Computer Lab 1

Digital Logic Circuits

Computer Architectures for MSc in Engineering

HT19, DT509G

Abshir Abdulrahman & Fria Khorshid

2019-09-19

Objective

The objective of this lab was to get familiar with building digital logic circuits on a breadboard and using a microcontroller to control the logic with different inputs.

In your report,

describe the steps you have taken to solve each task,

as well as the results you obtained

the methods you used to verify that the system functions properly.

Explain what experiments you performed,

how and why.

Provide your Arduino sketch as an attachment

demonstrate the final system to the lab assistant.

take pictures of the system in action,

and/or include screenshots.

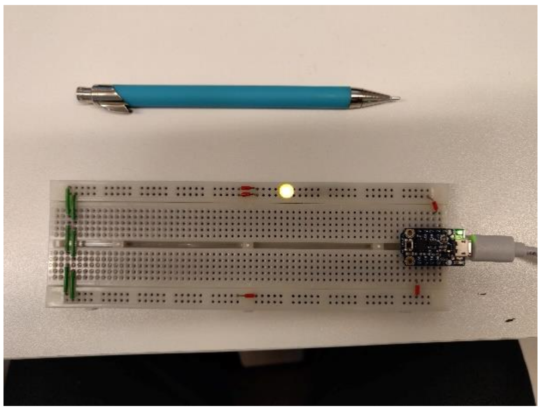
Send your lab report by e-mail to the lab instructor.

Preliminaries

We started by making sure that our software was up to date and that our microcontroller was connected correctly to our breadboard. We did this by using the tutorial code that made the led light on the microcontroller to blink. When this was confirmed we continued with the rest of the exercises.

Task 1: Breadboard Setup and a NAND gate

We started by making sure that our Arduino powered the whole breadboard by testing the connection with a LED-light. This was a simple circuit to get the current flowing around the board. Below is a picture showing the current going through the circuit and lighting the LED.



Next, we installed a nand gate array on the breadboard and weird the outputs from the microcontroller to a nand gate hub. After this we also attached led lights to the output from the nand gate and the inputs from the microcontroller.

Next, we sent a signal through the microcontroller in Arduino to the nand gate to see if it works as intended. As shown in the pictures below, the nand gate works as intended. We checked our results with a truth table for a nand gate.

Input 3, to the left, closest to the yellow lamp

Input 4, to the right of the first red lamp, farthest from the yellow lamp

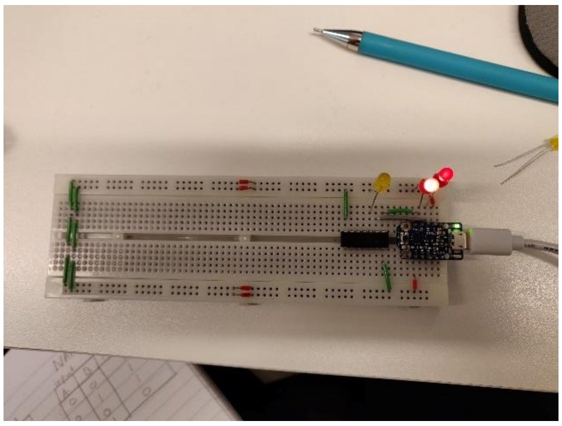
Red lamps:

Input 3 = 1, light on

Input 4 = 1, light on

Yellow lamp:

Output = 0, light off



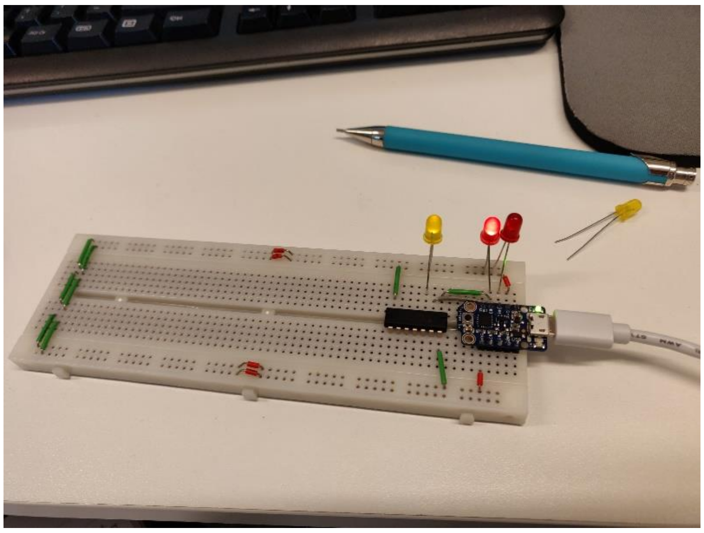
Red lamps:

