

Emran Yasser Moustafa – 20332041 – Digital Circuits Assignment 1

Introduction

In this report we were asked to build a circuit that implements a simple natural number adder. The functions that we were asked to implement can be found below :

$$S(A, B) = (A + B) \bmod N$$

$$C(A, B) = \begin{cases} 0, & A + B < N \\ 1, & \text{otherwise} \end{cases}$$

where :

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

The function $S(A, B)$ is the sum of two natural numbers. The carry of this operation is represented by the function $C(A, B)$. This function is 1 when a carry exists from the sum operation and 0 if not. The condition $N = 4$ means that both the input numbers and the sum can range from 0 to 3.

I constructed a two-bit adder to implement this function. I used a series of CMOS logic gates and an Arduino Nano 33 IoT to construct a half adder followed by a full adder. The Arduino was used to set the values of A and B . The Arduino was also used to read the output of the function; that being the sum, $S(A, B)$, and the carry, $C(A, B)$, of the operation.

My implementation of the function was successful and robust. I tested my circuit by manually setting the values of A and B , waiting a short period for the operation to complete and then reading the output values from the Arduino Serial Monitor. In all the test cases, the circuit output the correct value.

Logic

Each input, A and B , can be represented with a two bit binary number. Similarly, S can be represented as a two bit binary number also. Given the constraint $A + B < 4$, both inputs can range from zero up to three. The overflow of an operation like this can then be handled by the carry function $C(A, B)$.

In practice, this function can be seen as the addition two three bit numbers with a 0 in the most significant bit location. The output of this function can also be seen as a three bit number where the most significant bit is the carry, C , and S is the two other bits. This implementation can be seen in the logic table below.

A_1	A_0	B_1	B_0	C_0	S_1	S_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Figure 1 : Truth Table for $S(A, B)$ and $C(A, B)$

To implement this circuit on a logic level, I used a half adder and a full adder. A_0 and B_0 are the input to the half adder. The sum output of this logic module goes into S_0 while the carry is an input to the full adder, alongside A_1 and B_1 . The sum output of the full adder goes into S_1 while the carry out goes into C_0 . S_1 and S_0 represent the first and second bit of the $S(A, B)$ output. Similarly, C_0 is the one bit output of $C(A, B)$. A logic diagram for the function can be found below.

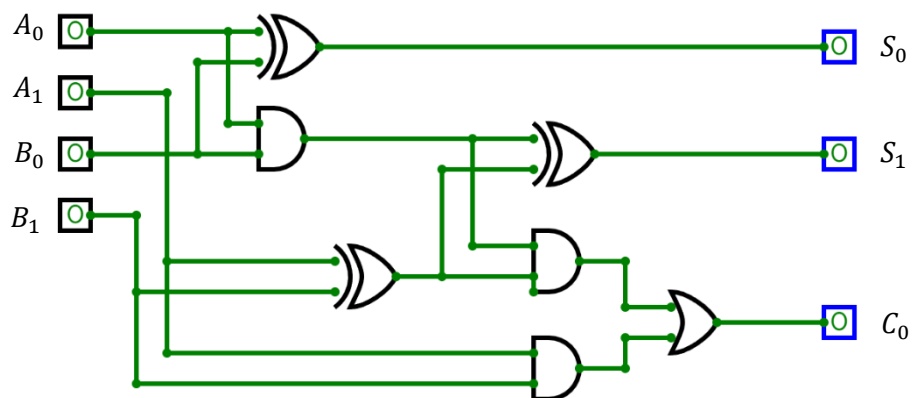


Figure 2 : Logic Circuit Diagram for $S(A, B)$ and $C(A, B)$

This circuit can be tested on a logic level on CircuitVerse via this link (<https://circuitverse.org/users/151118/projects/digital-circuits-assignment-1>) . The operation of this circuit is the same as the table found above.

Physical Implementation

I have implemented the function using an Arduino Nano 33 IoT, CD4070, CD 4071, CD4081 and a set of wires. I used the microcontroller to control the values of A and B as well as read the outputs of the function. I wired the circuit as shown in the diagram below. A circuit schematic can also be found below.

