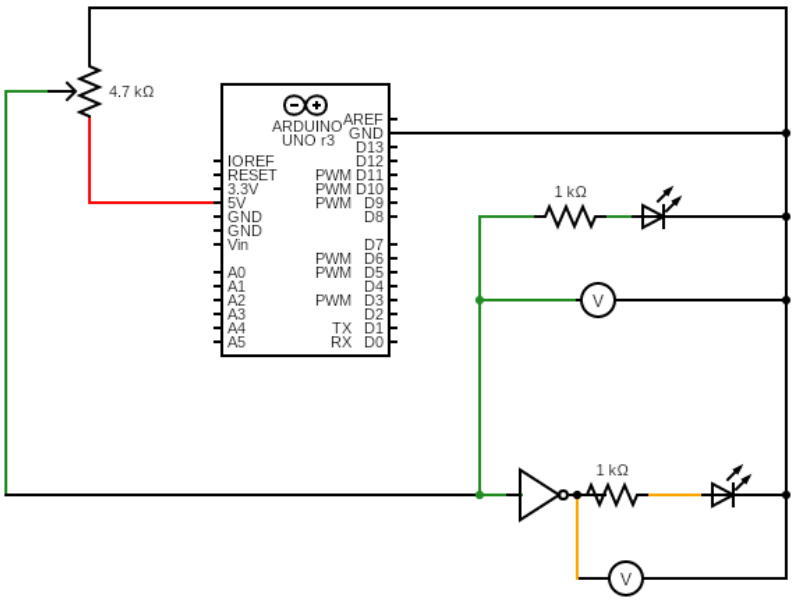
Arduino Uno R3, four LEDs, four resistances(1 kohm each), Hex Inverter 74HC04, QUAD OR 74HC32, QUAD NAND 74HC00 and connecting wires.

PART D:

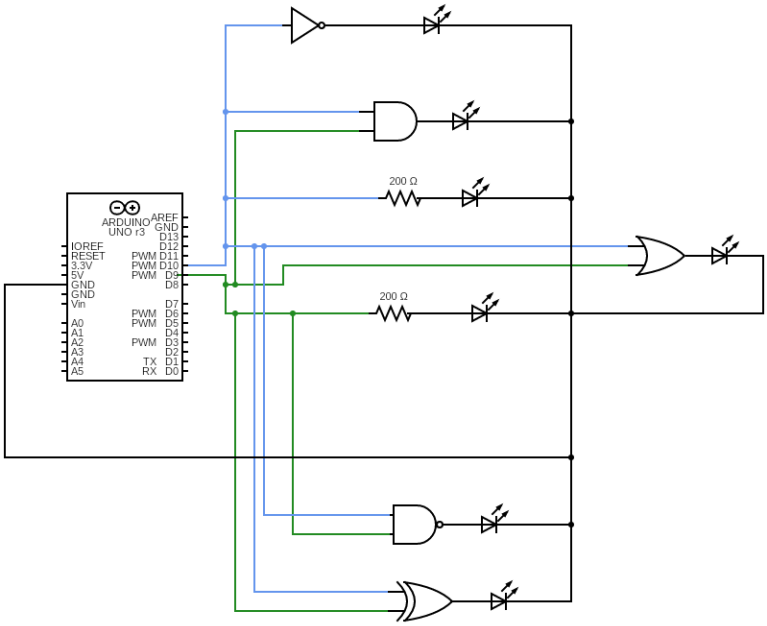
Arduino Uno R3, five LEDs, five resistances(1 kohm each), QUAD XOR 74HC86, QUAD AND 74HC08 and connecting wires.

REFERENCE CIRCUITS:

