Il Livello Logico-Digitale

Blocchi funzionali combinatori

18 -10-2015

Blocchi funzionali combinatori

- Esiste una ben nota e ormai stabilizzata libreria di blocchi funzionali predefiniti di tipo combinatorio che contiene i blocchi per tutte le funzioni combinatorie di base
  - La libreria contiene anche blocchi funzionali di tipo sequenziale

- I tipici blocchi funzionali combinatori sono:

- Multiplexer
- Demultiplexer
- Decoder (decodificatore)
- Confrontatore
- Shifter combinatorio
- Half adder e Full adder
- Addizionatore a n bit
- ALU or, and, not e somma

Multiplexer

- Il blocco funzionale multiplexer ha:
  - $n \geq 1$ ingressi di selezione
  - $2^n \geq 2$ ingressi dati
  - un'uscita
- Gli ingressi dati sono numerati a partire da 0: k = 0, 1, 2, …, $2^n - 1$
- Se sugli ingressi di selezione è presente il numero binario k, il k^{esimo} ingresso dati viene inviato in uscita

Multiplexer a 1 ingresso di selezione (1)

- 1 ingresso di selezione, 2 ingressi dati, un'uscita

I0	I1	Sel (Ctrl)	OUT
D1	D2	0	D1
D1	D2	1	D2

Tabella della verità

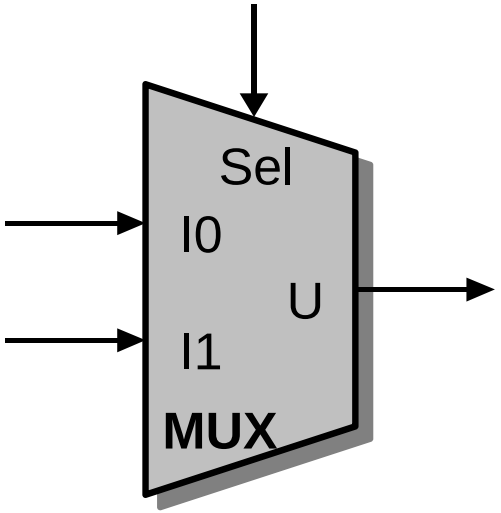
I0	I1	Sel (Ctrl)	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Multiplexer a 1 ingresso di selezione (2)

- 1 ingresso di selezione, 2 ingressi dati, un'uscita

OUT = !Sel I0 !I1 + !Sel I0 I1

+ Sel !I0 I1 + Sel I0 I1



Multiplexer a 2 ingressi di selezione

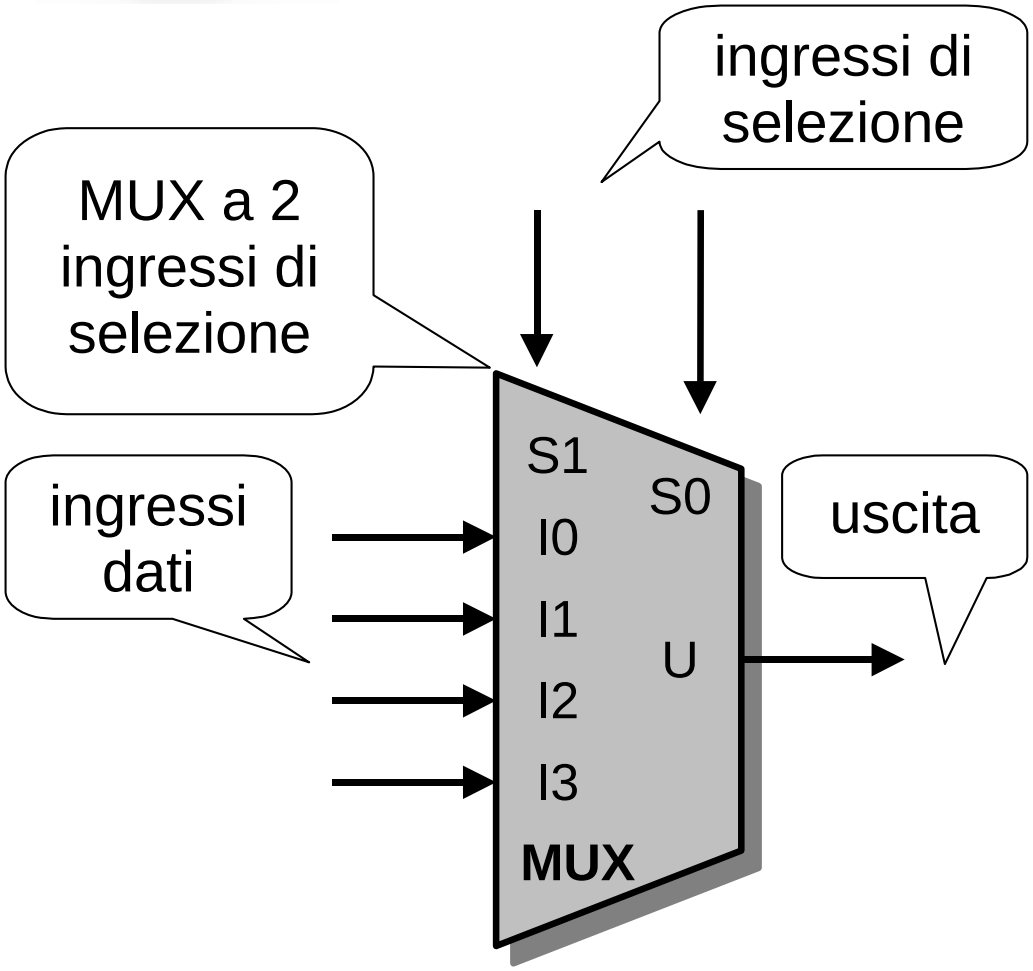
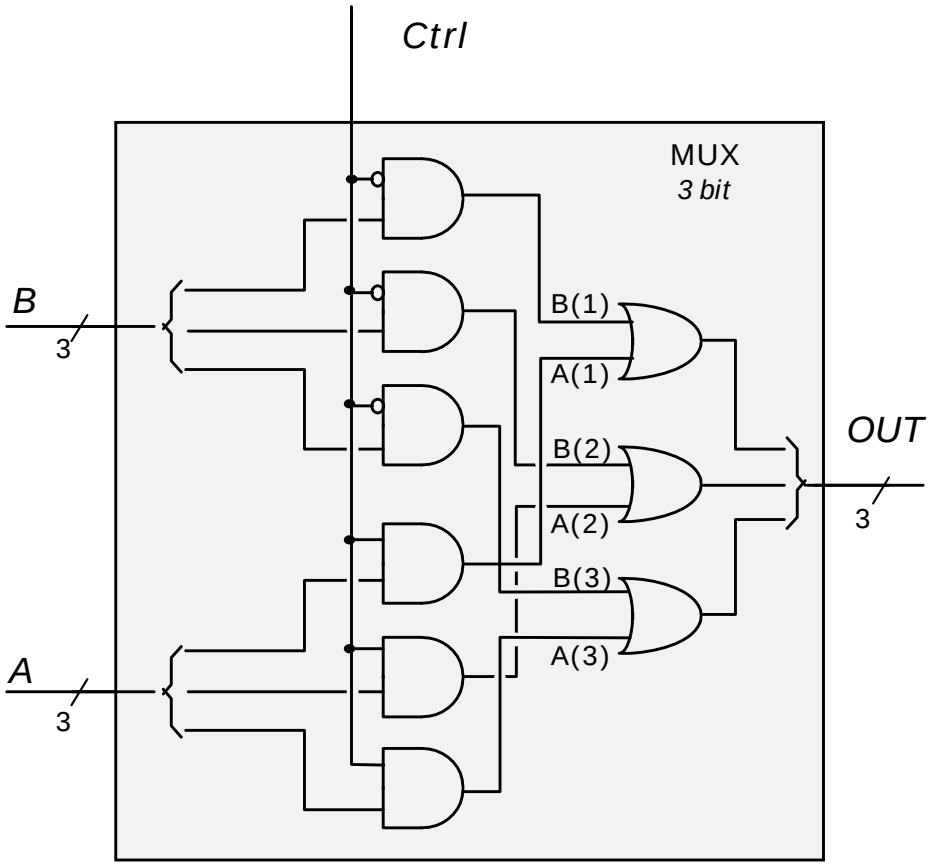


