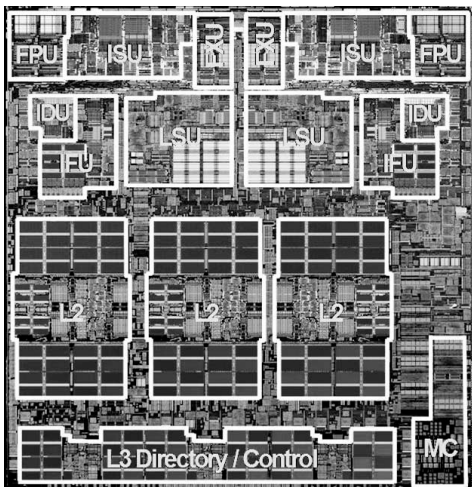




# Università degli Studi di Cassino



Corso di  
Calcolatori Elettronici I

*Reti combinatorie*

Anno Accademico 2007/2008

Francesco Tortorella

# Reti combinatorie

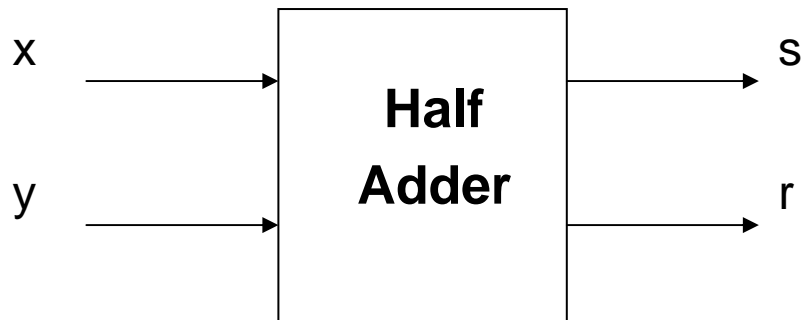
- una rete combinatoria è un circuito logico avente  $n$  ingressi  $(x_1, x_2, \dots, x_n)$  ed  $m$  uscite  $(y_1, y_2, \dots, y_m)$ , ciascuno dei quali assume valori binari (0/1), e tale che a ciascuna combinazione degli ingressi corrisponde un'unica combinazione delle uscite.
- da un punto di vista logico, ogni uscita può essere definita come una funzione booleana degli ingressi  
$$y_i = y_i(x_1, x_2, \dots, x_n).$$
- ad ogni istante, il valore delle uscite dipende unicamente dal valore assunto dagli ingressi nello stesso istante.

# Addizionatore

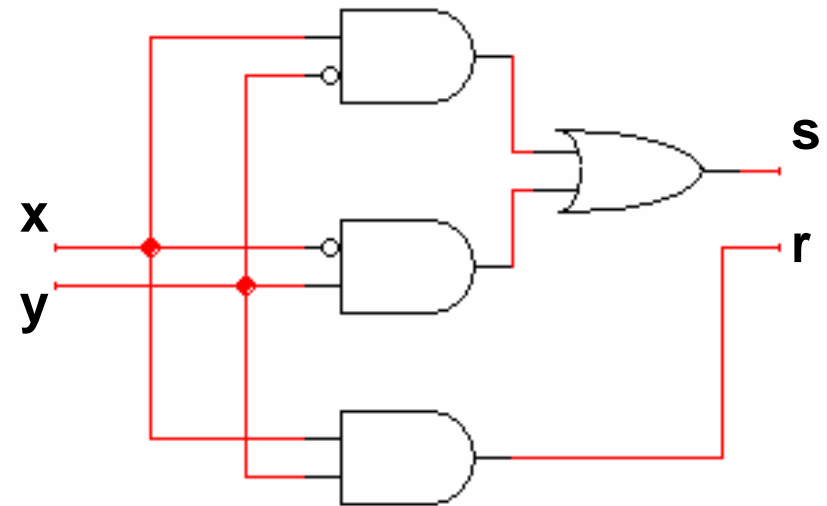
Esegue l'addizione di cifre binarie fornendo in uscita la cifra somma e la cifra riporto. Sono possibili due schemi:

- semiaddizionatore (half adder)
  - **2 cifre in ingresso**
- addizionatore completo (full adder)
  - **2 cifre in ingresso + carry in ingresso**

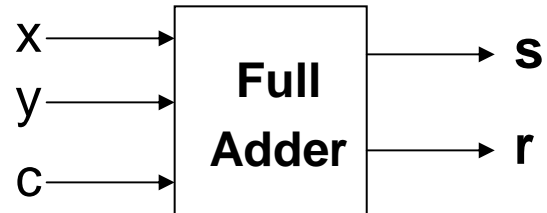
# Half adder



x	y	s	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



# Full Adder



x	y	c	s	r
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

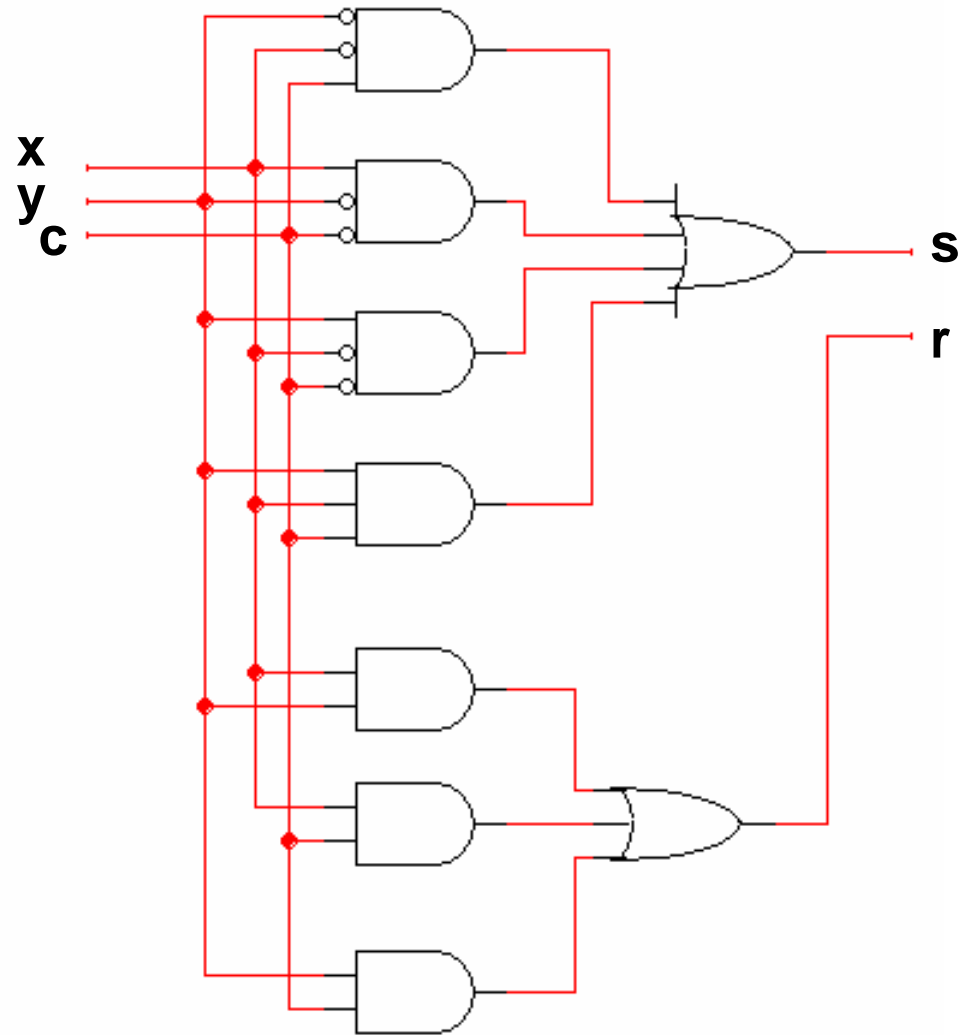
		xy			
		00	01	11	10
c	0		1		1
	1	1		1	

		xy			
		00	01	11	10
c	0			1	
	1		1	1	1

$$s = !x!yc + !xy!c + xyc + x!y!c$$

$$r = xy + yc + xc$$

# Full Adder – sintesi diretta



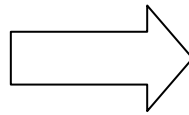
## Full Adder – sintesi per decomposizione

