EEE 3101: Digital Logic and Circuits

Adders & subtractors

Course Teacher: Nafiz Ahmed Chisty

Head, Department of EEE
Associate Professor, Department of EEE & CoE
Faculty of Engineering
Room# D0105, D Building
Email: chisty@aiub.edu
Website: http://engg.aiub.edu/faculties/nafiz
Website: www.nachisty.com

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

BASIC ADDERS

Adders are important in computers and also in other types of digital systems in which numerical data are processed. An understanding of the basic adder operation is fundamental to the study of digital systems. In this section, the half-adder and the full-adder are introduced.

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

Digital System Design Concept

Designing a digital system requires the following steps to be performed:

Topics covered in DLD

- Look for the problem statement
- Find out the inputs & outputs of the system
- Relate the inputs of your system with the outputs
- Develop a truth table/behavior table for your system using the input and output relationship that you have established
- From your truth table generate a standard output expression, relating the inputs to the output
- Reduce the output expression using either Boolean Algebra or k-MAP
- Write down your reduced expression (minimized expression)
- Design the system circuit with combinational logic gates
- **Convert the combinational logic gates to transistor level schematic (CMOS schematic)**

Topic covered in VLSI Circuit Design Course

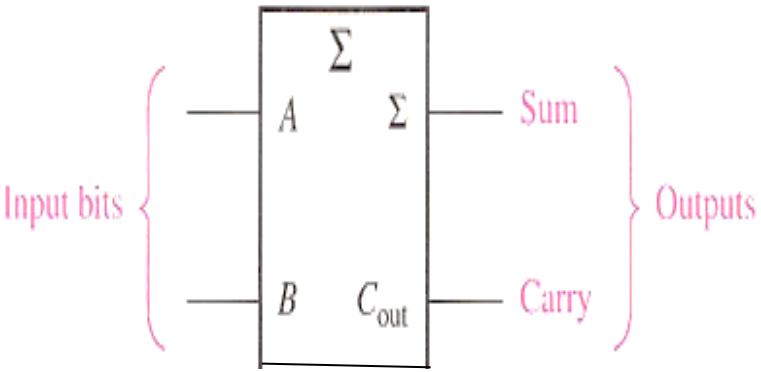
Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

The Half-Adder

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

A half-Adder accepts adds two bits and produces a Sum and a Carry output

Logic Symbol:



Truth Table:

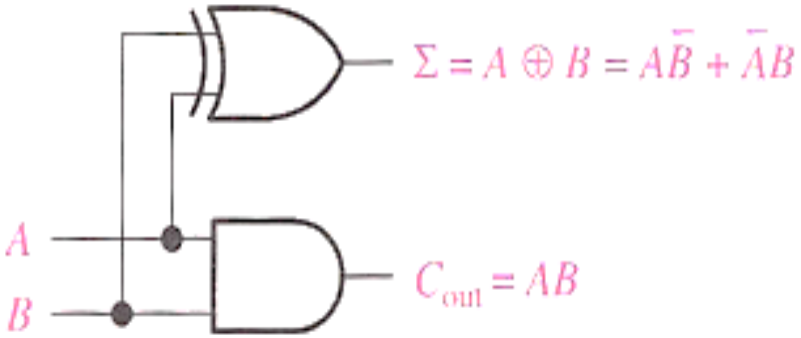
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum

C_{out} = output carry

A and B = input variables (operands)

Circuit Diagram:



Expression:

