Chapter 5

Binary unsigned numbers

$B = b_{n-1} b_{n-2} --- b_1 b_0$ $b_i \in \{0,1\}$

$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + --- + b_1 \times 2 + b_0$ binary to decimal

Decimal to binary : $30_{10} = \cdot_2$?

	quotient	remainder
		LSB
$30 \div 2$	15	0
$15 \div 2$	7	1
$7 \div 2$	3	1
$3 \div 2$	1	1
$1 \div 2$	0	1
		MSB

$= 11110_2$

Unsigned addition

2 inputs   2 outputs

$0_2 + 0_2 = 0_2$

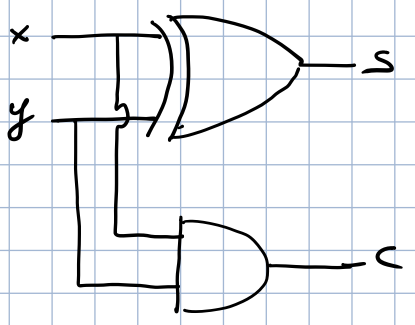
$0_2 + 1_2 = 1_2 + 0_2 = 1_2$

$1_2 + 1_2 = 10_2$ (need 2 bits)

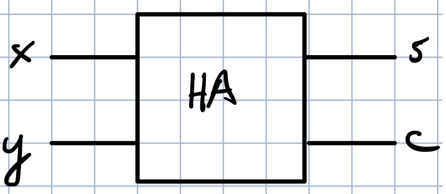
$$\begin{array}{r} \overset{\text{carry}}{1}\ \ \\ 17 \\ +\ 18 \\ \hline 35 \end{array}$$

Adding two 1 bit numbers

		carry	sum
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



half adder (HA)

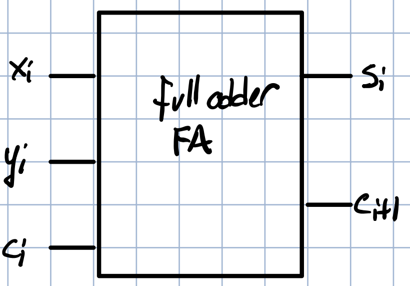


$c_3\ c_2\ c_1\ c_0$ ← carry in

Adding three 1 bit numbers

inputs output

x_i	y_i	c_i	s_i	c_{i+1}
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



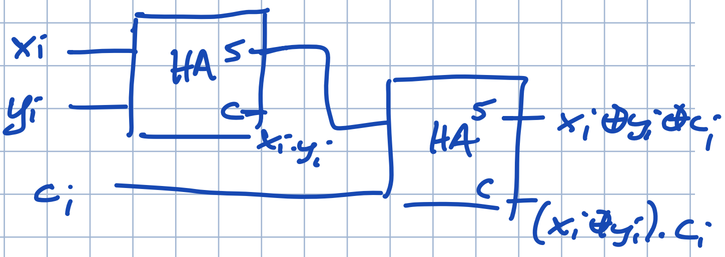
1 1 0 0

6 0 1 1 0

+2 0 0 1 0

8 1 0 0 0 sum

0 1 1 0 ← carry out



K-MAP

chessboard pattern => no simplification

s_i

$c_i \backslash x_i y_i$	00	01	11	10
0		1		1
1	1		1	

$$s_i = m_1 + m_2 + m_4 + m_7$$

$$s_i = \bar{x}_i \bar{y}_i c_i + \bar{x}_i y_i \bar{c}_i + x_i y_i c_i + x_i \bar{y}_i \bar{c}_i$$

$$= (x_i \oplus y_i) \oplus c_i = x_i \oplus y_i \oplus c_i$$

$$= (x_i \bar{y}_i + \bar{x}_i y_i) \oplus c_i$$

$$= (x_i \bar{y}_i + \bar{x}_i y_i) \bar{c}_i + \overline{(x_i \bar{y}_i + \bar{x}_i y_i)} c_i$$

$$= x_i \bar{y}_i \bar{c}_i + \bar{x}_i y_i \bar{c}_i + x_i y_i c_i + \bar{x}_i \bar{y}_i c_i$$

c_{i+1}

$x_i y_i$

$c_i \backslash x_i y_i$	00	01	11	10
0			1	
1		1	1	1

minimal SOP

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$= x_i y_i + (x_i \oplus y_i).c_i \quad \text{(not minimal SOP)}$$

