Computer Organization and Architecture
CEC 470

Module 02 :
Arithmetic and Logic Part 1
(Ch 10, Ch 11, Ch 12)

Last Module Key Ideas:

- Stored program computer/Von neumann architecture
- 5 components of computer
- ISA
- Computer architecture
- Moore's law
- Performance equations
- Amdahl’s law

This Module:

- Number system (decimal, binary, hexadecimal)
- Signed and unsigned integer representation
- Two’s complement
- Half adder
- Full adder
- Carry look ahead adder

Review…

Information → Data & Program

Data → Numeric & Non-numeric (names, address, etc.)

Numeric → Integers & Floating point

Integers → Signed integers & unsigned integers

Review: bits, bytes and nibbles…

- Bits:
  (8-bit binary)

  1 0 0 1 0 1 1 0

  most significant bit (MSB) — least significant bit (LSB)

- Bytes & Nibbles:
  (8-bit binary)

  byte (8 bits)

  1 0 0 1 0 1 1 0

  nibble (4 bits)

- Bytes:
  (32-bit hex)

  3 A C F 2 4 D 7

  MSbyte — LSbyte

Review: powers of 2…

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$
- $2^{16} = 65536$
- handy to ***memorize up to 2^{10}***

Review question?

a) If we have 2 bits word, how many possible values we have? Write all them

b) If we have 3 bits word, how many possible values we have? Write all values

c) If we have 4 bits word, how many possible values we have? Write all values

Number system

Decimal ($base_{10}$)

$$A = \sum_{i=0}^{n-1} a_i . 10^i$$

10^2	10^1	10^0
1	**5**	**7**

$= (1x100)+(5x10)+(7x1)$

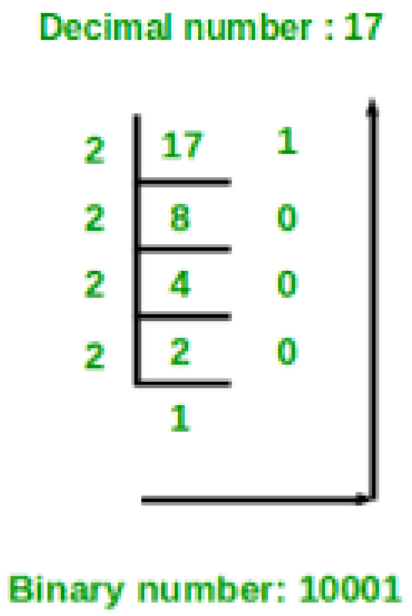
Binary ($base_2$)

$$A = \sum_{i=0}^{n-1} a_i . 2^i$$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	**0**	**0**	**1**	**1**	**1**	**0**	**1**

$=128+ 0+ 0 +16 +8 + 4 + 0 + 1 =157_{10}$

Decimal to binary



Binary to decimal

Position 4 … Position 0

$$10001 = 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0$$
$$= 16 + 1 = 17$$

Example 1: convert binary to decimal

Convert the following binary sequence to decimal:

a) 1 0 1 0 1 0 1 1
b) 1 0 0 0 0 1 0 1

Range of binary numbers

- N-digit decimal number
  - –How many values?
  - –Range?
  - –Example: 3-digit decimal number:

- N-bit binary number
  - –How many values?
  - –Range:
  - –Example: 3-bit binary number:

Hexadecimal numbers

- For humans, its clumsy to always work in binary
  - Just too many bits!
- Divide a binary number into 4-bit groupings and represent each 4-bits by a single hexadecimal ($base_{16}$) digit/symbol.

Binary:	0010	1001	0101	0111
Hex:	2	9	5	7

- But, in hexadecimal, each digit can have a value of 0 -15_{10} !!
- We need new symbols to represent the values 10_{10}–15_{10}
- Use symbols A, B, C, D, E and F

Hexadecimal numbers

- It is more compact than binary notation
- In most computers, binary data occupy some multiple of 4 bits, and hence some multiple of a single hexadecimal digit
- It is extremely easy to convert between binary and hexadecimal notation

Example 2: convert binary string to hexadecimal

Convert the following binary sequence to hexadecimal notation:

a) 1 1 1 1 1 1 1 1 1 1 1 1
b) 1 0 0 1 1 1 0 0 1 0 0 0
c) 0 0 1 0 1 1 0 0 1 0 1 1

Unsigned integer

- Positive or nonnegative numbers
- If we have a *3 bit word, 2^3= 8 values*
- Range is 0-7

