COMBINATIONAL LOGIC
HALF ADDER
FULL ADDER
RIPPLE CARRY ADDER
CARRY LOOKAHEAD ADDER
BINARY SUBTRACTOR
BINARY ADDER AND SUBTRACTOR

DIGITAL LOGIC DESIGN

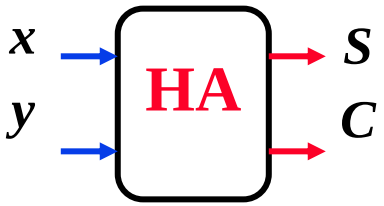
Iqra Chaudhary (Lecturer CS dept. NUML)

Binary Adder

★ Half Adder

- Adding two single-bit binary values X, Y produces a sum S bit and a carry out C-out bit.

$$\begin{array}{rr} & x \\ + & y \\ \hline C & S \end{array}$$



- **Adds 1-bit plus 1-bit**
- **Produces Sum and Carry**

- This operation is called half addition and it is called a half adder

Implement half Adder

« **Step 1:** Number of input=2 and Number of output=2

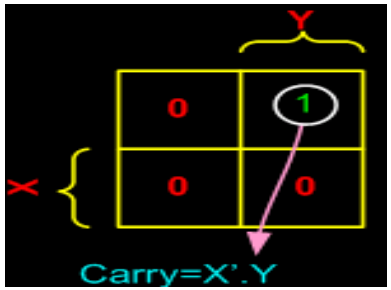
« **Step 2:** Drive the truth table

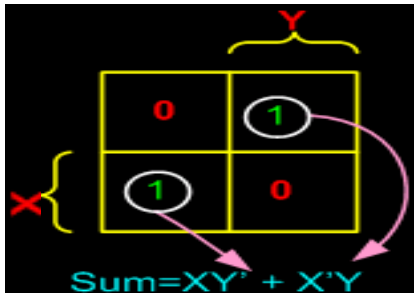
« **Step 3:** Obtain the equation from the truth table and simplify it

S = x'y + xy' = x⊕y

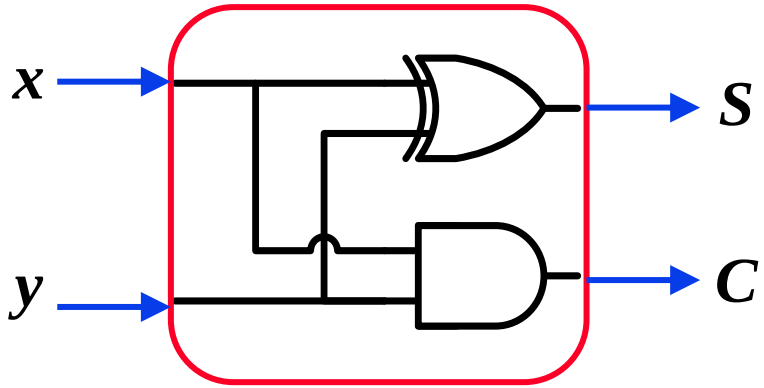
C= xy

x y	C S
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0





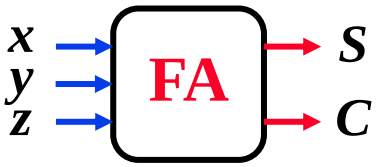
« **Step 4:** Draw the circuit diagram from simplified expression



Binary Adder

★ Full Adder

- Adds 1-bit plus 1-bit plus 1-bit
- Produces Sum and Carry



$$\begin{array}{rr} & x \\ + & y \\ + & z \\ \hline C & S \end{array}$$

Implement Full adder

- **Step 1:** Number of input=3 and Number of output=2
- **Step 2:** Drive the truth table
- **Step 3:** Obtain the equation from the truth table and simplify it

x	y	z	S	C
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

$S = xy'z'+x'yz'+x'y'z+xyz = x \oplus y \oplus z$

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$C = x'yz+xy'z+xyz'+xyz$

x \ yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C = xy + xz + yz$

$C = x'yz+xy'z+xyz'+xyz = z(x'y+xy')+xy(z+z') = z(x \oplus y)+xy$

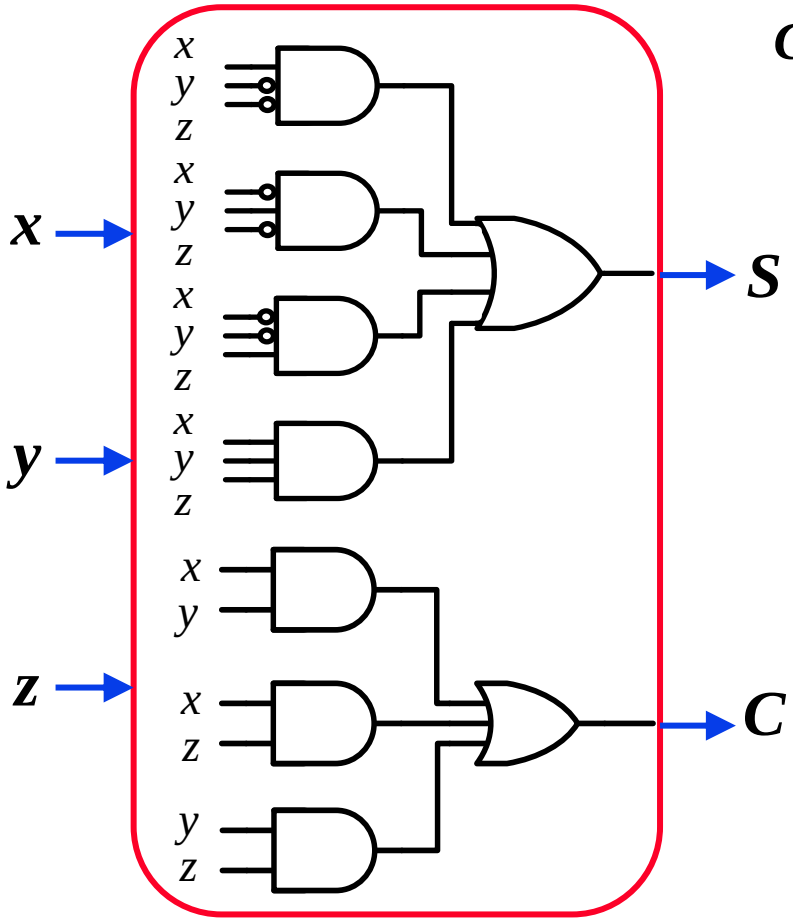
Step 4: Draw the circuit diagram from simplified expression