Alternative to c_{i+1}

	00	01	11	10
0			1	
1			1	

$x_i \cdot y_i$

+

	00	01	11	10
0		1		1
1		1		1

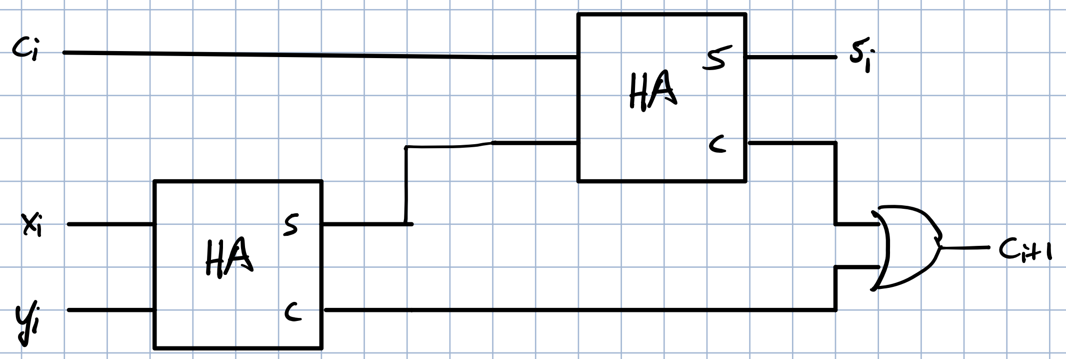
$x_i \oplus y_i$

·

	00	01	11	10
0				
1	1	1	1	1

c_i

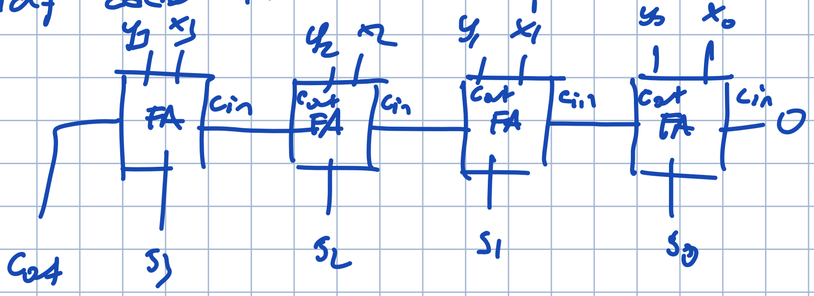
$$c_{i+1} = x_i y_i + (x_i \oplus y_i) \cdot c_i$$



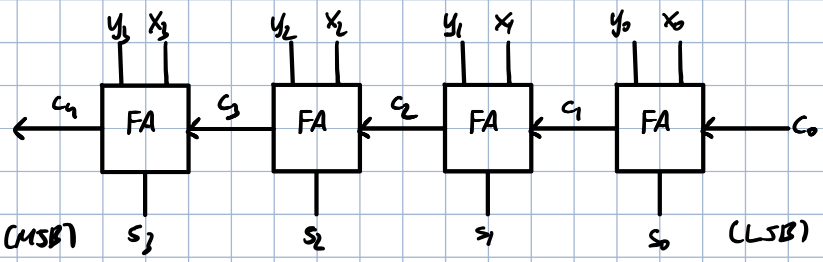
Unsigned addition (4-bit example)

half adder is also a possibility.