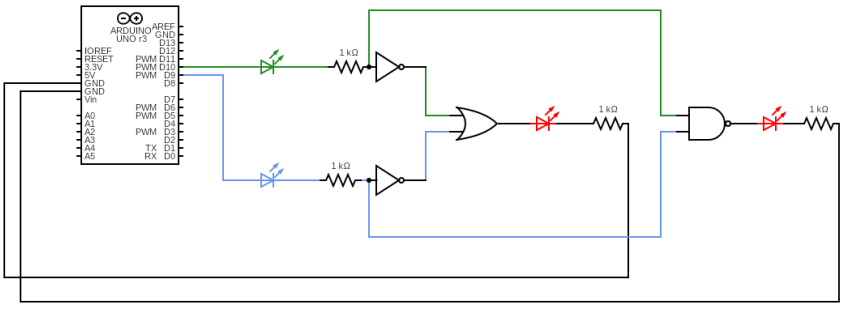
PART A: CIRCUIT DIAGRAM TO ILLUSTRATE FINDING OF THRESHOLD VALUE



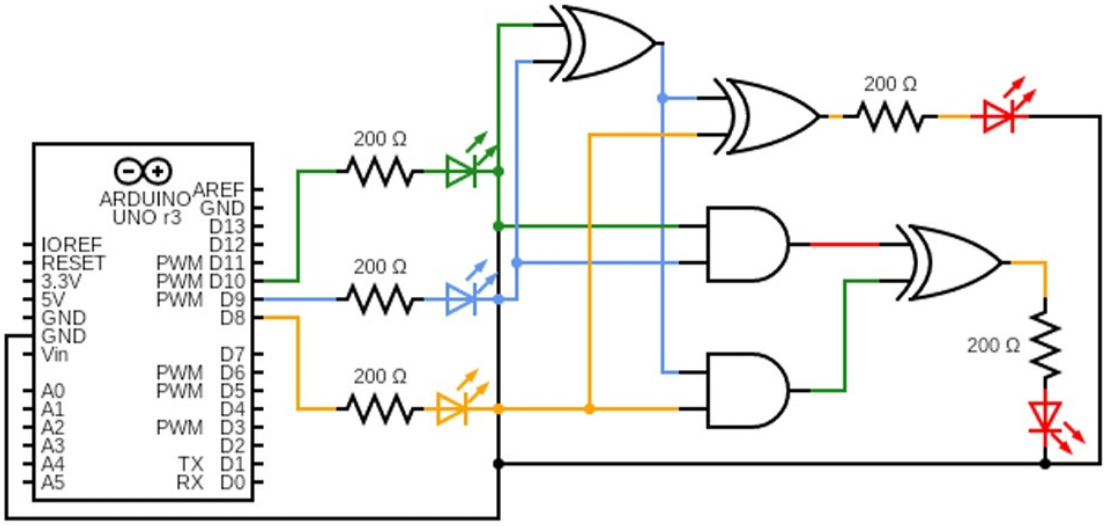
PART B: CIRCUIT DIAGRAM TO ILLUSTRATE THE TRUTH VALUES OF DIFFERENT LOGIC GATES



PART C: CIRCUIT DIAGRAM TO ILLUSTRATE THE WORKING OF DE MORGAN'S LAW



PART D: CIRCUIT DIAGRAM TO ILLUSTRATE THE BINARY FULL ADDER



PROCEDURE:

PART A:

1. Set up the circuit shown in the circuit diagram above for part A on the breadboard and turn the potentiometer shaft to one
   end so that the multimeter reads 0V for LED2 . In the circuit, the green wire represents A and the blue wire represents NOT(A).
2. Ground the NOT gate and write the appropriate code for the Arduino to function.
3. When the potentiometer shaft is towards extreme left, LED1 glows initially having potential difference of 4.99V and LED2 does not glow having potential difference of 0V.

4. Now rotate the potentiometer shaft gradually up to the other end and tabulate the transitions in both the LEDs.
5. On observing we find that LED1 stops glowing when the approx value of potential difference against it is 2.46V and at the same instant LED2 starts glowing when the potential difference against it attains a value of approx 4.87V.
6. This leads us to conclude that below 2.46V the INPUT has value LOW and has a value HIGH above it.

PART B:

1. Place the ICs on breadboard and give Vcc and Gnd connection to it as shown in the circuit.
2. Connect pin 10 of Arduino to LED A through a resistor and similarly pin 9 to LED B through a resistor.
3. Take inputs from the Serial Monitor for values of A and B and route them to the input pins of
   the IC.
4. Connect the required gates as mentioned in electronic components each to one LED through a resistor and ensuring proper grounding of the circuit throughout.
5. Follow this procedure for all the five mentioned logic gates by referring to the circuit diagram shown above.
6. Note the output of the chosen gate for different values of input in a truth table.

PART C:

1. Place the ICs on breadboard and give Vcc and Gnd connection to it as shown in the circuit.
2. Connect pin 10 of Arduino to LED A (green) through a resistor and similarly pin 9 to LED B(blue) through a resistor.
3. Connect the LEDs further to two separte NOT gates and then the output of these to a OR gate that connects to a LED(red) and a resistor.
4. Verify the above output of the circuit by builing a NAND circuit further.

5. Connect LED A and LED B to a NAND gate through another LED(red) and a resistor.
6. Ground the LEDs and the gates and write appropriate code for the Arduino to function.
7. Obtain the truth table of this circuit by noting the output of the function for different values of A and B. And ensure that the two circuits give the same output in order to verify the De Morgan's Law (A.B)' = A' + B'.

EXTRA QUESTION:

How would you realise the above circuit if you have only NAND gates instead of NOT gates? i.e
How would you use NAND gates to perform function of NOT gates?

ANS) NAND gate is a universal gate and can be used for making other gates as well. NAND gate unlike the NOT gate requires two inputs to generate an output. We can achieve this by applying the NAND gate to the input itself. Now only two outputs are possible in AND ie HIGH or LOW. The NAND gate will further invert HIGH to LOW and LOW to HIGH. The following truth table justifies the use of NAND gate that can be used in place of NOT gate.