$$C_{out} = AB$$

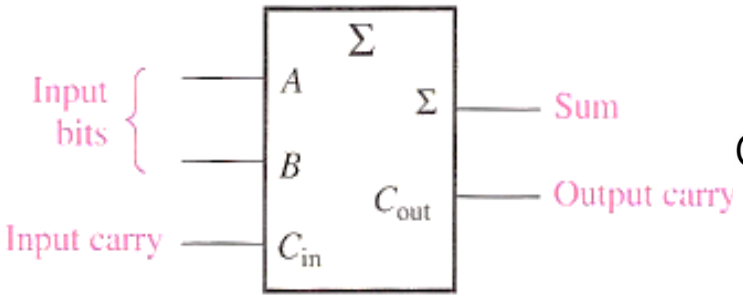
$$\Sigma = A \oplus B$$

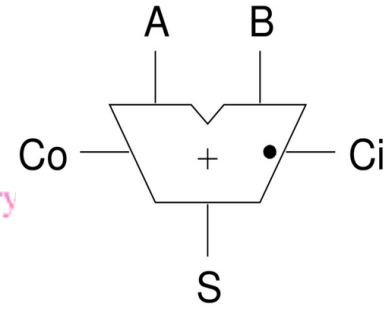
Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

The Full-Adder

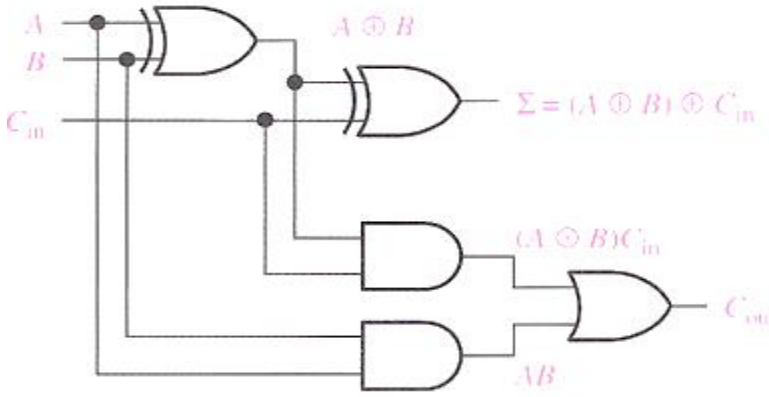
The Full-Adder accepts two input bits and an input carry and generates a sum output and an output carry.

Logic Symbol:





Circuit Diagram:



(b) Complete logic circuit for a full-adder (each half-adder is enclosed by a shaded area)

Expression:

$$\Sigma = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + (A \oplus B)C_{in}$$

Truth Table:

A	B	C_{in}	C_{out}	Σ
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

C_{in} = input carry, sometimes designated as *CI*

C_{out} = output carry, sometimes designated as *CO*

Σ = sum

A and *B* = input variables (operands)

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

The Full-Adder implementation with Half-Adders

Half Adder Expression:

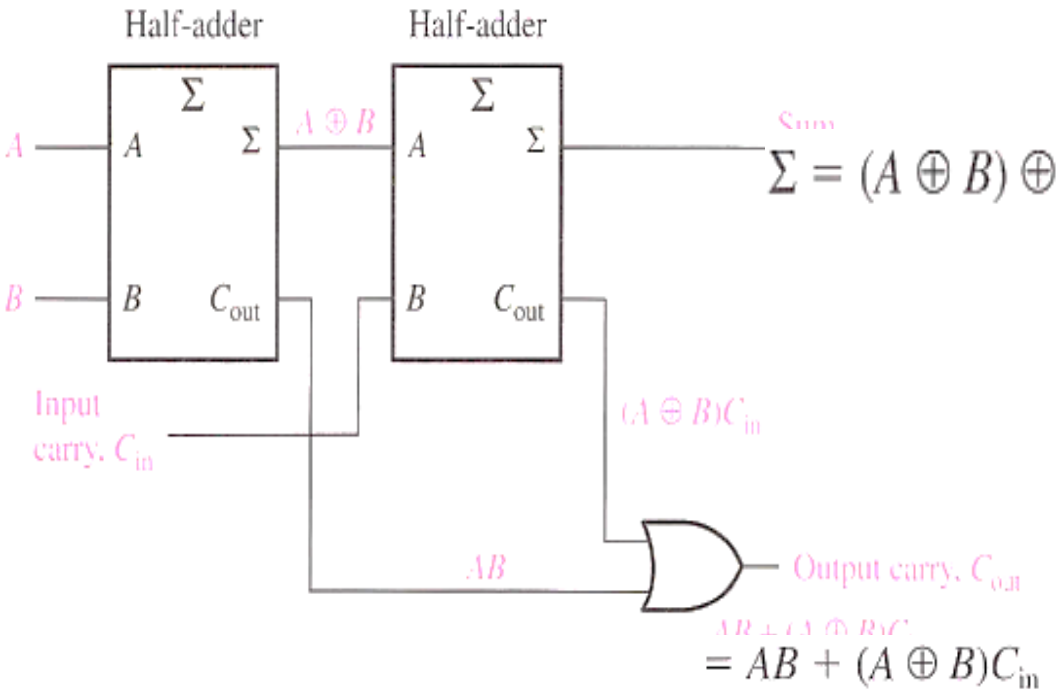
$$C_{out} = AB$$

$$\Sigma = A \oplus B$$

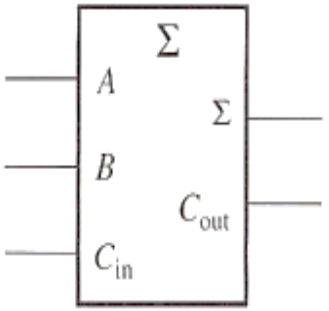
Full Adder Expression:

$$\Sigma = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + (A \oplus B)C_{in}$$



(a) Arrangement of two half-adders to form a full-adder



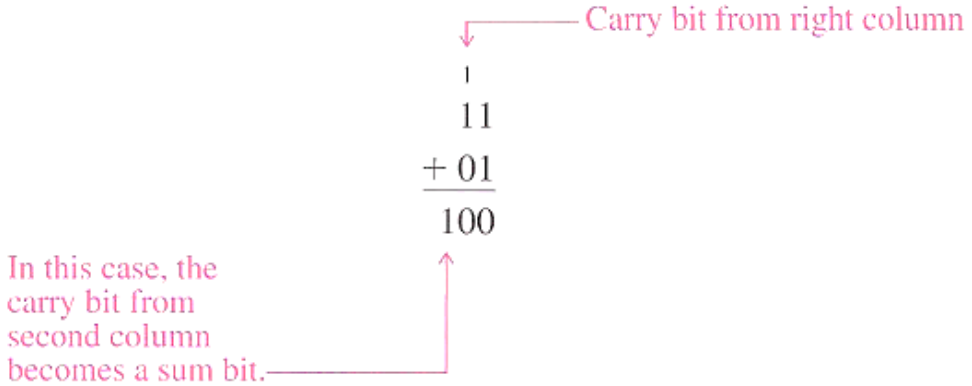
(b) Full-adder logic symbol

PARALLEL BINARY ADDERS

Two or more full-adders are connected to form parallel binary adders.

A single full-adder is capable of adding two 1 bit numbers and an input carry. To add binary numbers with more than one bit, additional full-adders are required.

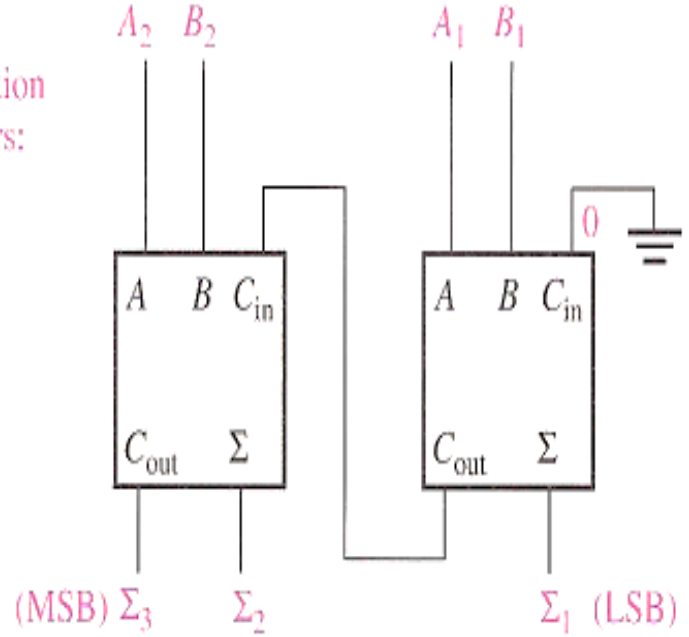
Example:



To add two binary numbers, a full-adder is required for each bit in the numbers. So for 2-bit numbers, two adders are needed; for 4-bit numbers, four adders are used; and so on. The carry output of each adder is connected to the carry input of the next higher-order adder.

