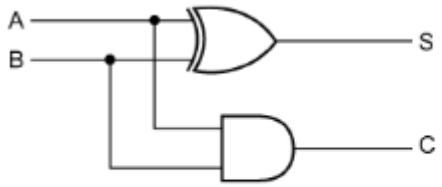
3.3.1 Logic gates and circuit design

In section 1.3.3, we talked about Logic gates and the different outputs they give when attached in a circuit. In this section, we will further discuss Logic gates and introduce a new concept of **Half adders** and **Full adders**.

Half-Adder

- A *half-adder* is a combinational arithmetic circuit block that can be used to add 2 bits. Such a circuit thus has 2 inputs that represent the 2 bits to be added and 2 outputs, with one producing the **SUM** output and the other producing the **CARRY** output.



a) Circuit Design

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

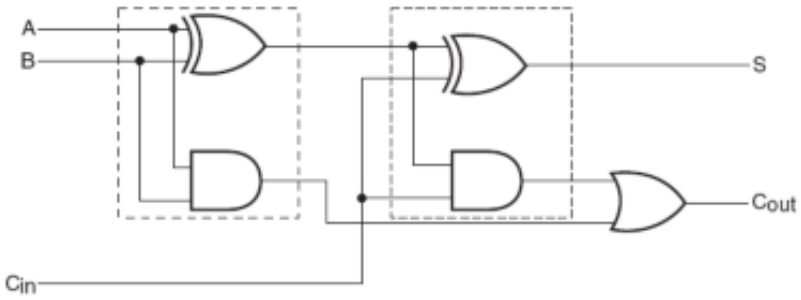
b) Truth Table

- Half-adders are the simplest of all adder circuits, **but it has a major disadvantage**. The half adder can add only 2 input bits (A and B) and has nothing to do with the carry if there is any as the input. So if the input to a half adder has a carry, then it will be neglected. Hence the binary addition process is not complete and that's why it is called a **HALF** adder

3.3.1 Logic gates and circuit design

Full-Adder

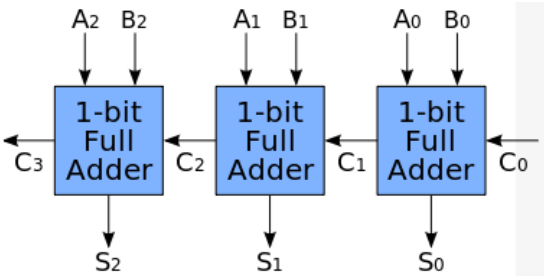
- A *full-adder* circuit is a combinational arithmetic circuit block that can be used to add **3 bits** to produce a **SUM** and **CARRY** output.
- Becomes a necessity when it comes to adding binary numbers with a large number of digits.
- Overcomes the limitation of the half-adder, which could only add 2 bits.
- While adding larger binary numbers, we also consider the carry bit from the last 2 binary numbers.
- A *full-adder* is therefore essential for the hardware implementation of an adder circuit capable of adding larger binary numbers.



Notice that a full-adder consists of 2 half-adders

A	B	CARRY (INPUT)	CARRY (OUTPUT)	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

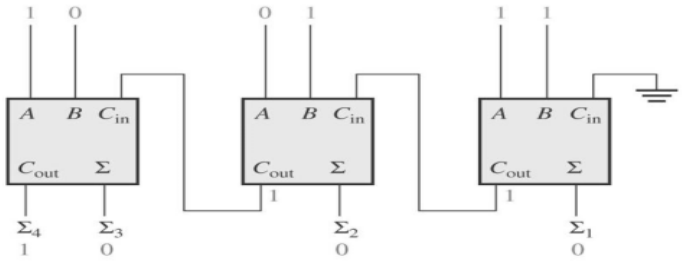
Therefore, you can concatenate one bit adders together, creating multiple bit adders (in this case 3 bit):



3.3.1 Logic gates and circuit design

Example: Determine the sum generated by the 3-bit parallel adder.

Add 101 + 011.



Result => 101 + 011 = 1000.

- Starting from the right side, the **CARRY INPUT** is going to be 0 because initially, there is no carry from before! (Hence grounded)
- The circuit starts by adding the LSB's of the 2 numbers 10**1** **(A)** and 01**1** **(B)** and adding 1+1 gives you "0" **(SUM)** and "1" **(CARRY OUTPUT)**
- The carry produced from the first full-adder, will be fed as **CARRY INPUT** into the second full-adder.
- The circuit then adds the next pair of numbers + the carry input (if any)
  1**0**1 **(A)** + 0**1**1 **(B)** + **1** **(CARRY INPUT)** adding 0+1+1 gives you "0" **(SUM)** and "1" **(CARRY OUTPUT)**
- Finally, the circuit adds the MSB's + the carry input (if any)
  101 **(A)** + **0**11 **(B)** + **1** **(CARRY INPUT)** adding 1+0+1 gives you "0" **(SUM)** and "1" **(CARRY OUTPUT)**
- The last carry generated will be ignored and regarded as **"OVERFLOW"**

So the answer can be written as **(1)000** and addition is complete**.**

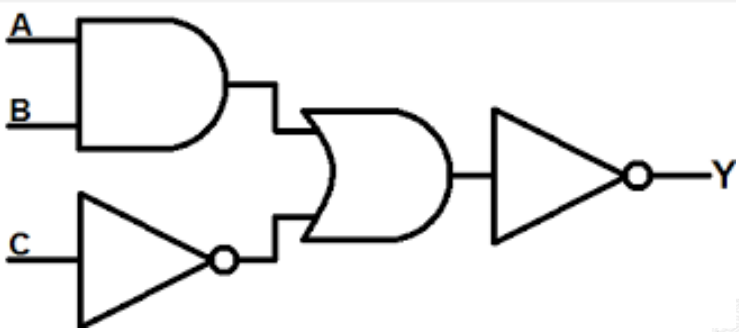
A truth table can also be generated to make calculations easier!

A	B	CARRY (INPUT)	CARRY (OUTPUT)	SUM
1 (LSB)	1 (LSB)	-	1	0
0	1	1	1	0
1 (MSB)	0 (MSB)	1	1 (Overflow)	0

3.3.1 Logic gates and circuit design

Example 1:

Derive a truth table for the given logic circuit



ANSWER:

INPUTS			OUTPUT
A	B	C	Y
1	1	1	0
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

Example 2:

Derive a truth table for the given logic circuit

ANSWER:

INPUTS			OUTPUT
A	B	C	Y
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1