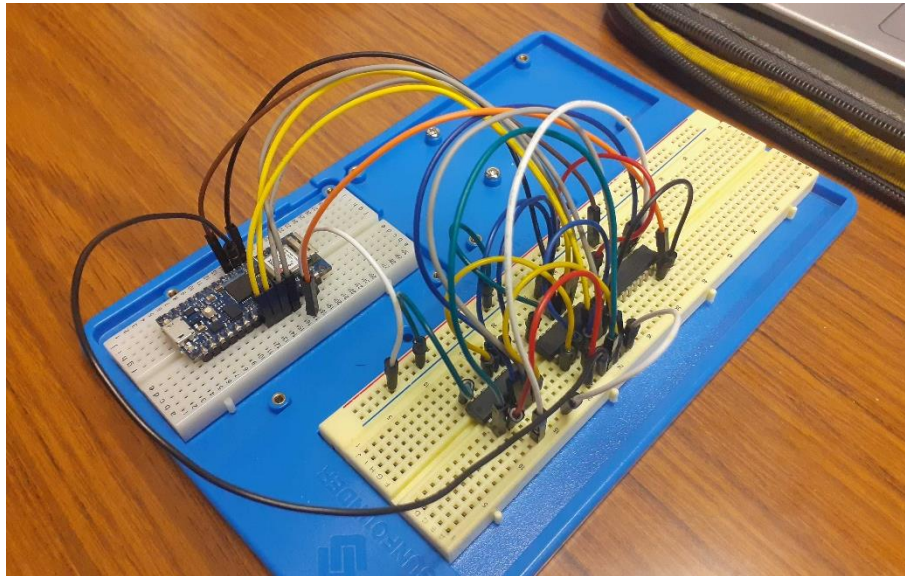


Figure 3 : Physical Implementation of $S(A, B)$ and $C(A, B)$

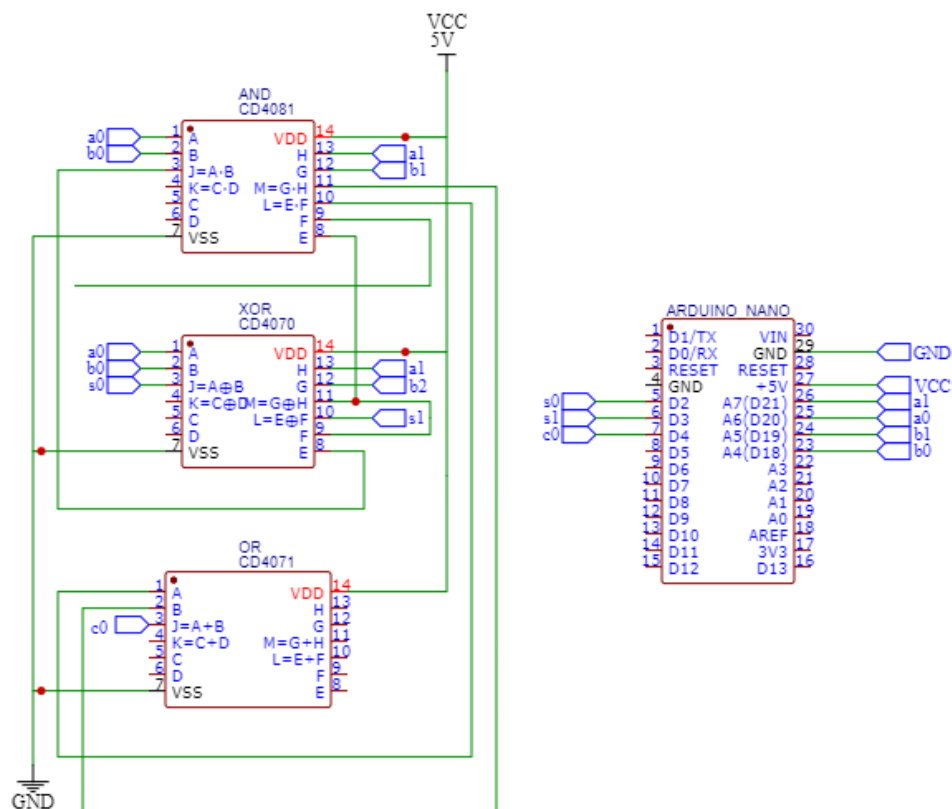


Figure 4 : Electrical Diagram of $S(A, B)$ and $C(A, B)$

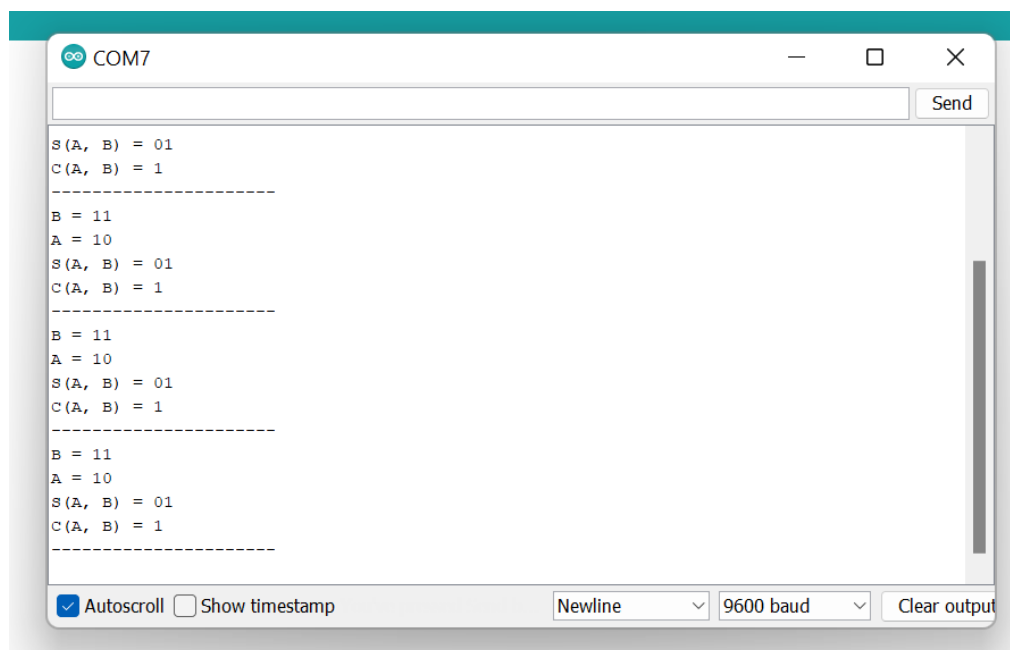
All gates have a noise margin of $1V$ at $5V$. The noise margins of these devices is the difference between the ICs valid logic outputs and inputs. The operation speed of the gates are slightly different however. The CD4080 and the CD4070 both have an operation time of $60ns$ at $10V$. However, the CD4071 has an operation time of $65ns$ at $10V$. To clarify, these values would be slightly different when supplied with $5V$, such as from the Arduino Nano 33 IoT. This issue is remedied however by including a delay when using the microcontroller to read the function output. In my implementation of this circuit, I added a delay of two seconds before reading the voltage on the input pins.

The cost of this implementation is relatively low. The microcontroller is the most expensive component by far. The ICs are roughly one euro each, however the Arduino 33 IoT is approximately forty to fifty euro. The cost of the wires is negligible.

Testing and Results

To test the circuit I wrote a short C script to be deployed on the Arduino Nano. This script used the Arduino's *digitalWrite* function to set the bits of A and B to on or off. Setting a pin to *HIGH* would represent a 1 and setting a pin to *LOW* would represent a 0. In this way the values for A and B could be set and passed into the function.

To read back the output I used the Arduino's *digitalRead* function. Again, the sum would be read back as two individual bits representing the two bit output $S(A, B)$. The one-bit carry, $C(A, B)$ would be read back the same way. To test the circuit, I would manually set the values of A and B and read the values of $S(A, B)$ and $C(A, B)$ off of the terminal. I used the Serial Monitor on the Arduino IDE to do this. The values would be printed in binary form. Below you will find an example of the output printed to the Serial Monitor.



```
COM7
S(A, B) = 01
C(A, B) = 1
-----
B = 11
A = 10
S(A, B) = 01
C(A, B) = 1
-----
B = 11
A = 10
S(A, B) = 01
C(A, B) = 1
-----
B = 11
A = 10
S(A, B) = 01
C(A, B) = 1
-----
```

