Moltiplicatori HW e ALU

Prof. Alberto Borghese
Dipartimento di Informatica
borghese@di.unimi.it

Università degli Studi di Milano

Riferimenti: Appendice B5 prima parte.
Per approfondimenti, Capitolo 7 del Fummi, Sami, Silvano.

Sommario

Moltiplicatori

ALU

Moltiplicazione mediante shift

Lo shift di un numero a destra, di k cifre, corrisponde a una divisione per la base elevata alla k-esima potenza.

Lo shift di un numero a sinistra, di k cifre, corrisponde a una moltiplicazione per la base elevata alla k-esima potenza.

Esempio nel caso decimale:

$213_{10} \text{ x } 10 = 2130_{10}$
$213_{10} = (2 \text{ x } 10^2 + 1 \text{ x } 10^1 + 3 \text{ x } 10^0) \textbf{ x } \mathbf{10^1} =$
$(2 \text{ x } 10^2 + 1 \text{ x } 10^1 + 3 \text{ x } 10^0) \textbf{ x } \mathbf{10^1} =$
$(2 \text{ x } 10^2 \textbf{ x } \mathbf{10^1} + 1 \text{ x } 10^1 \textbf{ x } \mathbf{10^1} + 3 \text{ x } 10^0 \textbf{ x } \mathbf{10^1}) =$
$(2 \text{ x } 10^3 + 1 \text{ x } 10^2 + 3 \text{ x } 10^1) = 2130$ cvd.

$213_{10} / 10 = 21{,}3_{10}$
$213_{10} = (2 \text{ x } 10^2 + 1 \text{ x } 10^1 + 3 \text{ x } 10^0) \mathbf{/ 10^1} =$
$(2 \text{ x } 10^2 + 1 \text{ x } 10^1 + 3 \text{ x } 10^0) \textbf{ x } \mathbf{10^{-1}} =$
$(2 \text{ x } 10^2 \textbf{ x } \mathbf{10^{-1}} + 1 \text{ x } 10^1 \textbf{ x } \mathbf{10^{-1}} + 3 \text{ x } 10^0 \textbf{ x } \mathbf{10^{-1}}) =$
$(2 \text{ x } 10^1 + 1 \text{ x } 10^0 + 3 \text{ x } 10^{-1}) = 21{,}3$ cvd.

Moltiplicazione mediante shift

Lo shift di un numero a destra, di k cifre, corrisponde ad una divisione per la base elevata alla k-esima potenza.
Lo shift di un numero a sinistra, di k cifre, corrisponde ad una moltiplicazione per la base elevata alla k-esima potenza.

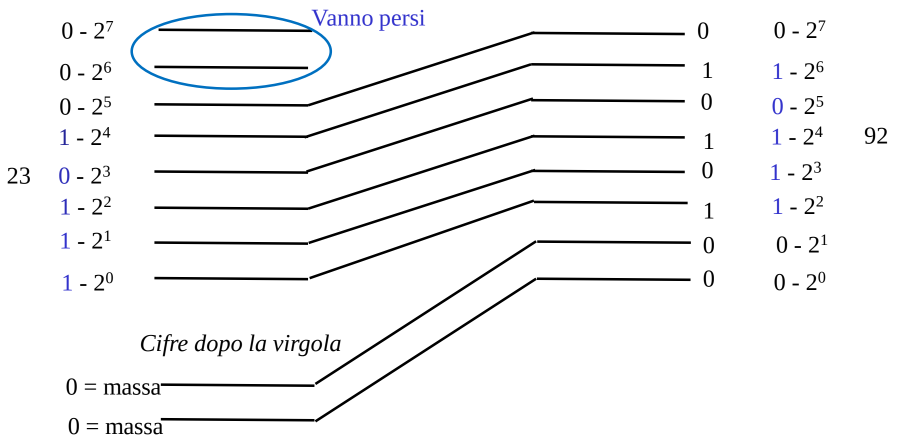
Esempio nel caso binario:
23 * 4 = 92 => 10111 * 100 = 1011100
Esprimendo l'operazione in decimale:
$(1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0) \text{ x } 2^2 =$
$(1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 1x2^2) = 64 + 16 + 8 + 4 = 92$ cvd.

23 / 4 = 5, 75 => 10111 / 100 = 101,11
Esprimendo l'operazione in decimale:
$(1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0) \text{ x } 2^{-2} =$
$(1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}) = 5{,}75$ cvd.

Moltiplicazione mediante shift, caso binario

Codifica su 8 + 2 cifre decimali:

23 * 4 = 92 => 10111 * 100 = 1011100 $23_{10} = 00010111{,}00_2$

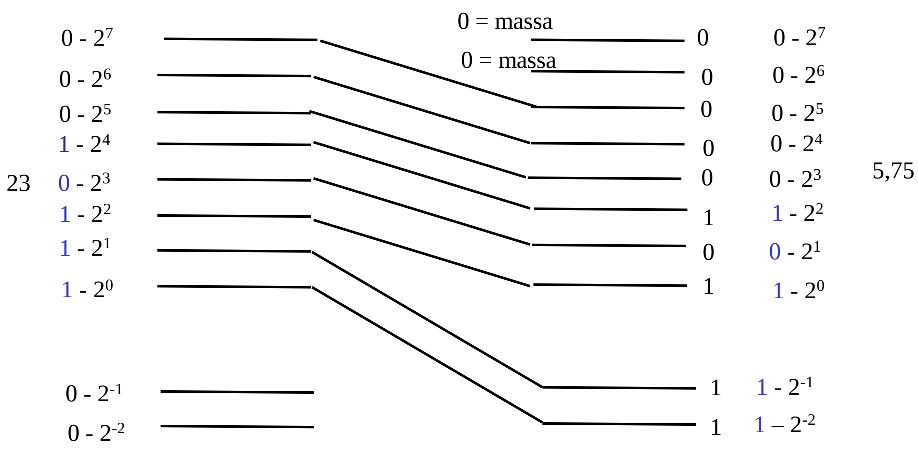


1+2+4+16 = 23 4+8+16+64 = 92

Divisione mediante shift

Codifica su 8 + 2 cifre decimali:

23 / 4 = 5,75 => 10111 / 100 = 101,1100 $23_{10} = 00010111{,}00_2$



1+2+4+16 = 23 *2 cifre dopo la virgola, vengono sovrascritte* 1/4+1/2+1+4 = 5,75

Moltiplicazione decimale

Moltiplicando ⟶ 2 7 8 x
Moltiplicatore ⟶ 4 2 3 =

Prodotti parziali ⟶

8 3 4 +
5 5 6 -
1 1 1 2 - -

----

prodotto ⟶ 1 1 7 5 9 4

$278 \times 423 = 278 \times (4 \times 10^2 + 2 \times 10^1 + 3 \times 10^0) =$
$278 \times (4 \times 10^2) + 278 \times (2 \times 10^1) + 278 \times (3 \times 10^0)$

Somma dei prodotti parziali

Moltiplicazione binaria - I

Moltiplicando ⟶ 1 1 0 1 1 x
Moltiplicatore ⟶ 1 1 1 =

----

1 1 0 1 1 x 27_{10}
1 1 1 = 7_{10}

----

1 1 1 1 1 1
1 1 0 1 1 +
1 1 0 1 1 –
1 1 0 1 1 - -

----

1 0 1 1 1 1 0 1 189_{10}

1° prodotto parziale 1 1 0 1 1 +

⟵ Prodotti parziali

⟵ prodotto

Somma dei prodotti parziali

Moltiplicazione binaria - II

Moltiplicando → 1 1 0 1 1 x
Moltiplicatore → 1 1 1 =
----

1 1 0 1 1 +
1 1 0 1 1 -
----

2 prodotti parziali

1 1 0 1 1 x 27_{10}
1 1 1 = 7_{10}
----
1 1 1 1 1 1
1 1 0 1 1 +
1 1 0 1 1 –
1 1 0 1 1 - -
----
1 0 1 1 1 1 0 1 189_{10}

Il secodo prodotto parziale è incolonnato alle potenze di 2^1 (la cifra del moltiplicatore ha peso 2^1)

Ho generato 2 addendi (2 prodotti parziali)
Provvedo subito alla loro somma

Moltiplicazione binaria - III

Moltiplicando → 1 1 0 1 1 x
Moltiplicatore → 1 1 1 =
----
1 1 1 1 1
1 1 0 1 1 +
1 1 0 1 1 -

1ª Somma parziale → 1 0 1 0 0 0 1 +

2 prodotti parziali

1 1 0 1 1 x 27_{10}
1 1 1 = 7_{10}
----
1 1 1 1 1 1
1 1 0 1 1 +
1 1 0 1 1 –
1 1 0 1 1 - -
----
1 0 1 1 1 1 0 1 189_{10}

$P = \mathbf{P_0 + P_1} + P_2 = \mathbf{(P_0+P_1)} + P_2 =$
$= S_0 + P_2$

Moltiplicazione binaria - IV

Moltiplicando → 1 1 0 1 1 x

Moltiplicatore → 1 1 1 =

1 1 0 1 1 x 27_{10}

1 1 1 = 7_{10}

----

1 1 1 1 1 1

1 1 0 1 1 +

1 1 0 1 1 –

1 1 0 1 1 - -

----

1 0 1 1 1 1 0 1 189_{10}

----

1 1 1 1 1

1 1 0 1 1 +

1 1 0 1 1 -

1ª Somma parziale → 1 0 1 0 0 0 1 +

1 1 0 1 1 - -

3 prodotti parziali

Il terzo prodotto parziale è incolonnato alle potenze di 2^2 (la cifra del moltiplicatore ha peso 2^2)

Moltiplicazione binaria - V

Moltiplicando → 1 1 0 1 1 x

Moltiplicatore → 1 1 1 =

1 1 0 1 1 x 27_{10}

1 1 1 = 7_{10}

----

1 1 1 1 1 1

1 1 0 1 1 +

1 1 0 1 1 –

1 1 0 1 1 - -

----

1 0 1 1 1 1 0 1 189_{10}

----

1 1 1 1 1

1 1 0 1 1 +

1 1 0 1 1 -

----

1 0 0 0 0

1ª Somma parziale → 1 0 1 0 0 0 1 +

1 1 0 1 1 - -

3 prodotti parziali

Somma finale = prodotto → 1 0 1 1 1 1 0 1

Moltiplicazione binaria - I

Moltiplicando	1 1 0 1 1 x	27 x
Moltiplicatore	1 0 1 1 =	11 =
	----------------------	-----
Riporto	1 1 1 1 1	
Prodotti parziali	1 1 0 1 1 +	27 +
	1 1 0 1 1 -	54 =
	-----------------------	-------
Somma parziale	1 0 1 0 0 0 1	81

Moltiplicazione binaria - II

Moltiplicando	1 1 0 1 1 x	27 x
Moltiplicatore	1 0 1 1 =	11 =
	----------------------	-----
Riporto	1 1 1 1 1	
Prodotti parziali	1 1 0 1 1 +	27 +
	1 1 0 1 1 -	54 =
	-----------------------	-------
Riporto	0 0 0 0 0	
Somma parziale	1 0 1 0 0 0 1 +	81 +
Prodotti parziali	**0 0 0 0 0** - -	0 =
	-----------------------	-----
Somma parziale	1 0 1 0 0 0 1	81

Moltiplicazione binaria - III

Moltiplicando → 1 1 0 1 1 x | 27 x

Moltiplicatore → 1 0 1 1 = | 11 =

```
                    11011 x        27 x
                     1011 =        11 =
---------------------------       -----
              11111
                11011 +            27 +
               11011 -             54 =
---------------------------       -------
           00000
             1010001 +             81 +
             00000 - -              0 =
---------------------------
        11010
            1010001 +              81 +
          11011 - - - =           216 =
---------------------------
        100 101001          -> 297|10
```

Prodotti parziali

Riporto

Somma parziale

Prodotto

Moltiplicazione binaria - IV

Moltiplicando →

Moltiplicatore →

```
                    11011 x        27 x
                     1011 =        11 =
---------------------------       -----
              11111
                11011 +            27 +
               11011 -             54 =
---------------------------       -------
           00000
             1010001 +             81 +
             00000 - -              0 =
---------------------------
        11010
            1010001 +              81 +
          11011 - - - =           216 =
---------------------------
        100 101001          -> 297|10
```