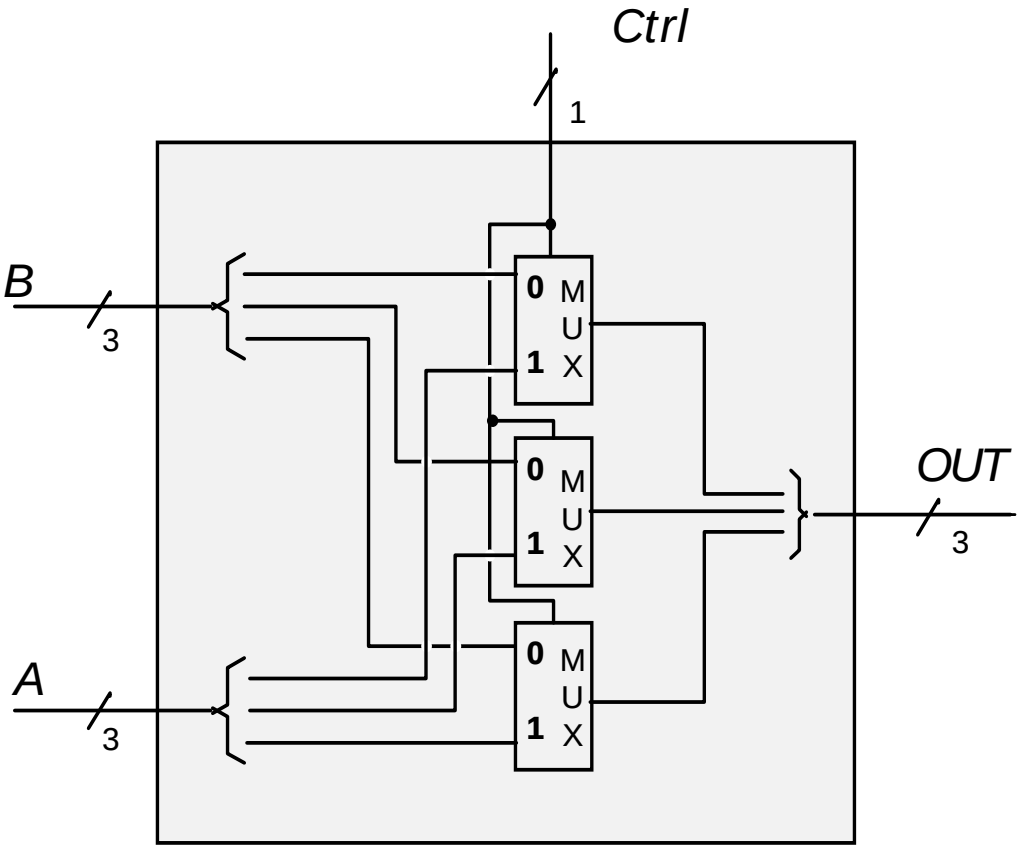
Tabella delle verità

# riga	S1	S0	I0	I1	I2	I3	U
0	0	0	0	X	X	X	0
1	0	0	1	X	X	X	1
2	0	1	X	0	X	X	0
3	0	1	X	1	X	X	1
4	1	0	X	X	0	X	0
5	1	0	X	X	1	X	1
6	1	1	X	X	X	0	0
7	1	1	X	X	X	1	1

Multiplexer a 2 ingressi dati da k bit

Esempio: k=3





Demultiplexer

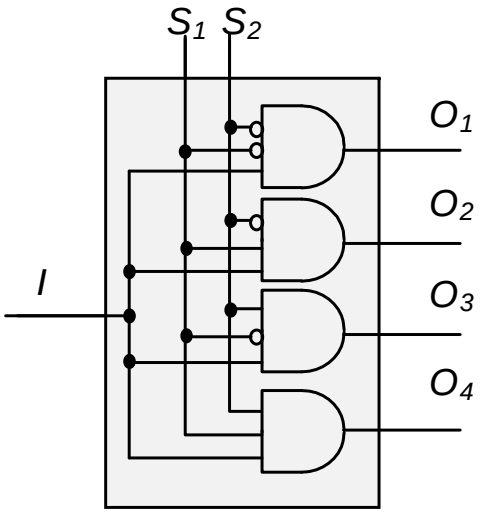
- Il blocco funzionale demultiplexer ha:
  - $n \geq 1$ ingressi di selezione
  - 1 ingresso dati
  - $2^n \geq 2$ uscite
- Le uscite sono numerate a partire da 0: $k = 0, 1, 2, \ldots, 2^n - 1$
- Se sugli ingressi di selezione è presente il numero binario k, l'ingresso dati viene inviato alla k^{esima} uscita, le rimanenti sono a 0

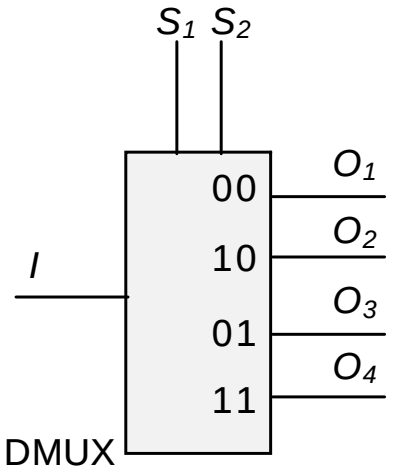
Demultiplexer a 2 ingressi di selezione

Ingressi	Selezione		Uscite			
I	S_1	S_2	O_1	O_2	O_3	O_4
D	0	0	D	0	0	0
D	1	0	0	D	0	0
D	0	1	0	0	D	0
D	1	1	0	0	0	D

$$O_1 = !S_1!S_2I \quad O_3 = !S_1S_2I$$

$$O_2 = S_1!S_2I \quad O_4 = S_1S_2I$$

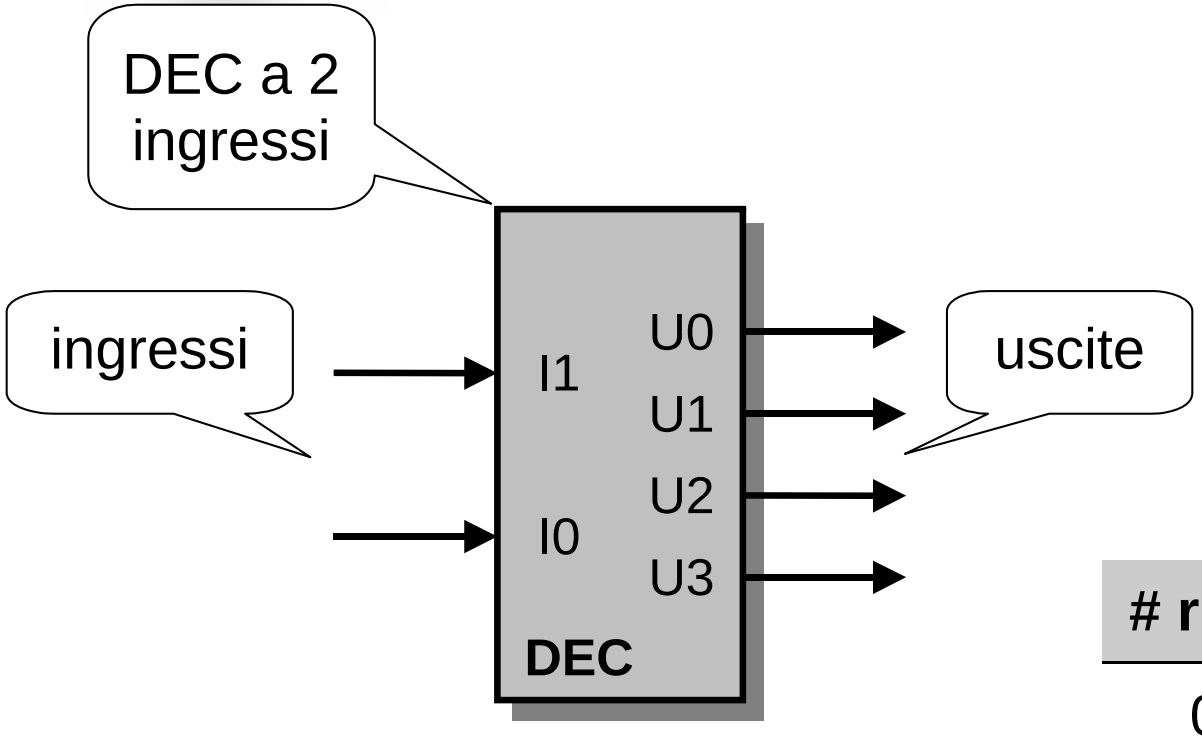




Decodificatore (decoder)

- Il blocco funzionale decodificatore ha:
  - $n \geq 1$ ingressi
  - $2^n \geq 2$ uscite
- Le uscite sono numerate a partire da 0: $k = 0, 1, 2, \ldots, 2^n - 1$
- Se sugli ingressi è presente il numero binario k, la k^{esima} uscita assume il valore 1 e le restanti uscite assumono il valore 0

Decodificatore



Decodificatore a 2 ingressi

Tabella delle verità

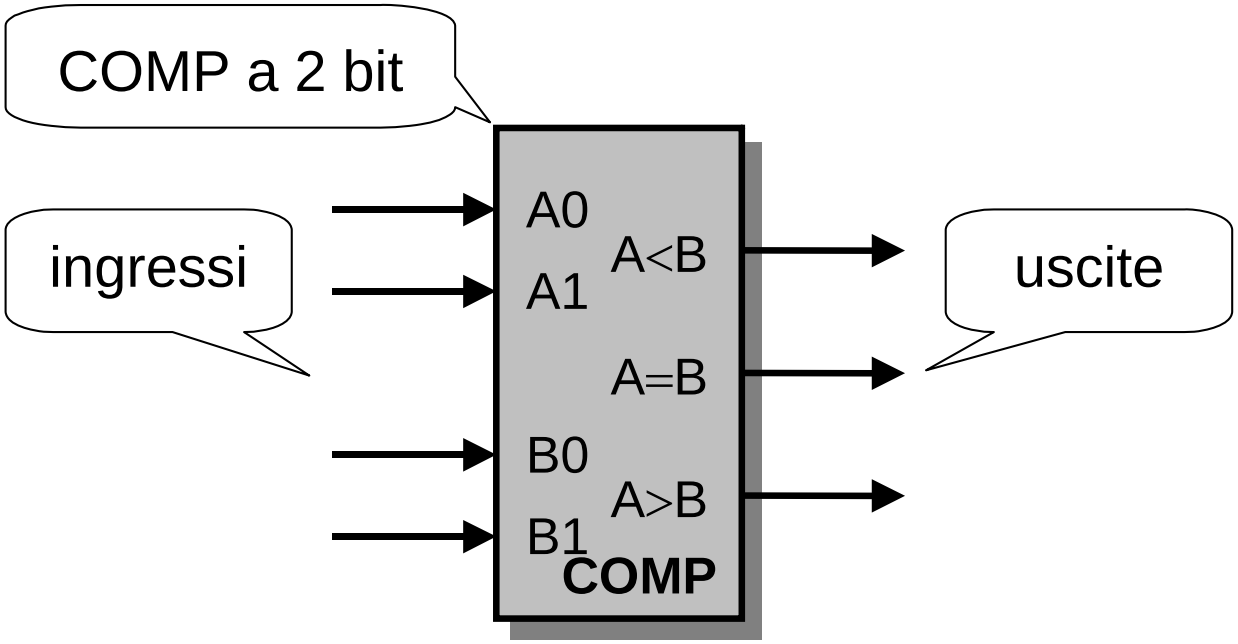
# riga	I1	I0	U0	U1	U2	U3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