Autoscroll ☐ Show timestamp Newline 9600 baud Clear output

Figure 5 : Serial Monitor output

To ensure the operation time of the gates doesn't affect the output of the function, I added a slight delay to the end of the *loop()* function. This delay ensures that the logic gates have all switched to the correct value before reading the output of the circuit.

The test script I used can be found in the following GitHub repo (<https://github.com/lilpharaoh1/3c2-assignment1>), labelled assignment1.ino.

Conclusion

To summarise, in this report I have documented my implementation of the functions $S(A, B)$ and $C(A, B)$. I did this by constructing a two bit adder, this being a half adder followed by a full adder. I implemented this function using off-the-shelf components; an Arduino Nano 33 IoT and a series of CMOS logic gates. I used a total of three gates to implement the function, coming to a total cost of roughly three euro.

In terms of possible work to be done in the future, there are a number of small and large changes that could be done to improve my implementation of the function. Some form of automated testing could be performed when evaluating the circuit. As stated above, to validate the operation of the function I manually altered the inputs and checked the values against a truth table. This could be automated such that the values are changed automatically and the script checks if the output of the circuit is as expected, perhaps outputting a pass or fail flag for the function.

Another viable improvement would be to match the calling of the *loop()* function to the operation time of the slowest gate. This would be done by including some sort of clock into the test script so as to ensure the output is being read only when all values are set correctly.

The circuit could also be implemented using an Field Programmable Gate Array. The FPGA would reconfigure itself. So as to synthesise the function into logic, using the devices logic blocks. The benefit of this would be very low power consumption per inference as well as extremely quick operation time. Using an FPGA however seem like overkill for such a simple function. Perhaps if the complexity of the function increased, an FPGA would provide a viable alternative to these CMOS logic gates.