0/1 0/1 0/1

___ ___ ___

* Note: / represents or

n-bits binary sequence

$$\boxed{A = \sum_{i=0}^{n-1} a_i . 2^i}$$

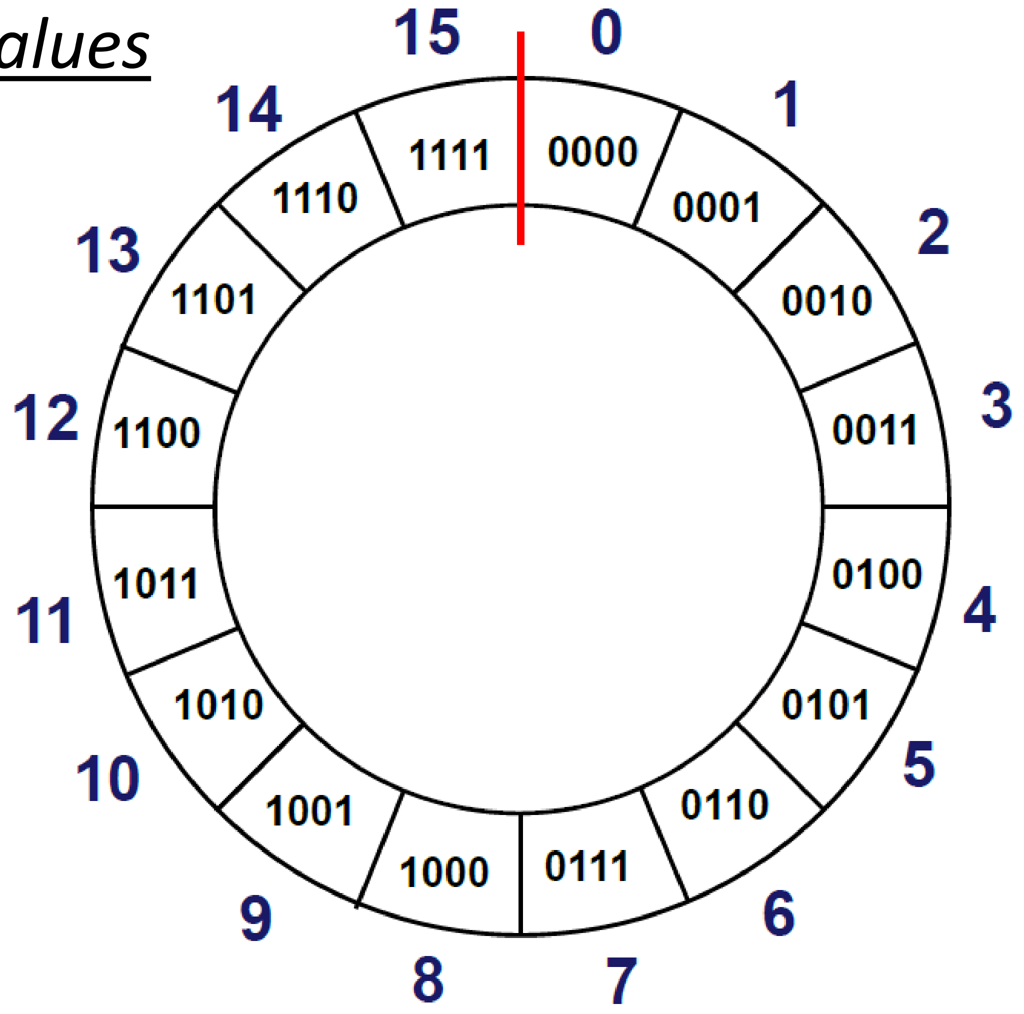
Unsigned integer

bit

Binary	Decimal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Unsigned integer wheel

4 bit word, 2^4= 16 values



Discontinuity at limits of numerical representation (0 and 15)

Question?

What would be the range of unsigned integer for 8 bit word?

Signed integer representation

Three common approaches to deal with *negative numbers*:

1. Signed magnitude number
2. One's compliment
3. Two’s compliment

1. Sign-magnitude representation

- One sign bit plus n-1 magnitude bits
- MSBit is the sign bit:
  - MSB=0 means positive number
  - MSB=1 means negative number

$$A = (-1)^{a_{n-1}} \times \sum_{i=0}^{n-2} a_i . 2^i$$

- *for example, for n=8:*

0 0 0 1 0 1 1 1

= +1 x (0 + 0 +16 +0 + 4 + 2 + 1) = 23

1 0 0 1 0 1 1 1

= –1 x (0 + 0 +16 +0 + 4 + 2 + 1) = – 23

- n-bit sign-magnitude number can take on values $-(2^{n}\quad)$ to $(2^{n-1}-1)$

