# Moltiplicatori HW e ALU

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@di.unimi.it

Università degli Studi di Milano

Riferimenti: Appendice B5 prima parte. Per approfondimenti e HW della moltiplicazione consultare il Fummi.



# Sommario

Moltiplicatori

ALU

# Moltiplicazione mediante shift

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

Esempio:
$213_{10} / 10 = 21.3_{10}$
$213_{10} = (2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0) / 10^1 =$
$(2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0) \times 10^{-1} =$
$(2 \times 10^2 \times 10^{-1} + 1 \times 10^1 \times 10^{-1} + 3 \times 10^0 \times 10^{-1}) =$
$(2 \times 10^1 + 1 \times 10^0 + 3 \times 10^{-1}) = 21.3$ cvd.

Esempio:
23 / 4 = 5, 75 => 10111 / 100 =
$(1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0) \times 2^{-2} =$
$(1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}) = 5{,}75$ cvd.



# 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

$278 \times 423 = 278 \times (4 \times 10^2 + 2x10^1 + 3\ x10^0) =$
$278 \times (4 \times 10^2) + 278 \times (2 \times 10^2) + 278 \times (3 \times 10^0)$

Somma dei prodotti parziali

# Moltiplicazione binaria

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
Somma parziale → 1 0 1 0 0 0 1 +
1 1 0 1 1 - -

prodotti parziali

-----------------------

prodotto → 1 0 1 1 1 1 0 1



# Moltiplicazione binaria

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

| | |
|---|---|
| 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 - | 27- = |
| ---------------------- | ------- |
| 0 0 0 0 0 | 297 |
| 1 0 1 0 0 0 1 + | |
| 0 0 0 0 0 - - | |
| ---------------------- | |
| 1 1 0 1 0 | |
| 1 0 1 0 0 0 1 + | |
| 1 1 0 1 1 - - - = | |
| ---------------------- | |
| 1 0 0 1 0 1 0 0 1 | -> $297|_{10}$ |

Prodotti parziali

Riporto

Somma parziale

Prodotto

# Somme parziali e prodotto

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 -

-----------------------

1

Somma parziale → 1 0 1 0 0 0 1 +

1 1 0 1 1 - -

-----------------------

prodotti parziali

prodotto → 1 0 1 1 1 1 0 1

$P = P_0 + P_1 + P_2 =$

$= (P_0 + P_1) + P_2 =$

$= S_0 + P_2$



# Moltiplicazione binaria (su 4 bit)

Moltiplicando →

Moltiplicatore →

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

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

-------------------------

0 0 0 0

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

0 0 0 0 – $1011* 0 *2^1 =$

---------------------------

1 0 1 1 +

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

--------------------------

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

Prodotti parziali (AND) →

Somma parziale (Sommatori) →

Prodotto →

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

# La moltiplicazione binaria

Possiamo vederla come:

Un primo stadio in cui si mette in AND ciascun bit del moltiplicatore con il moltiplicando.

Un secondo stadio in cui si effettuano le somme (full adder) dei bit sulle righe contenenti i prodotti parziali.



# La matrice dei prodotti parziali

Prodotti parziali

$$
\begin{array}{ccccccc|c}
 & & & a_3 & a_2 & a_1 & a_0 & \\
\hline
 & & & a_3 b_0 & a_2 b_0 & a_1 b_0 & a_0 b_0 & b_0 \\
 & & a_3 b_1 & a_2 b_1 & a_1 b_1 & a_0 b_1 & & b_1 \\
 & a_3 b_2 & a_2 b_2 & a_1 b_2 & a_0 b_2 & & & b_2 \\
a_3 b_3 & a_2 b_3 & a_1 b_3 & a_0 b_3 & & & & b_3
\end{array}
$$

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

Prodotti parziali

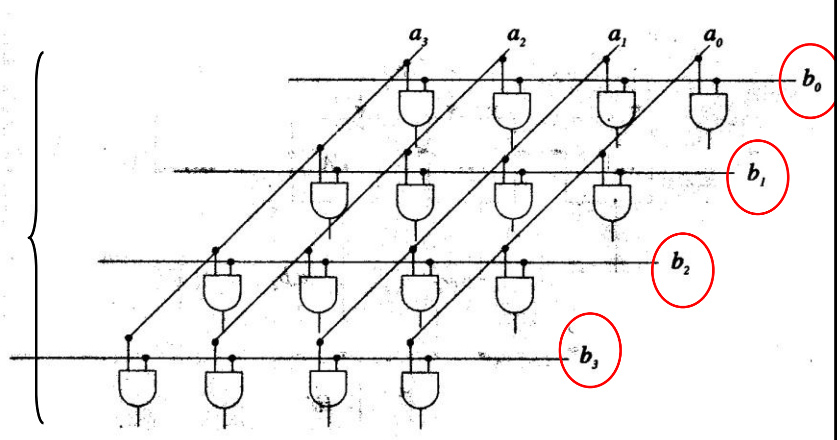
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | $a_3$ | $a_2$ | $a_1$ | $a_0$ | |
| | | | $a_3 b_0$ | $a_2 b_0$ | $a_1 b_0$ | $a_0 b_0$ | $b_0$ |
| | | $a_3 b_1$ | $a_2 b_1$ | $a_1 b_1$ | $a_0 b_1$ | | $b_1$ |
| | $a_3 b_2$ | $a_2 b_2$ | $a_1 b_2$ | $a_0 b_2$ | | | $b_2$ |
| $a_3 b_3$ | $a_2 b_3$ | $a_1 b_3$ | $a_0 b_3$ | | | | $b_3$ |

$b_0$ ($a_3$ $a_2$ $a_1$ $a_0$) 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 $P_3$
............