$$s = !x!yc + !xy!c + xyc + x!y!c = (!x!y + xy)c + (!xy + x!y)!c$$

$$r = xy + yc + xc = xy + !xyc + xyc + xyc + x!yc = xy + (!xy + x!y)c$$

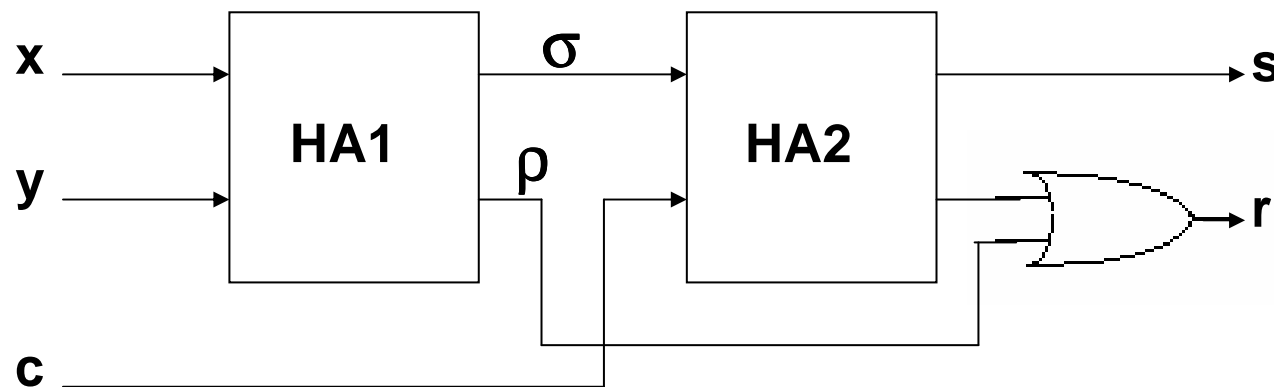
$$\sigma = !xy + x!y$$

$$\rho = xy$$



$$s = !\sigma c + \sigma!c$$

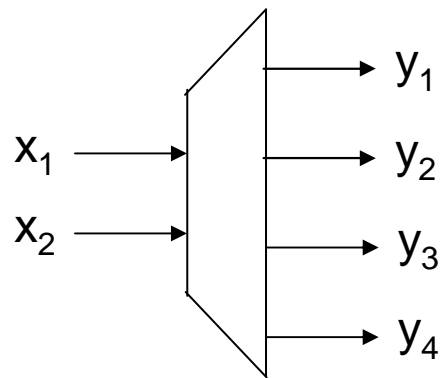
$$r = \rho + \sigma c$$



# Decodificatore

Rete combinatoria ad  $n$  ingressi ed a  $2^n$  uscite. Per ogni combinazione degli ingressi, solo una uscita assume valore 1 mentre le altre sono uguali a 0.

decodificatore 1/4



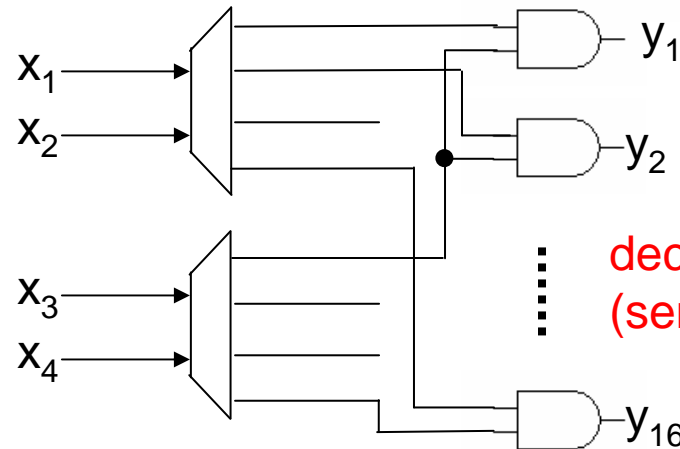
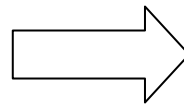
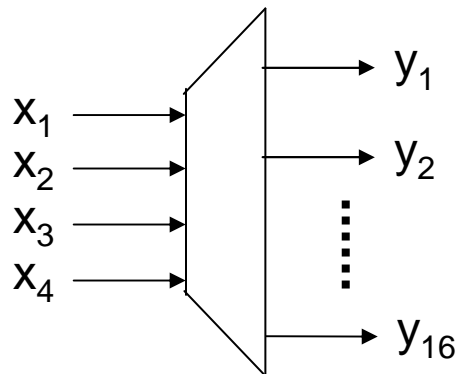
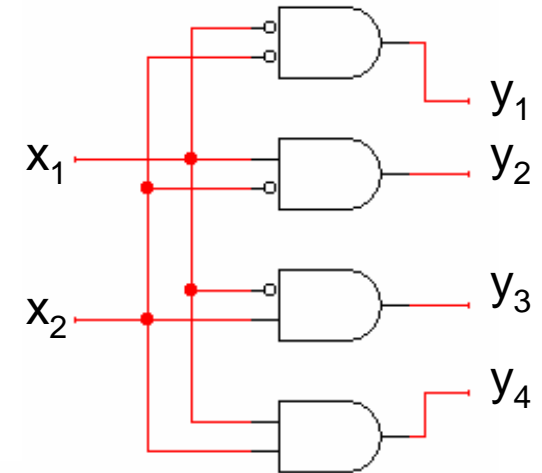
$x_1$	$x_2$	$y_1$	$y_2$	$y_3$	$y_4$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$y_1 = \neg x_1 \neg x_2$$

$$y_2 = \neg x_1 x_2$$

$$y_3 = x_1 \neg x_2$$

$$y_4 = x_1 x_2$$

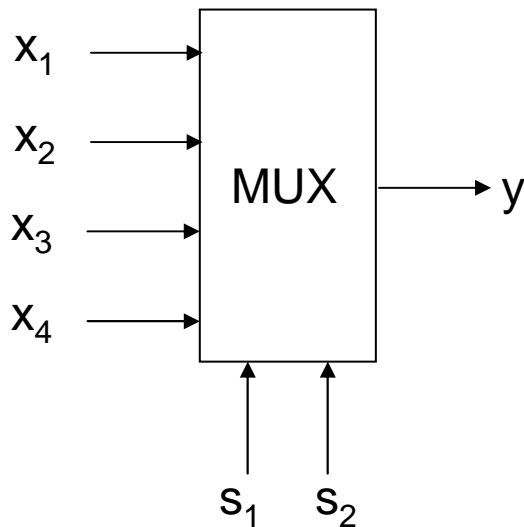


decodificatore 1/16  
(semiselezione)



# Multiplexer

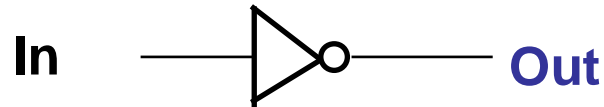
Rete combinatoria i cui ingressi sono divisi in *ingressi dati* ( $n$ ) e *ingressi selezione* ( $\lceil \log_2 n \rceil$ ), mentre l'uscita è unica ed è uguale ad uno degli ingressi dati, scelto sulla base degli ingressi selezione.



