Binary Arithmetic

- Addition
- Half and Full Adders
- Subtraction

decimal	hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

decimal addition

tens units

$$\begin{array}{r} 60 \\ +\ 43 \\ \hline 103 \\ \scriptstyle 1 \end{array}$$

hex addition

sixteens units

$$\begin{array}{r} 3C \\ +\ 2B \\ \hline 67 \\ \scriptstyle 1 \end{array}$$

C + B in decimal is **12 + 11 = 23**
This contains one lot of **16** with **7** units left over.
The sixteen is added into the sixteens column.

Binary Addition

Any time the sum of a column equals or exceeds the radix, a 1 is carried forward to the next column.

$$\begin{array}{r} 0111_2 \\ +\ 1110_2 \\ \hline 10101_2 \end{array} \qquad \begin{array}{r} 7_{10} \longleftarrow \text{augend} \\ +\ 14_{10} \longleftarrow \text{addend} \\ \hline 21_{10} \end{array}$$

decimal	hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Example with checking in hex:

$$\begin{array}{r} 0011\ 1100_2 \\ +\ 0010\ 1011_2 \\ \hline 0110\ 0111_2 \end{array} \qquad \begin{array}{r} 3C_{16} \\ +\ 2B_{16} \\ \hline 67_{16} \end{array}$$

Binary addition examples (check with hex).

(a)

$$\begin{array}{r} 1011_2 \\ +\ 0101_2 \\ \hline \end{array}$$

(b)

$$\begin{array}{r} 0111_2 \\ +\ 1000_2 \\ \hline \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ +\ 1111_2 \\ \hline \end{array}$$

Binary addition examples (check with hex).

(a)

$$\begin{array}{r} 1011_2 \\ +\ 0101_2 \\ \hline 10000_2 \end{array} \qquad \begin{array}{r} B_{16} \\ +\ 5_{16} \\ \hline 10_{16} \end{array}$$

(b)

$$\begin{array}{r} 0111_2 \\ +\ 1000_2 \\ \hline \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ +\ 1111_2 \\ \hline \end{array}$$

Binary addition examples (check with hex).

(a)

$$\begin{array}{r} 1011_2 \\ +\ 0101_2 \\ \hline 10000_2 \end{array} \qquad \begin{array}{r} B_{16} \\ +\ 5_{16} \\ \hline 10_{16} \end{array}$$

(b)

$$\begin{array}{r} 0111_2 \\ +\ 1000_2 \\ \hline 1111_2 \end{array} \qquad \begin{array}{r} 7_{16} \\ +\ 8_{16} \\ \hline F_{16} \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ +\ 1111_2 \\ \hline \end{array}$$

Binary addition examples (check with hex).

(a)

$$\begin{array}{r} 1011_2 \\ +\ 0101_2 \\ \hline 10000_2 \end{array} \qquad \begin{array}{r} B_{16} \\ +\ 5_{16} \\ \hline 10_{16} \end{array}$$

(b)

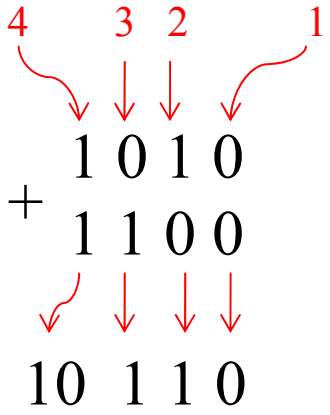
$$\begin{array}{r} 0111_2 \\ +\ 1000_2 \\ \hline 1111_2 \end{array} \qquad \begin{array}{r} 7_{16} \\ +\ 8_{16} \\ \hline F_{16} \end{array}$$

(c)

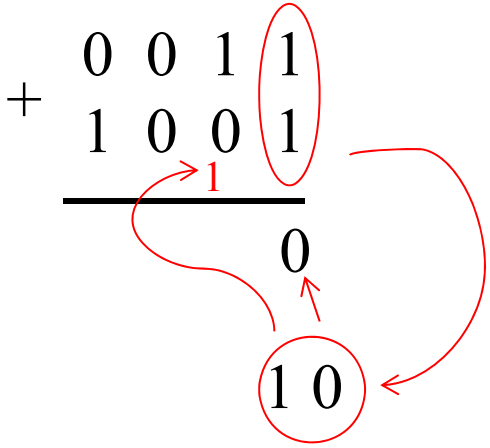
$$\begin{array}{r} 1111_2 \\ +\ 1111_2 \\ \hline 11110_2 \end{array} \qquad \begin{array}{r} F_{16} \\ +\ F_{16} \\ \hline 1E_{16} \end{array}$$

Column Addition of Bits

There are four possibilities when adding two bits. Cases 1,2,3 produce a single bit result. Case 4 produces a two bit result known as the sum and carry.
Two bits can be added using a **half adder**.