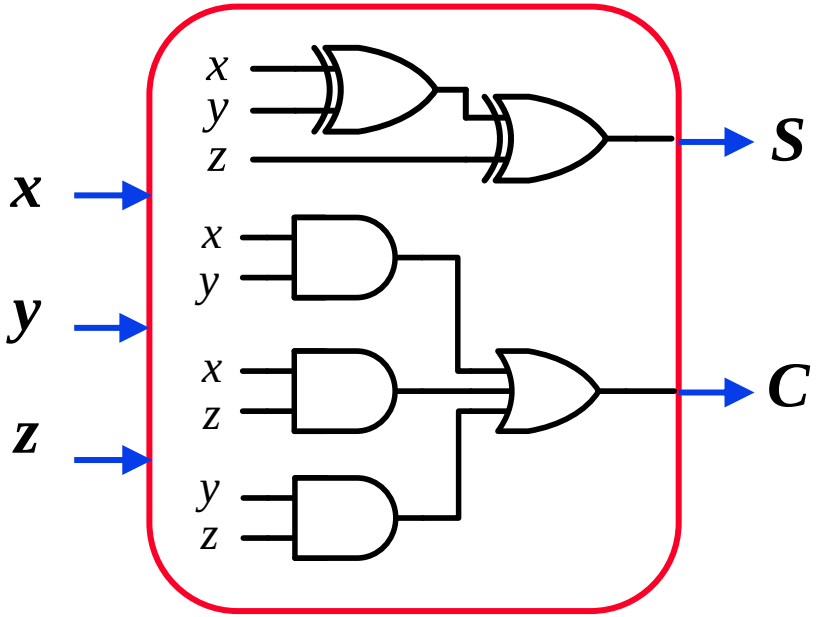
Implement Full adder

★ Circuit diagram:

$S = xy'z'+x'yz'+x'y'z+xyz = x \oplus y \oplus z$

$C = xy + xz + yz = z(x \oplus y) + xy$

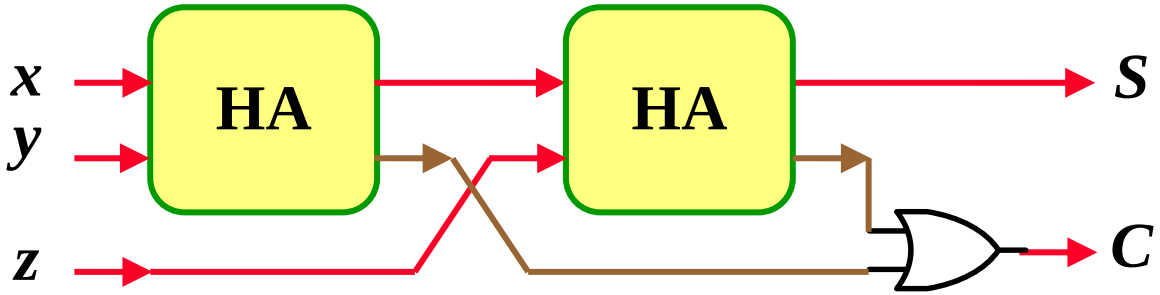


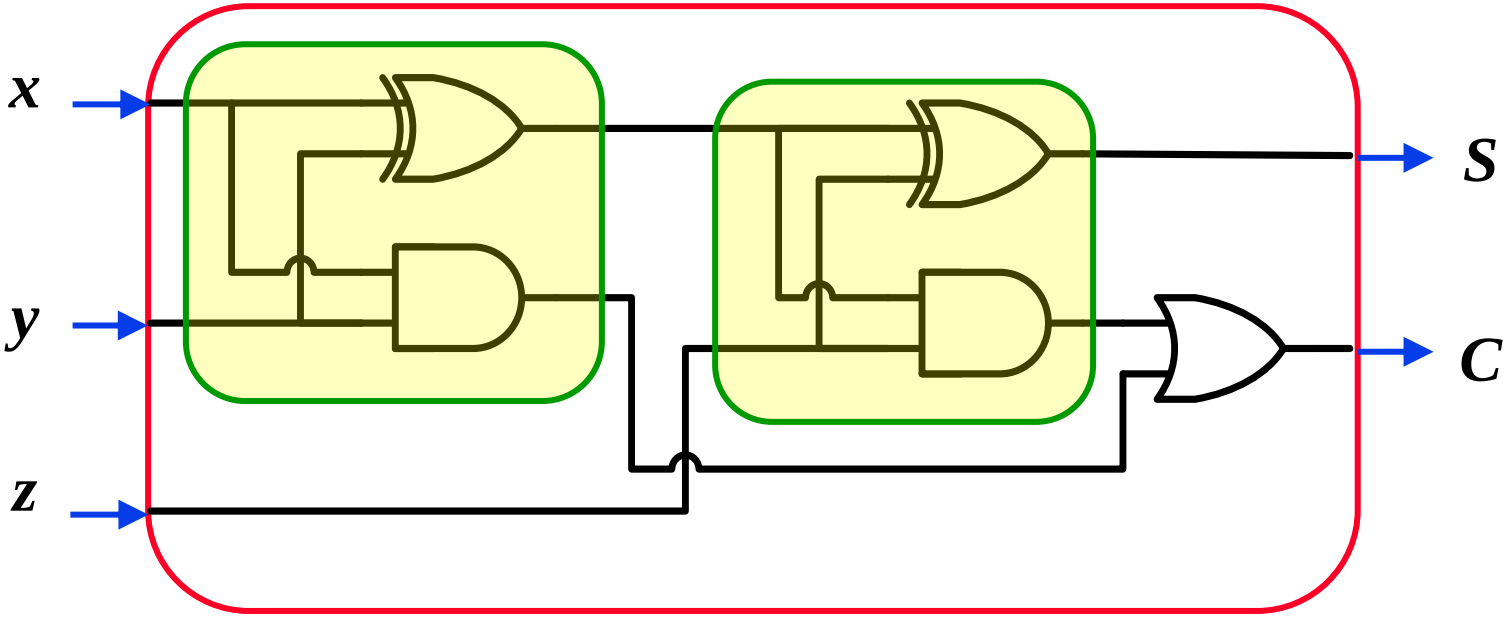


Multiple-variable exclusive-OR operation is defined as an *odd function*.

Implement Full adder

★ **Full Adder by using two half adders**





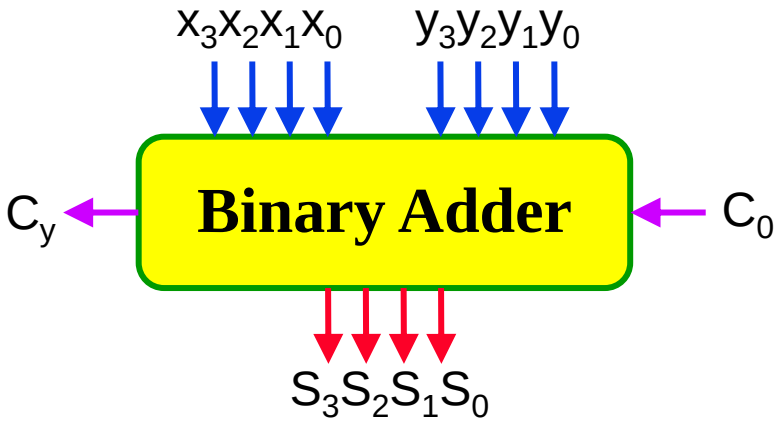
Binary Adder
Ripple carry adder/Carry propagate adder

$$
\begin{array}{rcccc}
 & c_3 & c_2 & c_1 & \\
+ & x_3 & x_2 & x_1 & x_0 \\
+ & y_3 & y_2 & y_1 & y_0 \\
\hline
Cy & S_3 & S_2 & S_1 & S_0
\end{array}
$$

Example: let x=1011 and y=0011

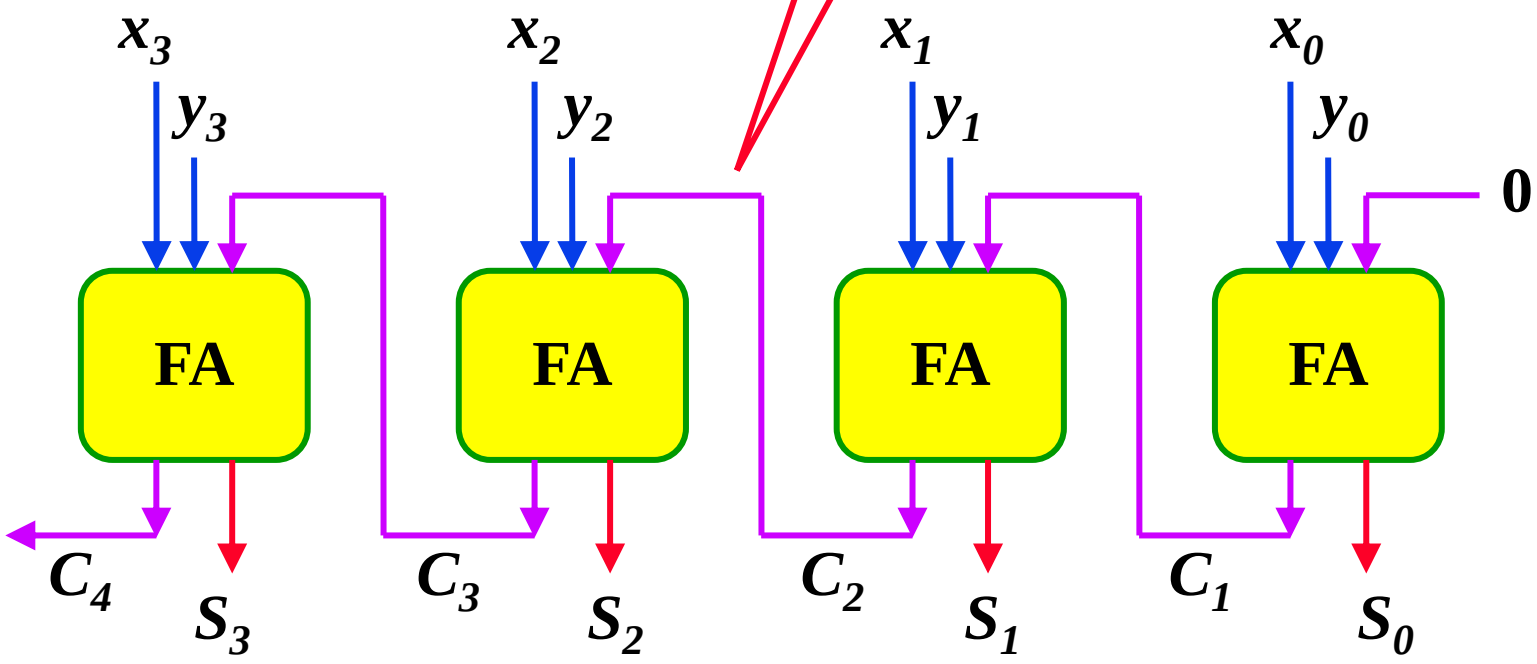
Subscript i:	3	2	1	0	
Input carry	0	1	1	0	x C_i
Augend	1	0	1	1	y A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

