EEU33C02 Digital Circuits: Logic Lab Assignment

Stephen Komolafe (21336975)

Abstract

The objective for this assignment was to create a logic circuit for natural number addition. This involved utilising predefined outputs $S(A, B)$ and $C(A, B)$ along with specific boundary conditions. This was achieved through the initial creation of a truth table, followed by the making Karnaugh maps to derive Boolean expressions for the outputs S_0, S_1, and C. Due to limited resources, the circuit had to be split into three parts for testing.

For S_0, a simple XOR gate was used, resulting in simple construction and successful testing using LEDs. The S_1 circuit was more complex due to wire management and multiple logic gates, but it also generated the expected outputs. The C circuit was the most intricate, involving three unique sets of inputs controlled by the microcontroller and interconnected gate outputs as inputs.

Despite challenges faced, the testing methodology was consistent, and all circuits produced the desired outputs for all input combinations. In the end, the project successfully created a logic circuit for natural number addition, with each section presenting its unique challenges. Ultimately the splitting of the circuit into parts allowed for a greater understanding of how the circuit works as a whole.

Function Implementation

To define the state of each input and output in a natural number addition function, a truth table was constructed by considering the equations and conditions provided in the assignment instructions for each variable.

- $A, B, S(A, B) \in \{x \in \mathbf{N} \mid 0 \leq x < N\}$.
- $C(A, B) \in \mathbf{B}$.
- $N = 4$.

*Therefore A, B and S are in decimal form to suit conditions

*C is in binary from (outputs only 1 or 0)

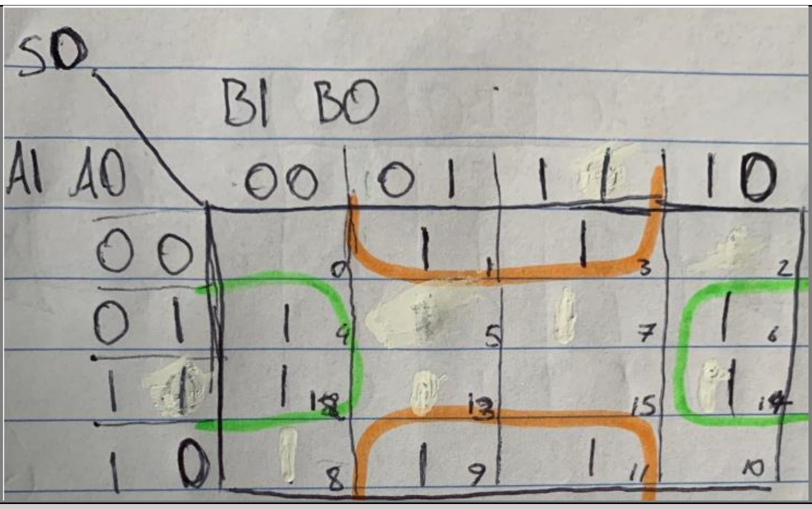
$(A)_{10}$	$(B)_{10}$	S(A, B) [A + B mod N]	$C(A, B)$ $\begin{cases} 0, A + B < N \\ 1, otherwise \end{cases}$
0	0	0	0
0	1	1	0
0	2	2	0
0	3	3	0
1	0	1	0
1	1	2	0
1	2	3	0
1	3	0	1
2	0	2	0
2	1	3	0
2	2	0	1
2	3	1	1
3	0	3	0
3	1	0	1
3	2	1	1
3	3	2	1

In order to find a way to incorporate this data into a logic level circuit, the input and output variables, with the exception of C, were converted from decimal form to binary form with A, B and S being split into two bits each. (A_1A_0, B_1B_0 and S_1S_0 respectively)

N	A_1	A_0	B_1	B_0	S_1	S_0	$C(A,B)$ $\{0, A+B<N$ $\{1, otherwise$
0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	1	0
4	0	1	0	0	0	1	0
5	0	1	0	1	1	0	0
6	0	1	1	0	1	1	0
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	1	0
10	1	0	1	0	0	0	1
11	1	0	1	1	0	1	1
12	1	1	0	0	1	1	0
13	1	1	0	1	0	0	1
14	1	1	1	0	0	1	1
15	1	1	1	1	1	0	1

Karnaugh maps were formed at this point and subsequently utilized to derive Boolean expressions for the outputs S_0, S_1 and C.