The carry is added into the next column. This means that we must be able to add three bits in a column to accommodate an incoming carry bit.

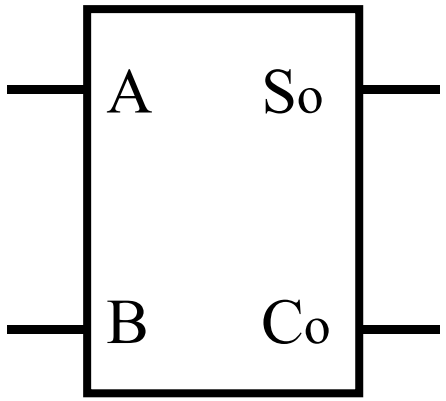


This is done with a **full adder**.

The Half Adder

The half adder has two binary input bits which it adds to produce a sum bit and a carry bit.

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



Truth table

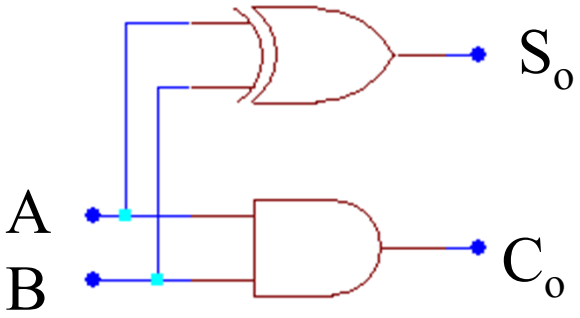
A	B	C_o	S_o
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Logic Equations

$$C_o = A.B$$

$$S_o = A \oplus B$$

Diagram



XOR of three variables

$$F = X \oplus Y \oplus Z$$

$$= X \oplus (Y.\overline{Z} + \overline{Y}.Z)$$

$$= X.\overline{(Y.\overline{Z} + \overline{Y}.Z)} + \overline{X}.(Y.\overline{Z} + \overline{Y}.Z)$$

$$= X.(\overline{Y}.\overline{Z} + Y.Z) + \overline{X}.(Y.\overline{Z} + \overline{Y}.Z)$$

F	G	XOR
0	0	0
0	1	1
1	0	1
1	1	0

F	G	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

The Full Adder

The full adder has two binary input bits and an input carry which it adds to produce a sum bit and a carry bit.

A	B	C	C_o	S_o
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

Intuitively … $S_o = A \oplus B \oplus C$

$C_o = A.B + (A \oplus B).C$

Alternatively ...

$$S_O = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.C$$

$$S_O = (\overline{A}.\overline{B} + A.B).C + (\overline{A}.B + A.\overline{B}).\overline{C}$$

$$S_O = \overline{(A \oplus B)}.C + (A \oplus B).\overline{C}$$

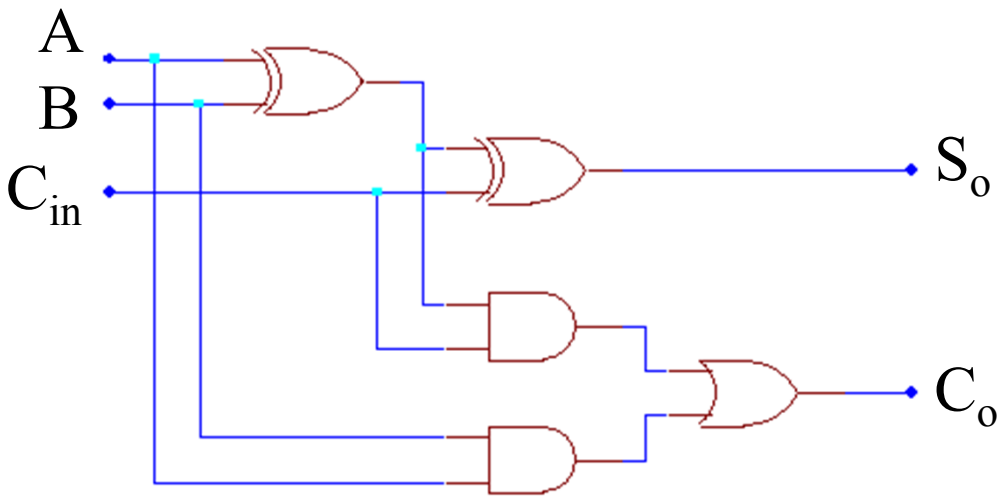
$$S_O = A \oplus B \oplus C$$

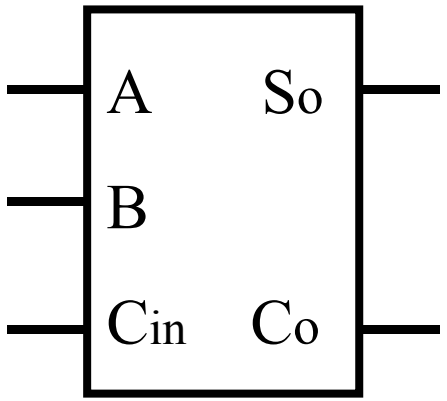
$$C_O = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$$

