# Digital systems: from bits to microcontrollers

- Introduction to digital systems fundamentals
- Combinatorial digital systems
- Sequential digital systems
- Introduction to microcontrollers
- Introduction to advanced implementation platforms



# Representation of signed numbers

Sign and magnitude representation

bit sign followed by natural binary representation

Two's complement

$N^{(2)} = 2^n - (N)_2$ where n is the number of bits of the number to represent

One's complement

$N^{(1)} = (2^n – 1) - (N)_2$

Universidade Nova de Lisboa FCT FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA UNINOVA CTS - Centre of Technology and Systems GRES

# Using complement representation

A - B    A + (- B)

# Two's complement

n = 4

A = $3_{10} = 0011_2$

B = $4_{10} = 0100_2$

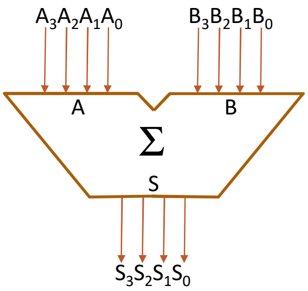
$$N^{(2)} = 2^n - (N)_2$$

$A^{(2)} = 2^4 - 0011_2 = 10000 - 0011 = 1101$

$B^{(2)} = 2^4 - 0100_2 = 10000 - 0100 = 1100$

```
+3 = 0011       +3 = 0011       -3  = 1101       -3  = 1101
+4 = 0100       -4 = 1100       +4  = 0100       -4  = 1100
+7   0111       -1   1111       +1 (1)0001       -7 (1)1001
```

# Adding signed number using two's complement




A, B, and S represented in two's complement

# One's complement

n = 4

A = $3_{10}$ = $0011_2$

B = $4_{10}$ = $0100_2$

$$N^{(1)} = (2^n - 1) - (N)_2$$

$A^{(1)} = (2^4 - 1) - 0011_2 = 1111 - 0011 = 1100$

$B^{(1)} = (2^4 - 1) - 0100_2 = 1111 - 0100 = 1011$

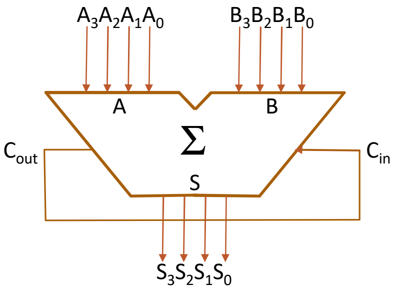
```
+3 = 0011
+4 = 0100
+7   0111
```

```
+3 = 0011
-4 = 1011
-1   1110
```

```
-3 = 1100
+4 = 0100
  (1)0000
   └──→ 1
+1   0001
```

```
-3 = 1100
-4 = 1011
  (1)0111
   └──→ 1
-7   1000
```

# Adding signed number using one's complement




A, B, and S represented in one's complement

---

Two's complement

$N^{(2)} = 2^n - (N)_2$

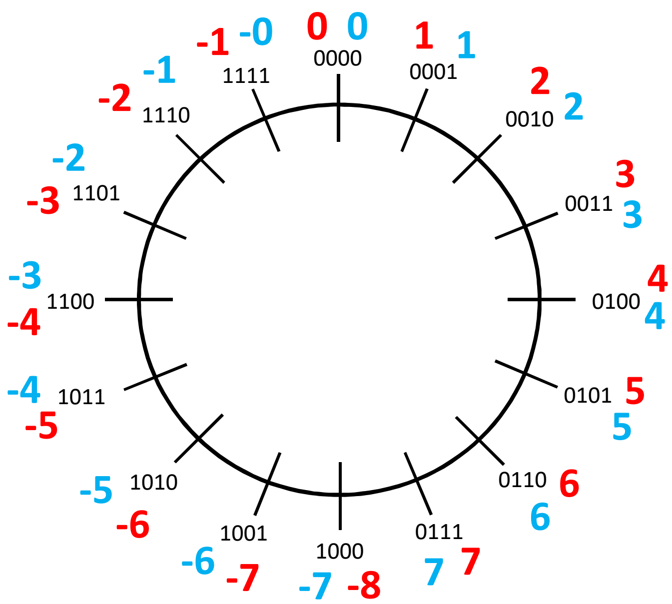
$-(2^{(n-1)}) <= N <= 2^{(n-1)}-1$

| | |
|---|---|
| n = 4 | -8 <= N <= 7<br>-7 <= N <= 7 |

$N^{(1)} = (2^n - 1) - (N)_2$

$-(2^{(n-1)}-1) <= N <= 2^{(n-1)}-1$

One's complement



# Adder/subtraction of two unsigned numbers

Data inputs:

A and B: unsigned numbers with 3-bit

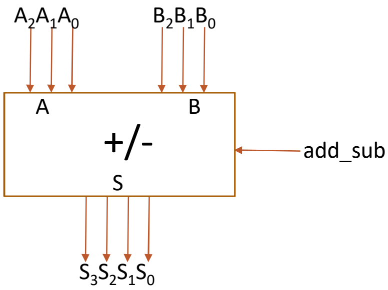
Data output:

S: 4-bit result using 1's or 2's complement

Control input:

add_sub: if add_sub=0 then S = A+B

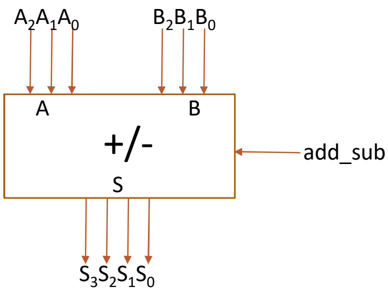
else S = A - B




# Adder/subtraction of two unsigned numbers using one's complement

if add_sub=0 then S = A+B

else S = A - B

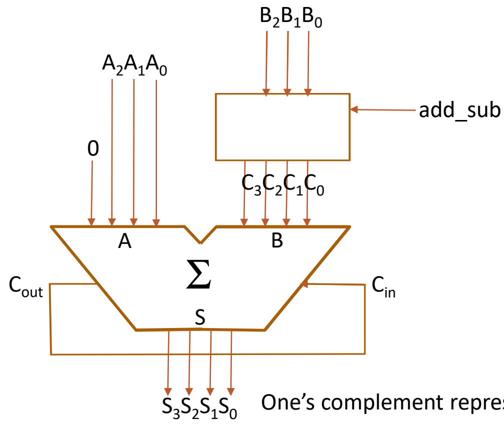


# Adder/subtraction of two unsigned numbers using one's complement

if add_sub=0 then S = A+B
else S = A - B

if add_sub=0 then $C_3C_2C_1C_0 = 0B_2B_1B_0$
else $C_3C_2C_1C_0 = (0B_2B_1B_0)^{(1)}$

$(0B_2B_1B_0)^{(1)} = 1\bar{B}_2\bar{B}_1\bar{B}_0$




One's complement representation

# Adder/subtraction of two unsigned numbers using one's complement

if add_sub=0 then S = A+B
else S = A - B

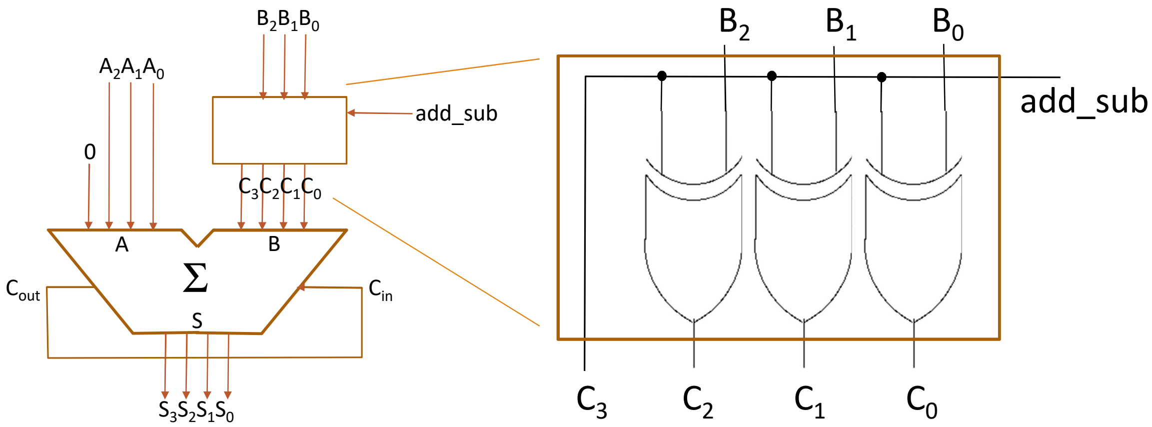
if add_sub=0 then $C_3C_2C_1C_0 = 0B_2B_1B_0$
else $C_3C_2C_1C_0 = (0B_2B_1B_0)^{(1)}$