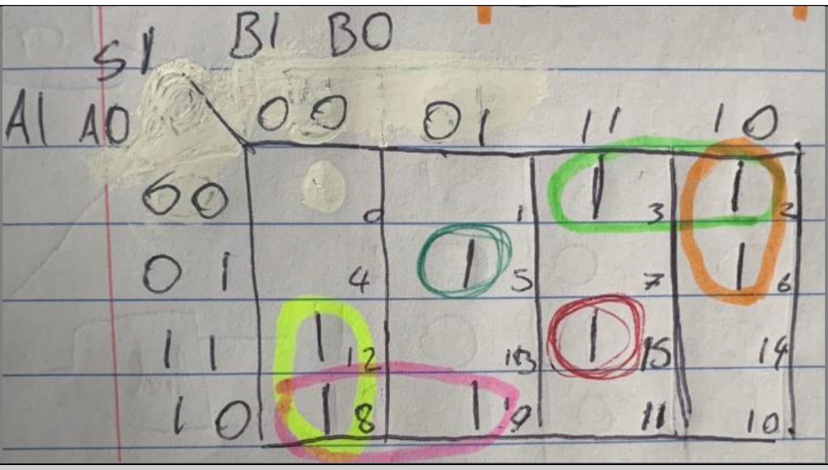
- **S_0 Output**



S0 A1 A0 \ B1 B0	00	01	11	10
00	0	1 1	1 3	2
01	1 4	5	7	1 6
11	1 12	13	15	1 14
10	8	1 9	1 11	10

$$S_0 = \overline{A_0}B_0 + A_0\overline{B_0}$$
$$\therefore S_0 = A_0 \oplus_{(xor)} B_0$$

- **S_1 Output**



$$S_1 = \overline{A_1}A_0\overline{B_1}B_0 + A_1A_0B_1B_0 + \overline{A_1}\,\overline{A_0}B_1 + \overline{A_1}B_1\overline{B_0} + A_1\overline{A_0}\,\overline{B_1} + A_1\overline{B_1}\,\overline{B_0}$$
$$= A_0B_0(\overline{A_1}\,\overline{B_1} + A_1B_1) + \overline{A_0}(\overline{A_1}B_1 + A_1\overline{B_1}) + \overline{B_0}(\overline{A_1}B_1 + A_1\overline{B_1})$$

$\bar{X}\,\bar{Y} + XY = 1$

$$\therefore \quad = A_0B_0 + \overline{A_0}(\overline{A_1}B_1 + A_1\overline{B_1}) + \overline{B_0}(\overline{A_1}B_1 + A_1\overline{B_1})$$
$$= A_0B_0 + (\overline{A_0} + \overline{B_0})(\overline{A_1}B_1 + A_1\overline{B_1})$$

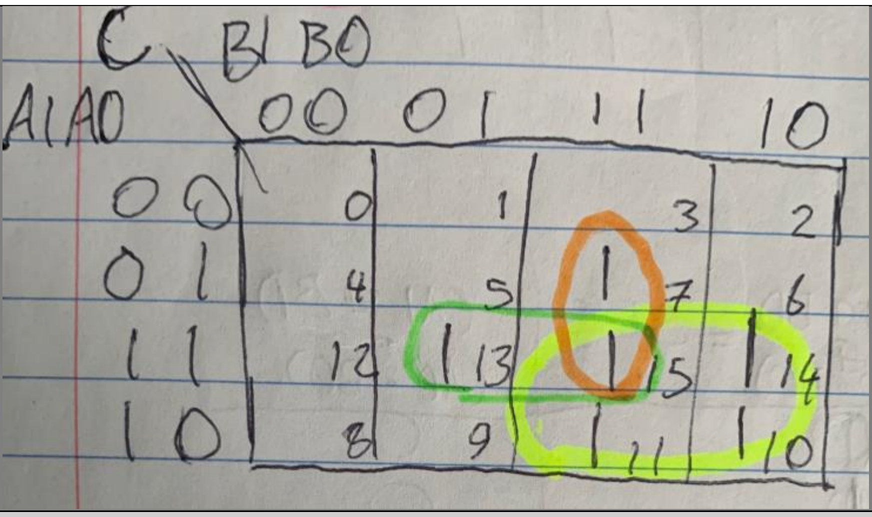
$X + Y = \overline{\bar{X}\,\bar{Y}}$

$$\therefore \quad = A_0B_0 + (\overline{\overline{\overline{A_0}}\,\overline{\overline{B_0}}})(\overline{A_1}B_1 + A_1\overline{B_1})$$