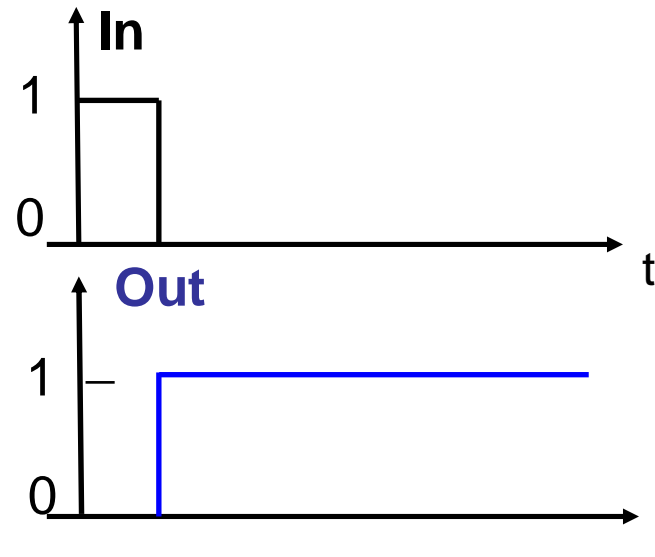
$s_1$	$s_2$	$y$
0	0	$x_1$
0	1	$x_2$
1	0	$x_3$
1	1	$x_4$

**realizzazione ?**

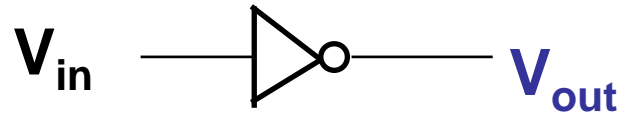
# Il ritardo nelle reti combinatorie



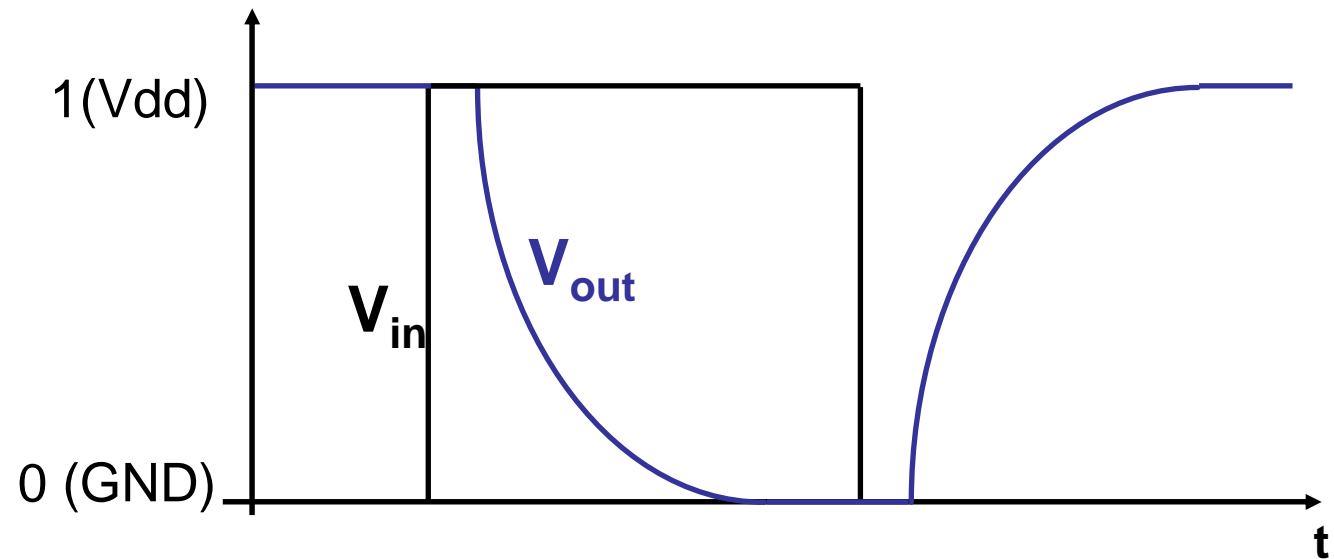
Comportamento ideale:

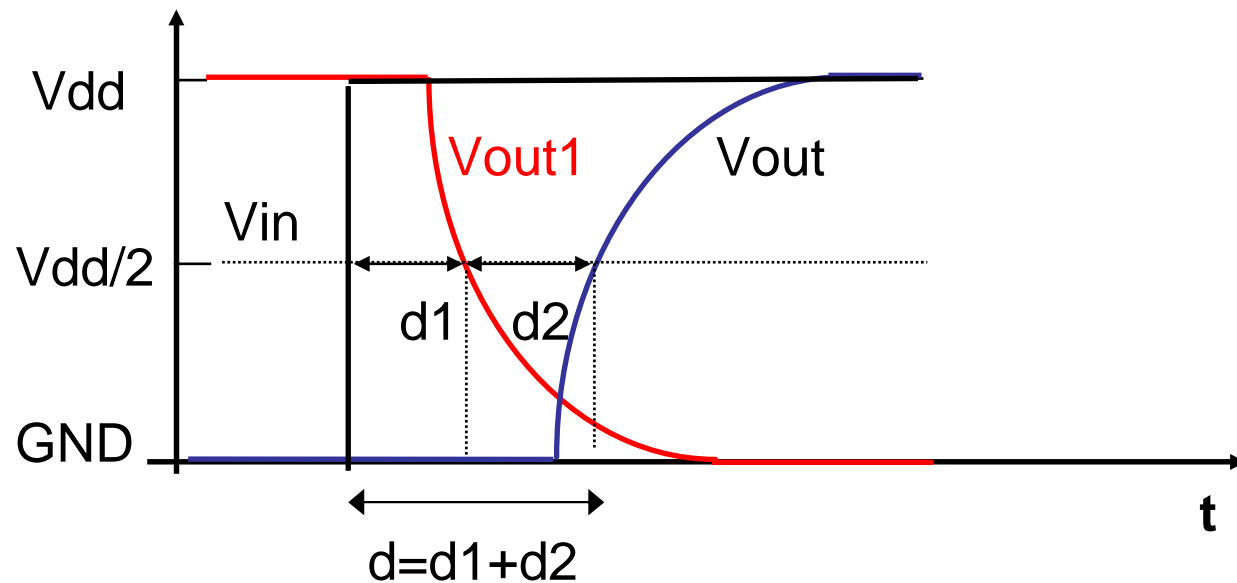
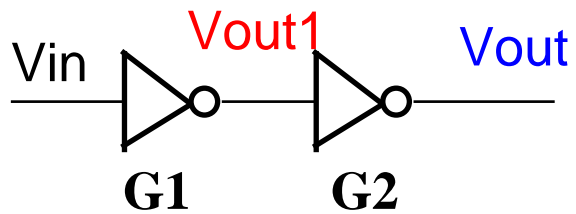


La realizzazione circuitale delle porte logiche non rispetta questo comportamento a causa degli effetti capacitivi presenti, che generano dei transitori.