Confrontatore (comparator)

- Il blocco funzionale confrontatore ha:
  - due gruppi A e B di ingressi da $n \geq 1$ bit ciascuno
  - tre uscite: minoranza $A < B$, uguaglianza $A = B$ e maggioranza $A > B$

- Il blocco confronta i due numeri binari A e B da n bit presenti sui due gruppi di ingressi, e attiva (a 1) l'uscita corrispondente all'esito del confronto

Confrontatore



Confrontatore di numeri a 2 bit

Tabella delle verità

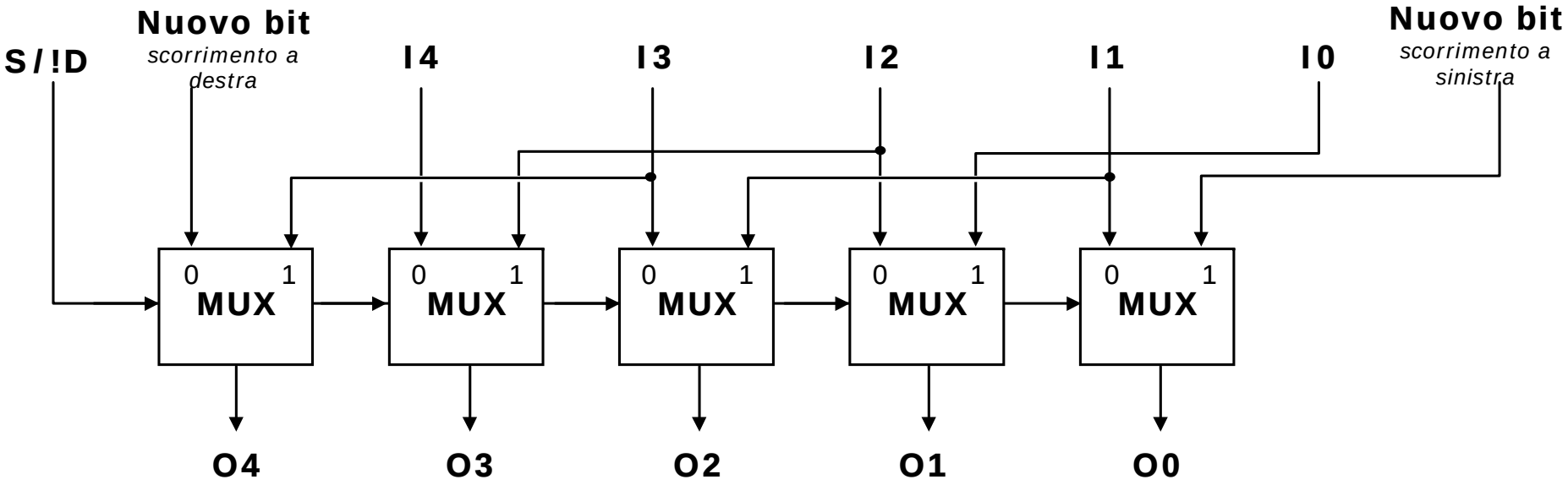
# riga	A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	0	1	0
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	0	0	1
9	1	0	0	1	0	0	1
10	1	0	1	0	0	1	0
11	1	0	1	1	1	0	0
12	1	1	0	0	0	0	1
13	1	1	0	1	0	0	1
14	1	1	1	0	0	0	1
15	1	1	1	1	0	1	0

Shifter combinatorio

- Il blocco funzionale shifter ha:
  - $n \geq 1$ ingressi
  - 1 ingresso per il bit aggiunto a dx (scorrimento a sinistra)
  - 1 ingresso per il bit aggiunto a sx (scorrimento a destra)
  - 1 ingresso di controllo che comanda lo scorrimento a destra o a sinistra
  - $n \geq 1$ uscite
- Uscite:
  - scorrimento a dx: bit aggiunto a sx + ingressi shiftati di una posizione a dx (viene "perso" il bit meno significativo degli ingressi)
  - scorrimento a sx: bit aggiunto a dx + ingressi shiftati di una posizione a sx (viene "perso" il bit più significativo degli ingressi)

- Si noti che se si considerano gli ingressi come un valore numerico espresso in binario naturale
  - lo scorrimento a dx (con bit aggiunto a sx = 0) equivale ad una **divisione per 2**
  - lo scorrimento a sx (con bit aggiunto a dx = 0) equivale ad una **moltiplicazione per 2**

Shifter combinatorio 5 ingressi



Blocchi aritmetici fondamentali

- Rappresentazione dei numeri in binario naturale intero su $k \geq 1$ bit
  - Addizionatore ad 1 bit
    - half adder
    - full adder
  - Addizionatore a k bit in binario naturale intero
  - Sottrattore a 1 bit
  - Sottrattore a k bit in binario naturale intero

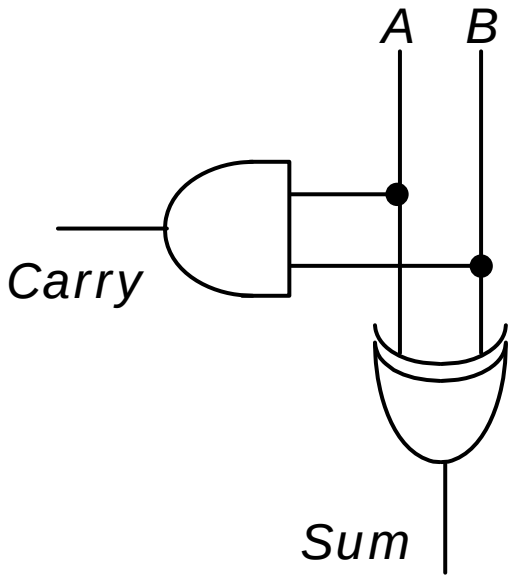
Half adder (semisommatore)

HALF-ADDER

A	*B*	*Carry*	*Sum*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$Sum = A \oplus B$

$Carry = AB$



Full adder (sommatore completo)

FULL ADDER

A	*B*	*Carry In*	*Sum*	*Carry Out*
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 = A xor B xor C_in

C_out = AB + C(A xor B)

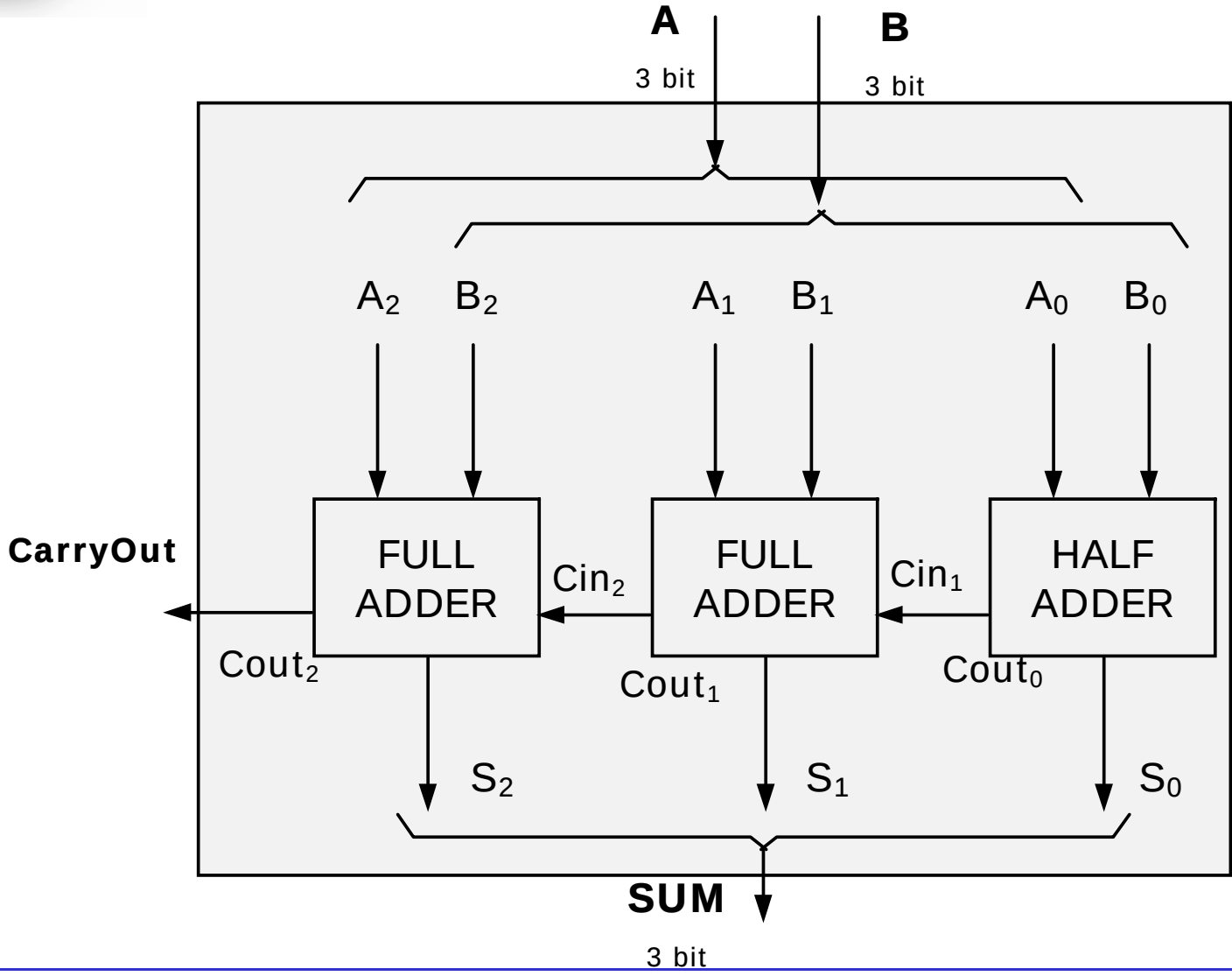
S = !A!BC + !AB!C + A!B!C + ABC =
= C (!A!B + AB) + !C (!AB + A!B)
= C !(A xor B) + !C (A xor B)