$X + (\bar{X}Y) = X + Y \; [where\ X = A_0B_0]$

$$\therefore \quad S_1 = A_0B_0 + (\overline{A_1}B_1 + A_1\overline{B_1}) \Rightarrow A_0B_0 + (A_1 \oplus B_1)$$

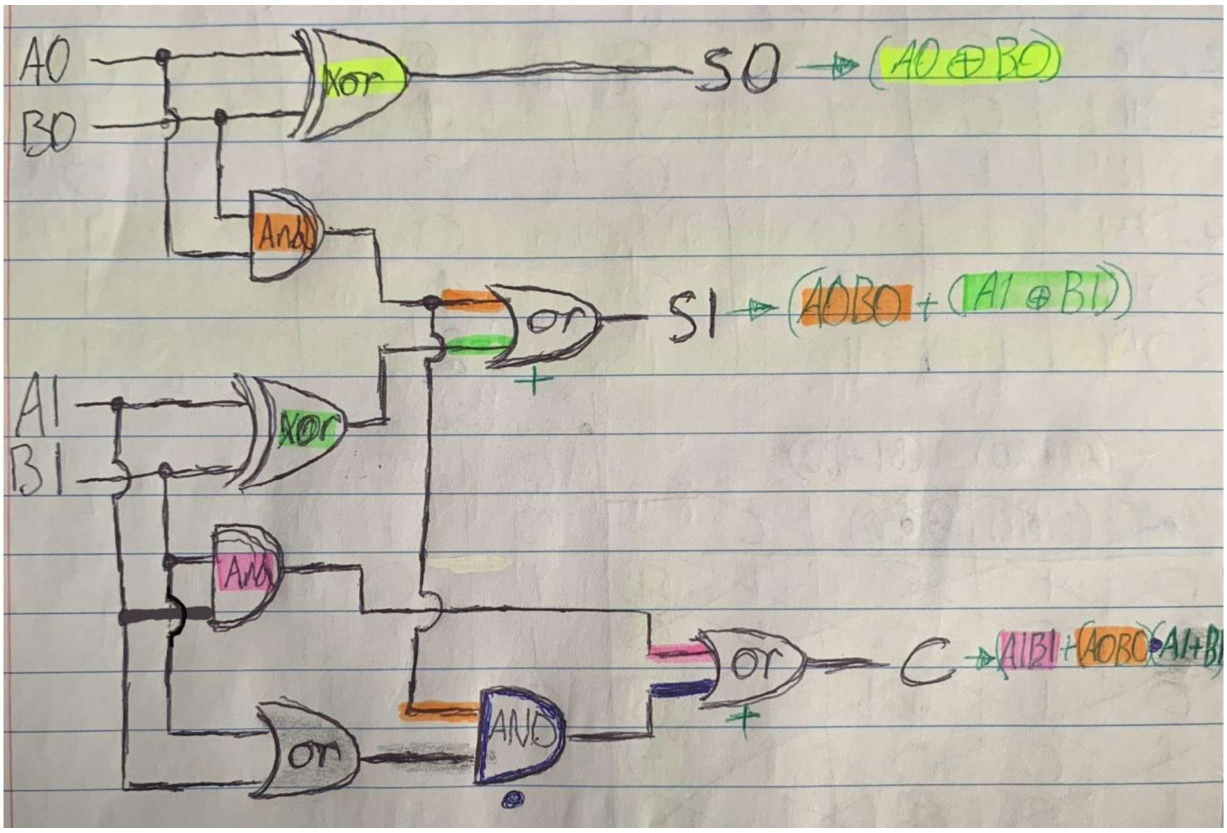
- ***C* Output**



$$C = A_0B_1B_0 + A_1A_0B_0 + A_1B_1 \Rightarrow A_1B_1 + A_0B_0(A_1 + B_1)$$

Outputs	Boolean Expressions
S_0	$A_0 \oplus B_0$
S_1	$A_0B_0 + (A_1 \oplus B_1)$
C	$A_1B_1 + A_0B_0(A_1 + B_1)$