$(0B_2B_1B_0)^{(1)} = 1\bar{B}_2\bar{B}_1\bar{B}_0$

| add_sub | $B_i$ | $C_i$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$C_i = add\_sub \oplus B_i$

# Adder/subtraction of two unsigned numbers using one's complement

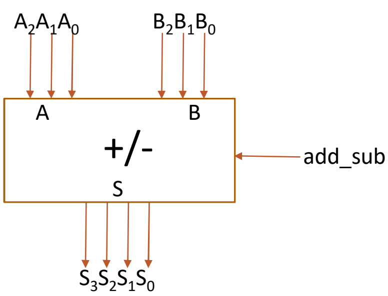



One's complement representation

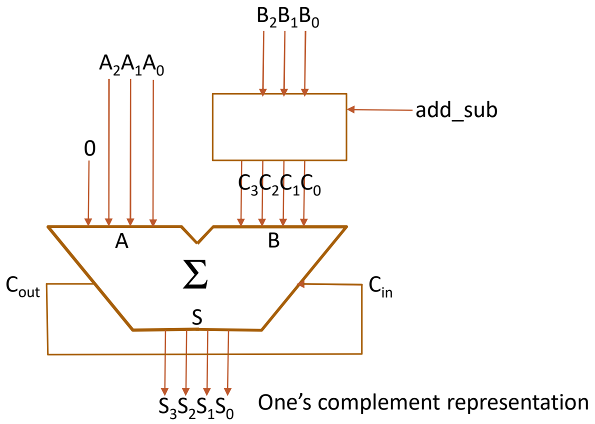
# Adder/subtraction of two unsigned numbers using two's complement

if add_sub=0 then S = A+B
else S = A - B

$(N)^{(2)} = (N)^{(1)} + 1$



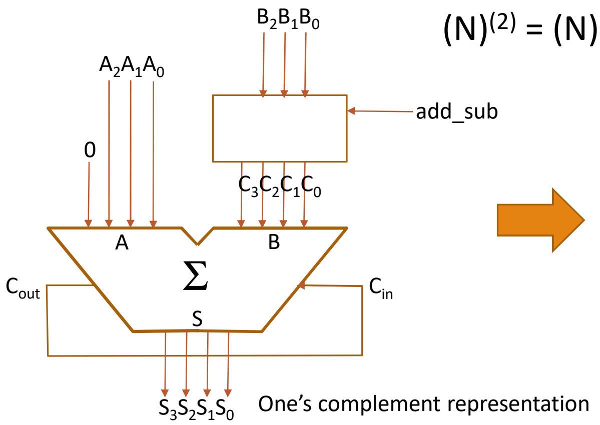
# Adder/subtraction of two unsigned numbers using two’s complement

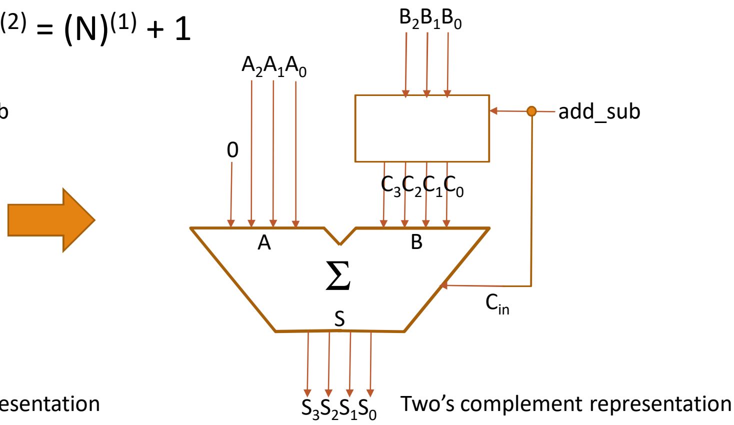



$$(N)^{(2)} = (N)^{(1)} + 1$$

# Adder/subtraction of two unsigned numbers using two’s complement

$$(N)^{(2)} = (N)^{(1)} + 1$$



# Adder/subtraction of two unsigned numbers using two's complement