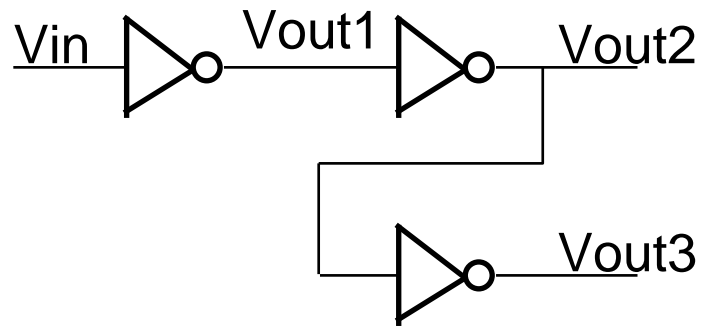


Comportamento  
reale:





Con due porte in cascata, il ritardo di propagazione è uguale alla somma dei ritardi.



Il ritardo sulle due uscite non è lo stesso.

$$\text{Ritardo}(V_{in} \rightarrow V_{out2}) = \text{Ritardo}(V_{in} \rightarrow V_{out1}) + \text{Ritardo}(V_{out1} \rightarrow V_{out2})$$

$$\text{Ritardo}(V_{in} \rightarrow V_{out3}) = \text{Ritardo}(V_{in} \rightarrow V_{out1}) + \text{Ritardo}(V_{out1} \rightarrow V_{out2}) + \text{Ritardo}(V_{out2} \rightarrow V_{out3})$$

Percorso critico (*Critical path*): il percorso a ritardo maggiore ( $V_{in} \rightarrow V_{out3}$ ), che caratterizza l'intero circuito.

# Progetto di una ALU

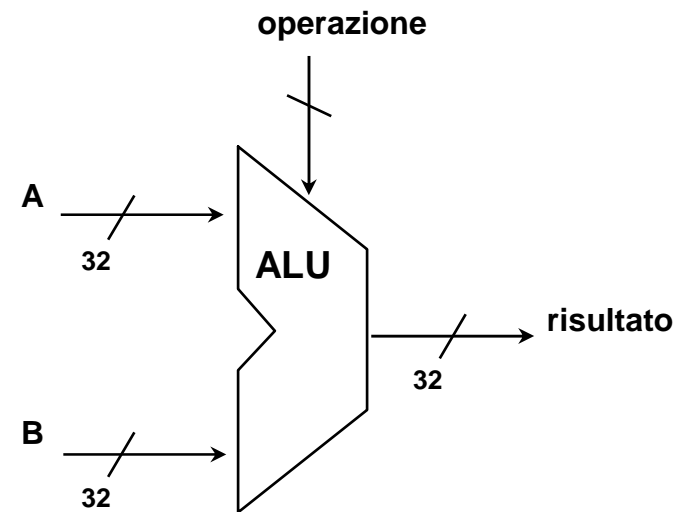
**ALU** n. [*Arthritic Logic Unit* or (rare) *Arithmetic Logic Unit*]

*A random-number generator supplied as standard with all computer systems.*

Stan Kelly-Bootle, *The Devil's DP Dictionary*

Compiti dell'ALU:

- esecuzione delle operazioni aritmetiche (addizioni, sottrazioni, moltiplicazioni,...)
- esecuzione delle operazioni logiche (AND, OR,...)
- produzione del risultato e modifica dei flag



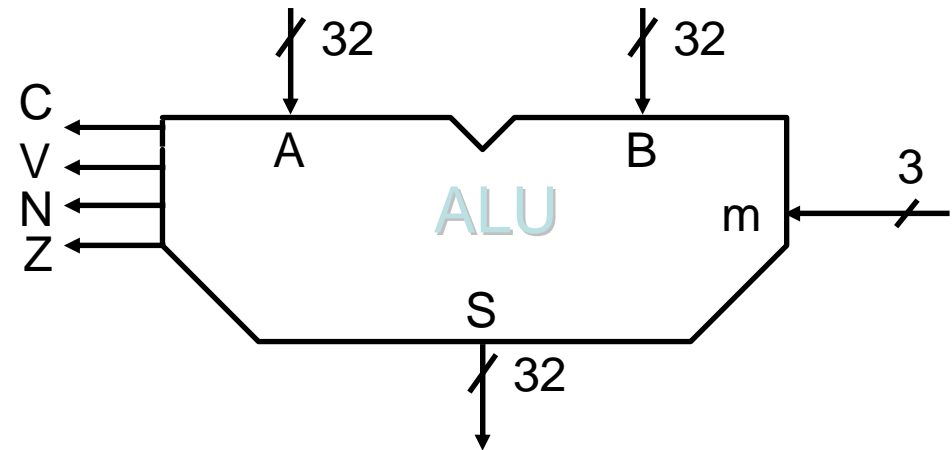
# Specifiche di progetto

- 2 ingressi da 32 bit
- 1 uscita da 32 bit
- almeno 6 operazioni possibili (add, sub, adc, and, or, not)
- gestione dei flag C, V, N, Z