Prodotti parziali

Riporto

Somma parziale

Prodotto

$P = P_0 + P_1 + P_2 + P_3 = ((P_0+P_1) + P_2) + P_3 = (S_0 + P_2) + P_3 = S_1 + P_3$

Moltiplicazione binaria (su 4 bit)

Moltiplicando

Moltiplicatore

1 0 1 1 x $\quad 11_{10}$ x

1 0 1 = $\quad 5_{10}$ =

----

0 0 0 0

Prodotti parziali

1 0 1 1 + $\quad 1011*1*2^0$ +

0 0 0 0 – $\quad 1011*0*2^1$ =

Somma parziale (Sommatori)

1 0 1 1 +

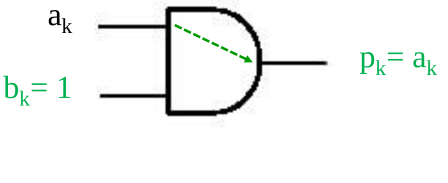
1 0 1 1 - - $\quad 1011*1*2^2$ =

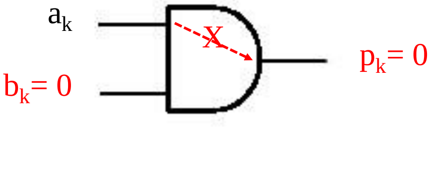
Prodotto

1 1 0 1 1 1 $\quad 55_{10}$

Il prodotto parziale è = $\begin{cases} \text{Moltiplicando incolonnato opportunamente} \\ 0 \end{cases}$

Moltiplicazione su 1 bit





AND non solo semaforo, ma anche moltiplicatore a 1 bit!

La moltiplicazione binaria

Possiamo vederla come:

Un primo stadio in cui si mette in AND ciascun bit del moltiplicatore con il moltiplicando (**prodotto parziale**).

Un secondo stadio in cui si effettuano le **somme dei prodotti parziali** (full adder): somma dei bit su due righe diverse.

La matrice dei prodotti parziali

A e B su 4 bit

1 1 0 1

Prodotti parziali

			a_3	a_2	a_1	a_0	
			$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$	b_0
		$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$		b_1
	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$			b_2
$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$				b_3
2^6	2^5	2^4	2^3	2^2	2^1	2^0	

1 0 1 1

In binario i prodotti parziali sono degli AND.

Sulla linea tanti AND quanto è la lunghezza di A
Tanti prodotti parziali quanto è la lunghezza di B

La matrice dei prodotti parziali

1 1 0 1

Prodotti parziali

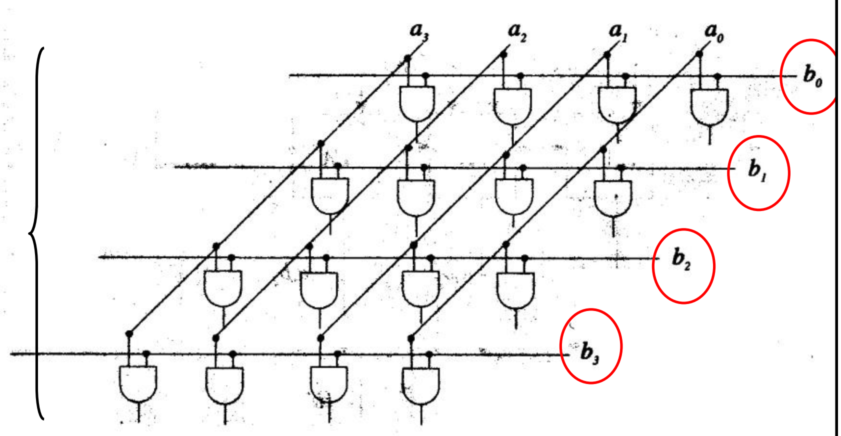
			a_3	a_2	a_1	a_0		
			$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$	b_0	
		$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$		b_1	1 0 1 1
	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$			b_2	
$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$				b_3	

Il bit i-esimo del moltiplicatore, b_i, fa passare 0 o il moltiplicando (prodotto parziale)
Il prodotto parziale viene incolonnato opportunamente.

b_0 (a_3 a_2 a_1 a_0) genera P_0
b_1 (a_3 a_2 a_1 a_0) genera P_1
b_2 (a_3 a_2 a_1 a_0) genera P_2
b_3 (a_3 a_2 a_1 a_0) genera P_3
....

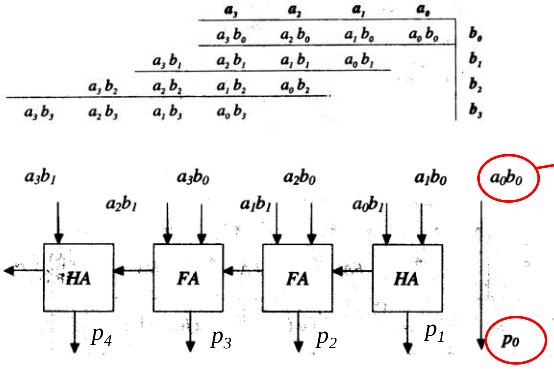
Il circuito che effettua i prodotti

Prodotti parziali



b_k agisce come interruttore, facendo passare 0 o A_k

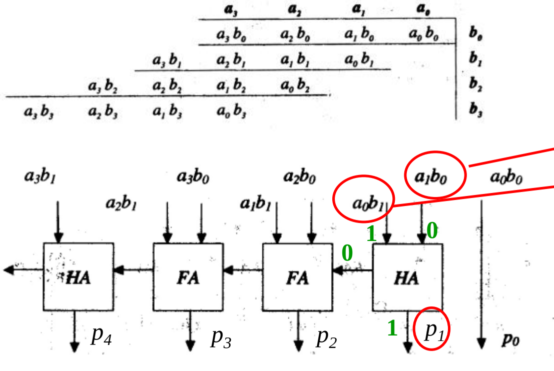
Somma delle prime 2 righe dei prodotti parziali - I