Input 3 = 1, light on

Input 4 = 0, light off

Yellow lamp:

Output = 1, light on



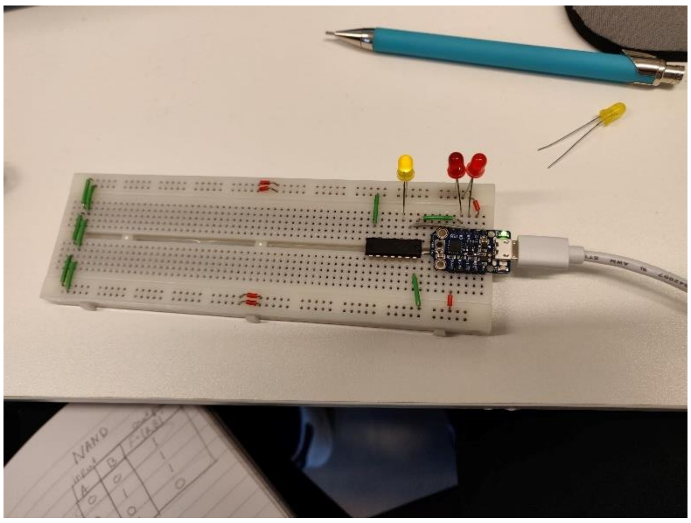
Red lamps:

Input 3 = 0, light off

Input 4 = 0, light off

Yellow lamp:

Output = 1, light on



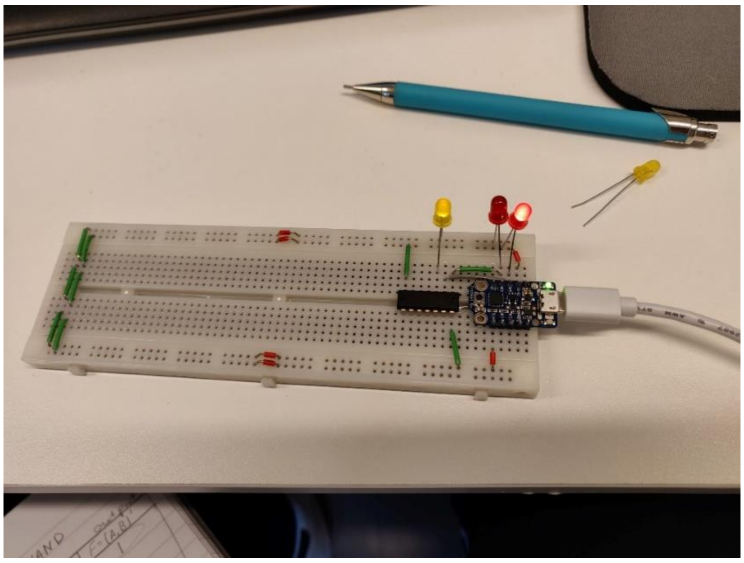
Red lamps:

Input 3 = 0, light off

Input 4 = 1, light on

Yellow lamp:

Output = 1, light on



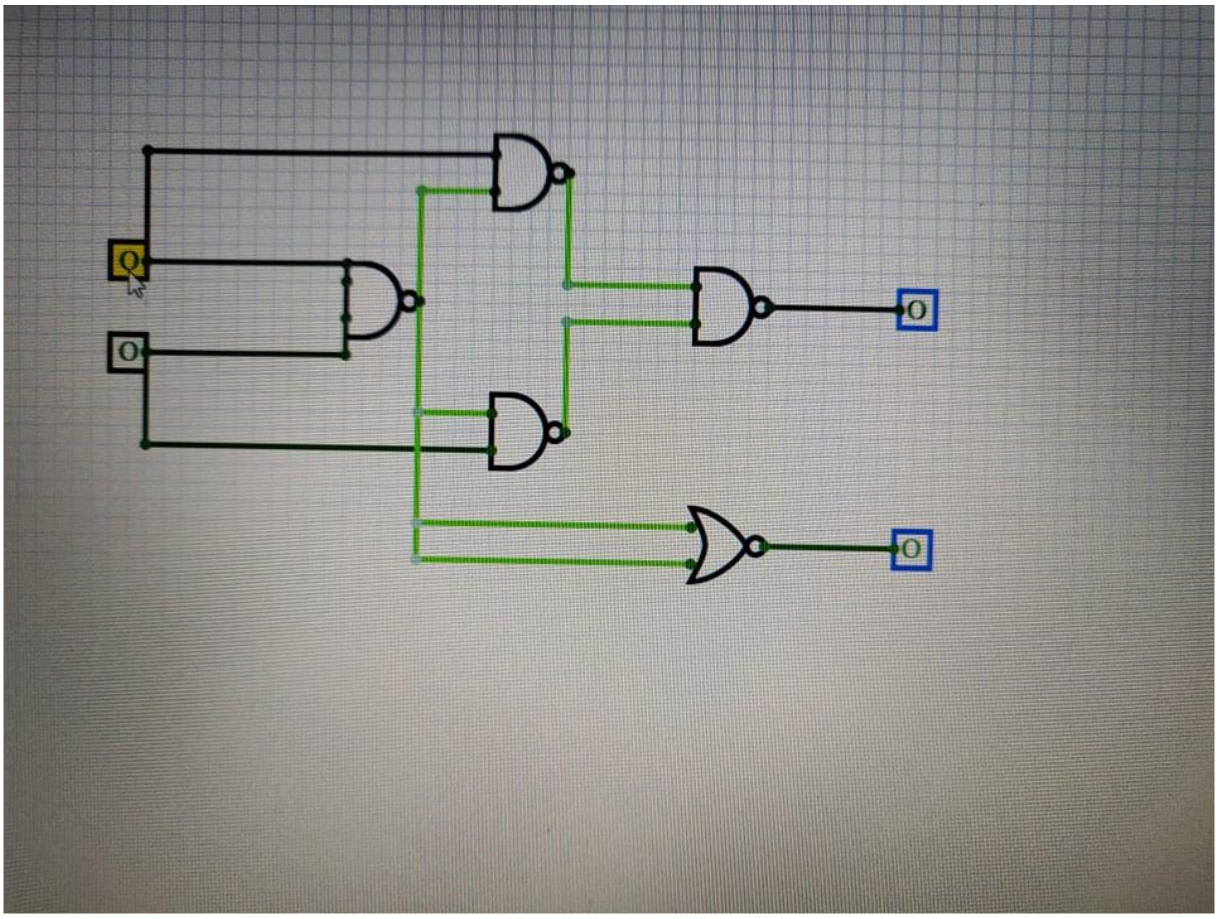
Task 2: Half-adder

In this exercise we were supposed to construct a half-adder circuit by using a maximum of 4 NAND gates and 1 NOR gate.

We started by sketching out the circuit by using some apps on the internet. These sketches and designs were used to get a feel for the problem at hand. We wrote a truth table and checked our design to see if it gives the correct outputs according to our truth table.

Next, we implemented our design on our breadboard and used the LED lights to check if our half-adder works.

This is the design we made before starting to implement the circuit. We could test our circuit by changing the input values to the left and get the output results to the right.



In the pictures below the yellow lamp 1 in the same “row” as the red lamps, shows the carry and the yellow lamp 2 basically by itself on the other side, shows the sum.

This is the code we wrote in Arduino to test our circuit.