NAND GATE			
A	**A**	**A.A**	**(A.A)'**
0	0	0	1
1	1	1	0

PART D:

1. Place the ICs on breadboard and give Vcc and Gnd connection to it as shown in the circuit.
2. Connect pin 10 of Arduino to LED A (green) through a resistor and similarly pin 9 to LED B(blue) AND PIN 8 TO LED C(orange).

3. Set up the circuit of a Half Adder using an XOR gate and an AND gate. Apply the inputs A and B from two input pins and observe the outputs S1 and C1 on two LED displays for all combinations of the inputs. Tabulate these values and verify the operation of the Half Adder.
4. Set up another Half Adder using another XOR and another AND gate out of the same ICs used in step 3, and connect the C input and the S1 output generated by the first Half Adder as
   its inputs to generate the final SUM output and the C2 output.
5. Generate the final CARRY output from the intermediate carry outputs C1 and C2, using the unused gates in the XOR and AND ICs deployed so far.
6. Ground the LEDs and the gates and write appropriate code for the Arduino to function.
7. Verify the truth table experimentally by applying the inputs A, B and C through three input pins and displaying the S1, C1, C2, SUM and CARRY outputs.

TRUTH TABLES:

PART A:

NOT GATE	
A	**NOT A**
1	0
0	1

PART B:

XOR GATE		
A	**B**	**A XOR B**
0	0	0
1	0	1
0	1	1
1	1	0

AND GATE		
A	**B**	**A.B**
0	0	0
1	0	0
0	1	0
1	1	1

OR GATE		
A	**B**	**A+B**
0	0	0
1	0	1
0	1	1
1	1	1

NAND GATE		
A	**B**	**(A.B)'**
0	0	1
1	0	1
0	1	1
0	0	0

PART C:

DE MORGAN'S TABLE (A.B)' = A' + B'						
A	B	A'	B'	A'+B'	A.B	(A.B)'
0	0	1	1	1	0	1
1	0	0	1	1	0	1
0	1	1	0	1	0	1
1	1	0	0	0	1	0

PART D:

A	B	C	S1	C1	C2	SUM	CARRY
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	1	0	0	1	0
0	1	1	1	0	1	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	1	0	1
1	1	0	0	1	0	0	1
1	1	1	0	1	0	1	1

CONCLUSION:

PART A:

In Part A, we built a circuit intending to find the tiping point of the voltage where LED1 stops glowing and LED2 starts glowing.We found the approximate value to be 2.46V which concludes us to the fact that below 2.46V the binary logic value will be 0 and above 2.46V the value will be 1. Hence we have converted the analog data of voltage in the range of (0-5)V to binary logic values.

The tipping voltage for the IC which it considers to be HIGH or LOW is:

i) **INPUT(low)** = 0 < Vil<2.46

ii) **INPUT(high)** = 2.46<Vih<5

iii) **OUTPUT(low)** = 0<Vol<2.27

iv) **OUTPUT(high)** = 2.35<Voh<5

PART B:

In Part B, we verified the actual working of AND,OR,NOT,XOR and NAND gates with the help of circuits by taking the input from the serial monitor and verifying the output by the proper blinking of LEDs for each gate.

PART C:

In Part C, we verified the De Morgan's Law for the equation (A.B)' = A' + B' by the help of NOT,OR and NAND gates by obtaining the NOT values of A and B first and passing them through OR gate.Then we verified the outcome of this by connecting the LEDs to the NAND gate and observed that the two circuits yield the exact same value for different inputs of A and B and hence proving the De Morgan's Law.

PART D:

In Part D, we built a circuit pertaining to prove the logic behind addition of two bits A and B along with a carry in C to generate SUM and CARRY bits as output. The first step to achieve this is to make a binary Half Adder, which adds two binary inputs A and B to give a sum S1 and a carry C1 according to the following Boolean expressions for the outputs S1 and C1:
$S1 = A' \cdot B + A \cdot B' = A \oplus B$ and $C1 = A \cdot B$

Another Half Adder is then used to generate the final SUM by adding the third binary input C to the S1 bit generated by the first Half Adder:

SUM = S1 $\oplus$ C

The carry bit generated by this Half Adder is given by:

C2 = S1 · C

CARRY = C1 +C2

We match the results of these observations by the output of our circuit and hence conclude the working of a full adder circuit.

Link of TinkerCAD simulation:

Link for PART A: https://www.tinkercad.com/things/7lZ3Bh9Br6Z

Link for PART B: https://www.tinkercad.com/things/6RxDcsGx5UD

Link for PART C: https://www.tinkercad.com/things/eCmAoU3QI2p

Link for PART D: https://www.tinkercad.com/things/7X7dejESuob

THANK YOU !