Binary Adder
Ripple carry adder/Carry propagate adder



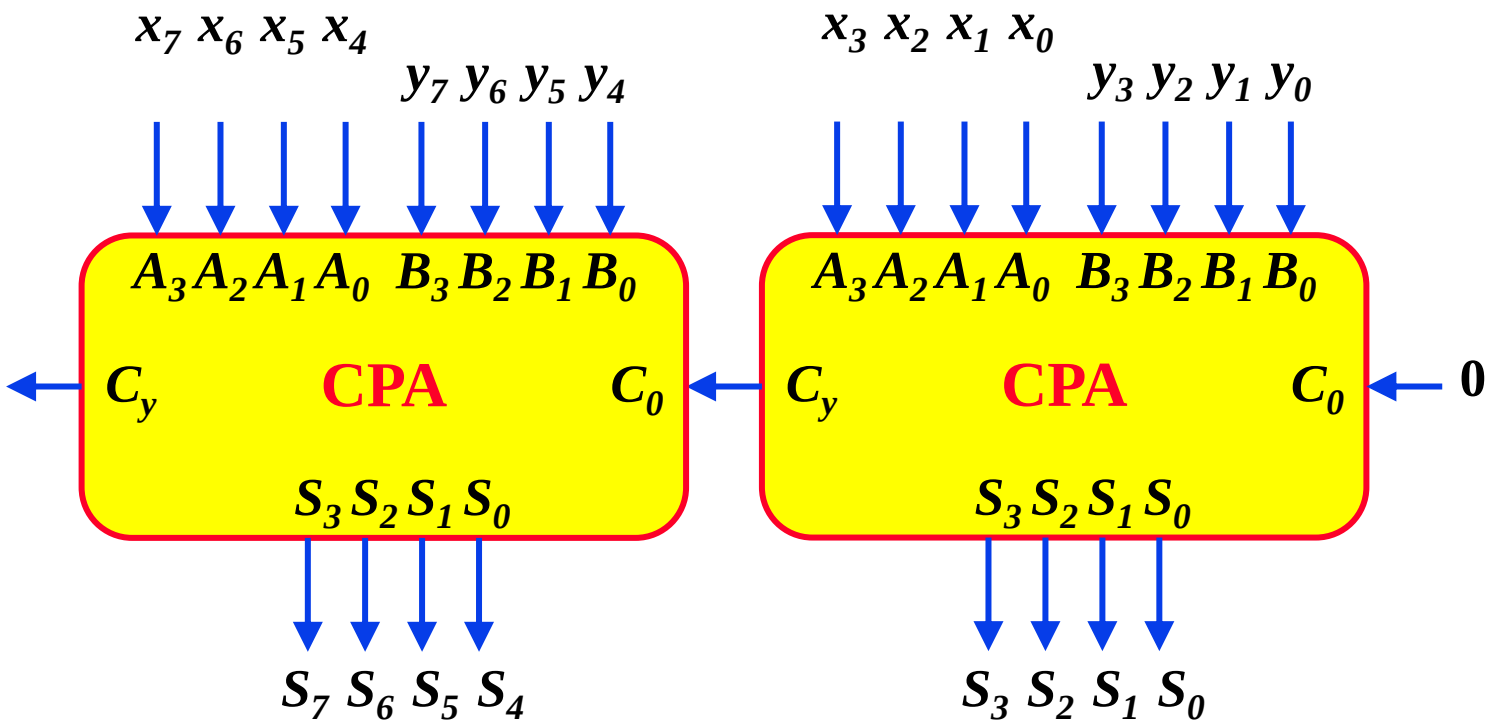
Carry Propagate Addition

$$\begin{array}{rcccc} & c_3 & c_2 & c_1 & \\ + & x_3 & x_2 & x_1 & x_0 \\ + & y_3 & y_2 & y_1 & y_0 \\ \hline Cy & S_3 & S_2 & S_1 & S_0 \end{array}$$