## Quale approccio progettuale seguire ?

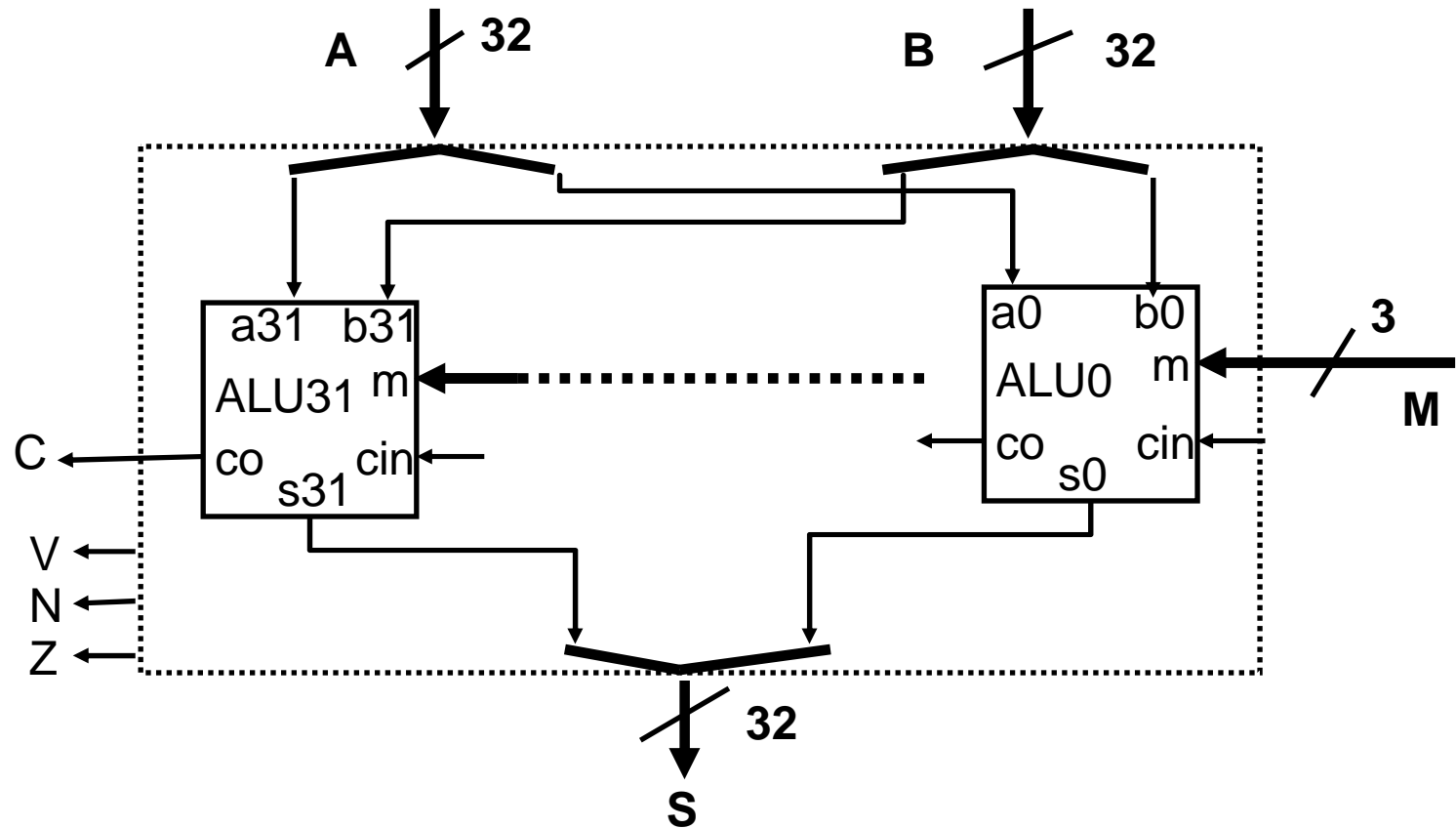
**Forza bruta:** rete combinatoria a 68 ingressi e 36 uscite.

**Approccio top-down:** decomposizione in sottoproblemi, impiego di componenti già noti.





## Decomponiamo l'ALU a 32 bit in 32 ALU da 1 bit (bit slice ALU)

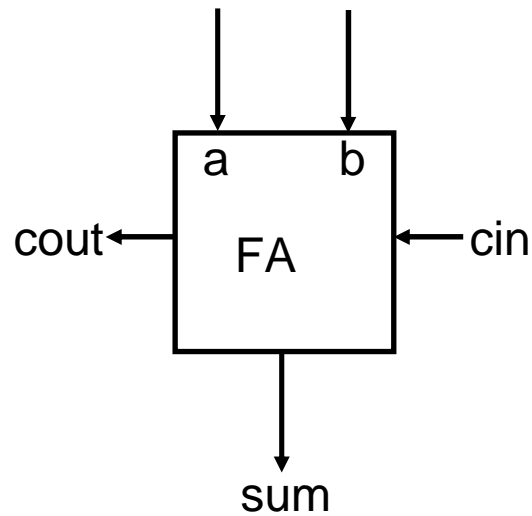




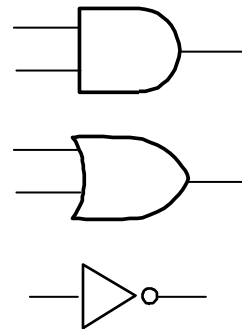
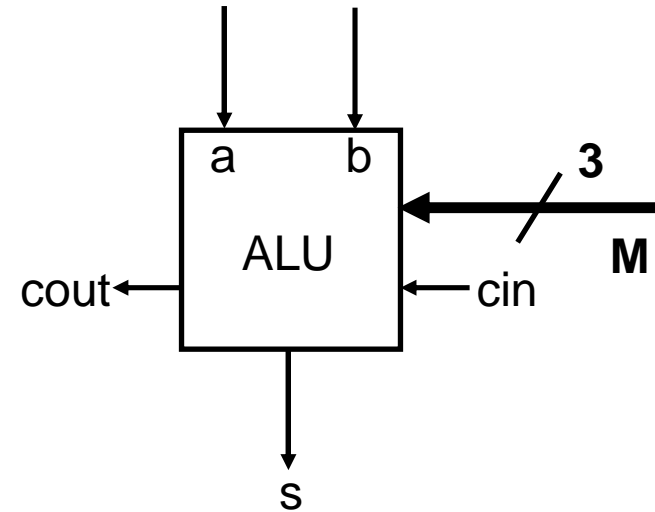
# ALU da 1 bit

Come realizzarla ?

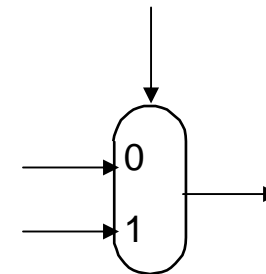
Quali tipi di componenti noti ci possono essere utili ?



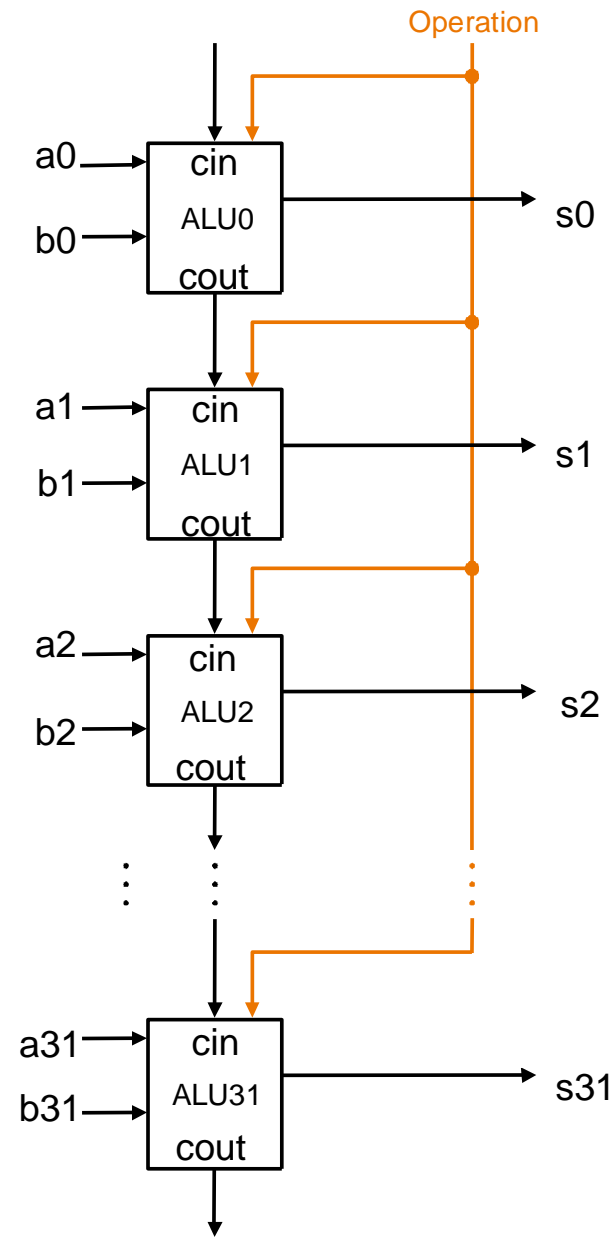
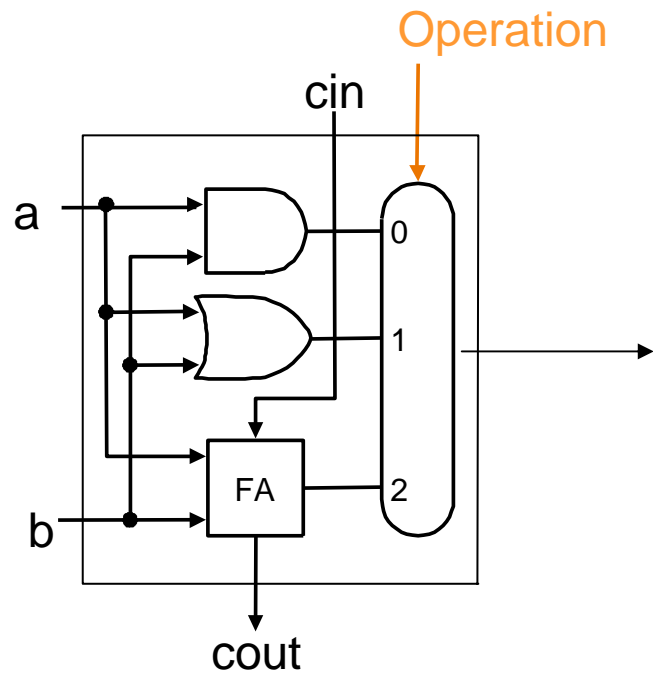
Full Adder