Problems with sign-magnitude

1. Addition doesn't work
   – for example, 4-bit addition of $(-5)\ and\ (+2)$

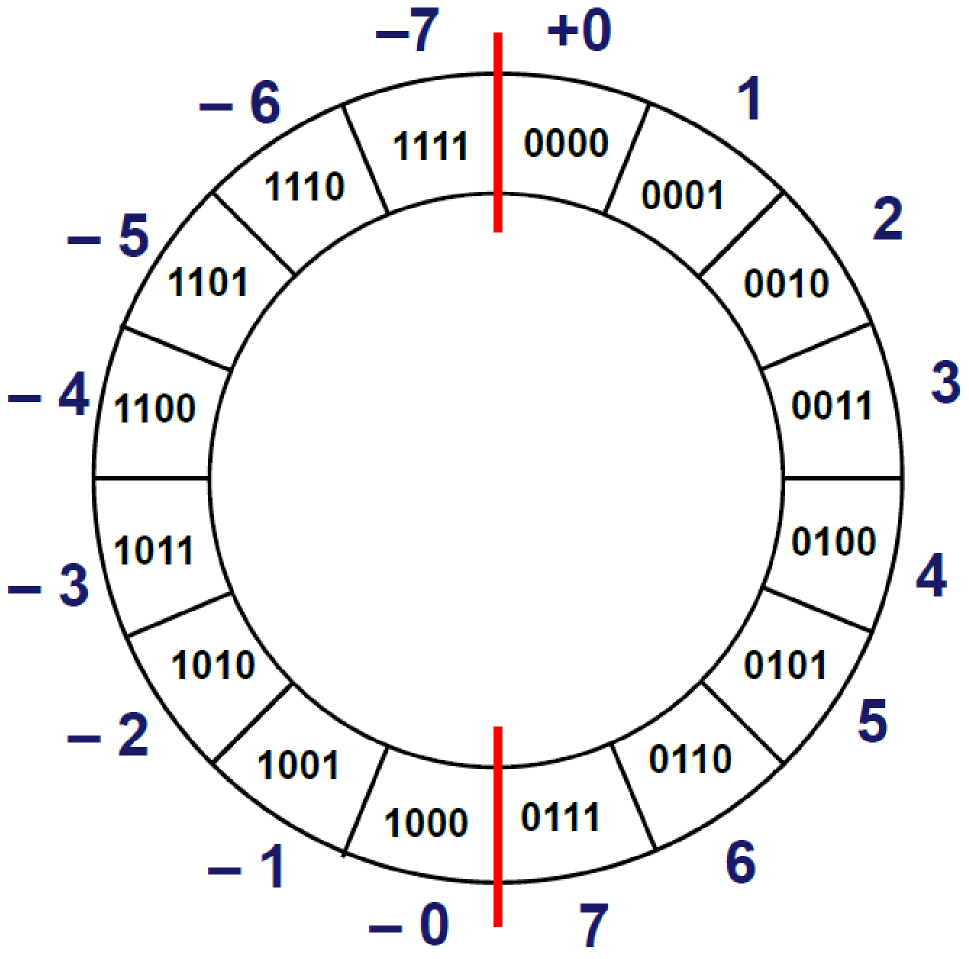
$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +\ 0\ 0\ 1\ 0 \\ \hline 1\ 1\ 1\ 1 \end{array} = -7_{10}\ (incorrect)$$

2. Two representations of zero (± 0):

0 0 0 0

1 0 0 0

Sign-magnitude number wheel



Two discontinuities:
at transitions around zero

2. One's complement representation

Complement (invert) all the bits!

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	**0**	**0**	**1**	**0**	**1**	**1**	**1**

=0+ 0+ 0+16+0+ 4+ 2+ 1=23

One's Complement →

1	**1**	**1**	**0**	**1**	**0**	**0**	**0**

Problems with one's complement

- Two representations of zero (+/- 0)

0 0 0 0 (+0)
1 1 1 1 (-0)

3. Two's complement representation

- MSBit has value (-2^{n-1}) :

$$A = -(a_{n-1}.2^{n-1}) + \sum_{i=0}^{n-2} a_i.2^i$$

- *for example, n=8:*

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	**0**	**0**	**1**	**0**	**1**	**1**	**1**

= 0 + 0 + 0 +16 +0 + 4 + 2 + 1 = 23

1	**1**	**1**	**0**	**1**	**0**	**0**	**1**

= -128 +64+32 +0 +8 + 0 + 0 + 1 = – 23

- n-bit two's complement number can take on values (-2^{n-1}) to $(2^{n-1}-1)$

Two's complement representation

- To form two's complement (i.e. flip the sign) of number A, *either*

- Working from LSB to MSB, complement (invert) all bits after (to the left of) first '1':
  – e.g. A = 0101 (= 5)
  complementing all bits to left of first '1' (occurs at bit 0):
  – A = 1011 (= – 5)

  OR

- Invert all bits in A and add 1:
  $-A = \overline{A} + 1 = 1010 + 1 = 1011 \; (= -5)$