$$\begin{array}{lcccc}
 & C_3 & C_2 & C_1 & C_0 \\
\text{Cin} & 1 & 1 & 0 & 0 \\
 & 0 & 1 & 1 & 0 \\
+ & 0 & 1 & 1 & 1 \\
\hline
\text{Sum} & 1 & 1 & 0 & 1 \\
\text{Cout} & 0 & 1 & 1 & 0 \\
 & C_4 & C_3 & C_2 & C_1
\end{array}$$



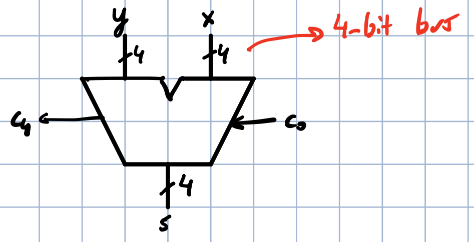
4-bit adder



$$\begin{array}{rl}
0011 & 3 \\
+\ 1001 & 9 \\
\hline
0\underbrace{1100}_{12} & ✓
\end{array}$$

$$\begin{array}{rl}
1111 & 15 \\
+\ 0010 & 2 \\
\hline
1\underbrace{0001}_{1} & \times
\end{array}$$

overflow



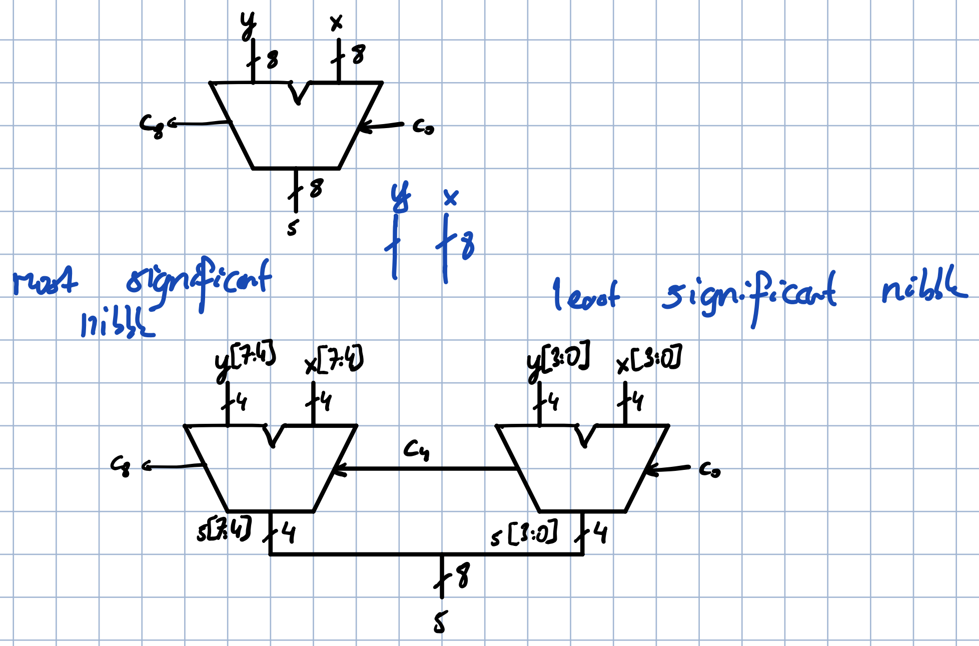
symbol of 4-bit adder

overflow detector = Cout

range of 4-bit unsigned : 0 - 15 , 0 to $2^4 - 1$

8-bit adder

4 bits = nibble



Signed numbers: two's complement

n-bits to represent negative and positive numbers.