Somma dei primi 2 prodotti parziali

```
      1 1 0 1 x        13 x
      1 0 1 1 =        11 =
---------------------
    1 1 0 0
      1 1 0 1 +
    1 1 0 1 -
---------------------
  0 0 0 0
    1 0 0 1 1 1 +
    0 0 0 0 - -
---------------------
 1 1 0 0
    1 0 0 1 1 1 +
  1 1 0 1 - - - =
---------------------
 1 0 0 0 1 1 1 1   -> 143|10
```


Somma delle prime 2 righe dei prodotti parziali - II

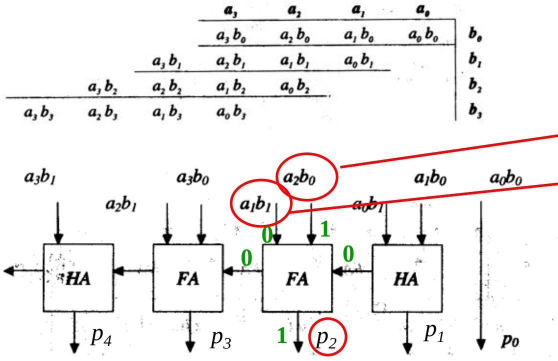


Somma dei primi 2 prodotti parziali → 1a somma parziale

```
      1 1 0 1 x        13 x
      1 0 1 1 =        11 =
---------------------
    1 1 0 0 0
      1 1 0 1 +
    1 1 0 1 -
---------------------
  0 0 0 0
    1 0 0 1 1 1 +
    0 0 0 0 - -
---------------------
 1 1 0 0
    1 0 0 1 1 1 +
  1 1 0 1 - - - =
---------------------
 1 0 0 0 1 1 1 1   -> 143|10
```

Somma delle prime 2 righe dei prodotti parziali - III

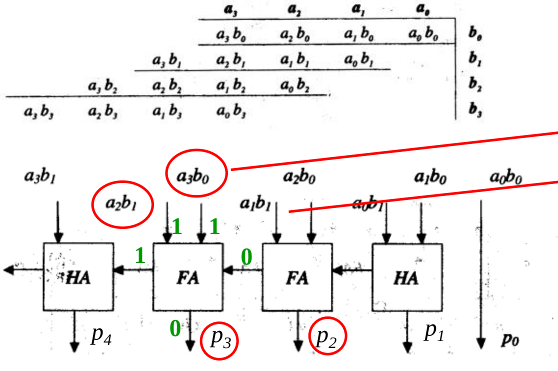
1 1 0 1 x   13 x
1 0 1 1 =   11 =
----
1 1 0 0 0
1 1 0 1 +
1 1 0 1 -
----
0 0 0 0
1 0 0 1 1 1 +
0 0 0 0 - -
----
1 1 0 0
1 0 0 1 1 1 +
1 1 0 1 - - - =
----
1 0 0 0 1 1 1 1   -> $143|_{10}$



Somma dei primi 2 prodotti parziali →
1a somma parziale

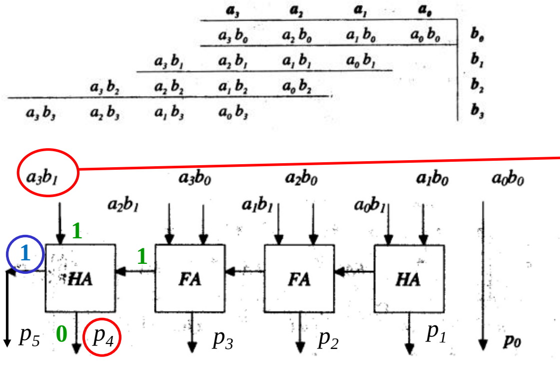
Somma delle prime 2 righe dei prodotti parziali - IV

1 1 0 1 x   13 x
1 0 1 1 =   11 =
----
1 1 0 0 0
1 1 0 1 +
1 1 0 1 -
----
0 0 0 0
1 0 0 1 1 1 +
0 0 0 0 - -
----
1 1 0 0 0
1 0 0 1 1 1 +
1 1 0 1 - - - =
----
1 0 0 0 1 1 1 1   -> $143|_{10}$



Somma dei primi 2 prodotti parziali →
1a somma parziale

Somma delle prime 2 righe dei prodotti parziali - V



1 1 0 1 x   13 x
1 0 1 1 =   11 =
----
1 1 0 0 0
1 1 0 1 +
1 1 0 1 -
----
0 0 0 0
1 0 0 1 1 1 +
0 0 0 0 - -
----
1 1 0 0
1 0 0 1 1 1 +
1 1 0 1 - - - =
----
1 0 0 0 1 1 1 1 -> $143|_{10}$

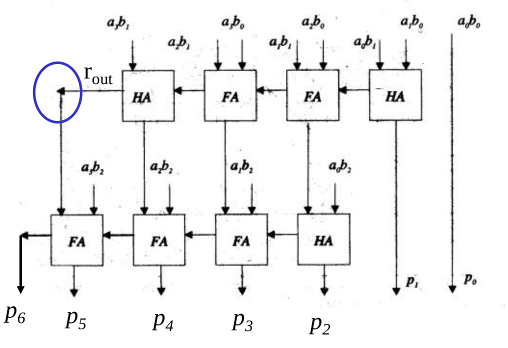
Somma dei primi 2 prodotti parziali → 1a somma parziale

Dove va il riporto in uscita all'ultimo FA?

Somma della terza riga

I primi due prodotti parziali sono ottenuti dalla prima batteria di sommatori.

Ogni altro prodotto parziale è sommato da un'ulteriore batteria di sommatori.