Convenience of two's complement

1. MSB still indicates sign

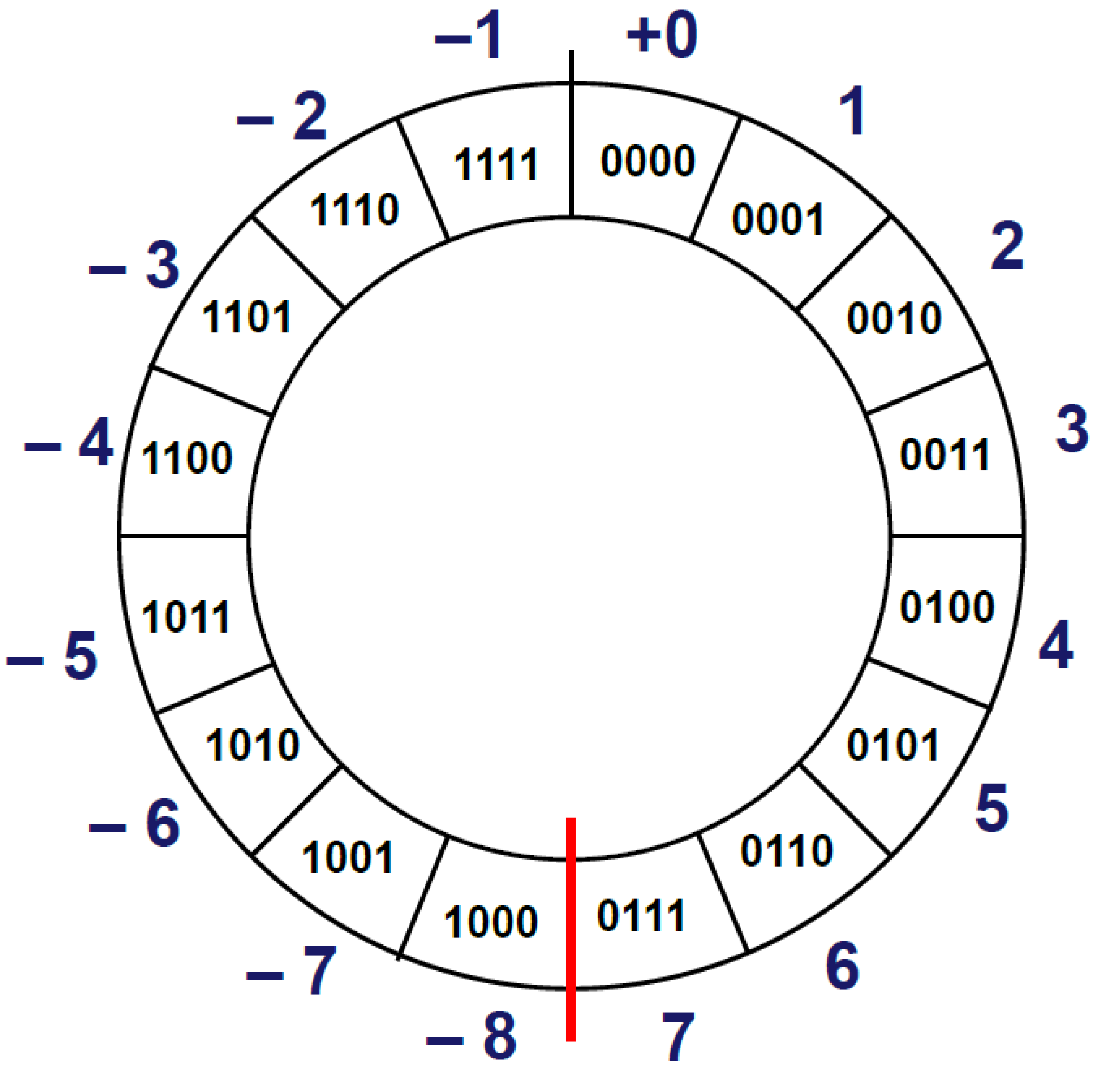
2. Addition *does* work

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 0\ 0\ 1\ 0 \\ \hline 1\ 1\ 0\ 1 \end{array} \qquad \begin{array}{r} -5_{10} \\ +2_{10} \\ \hline -3_{10}\ (correct!) \end{array} \qquad \begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 0\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 1\ 0 \end{array} \qquad \begin{array}{r} -5_{10} \\ +7_{10} \\ \hline +2_{10}\ (correct!) \end{array}$$

note: throw away the "overflow" bit

3. Only one representation of zero: 0 0 0 0

Two's complement number wheel



Discontinuity at limits of numerical representation (– 8 and +7)

Signed number representation

Three common approaches to deal with negative numbers:

3-bit number = 2^3 = 8 values

Value (decimal)	Sign Magnitude	1's Compliment	2's Compliment
3	0 1 1	0 1 1	0 1 1
2	0 1 0	0 1 0	0 1 0
1	0 0 1	0 0 1	0 0 1
0	0 0 0	0 0 0	0 0 0

Question? signed numbers range

3-bit word

4-bit word

8-bit word

16-bit word

Example 3

Represent the following numbers in 2s compliment
(use 8 bits).