example: n=4

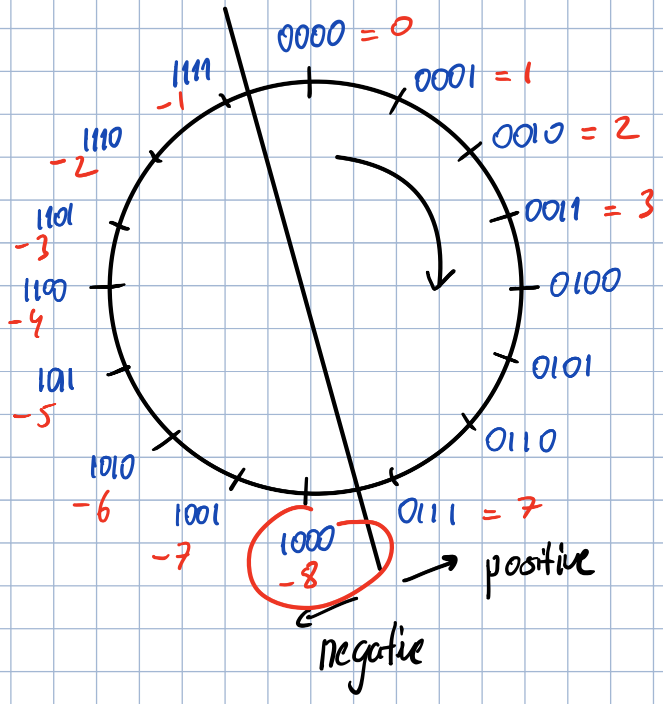
MSB = sign bit

two's complement

=> complement of a number w.r.t. 2^n

$2^4 = 10000$

$$\begin{array}{rr} 2 & 0010 \\ + \; -2 & 1110 \\ \hline & 10000 = 2^n \end{array}$$



range of representable numbers: −8 to 7

range of representable numbers (with n-bits): -2^{n-1} to $2^{n-1}-1$

range of representable numbers (with n-bits, unsigned): 0 to $2^n - 1$

Two's complement addition

$$
\begin{array}{llllll}
C_{in} & 0 & 0 & 0 & & \\
 & 1 & 1 & 1 & 0 & (-2) \\
+ & 0 & 0 & 0 & 1 & (+1) \\
\hline
Sum & 1 & 1 & 1 & 1 & (-1) \checkmark \\
C_{out} & 0 & 0 & 0 & 0 &
\end{array}
$$

★ hardware for addition in two's complement is the same as that of unsigned

Finding two's complement of a negative number

6_{10} = 00110
11001
11010
-6_{10} in 5-bit?

-6_{10} in 4-bit representation.

$$
\begin{array}{ll}
6_{10} = & 0110 \\
+ -6_{10} & xxxx \\
\hline
0_{10} & 0000
\end{array}
$$

⇒ bitwise complement: 1001
add 1 : 1010 = -6_{10}

-10_{10} in 5-bit representation

10_{10} = 01010
bitwise complement = 10101
add 1 = 10110 = -10_{10}

Shortcut: 010|10 = 10_{10}
101|10 = -10_{10}
complement copy from LSB until first 1.

n=12
011001011100
100110100100

n=5 bits $\overset{4}{1}\overset{3}{0}\overset{2}{1}\overset{1}{1}\overset{0}{1}$. Convert to decimal?

$= 1\times(-2^4) + 0.(2^3) + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$

$= -16+4+2+1 = -9$

Two's complement subtraction

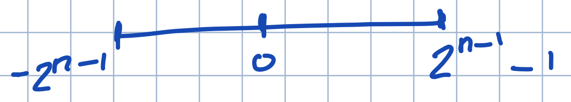
A − B = A + (−B)

= A + two's complement of B

A + bitwise complement of B + 1

n=4 bits

$$\begin{array}{r} +4 \\ - \; +3 \\ \hline \end{array} = \begin{array}{r} 0100 \\ - \; 0011 \\ \hline \end{array} \qquad \begin{array}{r} 0100 \\ 1100 \\ + \qquad 1 \\ \hline 0001 \end{array} = +1_{10}$$



Overflow

Happens when the result is not in the range of the number system.