1 1 0 1 x   13 x
1 0 1 1 =   11 =
----
1 1 0 0 0
1 1 0 1 +
1 1 0 1 -
----
0 0 0 0 0
1 0 0 1 1 1 +
0 0 0 0 - -
----
1 1 0 0
1 0 0 1 1 1 +
1 1 0 1 - - - =
----
1 0 0 0 1 1 1 1 -> $143|_{10}$

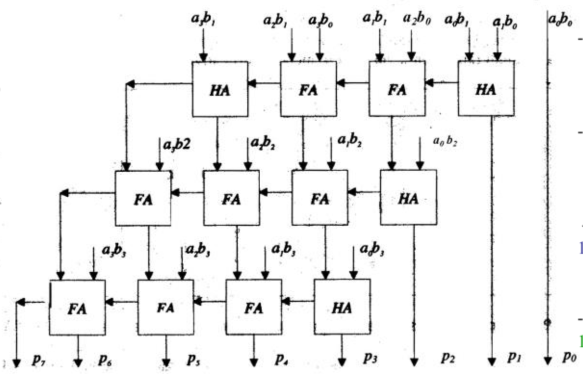
Circuito completo della somma dei prodotti parziali

N-1 batterie di sommatori

$P_0+P_1 \rightarrow S_0$

$S_0+P_2 \rightarrow S_1$

$S_1+P_3 \rightarrow P$

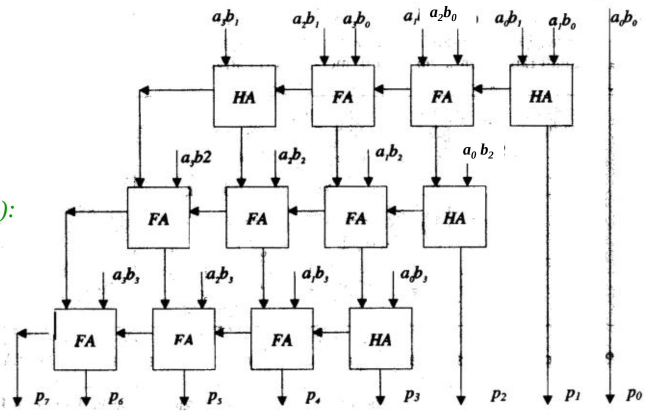


1 1 0 1 1 x	27 x
1 0 1 1 =	11 =
-----------------	-----
1 1 1 1 1	
1 1 0 1 1 +	27 +
1 1 0 1 1 -	54 =
-----------------	------
0 0 0 0 0	
1 0 1 0 0 0 1 +	81 +
0 0 0 0 0 - -	0 =
-----------------	
1 1 0 1 0	
1 0 1 0 0 0 1 +	81 +
1 1 0 1 1 - - - =	216 =
-----------------	
1 0 0 1 0 1 0 0 1	-> $297\|_{10}$

Problema: A e B su 4 bit => P su 8 bit (prodotto su 2N bit)

Valutazione della complessità

Complessità:
Half Adder: 2 porte
Full Adder: 5 porte



Stadio prodotti (AND):

A su N bit
B su N bit

N * N porte AND

Stadio Somme:

Se N = M = 4 numero totale di porte a 2 ingressi = 5

N sommatori per linea
N-1 linee

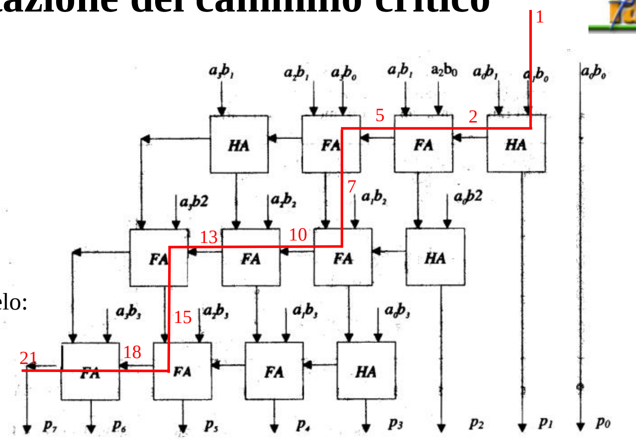
	Numero linee		Numero FA per linea	Numero HA per linea		Complessità Prima linea		Prodotti Parziali
CO_{Tot} =	(N-2)	*	[(N-1) * 5	+ 1 * 2]	+	(N-2)*5 +2*2	+	N * N

Valutazione del cammino critico

Cammini critici:
Half Adder:
Somma – 1 porta
Riporto – 1 porta
Full Adder:
Somma - 2 porte
Riporto - 3 porte

Gli AND operano in parallelo: ritardo 1.

HA e FA non sono equivalenti per il cammino critico



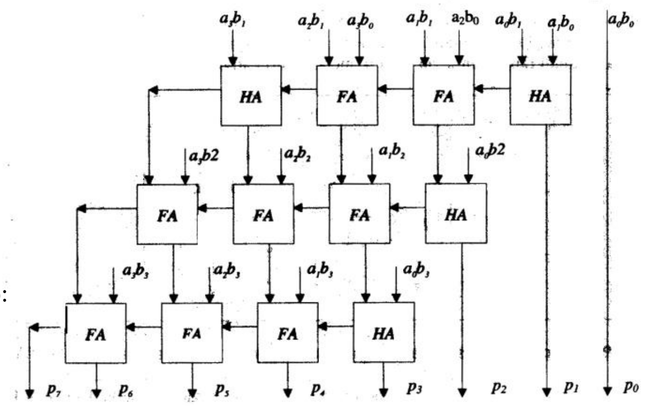
21 20 17 14 10 5 2 1

Se N = 4 cammino critico totale = 21

Osservazioni

Cammini critici:
Half Adder:
Somma – 1 porta
Riporto – 1 porta
Full Adder:
Somma - 2 porte
Riporto - 3 porte

Gli AND operano in parallelo: ritardo 1.



Architettura **modulare**, ogni schiera di sommatori lavora sul risultato della schiera superiore e fornisce l'input alla schiera inferior

Il prodotto è su 2*N cifre. Situazione da gestire da parte delle architetture che hanno registri su N bit.

Quanto si guadagna sostituendo ai sommatori a propagazione di riporto sommatori ad anticipazione di riporto?

Sommario

Moltiplicatori

ALU

Funzione della ALU

E' integrata nel processore, all'inizio degli anni 90 è stata rivoluzionaria la sua introduzione con il nome di co-processore matematico.

Esegue le operazioni aritmetico-logiche.