1) -5
2) -7
3) -26
4) -67
5) 85
6) -85

Positive and negative hexadecimal numbers

- If A is a 4-digit unsigned hexadecimal number
  - What is the smallest value (in hex) that A can be and what is its decimal equivalent ?
  - What is the largest value (in hex) that A can be and what is its decimal equivalent ?

- If B is a 4-digit signed hexadecimal number
  - What is the smallest value (in hex) that B can be and what is its decimal equivalent ?
  - What is the largest value (in hex) that B can be and what is its decimal equivalent ?

Try these…

1. What is 27_{10} in 8-bit binary?
2. What is -27_{10} in 8-bit binary?
3. What is 10011010 (unsigned) in decimal?
4. What is 10011010 (signed) in decimal?
5. What is 299_{10} in 16-bit hex?
6. What is 1A3F in decimal?

Addition

- Decimal:

$$\begin{array}{r} 3\,7\,3\,4 \\ +\,5\,1\,6\,8 \\ \hline \end{array}$$

- Binary:

$$\begin{array}{r} 1\,0\,1\,1 \\ +\,0\,0\,1\,1 \\ \hline \end{array}$$

- Hex:

$$\begin{array}{r} 1\,A\,3\,7 \\ +\,0\,9\,F\,6 \\ \hline \end{array}$$

Overflow

- Note that if we add two n-bit numbers, we will (in general) get an (n+1) bit result:

$$
\begin{array}{cccccc}
 & \mathit{1} & \mathit{1} & \mathit{1} & & \textit{carries} \\
 & & 1 & 0 & 1 & 0 \\
+ & & 0 & 1 & 1 & 1 \\
\hline
 & 1 & 0 & 0 & 0 & 1
\end{array}
$$