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize the 3rd and 4th pin as an output.
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop() {
  // INPUT BITS 1, 0. OUTPUT BIT 1
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  delay(1000);
 /*
  // INPUT BITS 0, 1. OUTPUT BIT 1
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  delay(1000);

  // INPUT BITS 1, 1. OUTPUT BIT 0
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  delay(1000);

  // INPUT BITS 0, 0. OUTPUT BIT 1
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  delay(1000);

}
```

Red lamps:

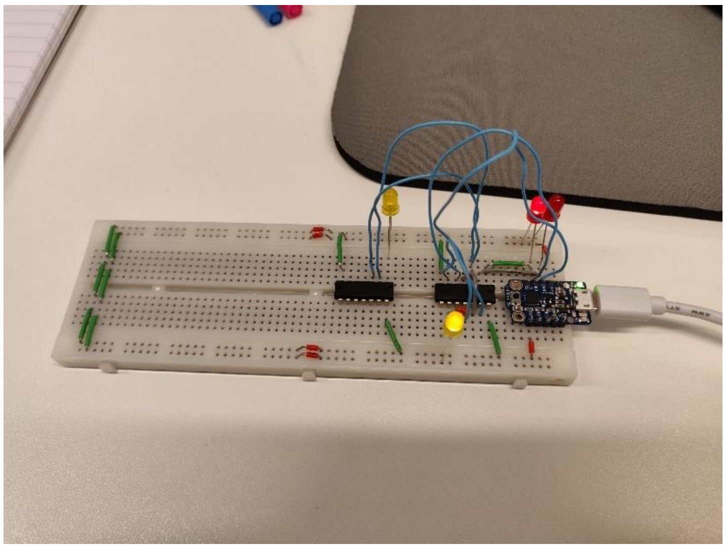
Input 3 = 1, light on

Input 4 = 0, light off

Yellow lamps:

Sum = 1, light on

Carry= 0, light off



Red lamps:

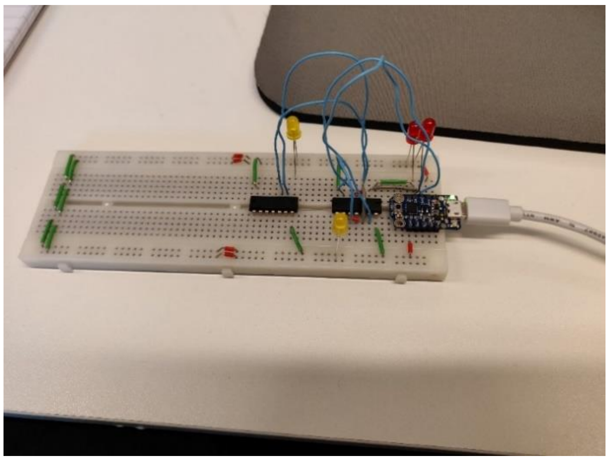
Input 3 = 0 light off

Input 4 = 0, light off

Yellow lamps:

Sum = 0, light off

Carry= 0, light off



Red lamps:

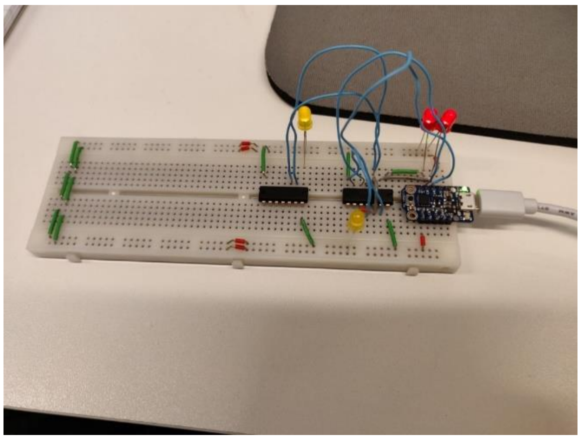
Input 3 = 1, light on

Input 4 = 1, light on

Yellow lamps:

Sum = 0, light off