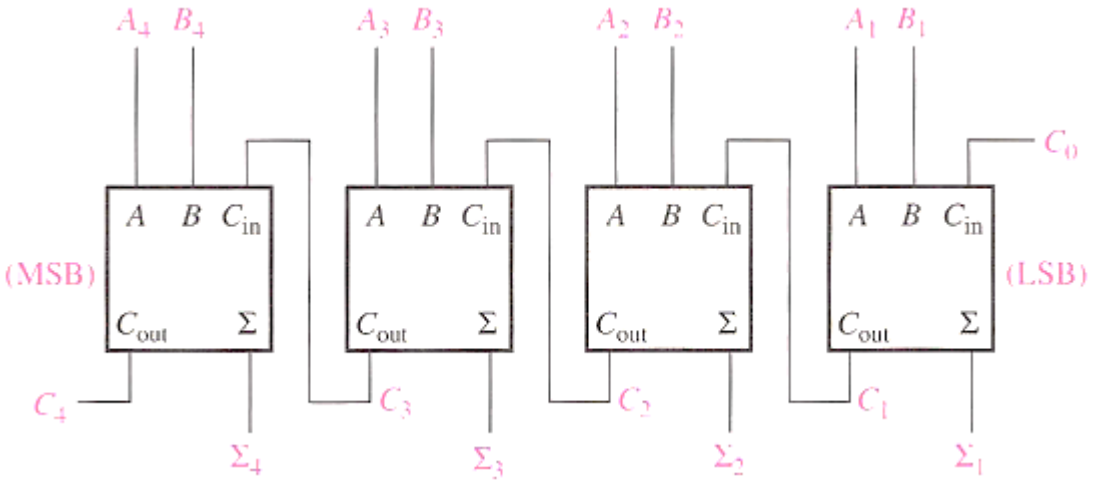
General format, addition of two 2-bit numbers:

$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$

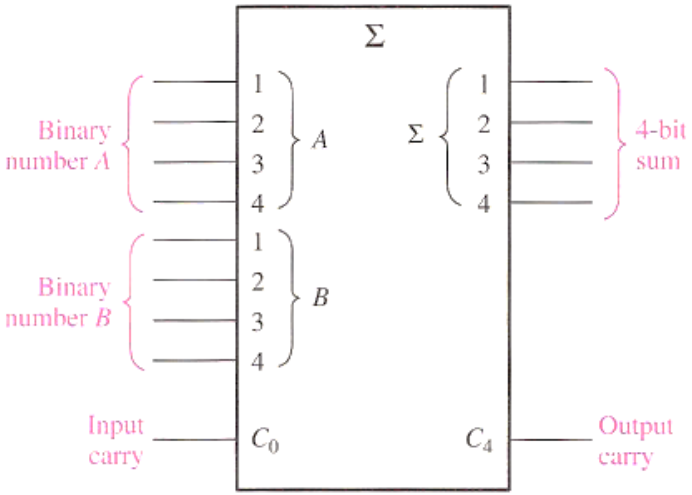


Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

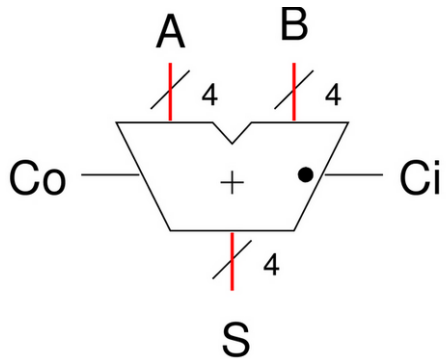
Four-Bit Parallel Adders



(a) Internal connection



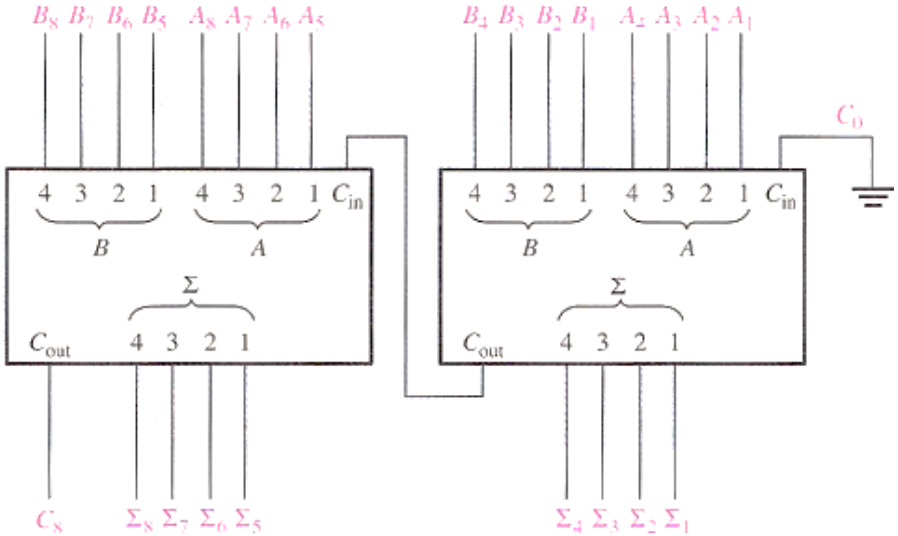
(b) Logic symbol



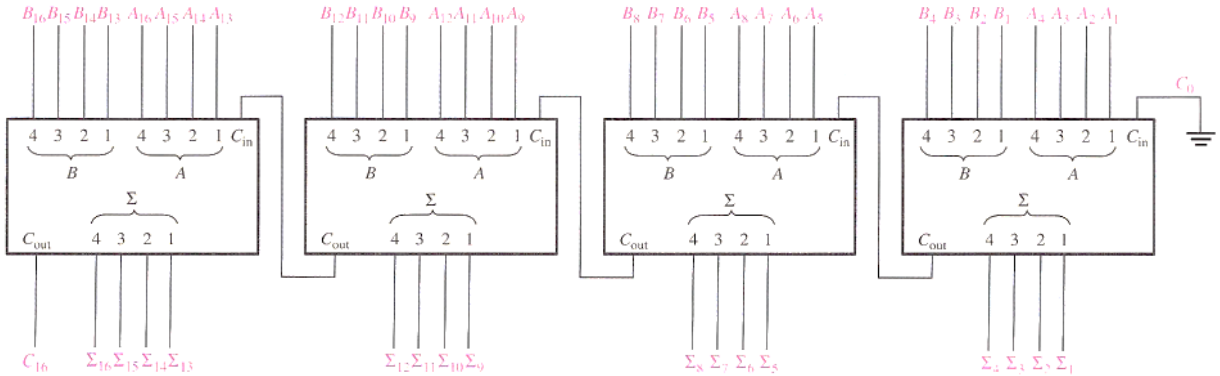
Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

Adder Expansion

The 4-bit parallel adder can be expanded to handle the addition of two 8-bit numbers by using two 4-bit adders.



(a) Cascading of two 4-bit adders to form an 8-bit adder



(b) Cascading of four 4-bit adders to form a 16-bit adder

Reference:

[1] Thomas L. Floyd, "Digital Fundamentals" 11th edition, Prentice Hall.
[2] M. Morris Mano, "Digital Logic & Computer Design" Prentice Hall.

Thanks


AIUB
EEE
FACULTY OF ENGINEERING
AIUB
Faculty of Engineering

Binary Subtractor:

a) To perform the subtraction A-B, we can use the 2's complements of B, so the subtraction can be converted to addition. Because, A - B = A + (-B)

b) 2's complement can be obtained by talking the 1's complement and adding 1 to the LSD bit.

   1)1's complement can be implemented with inventors.