sum of two positive numbers must be positive
sum of two negative numbers must be negative

n=4 bits

5 + 7

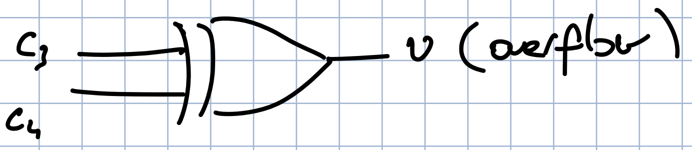
4 + 5

-8 + -7

$$\begin{array}{rl} 111 & \\ 0101 & \\ + \; 0111 & \\ \hline \boxed{1}100 & s \\ 0111 & c \end{array} \qquad \begin{array}{rl} 000 & \\ 1000 & \\ + \; 1001 & \\ \hline \boxed{0}001 & s \\ 1000 & c \end{array}$$

overflow detector?

overflow when $C_{out,n} \neq C_{out,n-1}$

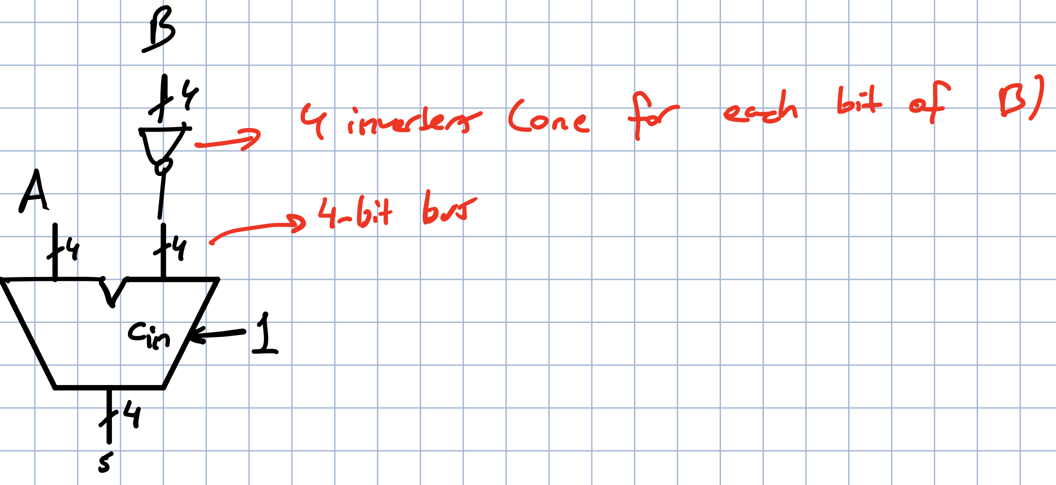


a_4	b_4	s_4	v
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$v = m_1 + m_6$

4-bit subtractor circuit

A − B



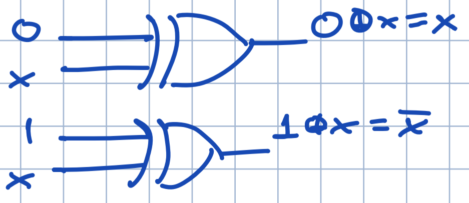
4 inverters (one for each bit of B)