Utilizza i blocchi di base già visti.

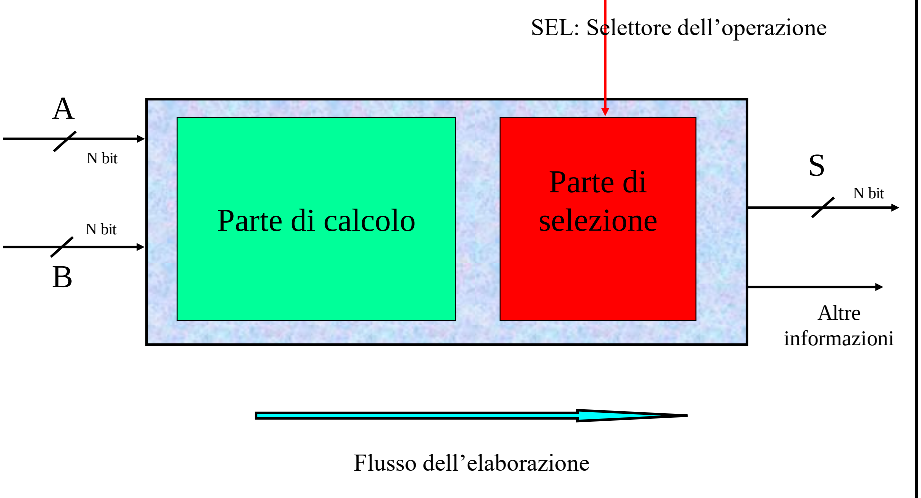
Opera su parole (MIPS 32 bit / 64 bit).

Le ALU non compaiono solamente nei micro-processori.

Problematiche di progetto

- Velocità (Riporto).
- Costo.
- Precisione.
- Affidabilità
- Consumo energetico.

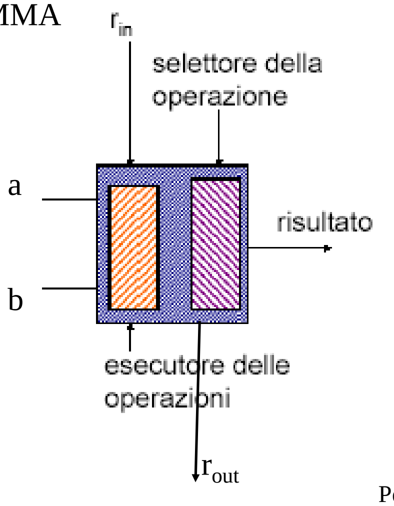
Struttura a 2 livelli di una ALU

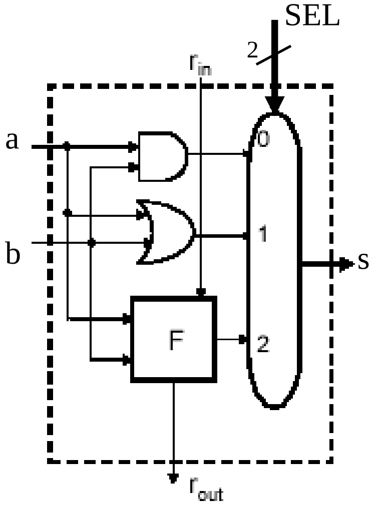


Esempio di esecuzione condizionata.

La nuova struttura della ALU – 1 bit

•AND
•OR
•SOMMA

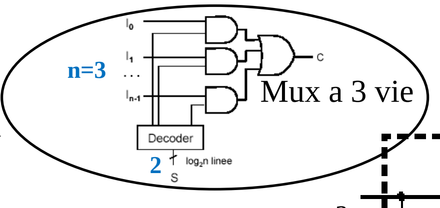


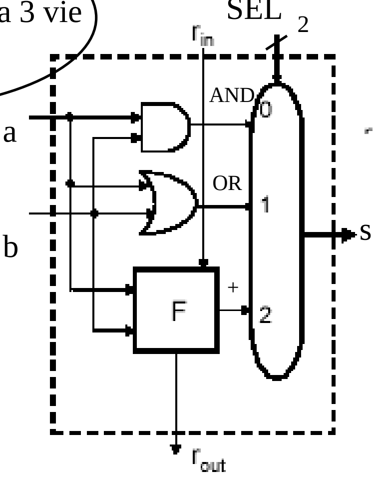


Perchè SEL non viene messo in ingresso?

Valutazione ALU a 1 bit

•AND
•OR
•SOMMA



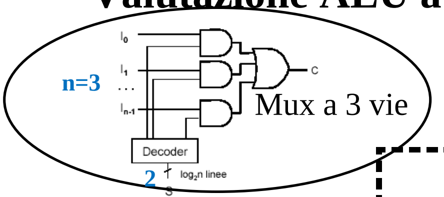


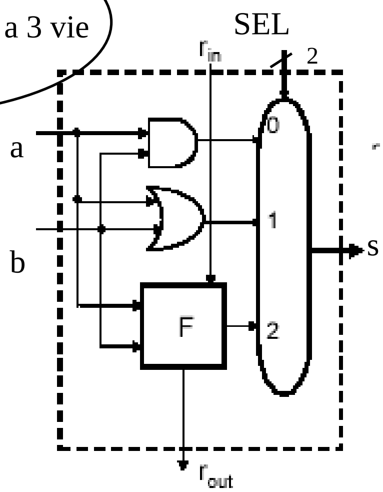
Complessità 1° livello (calcolo): 5+2 = 7
Complessità 2° livello (mux): 3*1+(3+2) = 8
(Decoder + AND (semaforo) + OR (congiunzione)