   2)1 can be added to the sum through the input carry.

The circuit for subtracting A-B consists of an adder with inverters placed between each data input B and the corresponding input of the full adder. The input carry C_0 must be equal to 1.

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

Binary Adder–Subtractor

The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full-adder. In order to convert a ripple–carry adder into a Subtractor, we employ the standard algebra trick: A – B is the same as A + (–B). In order to subtract B from A, it is necessary to negate B to produce –B, and then to add that number to A.

We now have to develop a circuit that will negate a binary value.
In order to do this, we must stipulate the method used to represent signed integers. As in earlier lecture, we used **two's–complement** notations.

In this method, in order to negate a binary integer, it is necessary to produce the two's–complement of that value.

a) First, take the one's–complement of the binary integer, and then
b) Add 1 to that value.

We have to develop a circuit to create the one's–complement of a binary integer. We shall develop two circuits to do this.

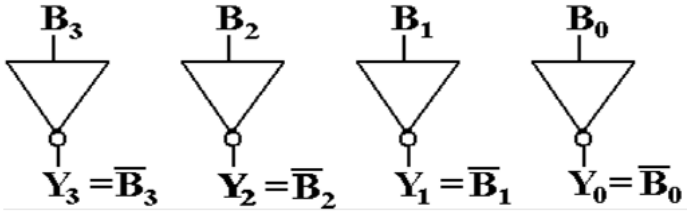
Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

i) The One's Complement of a Binary Integer

In order to take the one's–complement of an integer in binary form, just change every 0 to a 1, and every 1 to a 0. Here are some examples.

Original value	0110 0111	1010 0011
One's complement	1001 1000	0101 1100

The circuit that does this conversion is the NOT gate. The circuit below would be used for four–bit integers.



If the input is $B_3B_2B_1B_0 = 0110$, the output is $Y_3Y_2Y_1Y_0 = 1001$.

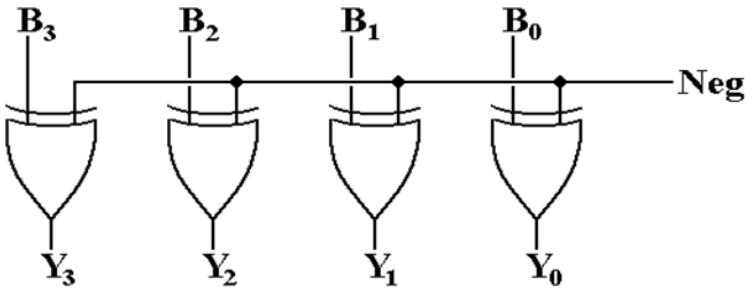
This circuit can be extended to any number of bits required.

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

Optional

ii) The XOR Gate as a NOT Gate

In order to make an adder/subtractor, it is necessary to use a gate that can either pass the value through or generate its one's–complement. The exclusive OR gate, XOR, is exactly what we need.



If Neg = 0 Then $Y_3 = B_3, Y_2 = B_2, Y_1 = B_1$, and $Y_0 = B_0$

If Neg = 1 Then $Y_3 = \overline{B}_3, Y_2 = \overline{B}_2, Y_1 = \overline{B}_1$, and $Y_0 = \overline{B}_0$

This is controlled by a single binary signal: **Neg**

Let B = 1011. If Neg = 0, then Y = 1011. If Neg = 1, then Y = 0100.

iii) Some Notation for Bit–Wise Operations

Taking the one's–complement of a binary integer involves taking the one's–complement of each of its bits. We use the NOT notation to denote the one's–complement of the entire integer.