overflow

One-bit adder circuit

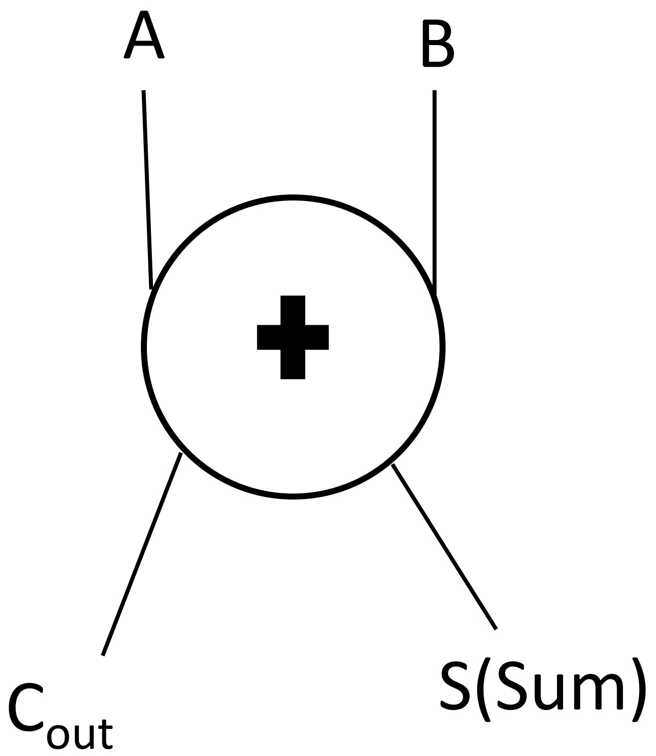
3	0 0 1 1
+2	0 0 1 0
5	0 1 0 1

S = $A \,.\, \bar{B} + \bar{A} \,.\, B$

And Or

$C_{out} = A \,.\, B$

1. Half adder:



One-bit adder circuit

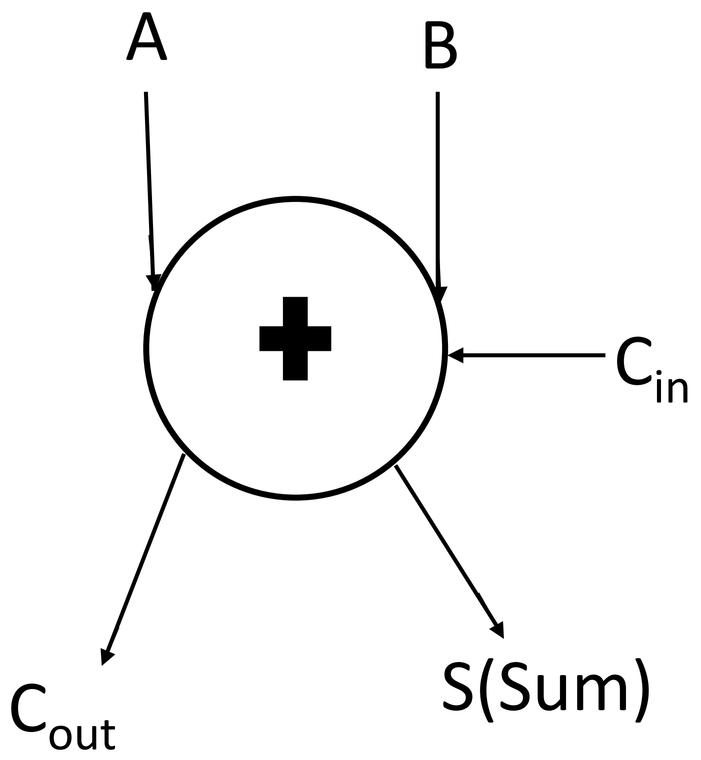
$$\begin{array}{r} 3 \\ +2 \\ \hline 5 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ 0\,0\,1\,0 \\ \hline 0\,1\,0\,1 \end{array}$$

S = $A.B.cin + A.\bar{B}.\overline{cin} + \bar{A}.B.\overline{cin} + \bar{A}.\bar{B}\ cin$