Complessità totale: 7+8 = 15

Valutazione ALU a 1 bit

- AND
- OR
- SOMMA





CC 1° livello: 2 per s_{out}, 3 per r_{out}

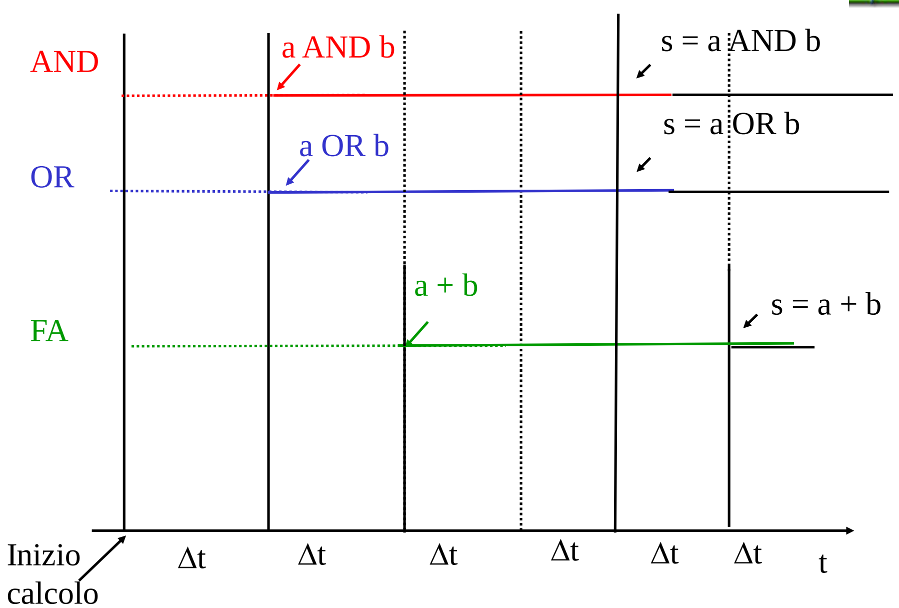
CC 2° livello: 4 (1 Decoder + (1 AND - semaforo + 2 OR (congiunzione)

CC complessivo: 2 (calcolo) + [1 AND (semaforo)+ 2 (OR – selezione)] = 5!!

Il CC del decoder non viene contato: gli AND del decoder interni al mux sono attivati in parallelo ai circuiti di calcolo.

Il CC considerato è quello della somma per valutare il tempo necessario perchè commuti s.

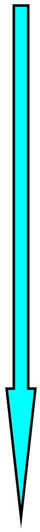
I cammini critici all'interno della ALU



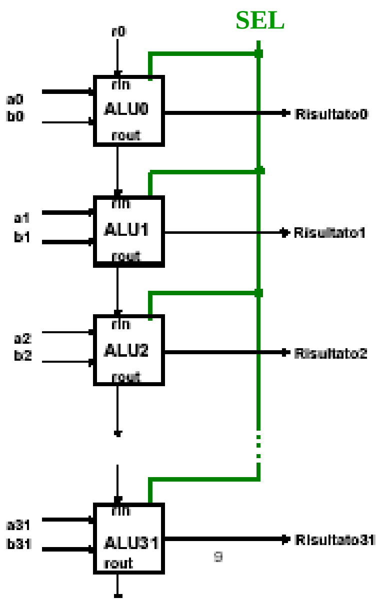
ALU a 32 bit

Come collegare le ALU ad 1 bit?

Flusso di calcolo



Perchè non si può parallelizzare?



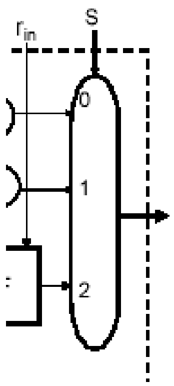
Valutazione ALU a 32 bit

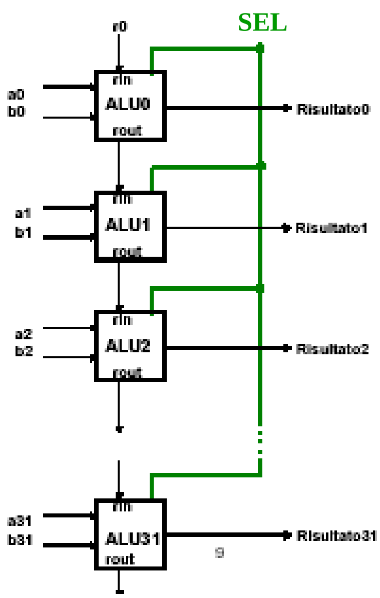
Complessità: 15 x 32 = 480 porte logiche

Cammino critico:
- 3 x 32 (propagazione riporti) del sommatore
- 3 (parte centrale del mux: semaforo + OR uscita di s_{31})

Totale = 99 porte logiche

per 4 operazioni su 32 bit





Sottrazione

In complemento a 2 diventa un'addizione: $a - b = a + \bar{b} + 1 = 1 + a + \bar{b}$

Esempio: s = 3 – 4; su 3 bit

3 -> 011	011 +
-4 -> 100 in complemento a 2	100 =
-1 -> 111 in complemento a 2	111

Posso scrivere il numero negativo in complemento a 2 come somma:

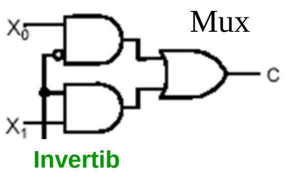
	4 -> 100	numero positivo:	b
Passo I – Complemento a 1	011+	complemento a 1:	$\bar{b}$+
Passo II – Sommo + 1	1=	sommo 1:	1=
Risultato – Complemento a 2	100	risultato	-b

Posso quindi scrivere: $-b = \bar{b} + 1$

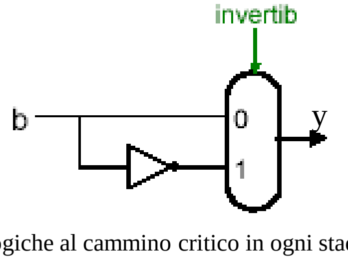
Sottrazione

In complemento a 2 diventa un'addizione: $a - b = a + \bar{b} + 1$

Serve:
a) un inverter (NOT).
b) la costante 1



a)