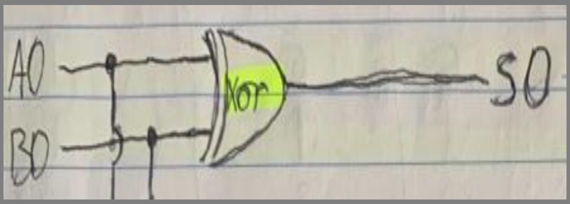
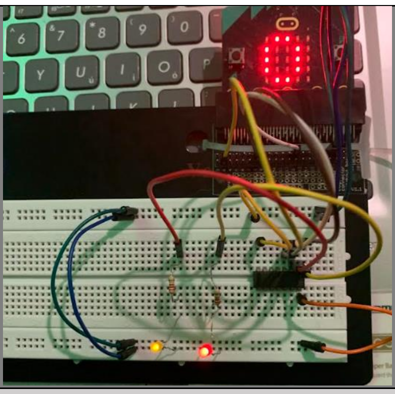
With the Boolean expressions for each output derived, a logic gate circuit which incorporates all these expressions can now be formed.



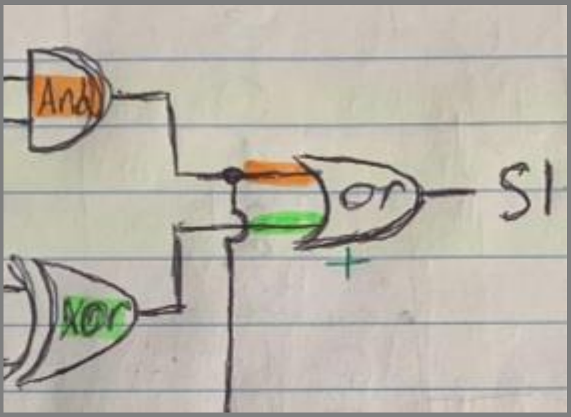
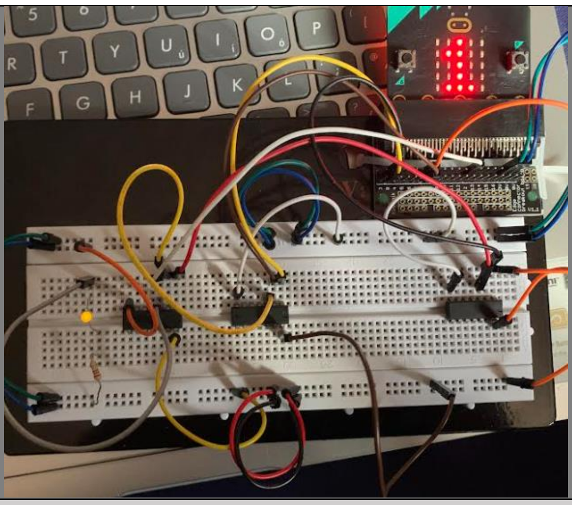
Physical Circuit Implementation

Due to the insufficient number of wires on hand, it was not possible to construct a full physical implementation of the logic gate circuit along with all its components. Consequently, three separate implementations were created, each showcasing a different output.