ponendo Z = A xor B
= !Z C + Z !C = Z xor C
= A xor B xor C_in

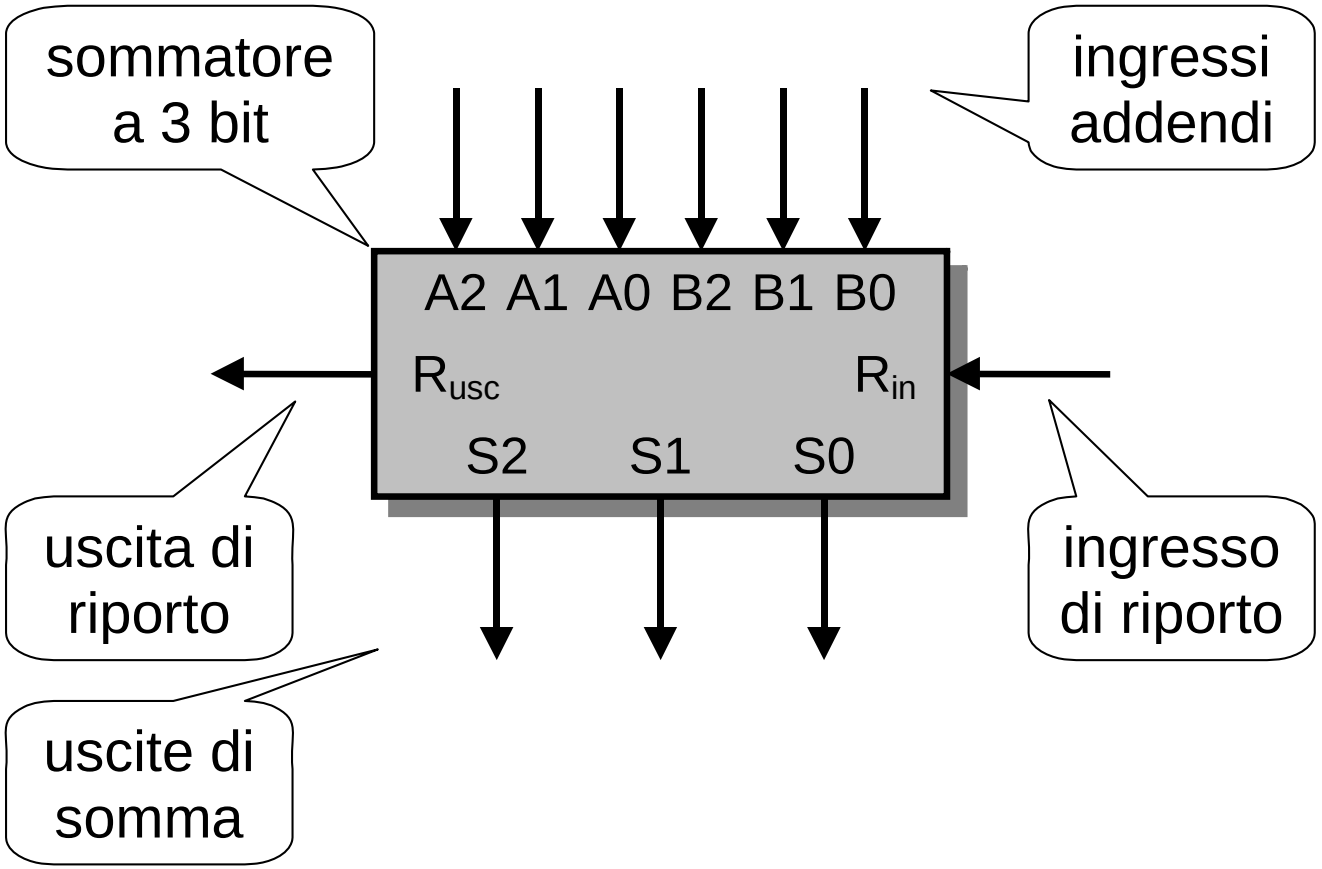
C_out = !ABC + A!BC + AB!C + ABC
= C (!AB + A!B) + AB(!C + C)
= C (A xor B) + AB

……. schema rete da ricavare …..

Addizionatore a k bit in binario naturale intero



Sommatore intero a n bit



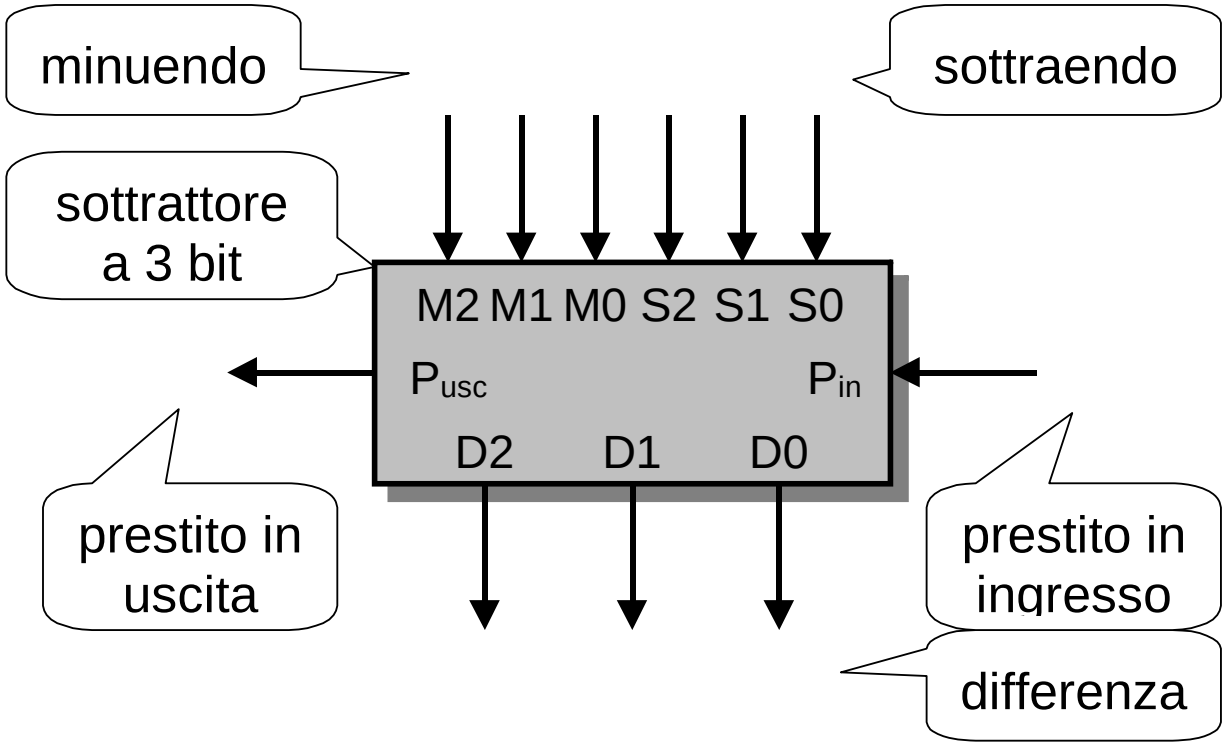
Sommatore intero binario naturale a 3 bit

Sottrattore a 1 bit e a k bit in binario naturale intero

- Ricavare le tabelle delle verità, l'espressione logica minima e la rete combinatoria che realizza
  - un semi-sottrattore a 1 bit
  - un sottrattore completo a 1 bit