And Or

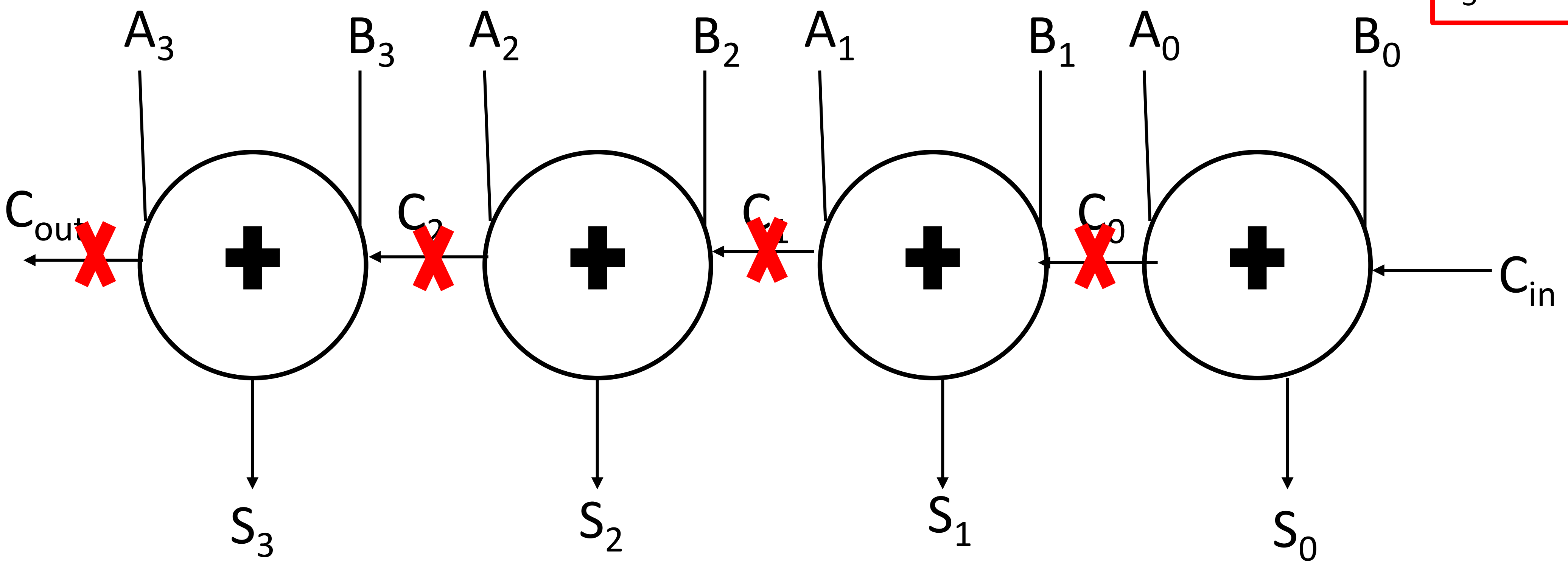
$C_{out} = A.B + A.cin + B.cin$

2. Full adder:



Multiple-bit adder circuit

3	0 0 1 1
+2	0 0 1 0
5	0 1 0 1



Inputs:
A0 A1 A2 A3
B0 B1 B2 B3
Cin

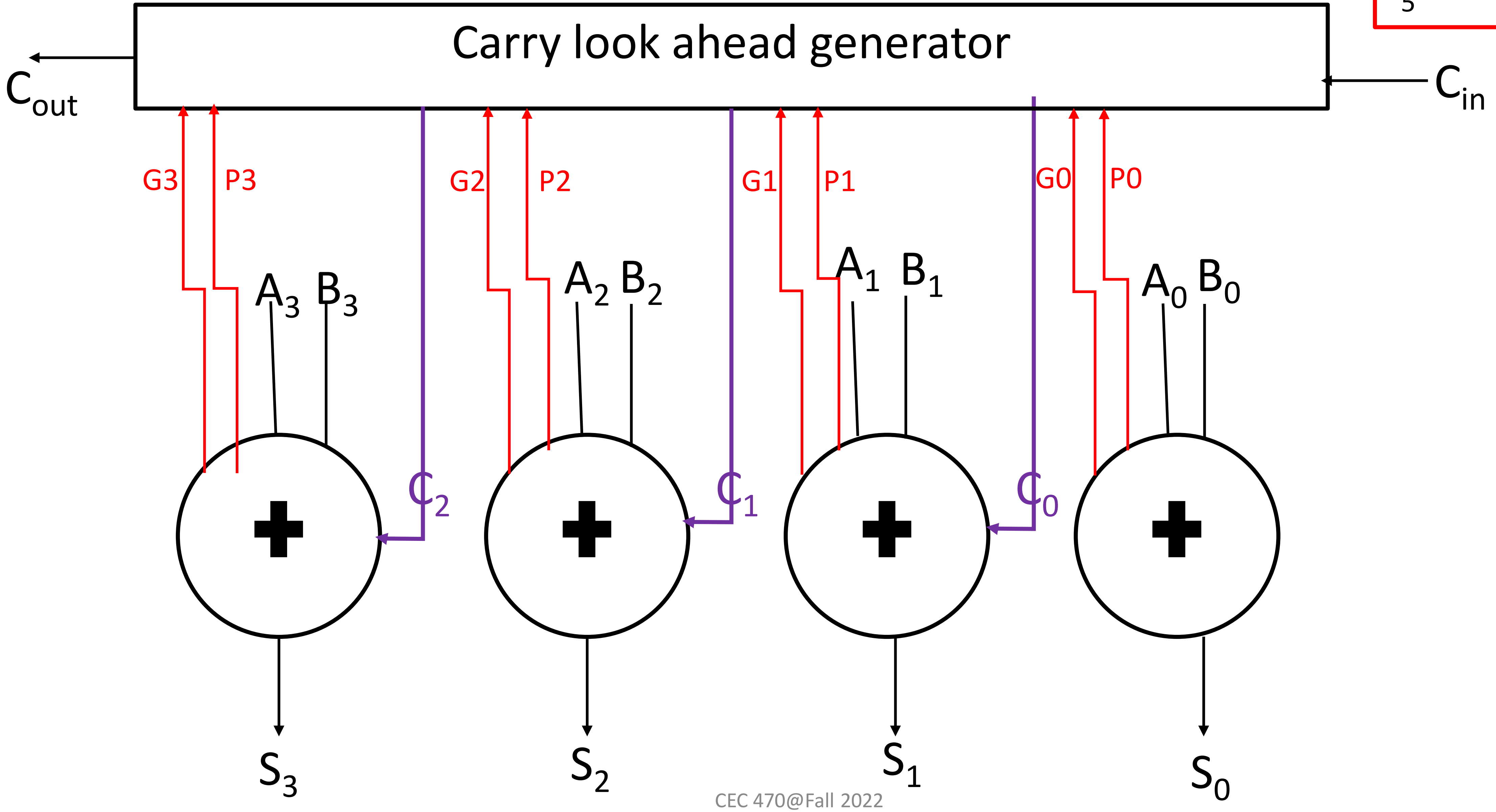
Output:

S0 S1 S2 S3
Cout

- Slow adder/Serial adder/Ripple carry adder
- 4 clock cycles required to add two 4-bit numbers
- 8 clock cycles to add two 8-bit numbers
- 64 clock cycles to add two 64-bit numbers

Carry look ahead adder circuit

3	0 0 1 1
+2	0 0 1 0
5	0 1 0 1



Carry look ahead adder

$C_0 = A_0 \,.\, B_0 + A_0 .\, cin + B_0 .\, cin$

$C_0 = A_0 \,.\, B_0 + cin(A_0 + B_0)$

$C_0 = G_0 + cin\ P_0$

$C_1 = A_1 \,.\, B_1 + A_1 .\, c_0 + B_1 .\, c_0$

$C_1 = A_1 \,.\, B_1 + c_0(A_1 + B_1)$

$C_1 = G_1 + c_0\ P_1$

$C_1 = G_1 + (G_0 + cinP_0)\ P_1$

$C_1 = G_1 + P_1G_0 + cinP_0\ P_1$

$C_2 = A_2 \,.\, B_2 + A_2 .\, c_1 + B_2 .\, c_1$

$C_2 = A_2 \,.\, B_2 + c_1(A_2 + B_2)$

$C_2 = G_2 + C_1\ P_2$

$C_2 = G_2 + P_2G_1 + P_2P_1G_0 + cinP_0\ P_1P_2$

Similarly,

$C_{out} = ?$

Carry look ahead adder

1. In 1 clock cycle : generate all Ps and Gs
2. In 2^{nd} clock cycle: generate carries
3. In 3^{rd} clock cycle : add the input bits along with the carries simultaneously