$$C_O = (\overline{A}.B + A.\overline{B}).C + A.B.(C + \overline{C})$$

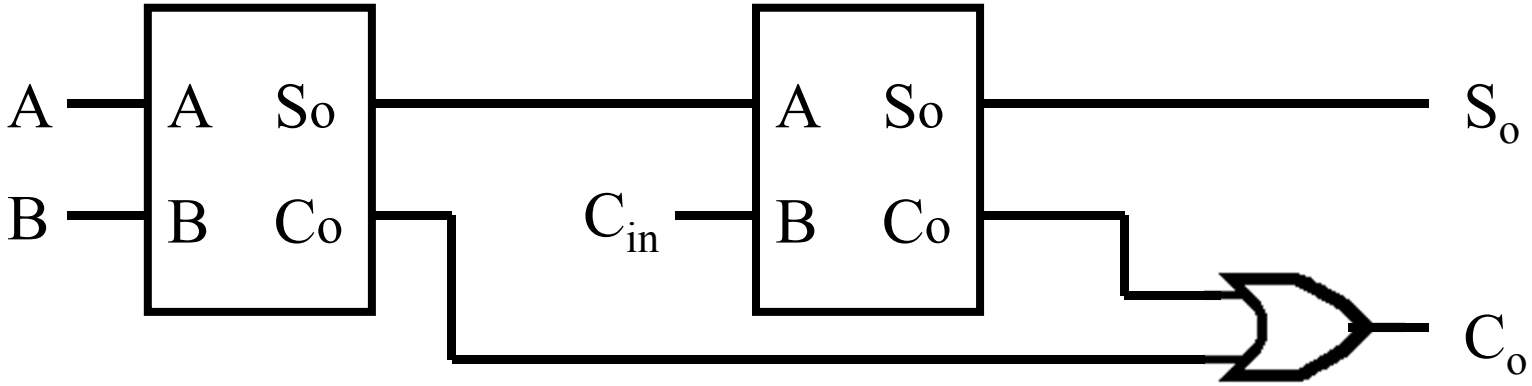
$$C_O = (A \oplus B).C + A.B$$

Full Adder Schematic and Symbol



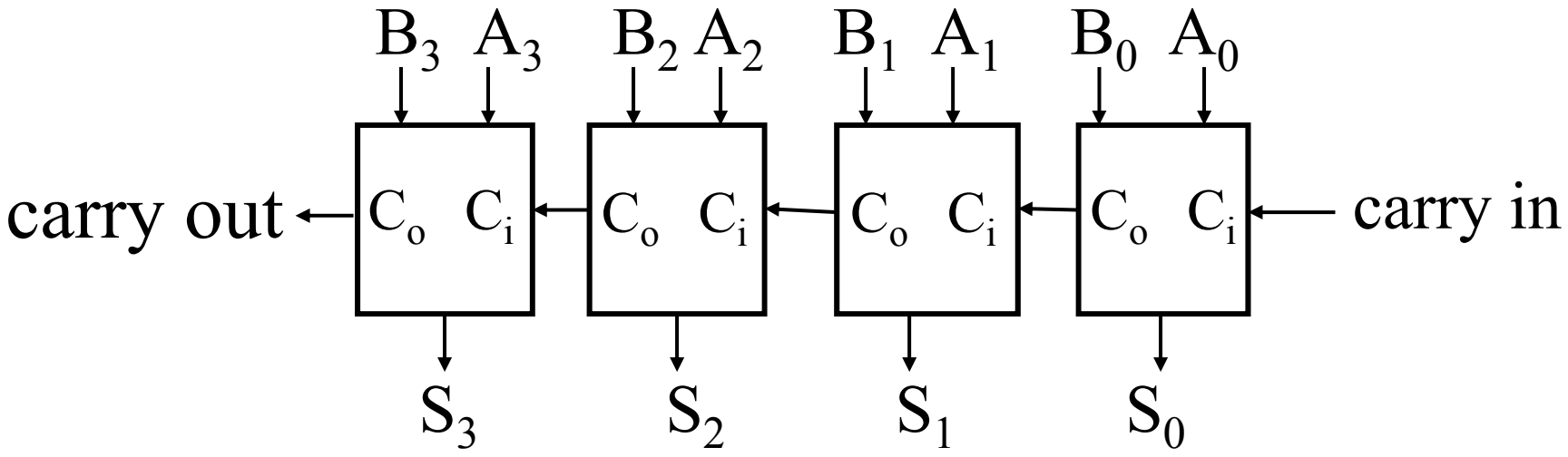


Alternatively, with half adders and an OR gate.