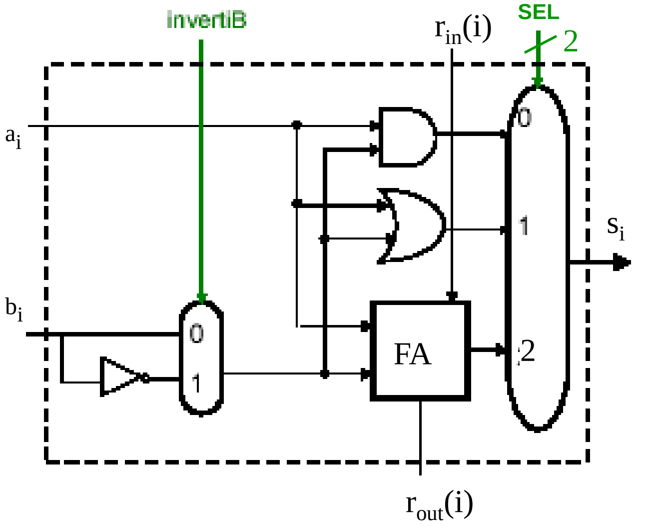
Iff invertib
y = !b

Aggiunge 2 porte logiche al cammino critico in ogni stadio. y viene calcolato in parallelo su tutti gli stadi.
Aggiunge 2 porte per ogni stadio.

b) Da dove prendo la costante 1?

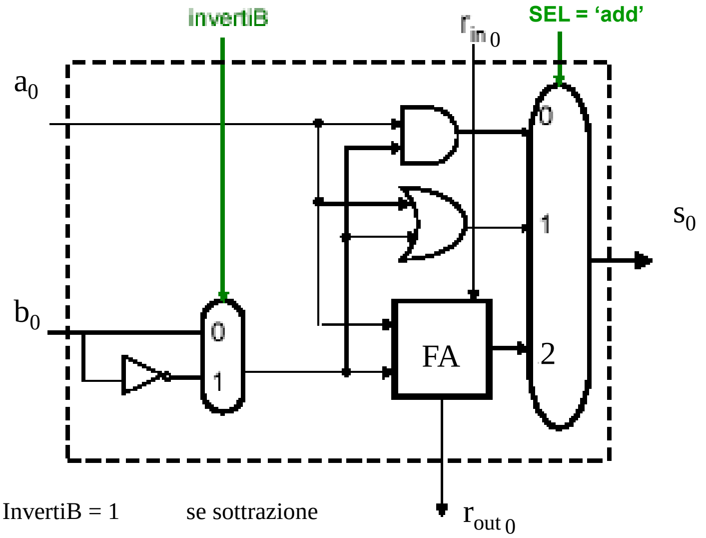
Sottrazione - ALU_i

Operazioni:
- •AND
- •OR
- •SOMMA
- •SOTTRAZIONE



$r_{in}(i) = r_{out}(i-1)$ $i = 1,2,3,....31$ **i ≠ 0** (tranne che nel primo stadio)
InvertiB = 1 se sottrazione

Sottrazione – primo stadio: ALU_0



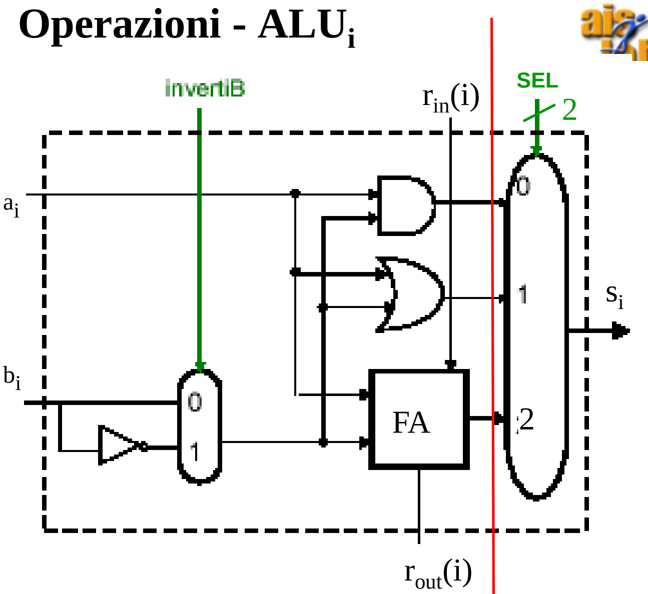
$r_{in}(0)$ = InvertiB = 1 se sottrazione

(occorre utilizzare un full adder anche per il bit meno significativo con $r_{in0} = 1$).
Effettuo quindi la somma di 1 con la somma della prima coppia di bit.

Operazioni - ALU_i

E' possibile programmare questa ALU per eseguire:

1) a AND !b
2) a OR !b
3) a + b
4) a - b



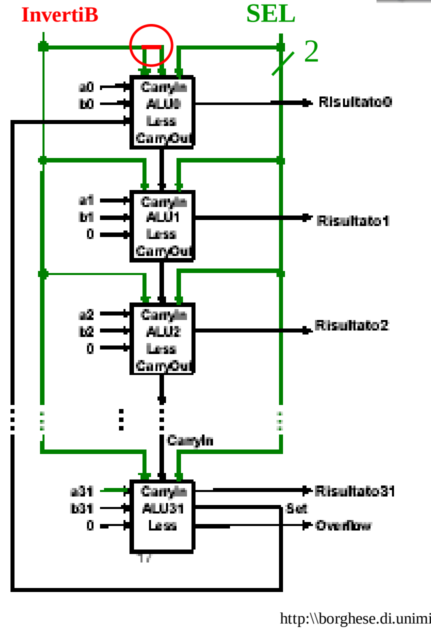
La parte di calcolo è comunque separata dalla parte di selezione

Sottrazione: ALU a 32 bit

$r_{in}(0)$ = InvertiB = 1
se sottrazione

- •AND
- •OR
- •SOMMA
- •SOTTRAZIONE

From_UC	SEL	r_0	InvertiB
And	And	0	0
Or	Or	0	0
Somma	Add	0	0
Sottr.	Add	1	1



InvertiB e r_0 sono lo stesso segnale, si può ancora ottimizzare.

$r_{in}(0)$ entra solo in ALU_0

InvertiB entra in tutte le ALU_i

ALU a 32 bit con CLA

- Come realizzare una ALU a 32 bit con:
  - Porte OR
  - Porte AND
  - CLA a 4 bit?

Definire complessità e cammino critico

Notate che l'inverter su b aggiunge complessità e cammino critico.

Sommario

Moltiplicatori

ALU