If $B = B_3B_2B_1B_0$ is a four–bit binary number, its one's–complement is

$$\overline{B} = \overline{B}_3\,\overline{B}_2\,\overline{B}_1\,\overline{B}_0$$

Remember that to get the two's–complement, one takes the one's–complement and adds one. In two's–complement arithmetic we have:

$$-B = \overline{B} + 1$$

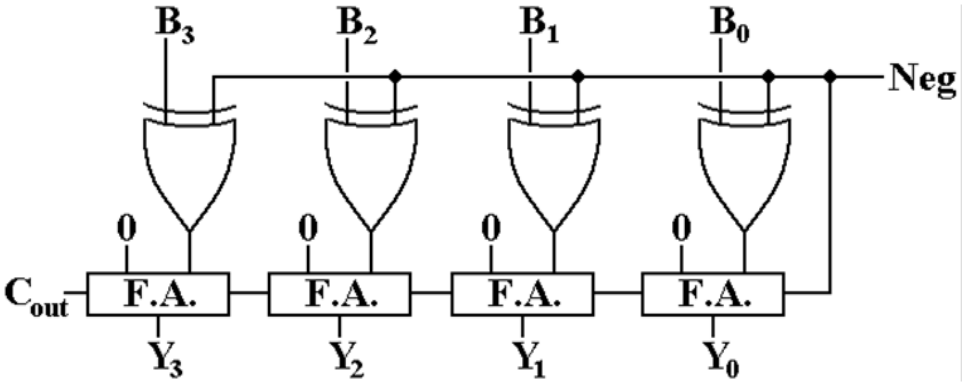
This suggests the use of an adder to produce the negative. For $B = B_3B_2B_1B_0$, we take the one's complement to start the negation process. The addition of 1, necessary to take the two's–complement, is achieved by setting the carry–in of the unit's full–adder to 1.

This is the reason for using a full–adder in the unit's position. We can alternate between an adder and subtractor.

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

iv) The Negator Circuit

Here is the circuit to produce the negative of a 4–bit number $B = B_3B_2B_1B_0$.



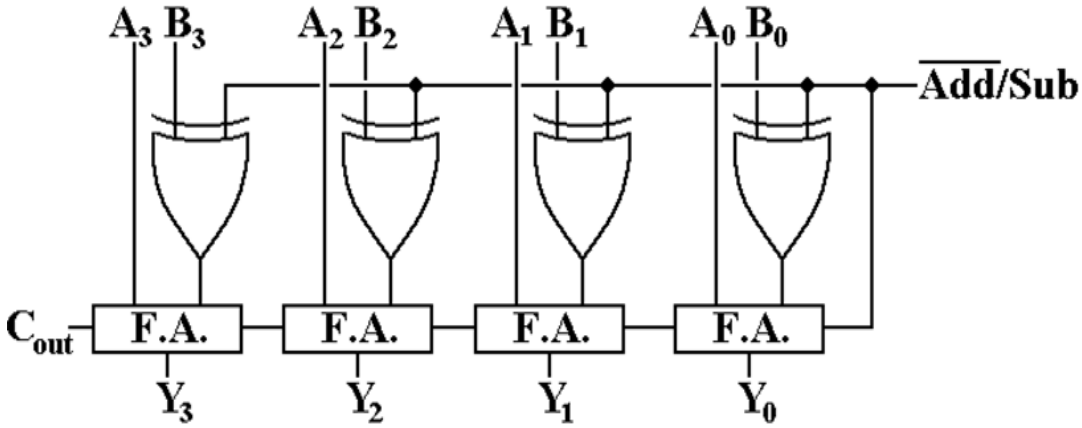
If Neg = 0, then the XOR gates pass the values $B_3B_2B_1B_0$ unchanged and the ripple–carry adder adds 0 to that. The result is Y = B.

If Neg = 1, then the XOR gates cause the one's–complement of $B = B_3B_2B_1B_0$ to be computed. Note that the carry–in of the full adder is set to 1, so that it adds 1 to the one's–complement, thus producing the negative: Y = – B.

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE

v) The Full Adder/Subtractor

We now have the circuit that either adds or subtracts. This is a 4–bit circuit that produces either (A + B) or (A – B).



$\overline{\text{Add/Sub}}$

This notation is used for a two–valued control signal. When the value is 0, the circuit is to add, when it is 1 the circuit is to subtract.

This adder/subtractor can be extended to any number of binary bits.

Nafiz A. Chisty| Head, Dept of EEE; Associate Professor, Dept. of EEE & CoE, FE