Adding two 4 bit numbers $\approx$ 3 clock cycles
Adding two 8 bit numbers $\approx$ 3 clock cycles
Adding two 64 bit numbers $\approx$ 3 clock cycles
Adding two 128 bit numbers $\approx$ 3 clock cycles

Subtraction

A – B= A+Twos complement of B

A= -5

B= 2

A-B , Subtract -5 and 2 ==> -5 - 2

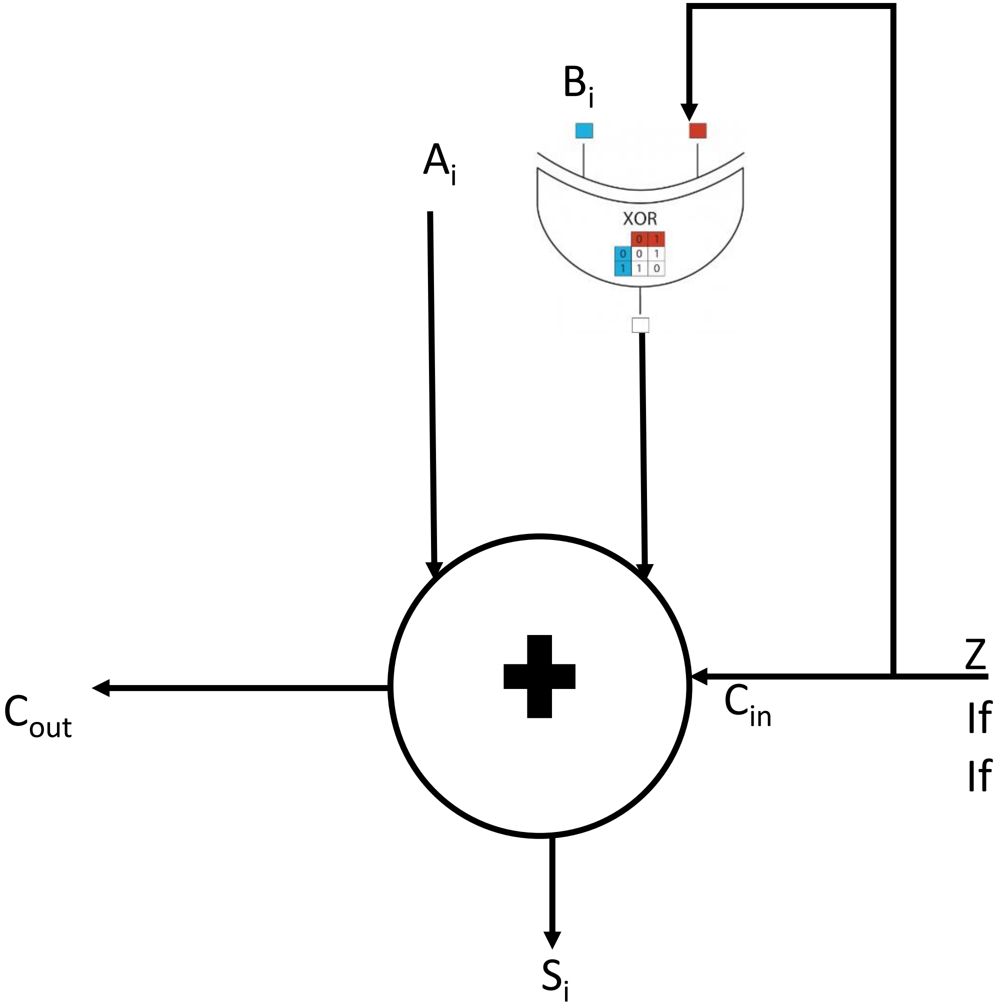
==> -5 + (-2) $\boxed{\text{(A + Two's Complement of B)}}$

-5 = 1 0 11

+2 = 0 0 1 0

-2 = 1 1 1 0

Subtraction



If Z=0, S = A+B

If Z=1, S = A+($\overline{B}$+1) = A−B

Key Ideas

- MSB
- LSB
- Bit
- Byte
- Word
- Nibble
- Decimal, binary, hexadecimal number system
- Unsigned and signed integers
- Signed magnitude
- Ones complement
- Twos complement
- One-bit half adder
- Full adder
- Carry look ahead adder
- Subtraction circuit