Binary Adder
Ripple carry adder/Carry propagate adder

★ We can also combine 2 or more CPA



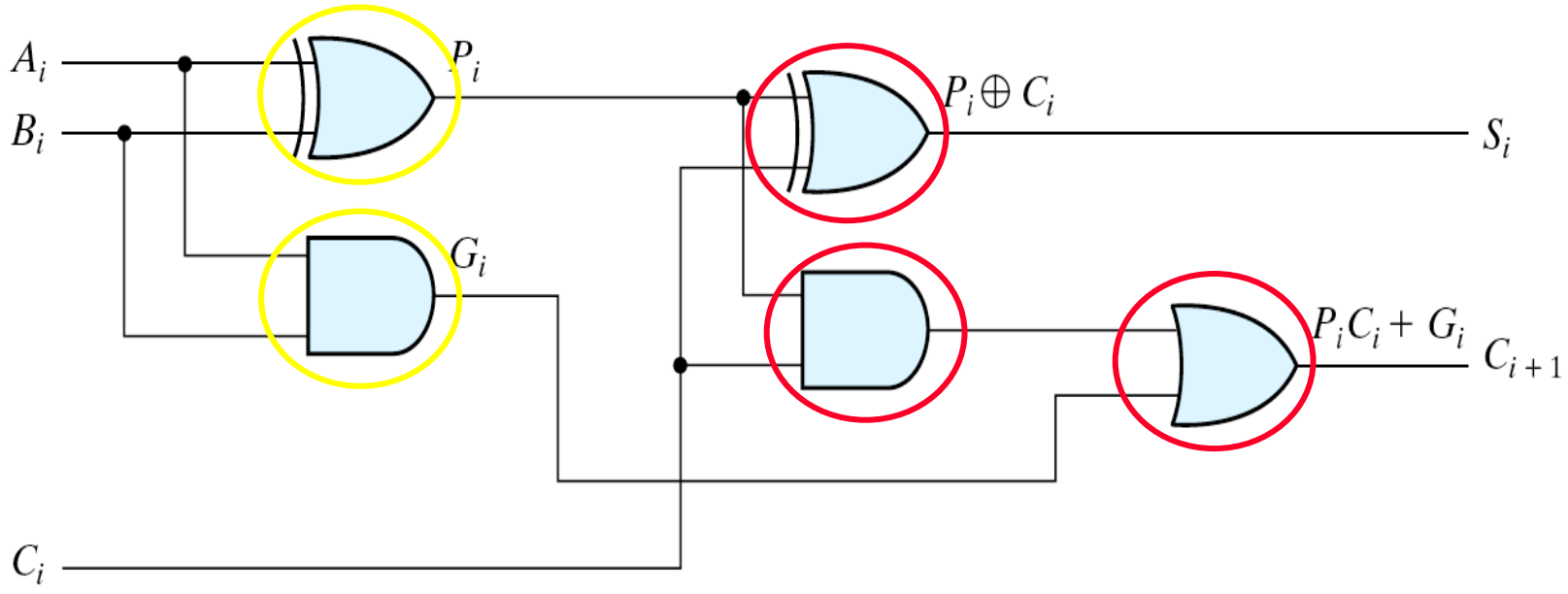
Binary Adder
Ripple carry adder/Carry propagate adder

- This kind of adder is also called ripple carry adder, since each carry bit "ripples" to the next full adder.
- Note that the first (and only the first) full adder may be replaced by a half adder.
- The layout of a ripple carry adder is simple.
- However, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder.
- The gate delay can easily be calculated by inspection of the full adder circuit. Following the path from C_{in} to C_{out} shows 2 gates per full adder that must be passed through.

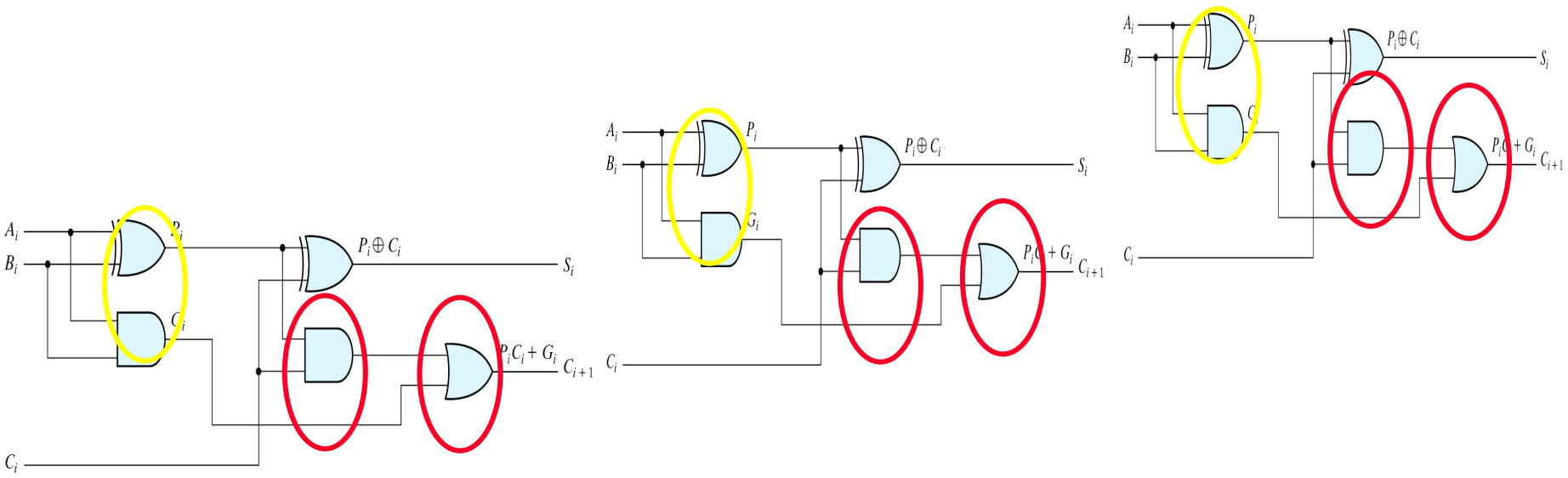
Ripple carry adder/Carry propagate adder

★ Carry propagation delay time

- **The critical path counts (the worst case)**
- **$(A_1, B_1, C_1) \rightarrow C_2 \rightarrow C_3 \rightarrow C_4 \rightarrow (C_5, S_4)$**
- **When 4-bits full-adder → 8 gate levels (n-bits: $2n$ gate levels)**



★ Carry propagation delay time



When 3-bits full-adder → 6 gate levels (*n*-bits: 2*n* gate levels)