- Disegnare la struttura modulare di un sottrattore a k bit

Sottrattore intero a n bit



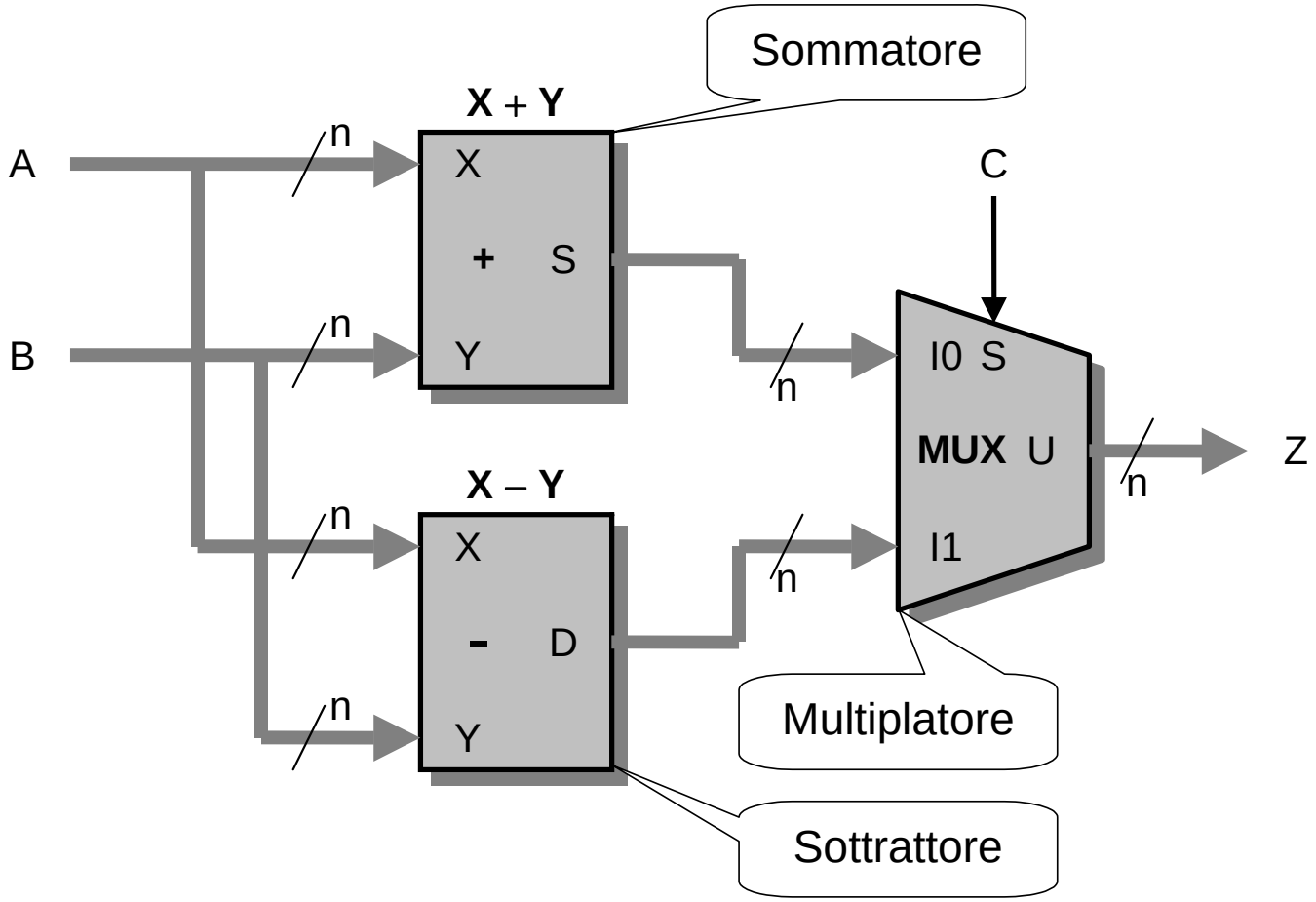
Sottrattore intero binario naturale a 3 bit

Semplice esempio di progetto in stile funzionale

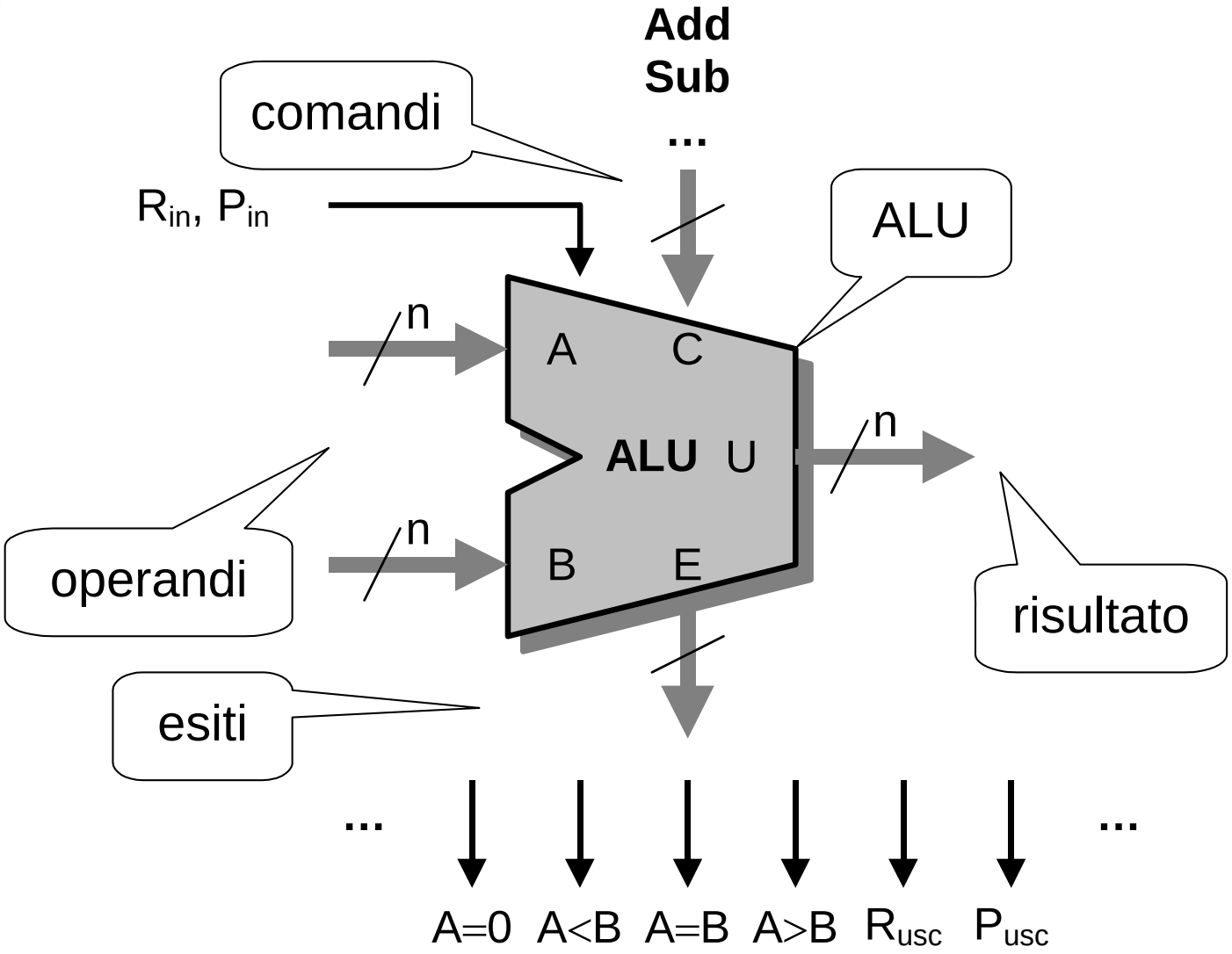
- Si chiede di progettare un circuito digitale combinatorio, che abbia:
  - in ingresso due numeri interi binari naturali (positivi) A e B da n ≥ 1 bit ciascuno
  - in ingresso un segnale di comando C
  - in uscita un numero intero binario naturale Z da n ≥ 1 bit