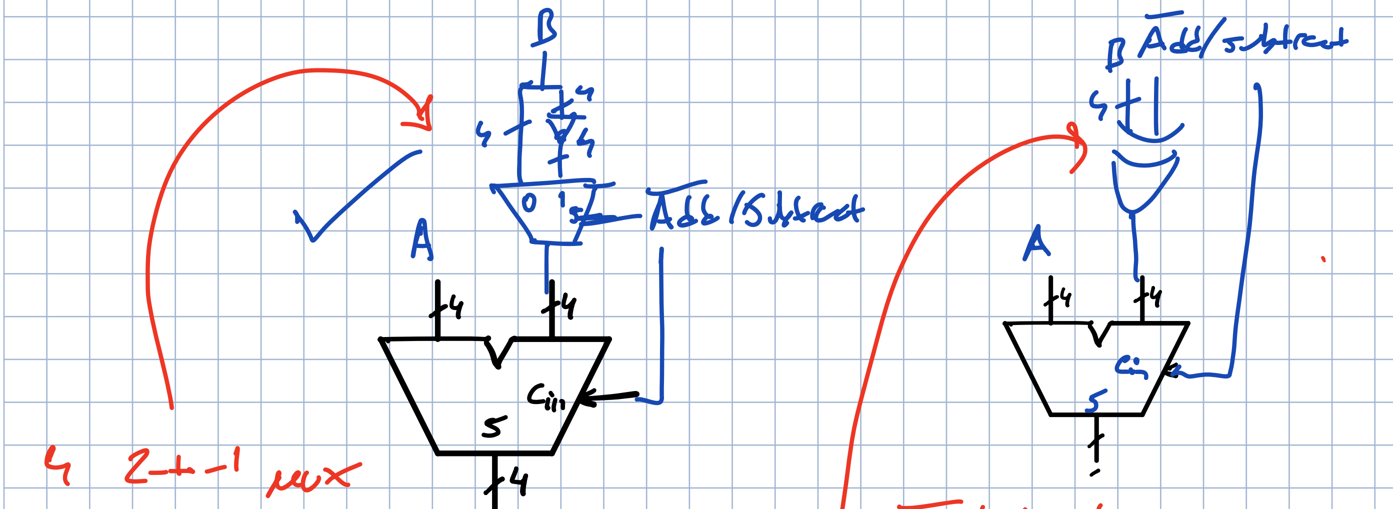
4-bit bus

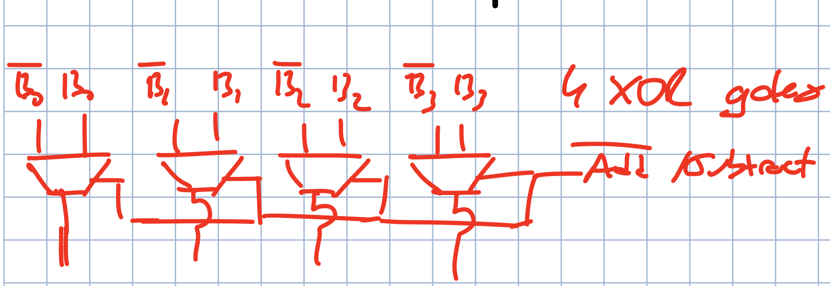
4-bit adder/subtractor circuit

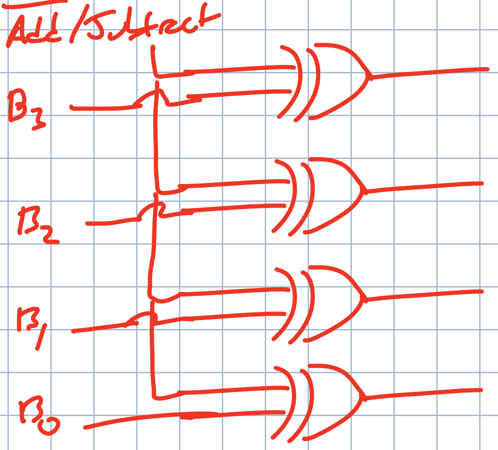
$\overline{Add}/Subtract = 0 \Rightarrow Y = A+B$

$\overline{Add}/Subtract = 1 \Rightarrow Y = A-B$









```
library ieee;
use ieee.std_logic_signed.all

entity addnibble is
    port(a,b: in std_logic_vector(3 downto 0);
        y: out std_logic_vector(3 downto 0));
end addnibble;

architecture addarch of addnibble is
begin
    y <= a+ b;
end addarch;
```

$b_3b_2b_1b_0$

$y_3y_2y_1y_0$

addition operator in VHDL

4 bit adder