Carry= 1, light on



Red lamps:

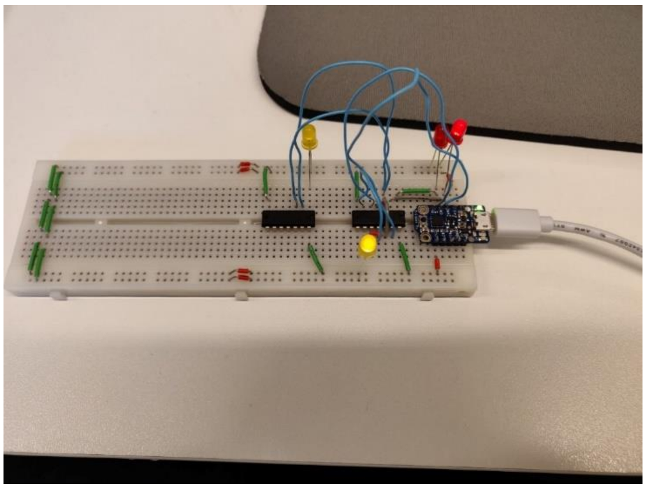
Input 3 = 0, light off

Input 4 = 1, light on

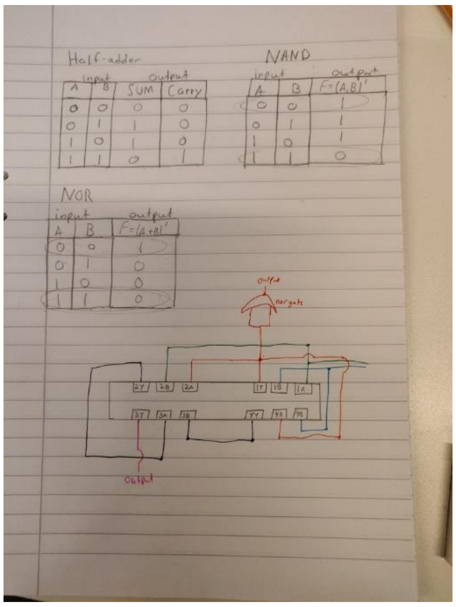
Yellow lamps:

Sum = 1, light on

Carry= 0, light off



These are the truth tables and our nand gate design we used to implement our circuit.



Half-adder

input		output	
A	B	SUM	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

NAND

input		output
A	B	F=(A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

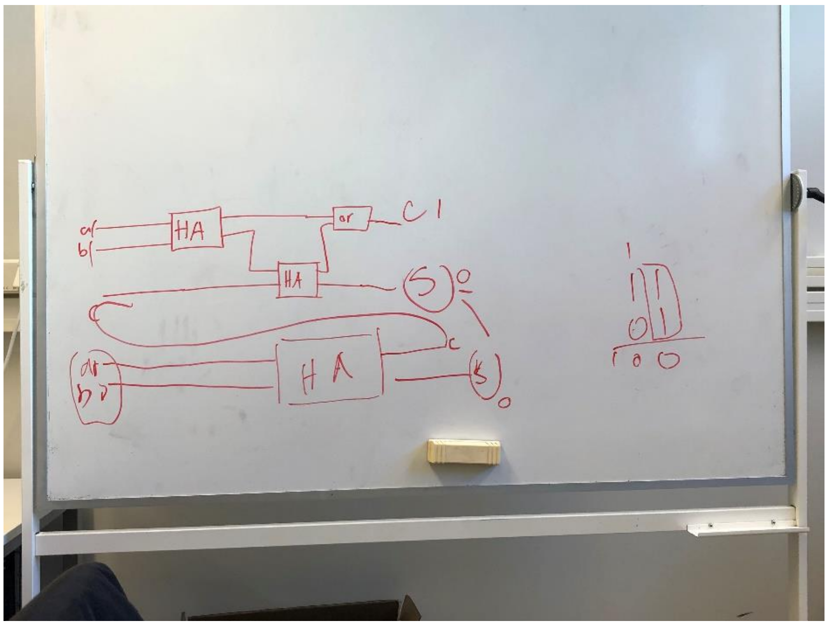
NOR

input		output
A	B	F=(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

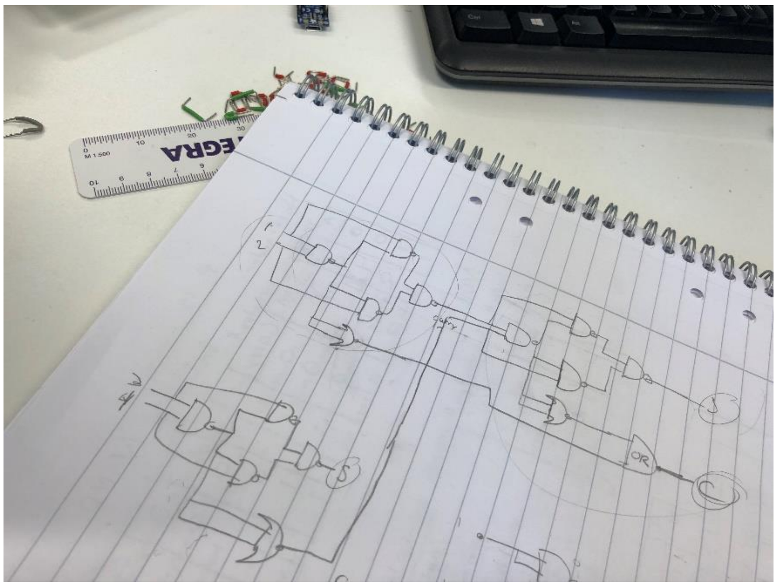
Task 3: 2-bit adder circuit

In this exercise we implemented a 2-bit adder circuit. Unfortunately, we did not manage to make it work as intended.

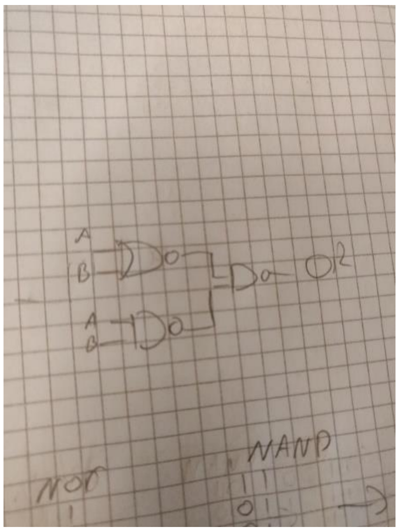
We started by drawing a sketch of how we were thinking of implementing the circuit. This is how we planned on wiring the circuit.



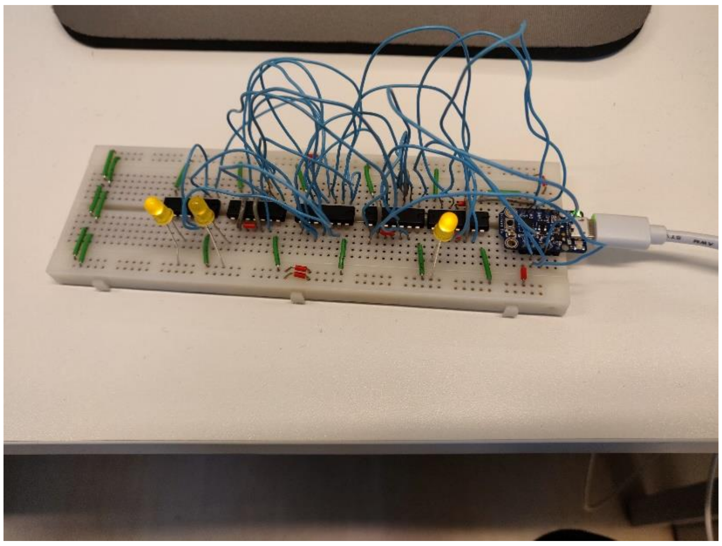
We checked if the design works by trying out some 2-bit addition. According to our calculations this design should give us the correct answers. After we were satisfied with the design we drew it with more detail. Specifically, we drew it again with the gates. This is the result that we got.



We also implemented an OR gate by using 2 NAND gates and one NOR gate. This was done by setting up the circuit in the following way. This part is needed in the last bit of the circuit. It calculates the last carry bit.



This is how the circuit looks like when set up. The LED from the right to the left represents the first sum from the first adder, the second sum and the carry bit.



We tried to implement our design, but our final circuit did not work as intended. This was because of a few factors. The first one being time. We ran out of time when trying to implement the solution, this led us to making hasty mistakes in the wiring of the circuit. The easiest way of finding the error is to start over with the wiring.