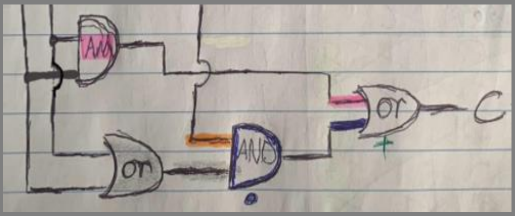
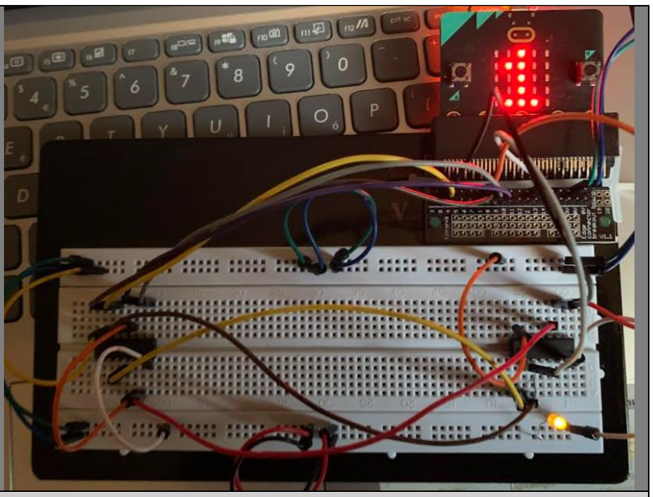
- **S_0 Output**

	
The state of inputs $A_0$and B_0 are automatically controlled by the microcontroller. The current state of each input displayed via LEDs – in this case both inputs are 1. These inputs are plugged into their respective positions on the XOR logic chip. The output is read by the microcontroller and is displayed on its screen.	