- Su Z deve presentarsi la somma A + B se C = 0, la differenza A – B se C = 1

Schema logico della soluzione



Unità Aritmetico-Logica

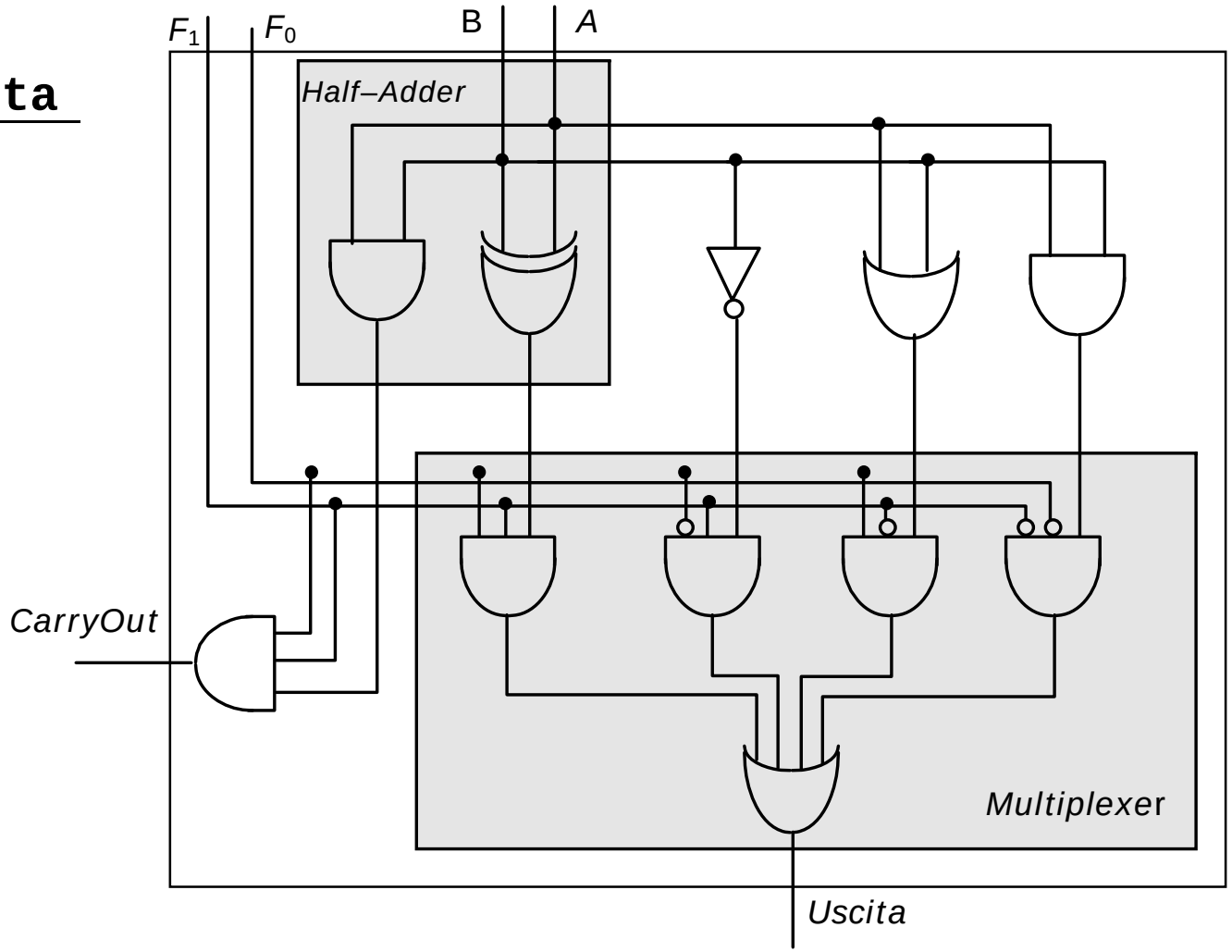


Unità Aritmetico-Logica: esempio di comandi

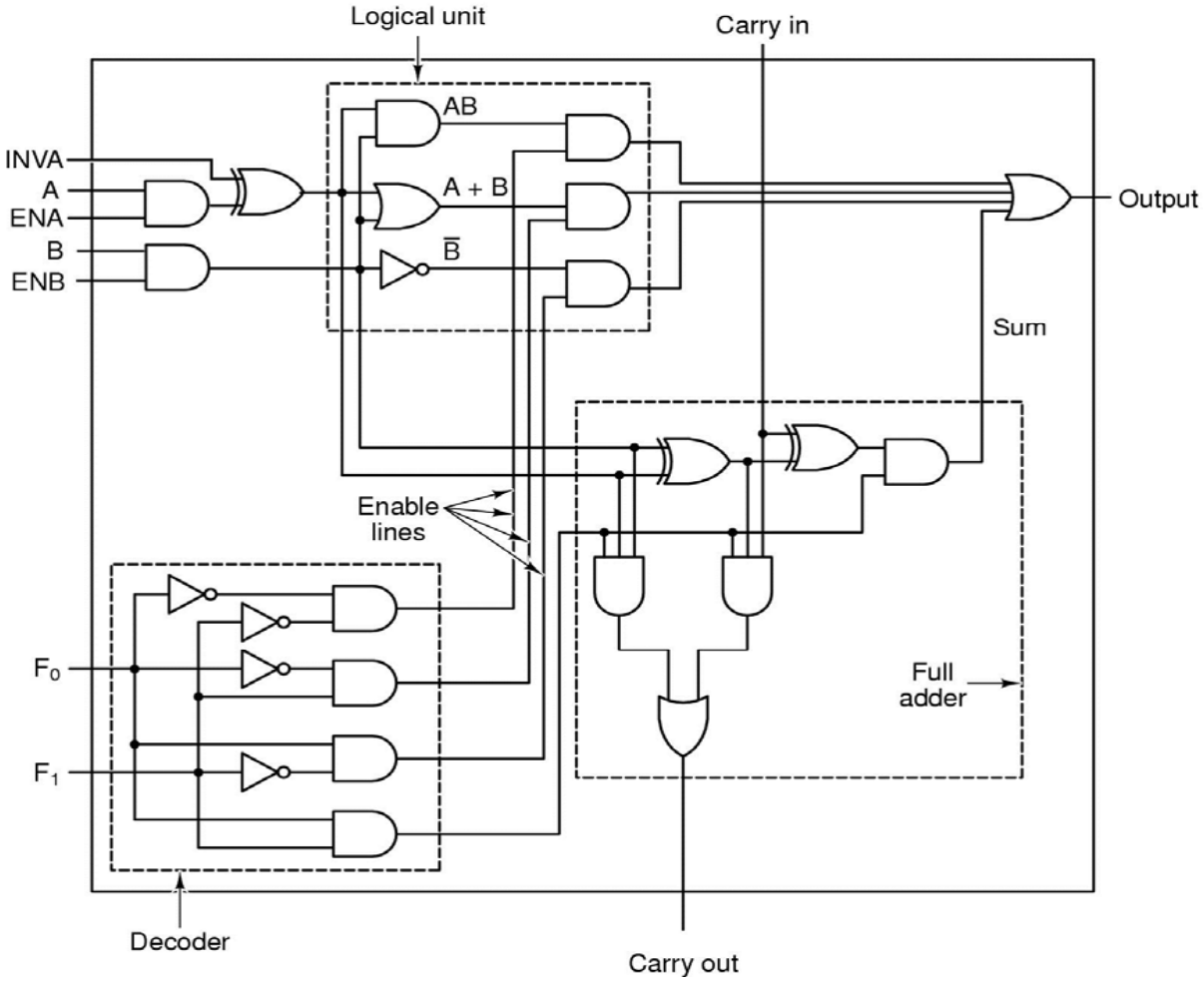
# riga	Comando	Operazione	R	Esito
0	Add	somma A e B	$A + B + R_{in}$	riporto in uscita R_{usc}
1	Sub	sottrae B da A	$A - B - P_{in}$	prestito in uscita P_{usc}
2	Pass A	A passa in uscita	A	-
3	Pass B	B passa in uscita	B	-
4	Zero	annulla uscita	0	-
5	Shift Left A	A scorre a SX	2A	bit più significativo di A
6	Shift Right A	A scorre a DX	A / 2	bit meno significativo di A
7	Null	Confronta A con 0	-	$A = 0$
8	Compare	Confronta A con B	A <,=,> B	$A < B$, $A = B$, $A > B$
9	Multiply	prodotto di A e B	$A \times B$	riporto in uscita
10	Divide	divisione A / B	A / B	divisione per 0 ?
…	…	…	…	…

ALU and, not, or e somma (1): realizzazione con MUX

F_0F_1	Operazione svolta
0 0	*A* and *B*
1 0	*A* or *B*
0 1	!*B*
1 1	*A* + *B*



ALU and, not, or e somma (2): realizzazione con DECODER



Schema logico di una ALU da 1 bit

ALU e MIPS

- Istruzioni aritmetico-logiche
  - add
  - sub
  - or
  - and
  - nor

Un altro schema …… somma e sottrazione in complemento a 2

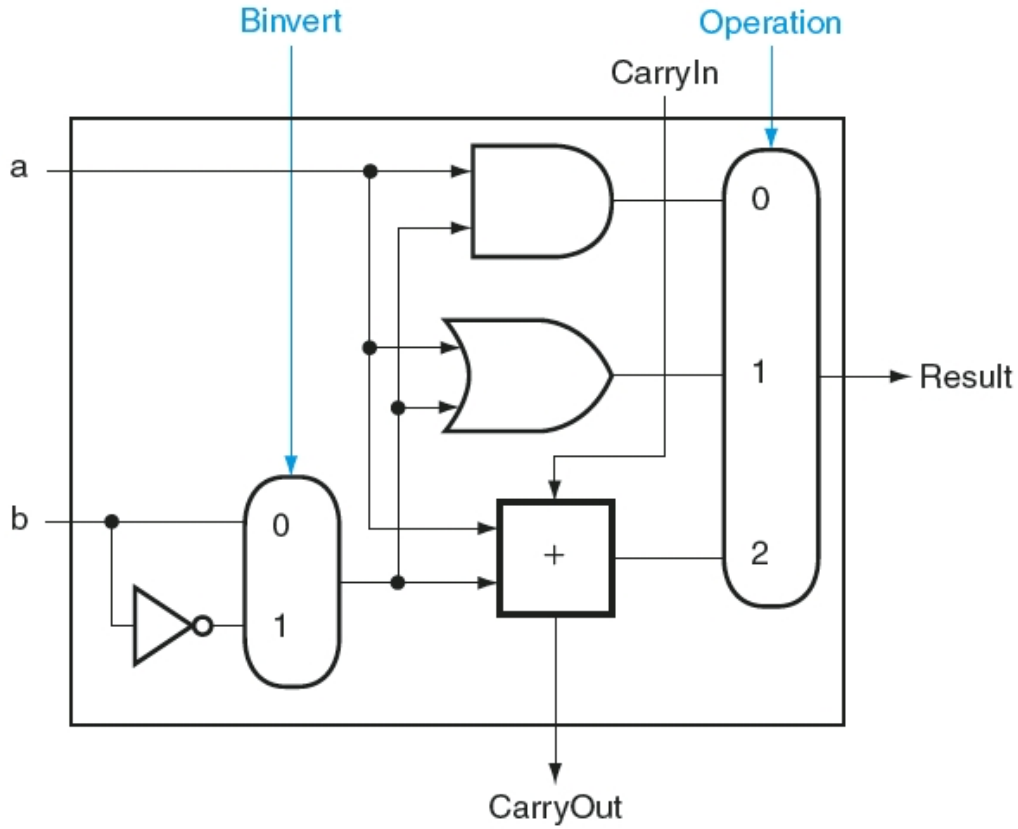


FIGURE B.5.8 A 1-bit ALU that performs AND, OR, and addition on a and b or a and $\overline{b}$. By selecting $\overline{b}$ (Binvert = 1) and setting CarryIn to 1 in the least significant bit of the ALU, we get two's complement subtraction of b from a instead of addition of b to a.