Parallel Adders

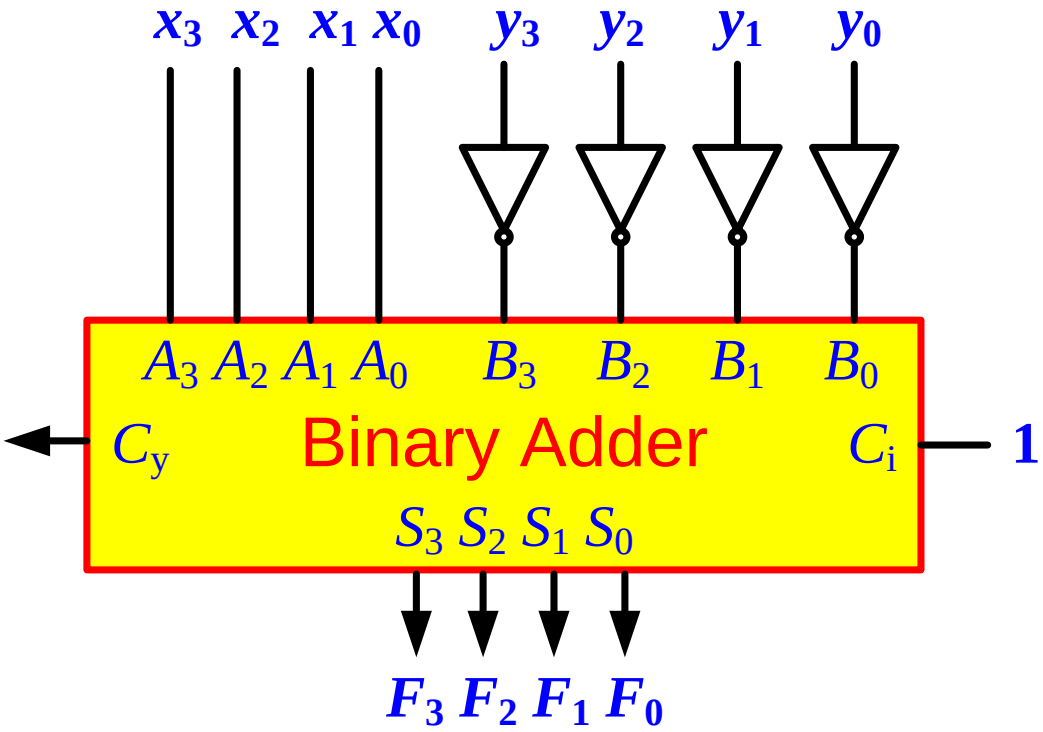
- **Reduce the carry propagation delay**
  1. **Employ faster gates**
  2. **Look-ahead carry (more complex mechanism, yet faster)**
- **We will convert the given ripple carry adder into carry lookahead adder by simultaneously generating carries**

Binary Subtractor

★ **Subtraction using 2's complements**

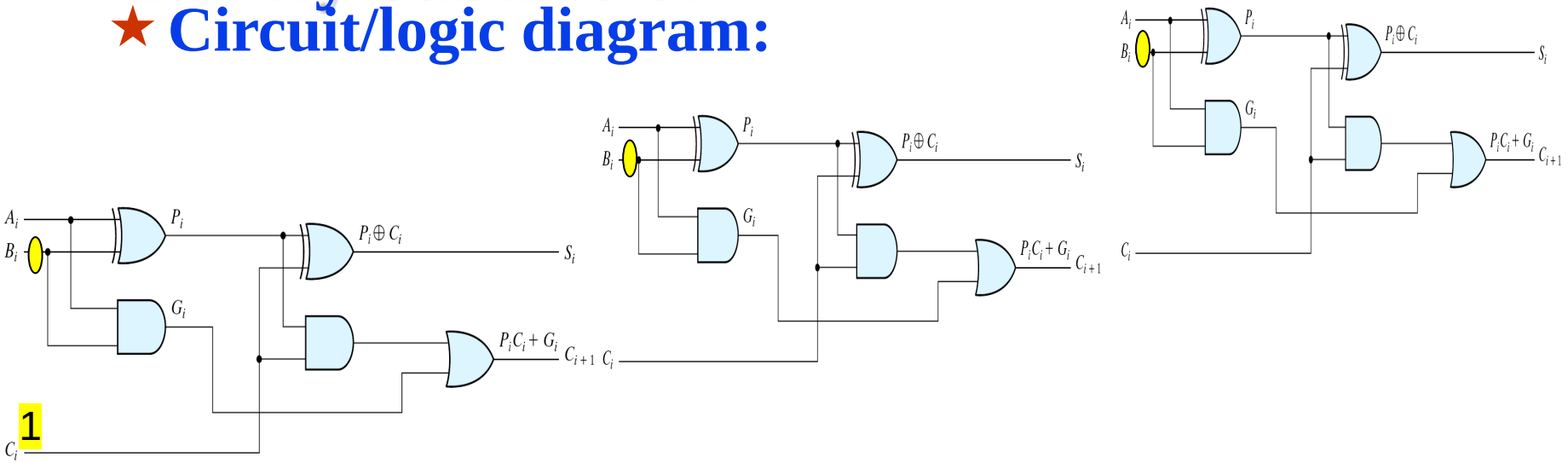
$$x - y = x + (-y) = x + (y\ 2\text{'s comp})$$