Porte logiche



Multiplexer

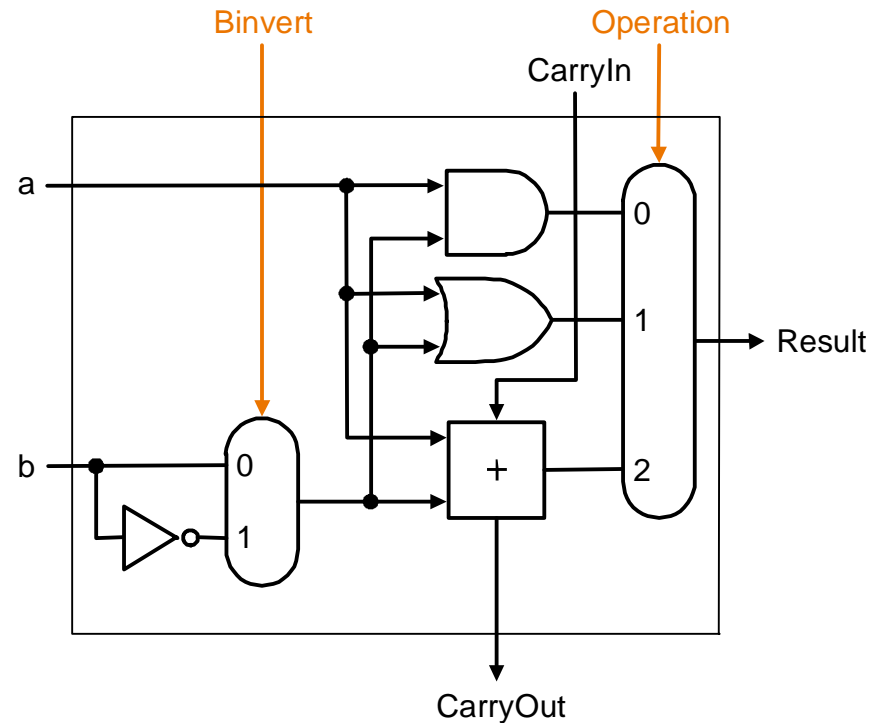


Operazioni possibili: somma,  
AND, OR.

# Realizzazione della sottrazione

uso della rappresentazione per complementi

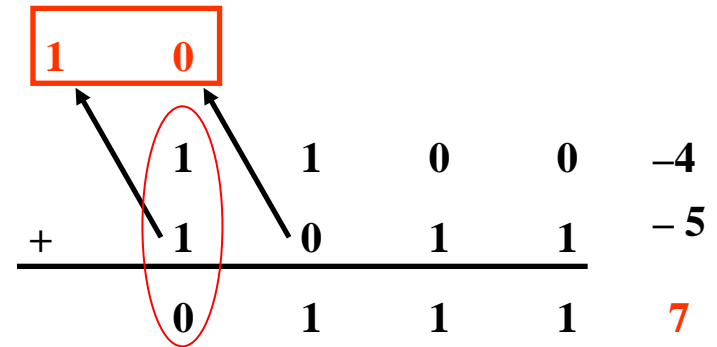
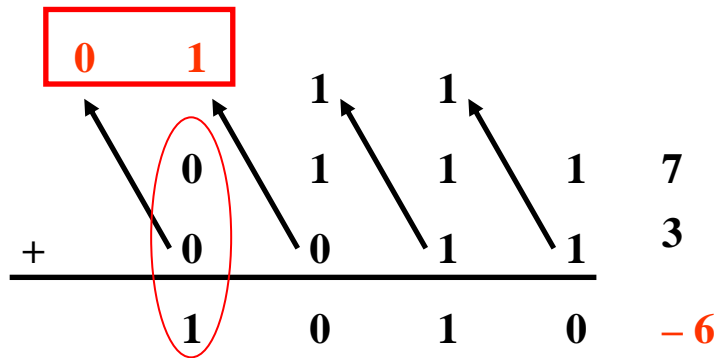
- $A - B = A + \bar{B}$
- $\bar{B} = !B + 1$



# Gestione dell'overflow

- OVERFLOW: il risultato dell'operazione non è rappresentabile ( $> \text{MAX}$  o  $< \text{MIN}$ )
- Si verifica solo in presenza di operandi con lo stesso segno
- Criteri per la rilevazione:
  - segno del risultato diverso dal segno degli operandi
  - riporto entrante nel MSB diverso dal riporto uscente dal MSB

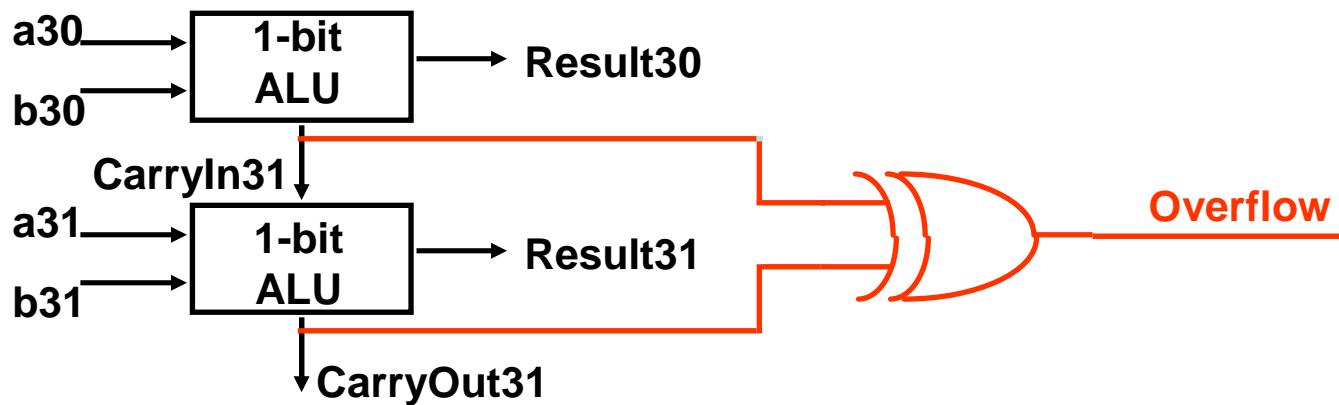
Esempio (a 4 bit):