ALU a 32 bit

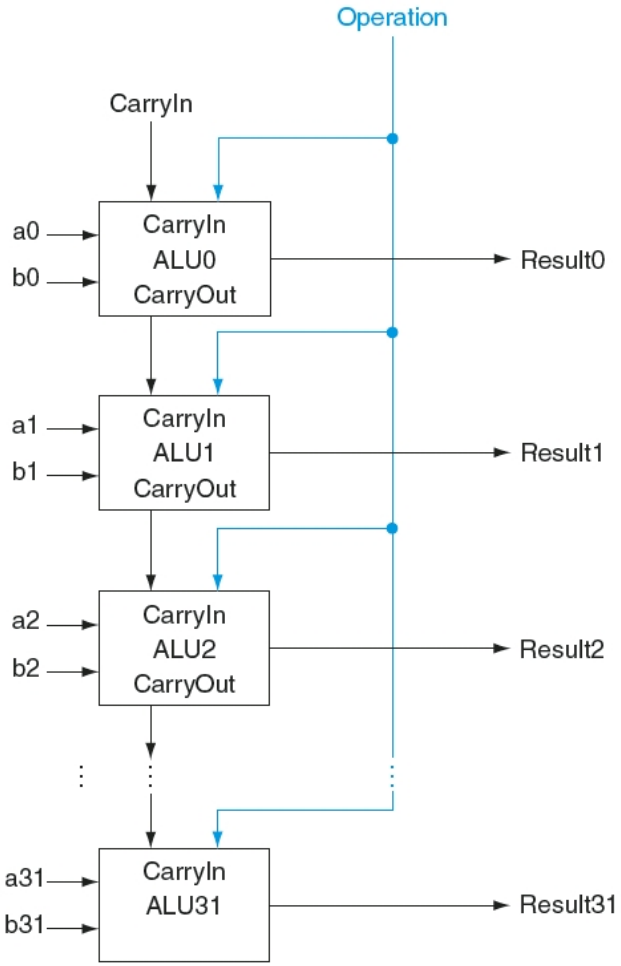


FIGURE B.5.7 A 32-bit ALU constructed from 32 1-bit ALUs. CarryOut of the less significant bit is connected to the CarryIn of the more significant bit. This organization is called ripple carry.

Si aggiunge il ….. NOR

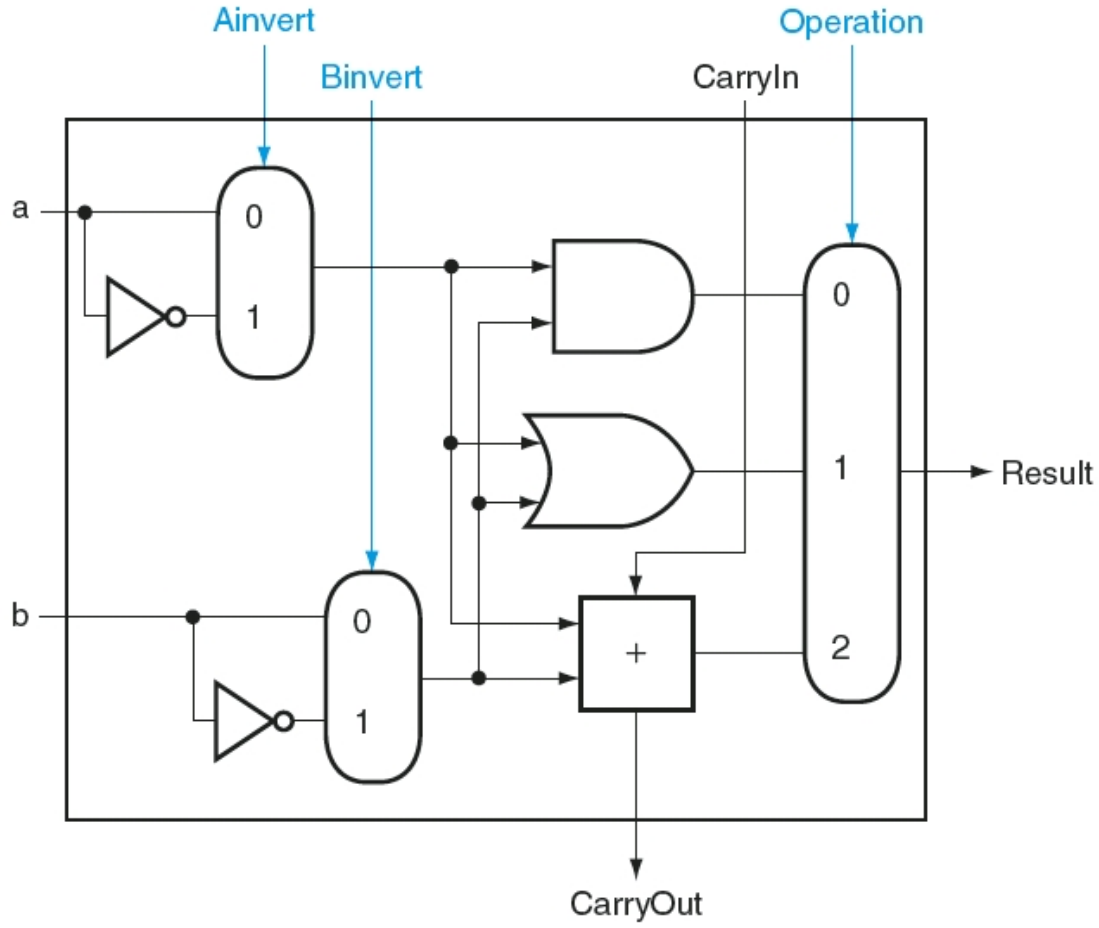


FIGURE B.5.9 A 1-bit ALU that performs AND, OR, and addition on a and b or $\overline{a}$ and $\overline{b}$. By selecting $\overline{a}$ (Ainvert = 1) and $\overline{b}$ (Binvert = 1), we get a NOR b instead of a AND b.

L'indispensabile «Overflow Detection»