Parallel Binary Adders

To add two four bit numbers, four full adders are connected as shown.



This is also known as a ripple carry adder because of the way the carry signal is passed along, or ripples from one adder to the next.

Its main disadvantage is that the final sum output bit may depend on a carry which has rippled the entire length of the adder.

Overflow

The addition of the two four bit numbers shown has produced a five bit result.

This is not a problem for a pencil and paper calculation, but is a problem for digital hardware.

$$\begin{array}{r} 1011_2 \\ +\,0101_2 \\ \hline (1)0000 \end{array}$$

When the addition of two n-bit numbers gives an (n + 1) bit result, we say that an overflow has occurred.

In a digital computer, the number of bits that hold the result is finite and an (n + 1) bit result cannot be accommodated for n bit arithmetic.

In this case, an overflow is detected by the carry out of the adder at the most significant bit position.

Subtraction

$$\begin{array}{rl} & 875_{10} \\ - & 329_{10} \\ \hline & 546_{10} \end{array}$$

this is equivalent to 800 - 300

70 - 20

5 - 9

The problem is 5 - 9. We overcome this by taking 10 from the 70 and adding it to the 5 giving:

800 - 300 = 500

60 - 20 = 40

15 - 9 = 6

500 + 40 + 6 = 546

We are borrowing from the next highest power. The same method can be used for binary, but in this case we borrow a 2 from the next highest bit position.

Binary Subtraction

The rules for binary subtraction are similar to the rules for decimal subtraction.

If the subtrahend digit is greater than the minuend digit in a column, then a 2 is borrowed from the next column.

$$\begin{array}{rl} {}^{2}1001_2 & 9_{10} \leftarrow \text{minuend} \\ -\,{}_{1}0100_2 & -\,4_{10} \leftarrow \text{subtrahend} \\ \hline 0101_2 & 5_{10} \end{array}$$

Binary subtraction examples:

(a)

$$\begin{array}{r} 1011_2 \\ -\ 0101_2 \\ \hline \end{array}$$

(b)

$$\begin{array}{r} 1000_2 \\ -\ 0111_2 \\ \hline \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ -\ 1101_2 \\ \hline \end{array}$$

Binary subtraction examples:

(a)

$$\begin{array}{r} 1011_2 \\ -\ 0101_2 \\ \hline 0110_2 \end{array} \qquad \begin{array}{r} B_{16} \\ -\ 5_{16} \\ \hline 6_{16} \end{array}$$

(b)

$$\begin{array}{r} 1000_2 \\ -\ 0111_2 \\ \hline \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ -\ 1101_2 \\ \hline \end{array}$$

Binary subtraction examples:

(a)

$$\begin{array}{r} 1011_2 \\ -\ 0101_2 \\ \hline 0110_2 \end{array} \qquad \begin{array}{r} B_{16} \\ -\ 5_{16} \\ \hline 6_{16} \end{array}$$

(b)

$$\begin{array}{r} 1000_2 \\ -\ 0111_2 \\ \hline 0001_2 \end{array} \qquad \begin{array}{r} 8_{16} \\ -\ 7_{16} \\ \hline 1_{16} \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ -\ 1101_2 \\ \hline \end{array}$$

Binary subtraction examples:

(a)

$$\begin{array}{r} 1011_2 \\ -\ 0101_2 \\ \hline 0110_2 \end{array} \qquad \begin{array}{r} B_{16} \\ -\ 5_{16} \\ \hline 6_{16} \end{array}$$

(b)

$$\begin{array}{r} 1000_2 \\ -\ 0111_2 \\ \hline 0001_2 \end{array} \qquad \begin{array}{r} 8_{16} \\ -\ 7_{16} \\ \hline 1_{16} \end{array}$$

(c)

$$\begin{array}{r} 1111_2 \\ -\ 1101_2 \\ \hline 0010_2 \end{array} \qquad \begin{array}{r} F_{16} \\ -\ D_{16} \\ \hline 2_{16} \end{array}$$

Summary

- Binary addition and subtraction follow the same rules as decimal but with a radix of 2.
- The addition of two or more 1s will form a carry.
- In subtraction we borrow a 2 if the subtrahend bit is greater than the minuend.
- A full adder can be used to add two bits plus the carry in from a previous stage. It will produce a sum and a carry out.