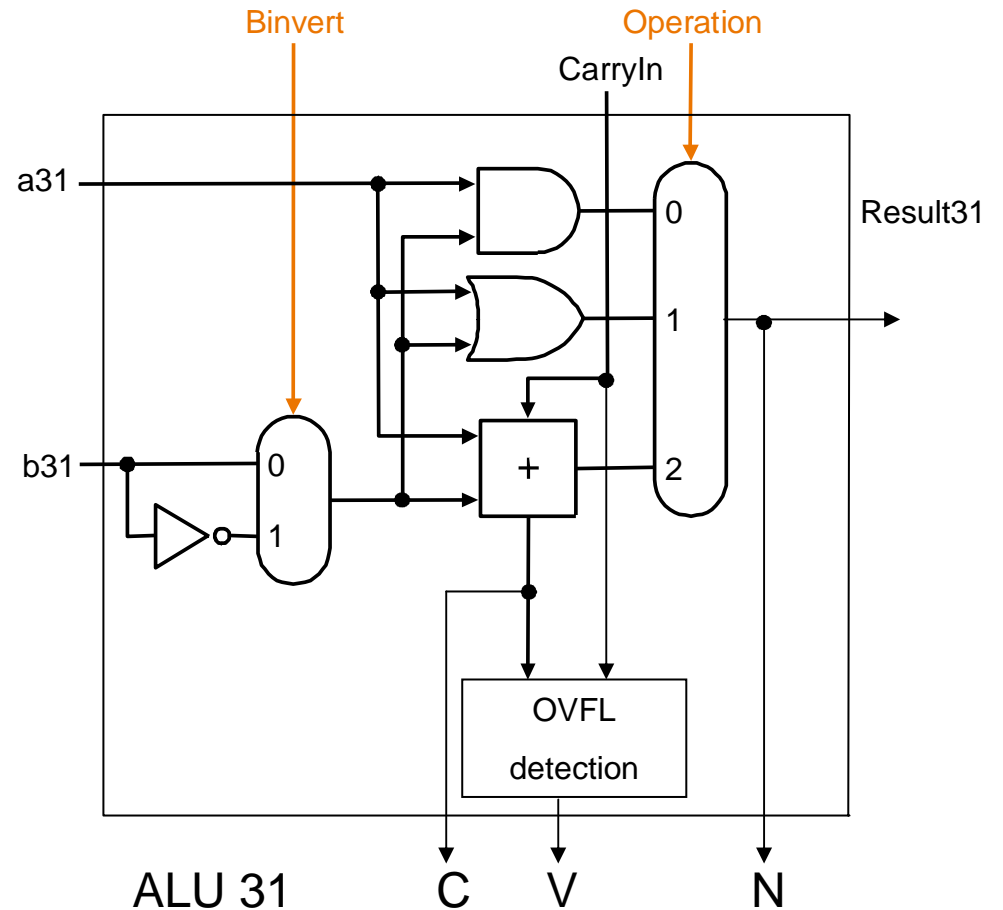
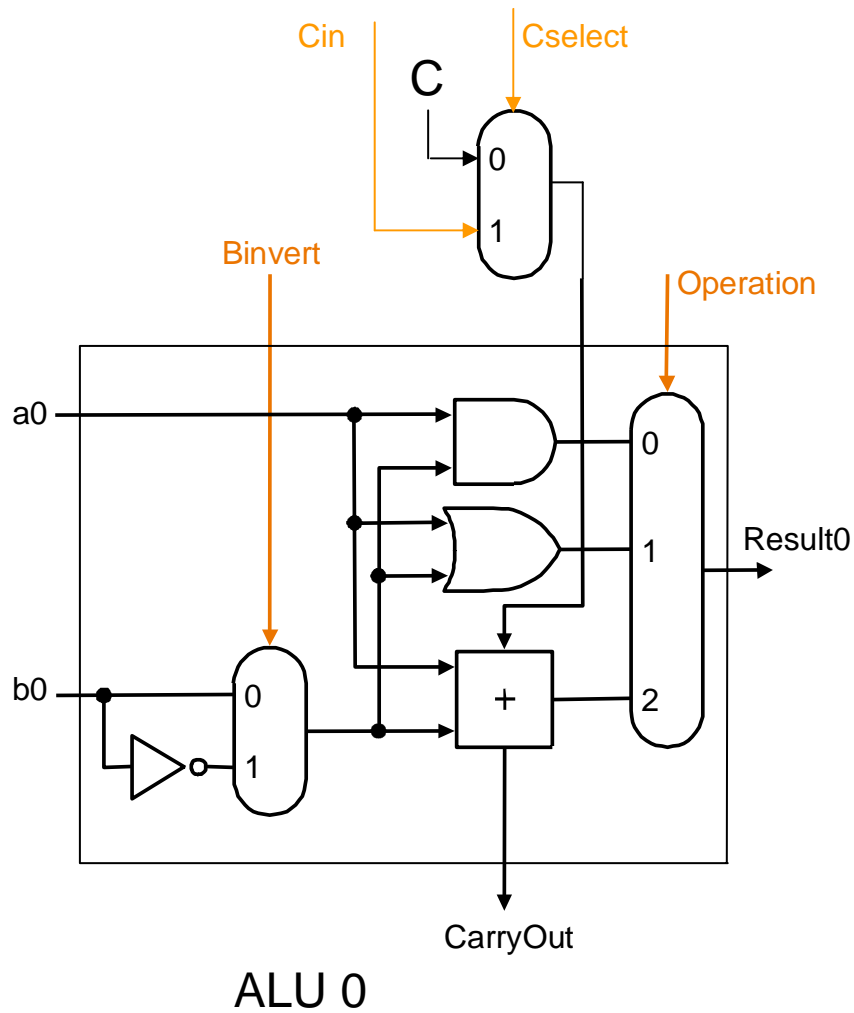
# Gestione dell'overflow