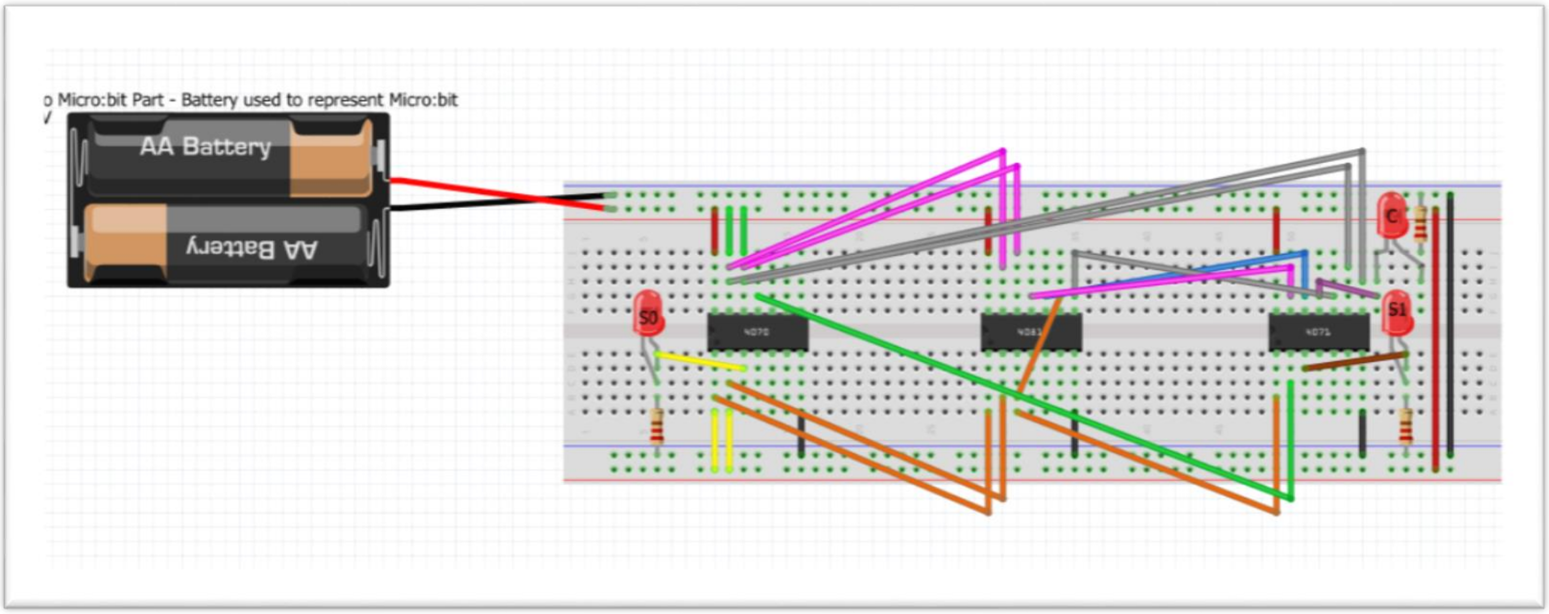
- **S_1 Output**

	
The state of inputs $A_0, B_0, A_1$and B_1 are automatically controlled by the microcontroller. $A_1$and B_1 are connected to their respective positions on the XOR logic chip (on the right). Its output is used as the first input on the OR logic chip (on the left). A_0 and B_0 are connected to their respective positions on the AND logic chip (in the middle). Its output is used as the second input on the OR logic chip. The output of the OR logic chip is read by the microcontroller and is displayed on its screen. An LED was also added to indicate and verify the current state of the output.	

- ***C* Output**

	
The state of inputs $A_0, B_0, A_{1(OR)}, B_{1(OR)}, A_{1(AND)}$and $B_{1(AND)}$ are automatically controlled by the microcontroller. $A_{1(OR)}$ and $B_{1(OR)}$ are connected to their respective positions on OR gate A (OR logic chip on the right). Its output is used as the first input of AND gate A (AND logic chip on the left). A_0 and B_0 are connected to their respective positions on AND gate B. Its output is used as the second input of AND gate A. The output of AND gate A is used as the first input on OR gate B. $A_{1(AND)}$and $B_{1(AND)}$ are connected to their respective positions on AND gate C. The output of AND gate C is used as the second input on OR gate B. The output of OR gate B is read by the microcontroller and is displayed on its screen. An LED was also added to indicate and verify the current state of the output.	

An electrical schematic was made using Fritzing in order to depict a complete physical implementation of the circuit containing all its outputs



Testing

The automation of testing for these circuits involves the continuous testing of all potential input combinations {[i.e. X_1, X_0] $(0,0);\ (0,1);\ (1,0);\ (1,1)$} This process is executed by the microcontroller, which then presents the culminating output of S_0, S_1 or C depending on the specific circuit built. The microcontroller code used for testing each output can be viewed in the appendix.

Noise Margins

According to the CD4070 XOR, CD4071 OR, and CD4081 AND datasheets the noise margin (Over Full Package Temperature Range) is as follows:

@ VDD = 5V	1V
@ VDD = 10V	2V
@ VDD = 15V	2.5V

The noise margin of the circuit exhibits a direct relationship with the VDD voltage. Specifically, as VDD increases, the noise margin also increases, and conversely, it decreases as VDD decreases. Given that the VDD applied to the physical implementation was set at 3V, it follows that the circuit's ideal noise margin should be less than 1V.

Discussion

As previously mentioned, due to a shortage of wires and other components, I had to divide the circuit into three distinct sections, conducting tests on each part independently. While this approach presented challenges, such as working within wire limitations and repeatedly dismantling and reconstructing the circuit, it ultimately deepened my comprehension of how each segment of the logic circuit works individually and would perform when integrated into a unified circuit.

- **S_0 Circuit**

The construction and testing of the circuit used for the S_0 output proved to be a straightforward task. This is due to the fact that it involved only two inputs connected to a single XOR logic gate chip, which would then generate the desired output, S_0. To validate the S_0 outputs, I used LEDs to provide an automatic visual representation of the state of each input, validating the generated S_0 outputs.

- **S_1 Circuit**

The construction and testing of the circuit used for the S_1 output was demanding in terms of wire management. Simultaneously managing the inputs and outputs of the AND, OR, and XOR logic gates quickly became complex. As I did not have enough LEDs, I could not check the state of the inputs and outputs of each gate to ensure everything was working as it should. Despite these difficulties, the circuit did manage to generate the expected S_1 outputs for all possible input combinations.

- **C Circuit**

The construction and testing of the circuit used for the C output presented the most complexity among the rest of the circuits. This was due to the need to automatically control three distinct sets of inputs using the microcontroller. The output of each input gate served as the input for another gate. As I progressed with the circuit construction, it became more challenging to manage and trace each set of inputs and their respective outputs. Testing followed the same methodology as previous circuits, with the microcontroller cycling through all input combinations and displaying the C output on its screen.

