EXPERIMENT-4

Objective: To study the half/full adder/subtractor.

Resources Required:

Half/full adder/subtractor verification kit

Theory:

A Half-Adder

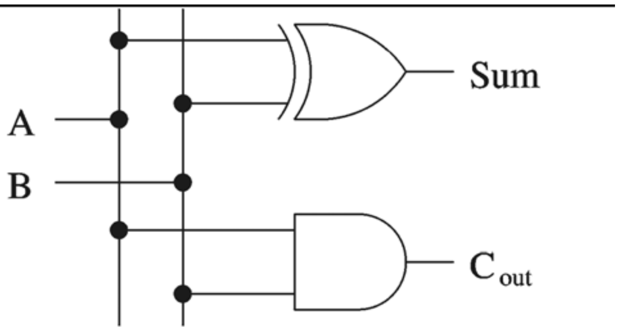
As a first example of useful combinational logic, let's build a device that can add two binary digits together. We can quickly calculate what the answers should be:

0 + 0 = 0   0 + 1 = 1   1 + 0 = 1   1 + 1 = 10

So we well need two inputs (a and b) and two outputs. The low order output will be called Σ because it represents the sum, and the high order output will be called Cout because it represents the carry out.

The truth table is

A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(a) Half-adder truth table and implementation

Simplifying boolean equations or making some Karnaugh map will produce the same circuit shown below, but start by looking at the results. The Sum column is our familiar XOR gate, while the Cout column is the AND gate. This device is called a half-adder for reasons that will make sense in the next section.

A Full-Adder:

The half-adder is extremely useful until you want to add more that one binary digit quantities. The slow way to develop a two- binary digit adder would be to make a truth table and reduce it. Then when you decide to make a three binary digit adder, do it again. Then when you decide to make a four-digit adder, do it again. Then when ... The circuits would be fast, but development time would be slow.

Looking at a two binary digit sum shows what we need to extend addition to multiple binary digits.

11

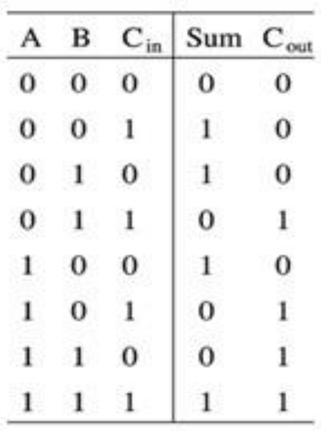
11

110

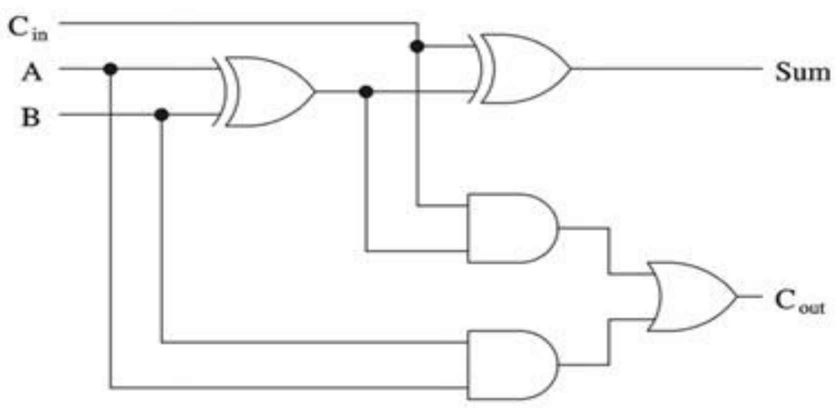
Look at how many inputs the middle column uses. Our adder needs three inputs; a, b, and the carry from the previous sum, and we can use our two-input adder to build a three input adder.

Σ is the easy part. Normal arithmetic tells us that if Σ = a + b + Cin and Σ1 = a + b, then Σ = Σ1 + Cin.

In order to calculation the high order bit, notice that that it is 1 in both case when a+b producer a C_1. Also, the high order bit is 1 when a+b produces a sum and C_{in} is a 1. So we will have a carry when C_1 OR (Sum AND C_{in}). Our complete three input adder is:



A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(b) Full-adder truth table and implementation

For some designs, being able to eliminate one or more types of gates can be important, and you can replace the final OR gate with an XOR gate without changing the results.

A Half-Subtractor:

A half subtractor is a combinational circuit that subtracts two bits and produces their difference. It also has an o/p to specify if a1 has been borrowed. Designate the minuend bit by X and the subtrahend bit by Y. to perform X-Y we have three possibilities 0-0=0,1-0=1,0-1=1,1-1=0.the half subtractor needs two o/p's. One o/p generates the difference and will be designed by the symbol D. The second o/p designated by B for borrow, generates the binary signal that informs the next stage that 1 has been borrowed.

Expression for half subtractor:

Difference= X'Y+XY' Borrow= X'Y

Truth Table: -

INPUT		OUTPUT	
X	**Y**	**Difference**	**Borrow**
0	0	0	0
0 1	1 1	0	1 0
1	0	1	0

A Full- Subtractor:

A Full Subtractor is a combinational circuit that performs a subtraction between two bits; taking into account that a1 may have been borrowed by a lower significant stage. This circuit has two inputs and two outputs. The three inputs, Y and Z, denotes the minuend, subtrahend and previous borrow respectively. The two outputs and B represents the difference and output borrow respectively.

Expression for full subtractor:

Difference= X'Y'Z+X'YZ'+XY'Z'+XYZ

Borrow = X'Y+YZ+ZX'

Truth Table: -

INPUT			OUTPUT	
X	**Y**	**Z**	**Difference**	**Borrow**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Result: The operation of half adder, half subtractor, full adder, full subtractor has been verified.

Precautions: -

1. Connection should be tight.
2. O/P should be finding sequentially.