$$\text{Overflow} = \text{CarryIn}[31] \text{ XOR } \text{CarryOut}[31]$$

Cin31	Cout31	Overflow
0	0	0
0	1	1
1	0	1
1	1	0

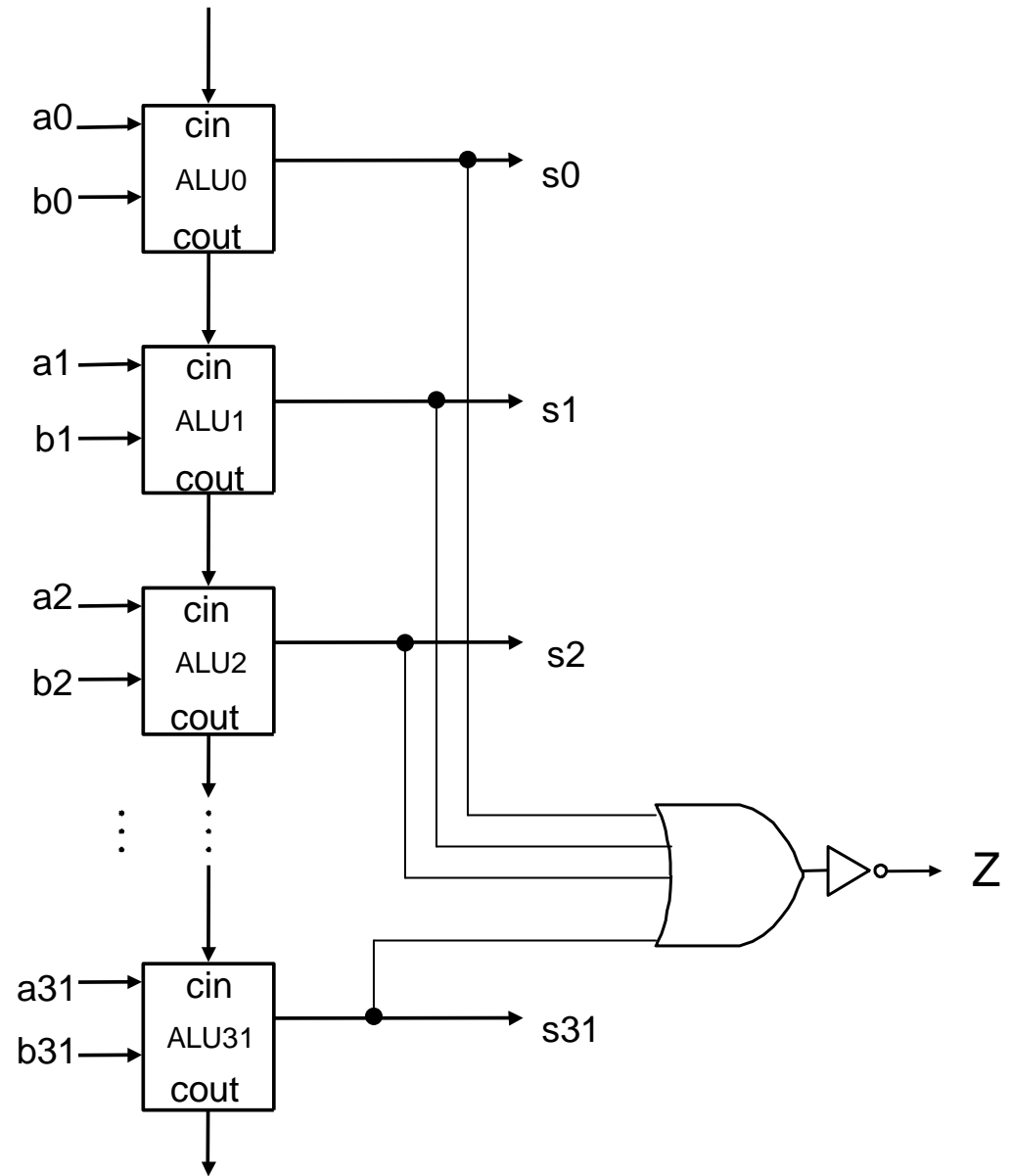


# Modifiche da apportare alle ALU 0 e 31



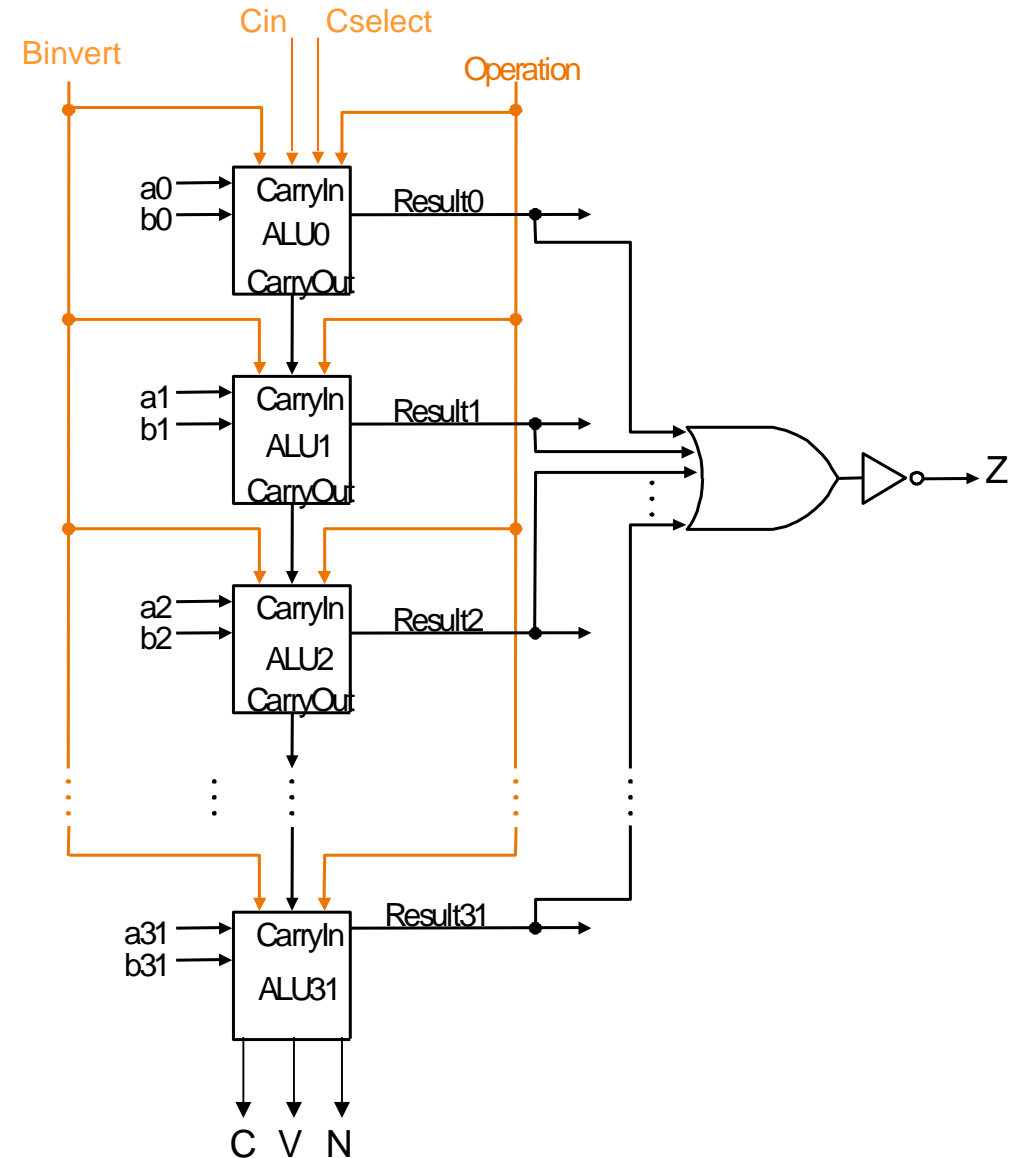
## Gestione del flag Z

Il flag Z vale 1  
quando il risultato  
è nullo!



# Progetto complessivo

	Op	Binvert	Cin	Csel
add	10	0	0	1
adc	10	0	X	0
sub	10	1	1	1
and	00	0	X	X
or	01	0	X	X
not	01	1	X	X





# Valutazione del progetto

Quanto vale il percorso critico?  $n \cdot CP$

Problema:  
il ripple carry adder è lento.