Final Bill of Materials

Component	Cost (€)*
CD4070 XOR Gate	1.79
CD4071 OR Gate	0.79
CD4081 AND Gate	0.40
BBC Micro:Bit V2 (Board Only)	23.90
Edge Connector Breakout Board	7.38
Velleman VTBB2 Solderless Breadboard	6.25
AWG Jumper Wires Set	4.95
Jumper Wires M/F	4.49
Resistor 22 Ω	1.35
LEDs (1 green + 1 red + 2 yellow)	0.64
USB Type-A to Micro-B USB Cable	4.98
Total Cost:	€56.92

(*prices found on https://www.irishelectronics.ie)

Conclusion

A truth table for natural number addition was used to create Boolean expressions through Karnaugh maps. These expressions were used to design a logic gate circuit. Due to limited resources, the circuit was divided into three sections for independent testing. $S_0{}'$s circuit was easily tested with a simple XOR gate and LEDs to verify outputs. S_1's circuit had more complex wire management and used more logic gate types but produced the expected outputs for all inputs. The C circuit was the most complex, requiring the simultaneous control of three input sets and interconnecting gate outputs as inputs. Despite challenges, all circuits successfully produced the desired outputs for all input combinations. In conclusion the division of the circuit into three parts allowed for an in-depth understanding of how the circuit would function when unified. This, in turn, demonstrates the viability of implementing a natural number addition function using a logic circuit.

References

1. *CD4070B, CD4077B datasheet (rev. E) - texas instruments india*. Available at: https://www.ti.com/lit/ds/symlink/cd4077b.pdf (Accessed: 17 October 2023).

2. *CD4071B Quad 2-input or Gate CD4072B dual 4-input or Gate CD4075B ...* Available at: https://www.ti.com/lit/ds/symlink/cd4071b.pdf (Accessed: 17 October 2023).

3. *CD4081B CD4081B data sheet, product information and support | TI.com*. Available at: https://www.ti.com/product/CD4081B (Accessed: 17 October 2023).

Appendix

S_0 Output Code

```
from microbit import *
# define pins to use for input and output
input_pins = [pin0, pin1]
output_pin = pin2

# set pins as inputs and output
for pin in input_pins:
    pin.write_digital(0)
    pin.set_pull(pin.PULL_DOWN)
    output_pin.write_digital(0)

# function to test XOR gate + display output
def test_xor_gate():
    for i in range(4):
        input_pins[0].write_digital(i & 1)
        input_pins[1].write_digital((i >> 1) & 1)
        sleep(100)
        result = output_pin.read_digital()
        display.show(result)
        sleep(1000)

while True:
    test_xor_gate()
```

S_1 Output Code

```
from microbit import *

and_gate_output = pin0
xor_gate_output = pin1
or_gate_output = pin2

and_gate_output.write_digital(0)
xor_gate_output.write_digital(0)
or_gate_output.write_digital(0)

# function to test OR gate with AND & XOR gate inputs + display output
def test_or_gate():
    for i in range(4):
        and_gate_output.write_digital(i & 1)
        xor_gate_output.write_digital((i >> 1) & 1)
        sleep(100)
        result = or_gate_output.read_digital()
        display.show(result)
        sleep(1000)
        display.clear()
        sleep(10)
while True:
    test_or_gate()
```

***C* Output Code**

```
from microbit import *

input_A1 = pin0
input_B1 = pin1
input_A0 = pin8
input_B0 = pin9

for pin in [input_A0, input_A1, input_B0, input_B1]:
    pin.write_digital(0)
    pin.set_pull(pin.PULL_DOWN)

output_OR = pin2

output_OR.set_pull(pin.PULL_DOWN)

# function to test gate inputs + display output
def test_or_gate():
    for i in range(4):
        input_A1.write_digital(i & 1)
        input_B1.write_digital((i >> 1) & 1)
        sleep(100)

        result_A0_B0 = input_A0.read_digital() & input_B0.read_digital()
        result_A1_B1 = input_A1.read_digital() & input_B1.read_digital()

        result = result_A0_B0 | result_A1_B1
        display.show(str(result))
        sleep(1000)
        display.clear()
        sleep(10)

while True:
  test_or_gate()
```