… e anche la realizzazione di

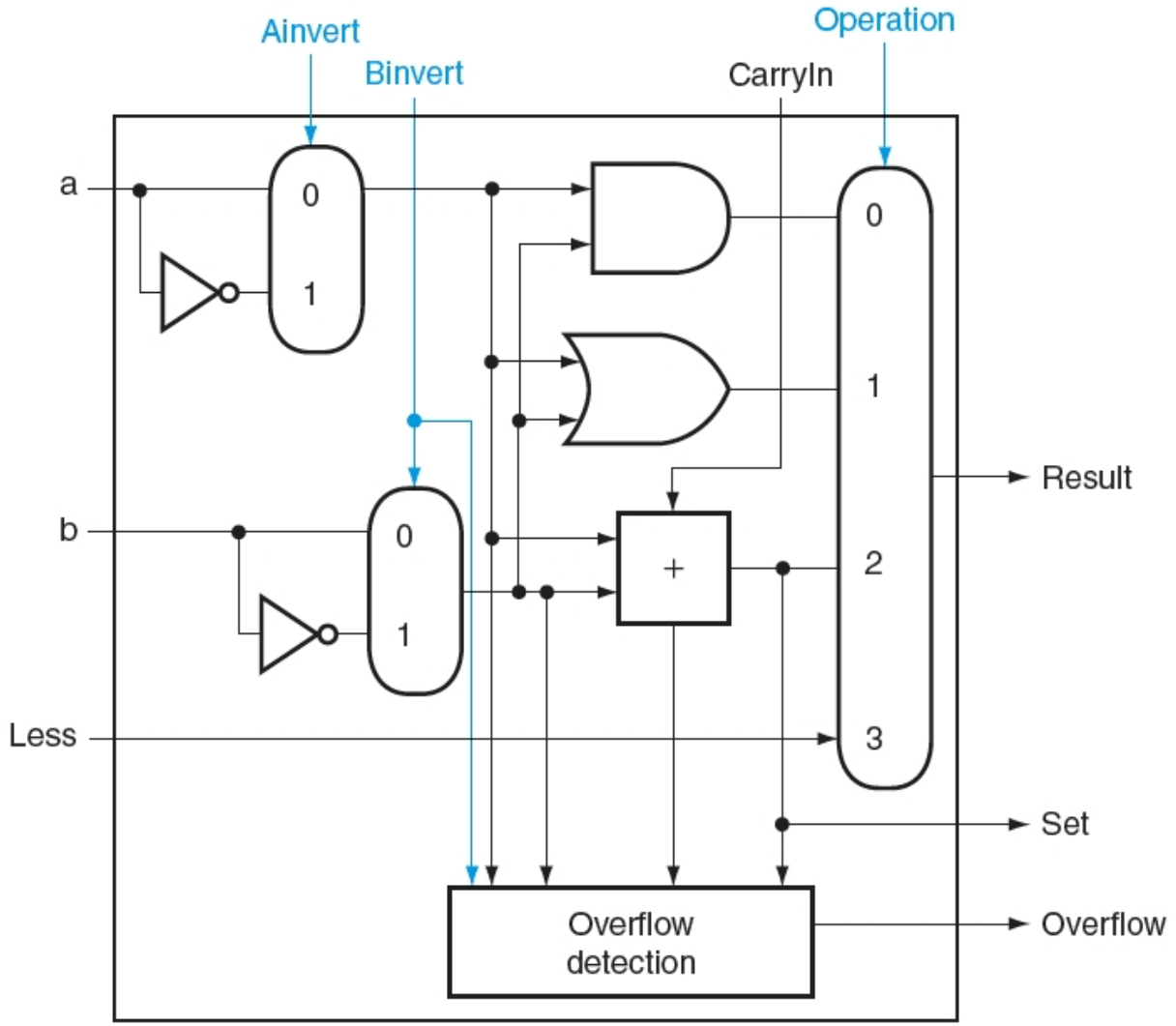
`slt $s1, $s2, $s3`

se s2<s3 allora s1=1 altrimenti s1=0

s2 < s3 equivale a s2 – s3<0

se s2 – s3<0 allora il bit di segno del sommatore è 1 altrimenti è 0

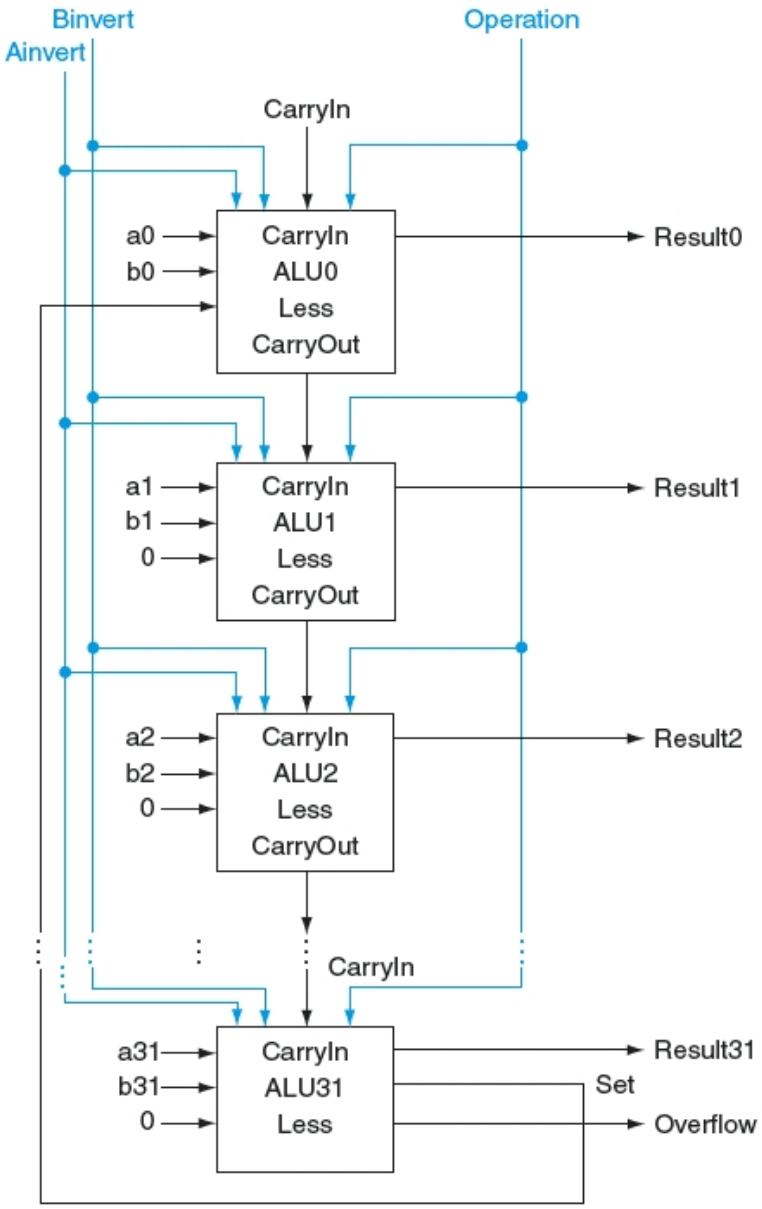
L'ingresso ***less*** viene portato direttamente in uscita tramite il multiplexer e deve essere
0 per i 31 bit più significativi di $s1 e
1 oppure *0* per il bit meno significativo



... a 32 bit

Overflow
in uscita solo quello relativi al bit più significativo

Slt
ecco i valori di tutti i bit



Test zero e sottrazione

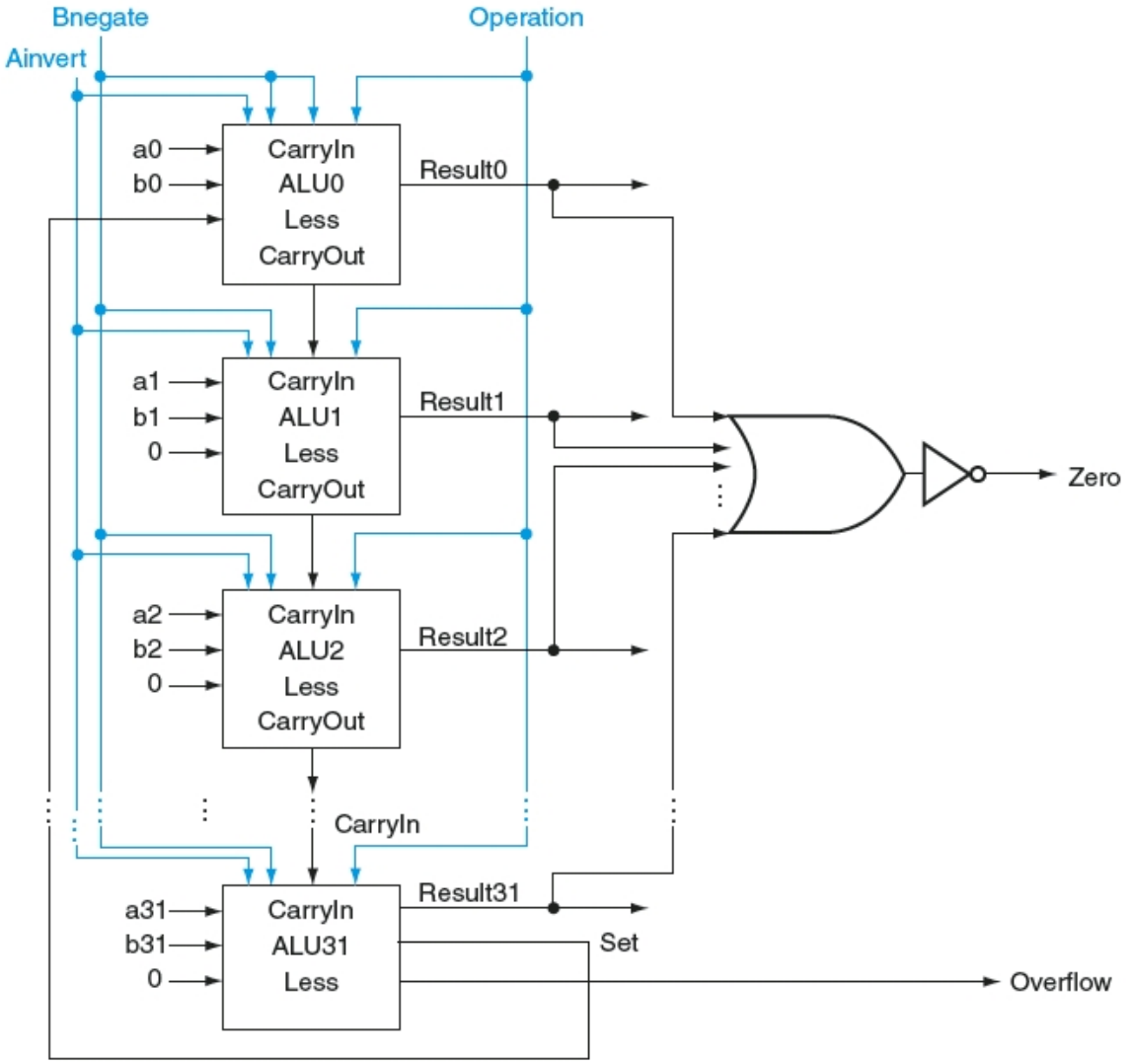
bit ***Zero***

- posto a 1 se il risultato dell'ALU vale zero
- altrimenti a 0

Bnegate

Per la sottrazione in cpl2 è necessario avere i segnali ***Binvert*** e ***Carryin*** a 1 mentre per le altre operazioni considerate, eccetto il NOR, sono entrambi a 0

Vengono sostituiti dall'unico segnale ***Bnegate***

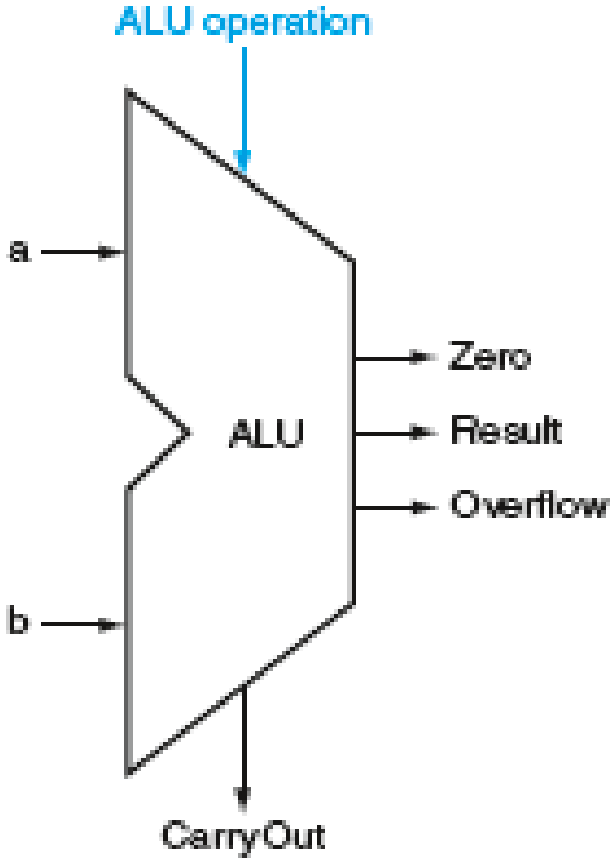


ALU del MIPS e linee di controllo

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Le linee di controllo interne dell'ALU sono:

Ainvert, Bnegate, Operation(2)



... come blocco funzionale

Numeri relativi e numeri reali

- I numeri relativi sono rappresentabili tramite sequenze di bit, proprio come i numeri interi naturali (sempre positivi)
  - La tecnica più usata per rappresentare i numeri interi relativi è il complemento a due (two's complement)
  - Le ALU sono normalmente in grado di operare sia con numeri interi naturali sia con numeri interi relativi rappresentati in complemento a due

- I numeri reali sono rappresentabili tramite sequenze di bit, proprio come i numeri interi
  - Esiste uno standard internazionale per la rappresentazione binaria di numeri reali: lo standard IEEE 754 per la rappresentazione in virgola mobile
  - Esistono ALU in grado di effettuare i calcoli aritmetici con i numeri reali, oltre che con i numeri interi