$$= x + (y\ 1\text{'s comp}) + 1$$

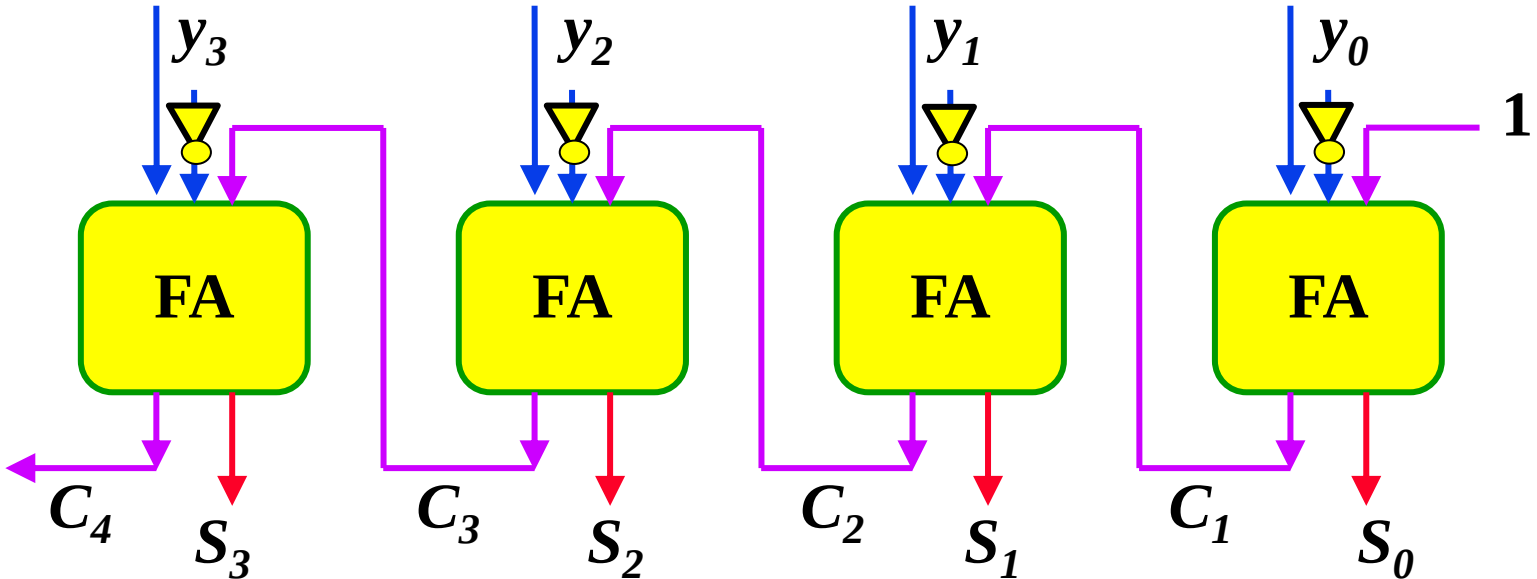


Binary Subtractor

★ Circuit/logic diagram:



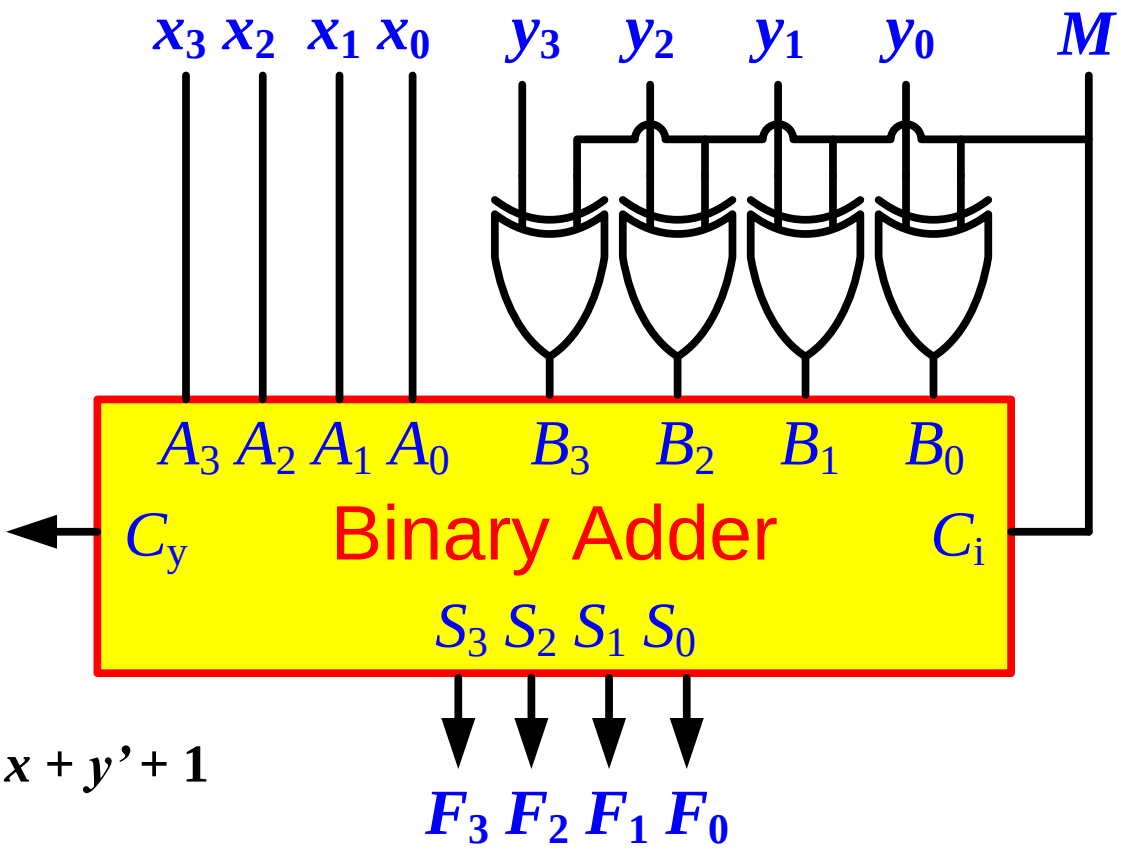
★ Block diagram:



Binary Adder/Subtractor

★ ***M*:Control Signal (Mode)**

- ***M*=0 ➔ *F* = *x* + *y***
- ***M*=1 ➔ *F* = *x* – *y***



★ ***Use xor gate***

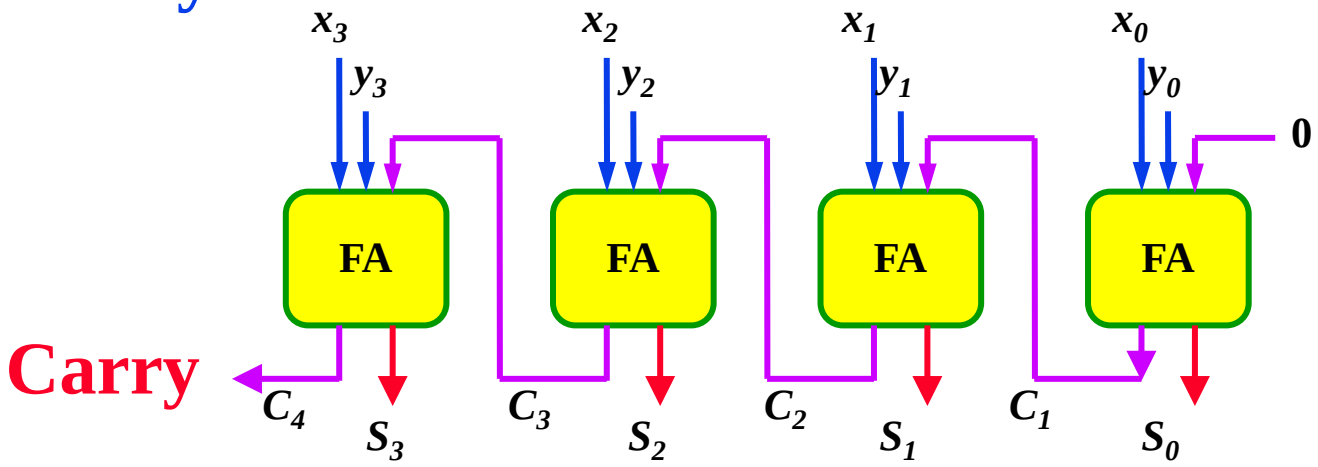
- ***y*⊕0 = *y*** ➔ ***F* = *x* + *y***
- ***y*⊕1 = *y*'** ➔ ***F* = *x* – *y* = *x* + *y*' + 1**

Overflow

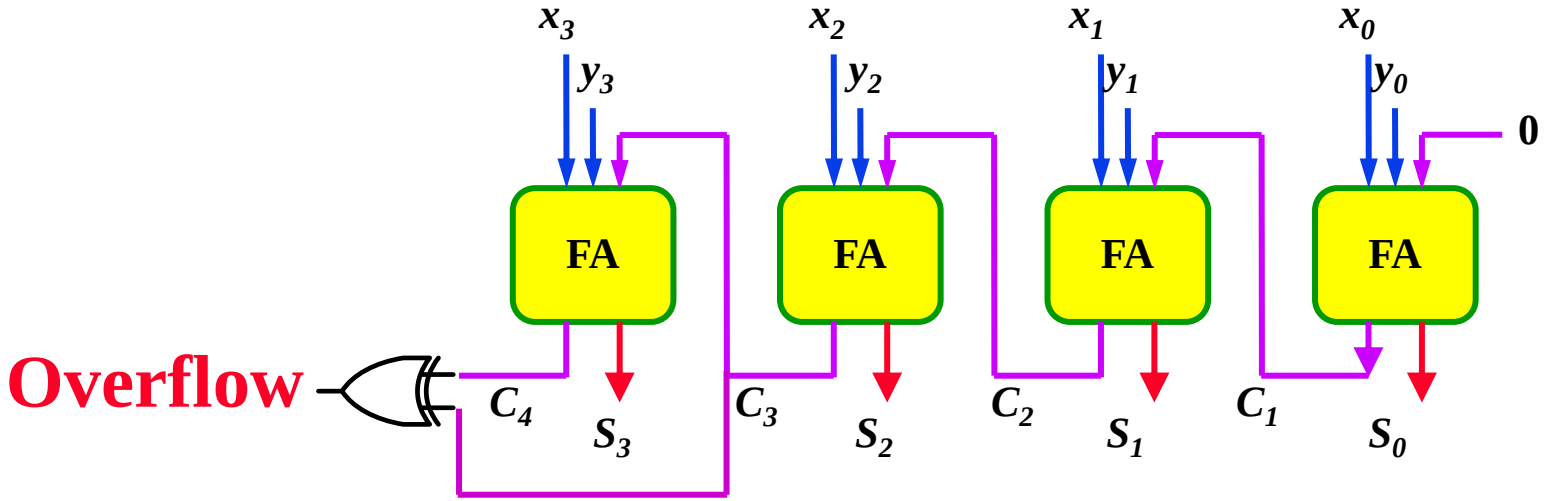
- ★ If we start with two n-bit numbers and we end up with a number that is n+1 bits, we say an overflow has occurred.
- Two cases of overflow for addition of signed numbers
  - Two large positive numbers overflow into sign bit
    - Not enough room for result
  - Two large negative numbers added
    - Same – not enough bits

Overflow

★ Unsigned Binary Numbers



★ Signed binary numbers/2's Complement Numbers



Examples

Two signed numbers +70 and +80 are stored in 8-bit registers.

The range of binary numbers, expressed in decimal, that each register can accommodate is from +127 to -128.

Since the sum of the two stored numbers is 150, it exceeds the capacity of an 8-bit register.

The same applies for -70 and -80.

Overflow Detection

Carries :	01 000000		10110000
+70	0 1000110	-70	1 0111010
+80	0 1010000	-80	1 0110000
+150	1 0010110	-150	0 1101010

The addition of +70 and +80 resulted in a negative number!

An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position.

If these two carries are not equal an overflow has occurred.