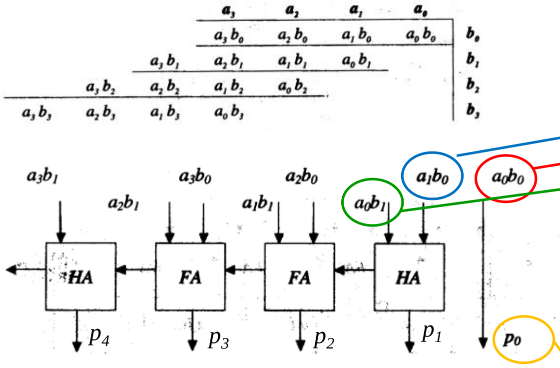
# Il circuito che effettua i prodotti

Prodotti parziali




$b_k$ agisce come interruttore, facendo passare 0 o A

# Somma delle prime 2 righe dei prodotti parziali




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

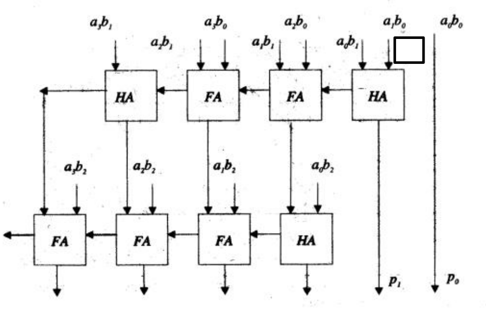
Somma dei primi 2 prodotti parziali:
Aggiunge il terzo prodotto parziale:

HA e FA non sono equivalenti per i diversi cammini critici.



# Somma della terza riga

I primi due prodotti parziali sono sommati 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
1 1 0 1 +
1 1 0 1 -
----------------------
0 0 0 0 0
1 0 0 1 1 1 +
0 0 0 0 0 - -
----------------------
1 1
1 0 0 1 1 1 +
1 1 0 1 - - - =
----------------------
1 0 0 0 1 1 1 1     -> $143|_{10}$

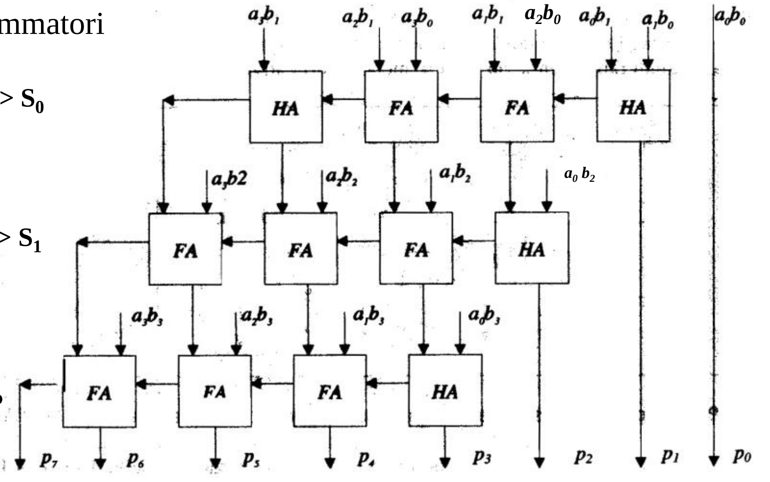
# Circuito completo della somma dei prodotti parziali

N-1 batterie di sommatori

$P_0+P_1 \rightarrow S_0$

$S_0+P_2 \rightarrow S_1$

$S_1+P_3 \rightarrow P$

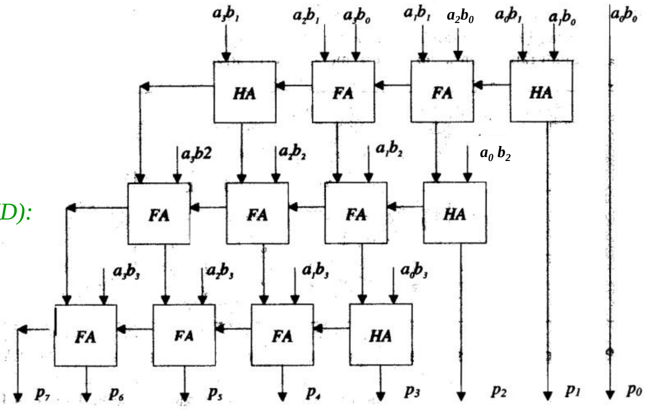



Problema: overflow: A e B su 32 bit => P su 64 bit.



# Valutazione della complessità

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




*Stadio prodotti (AND):*

A su N bit
B su M bit

N * M porte AND

*Stadio Somme:*

Se N = M = 4 numero totale di porte a 2 ingressi = 60

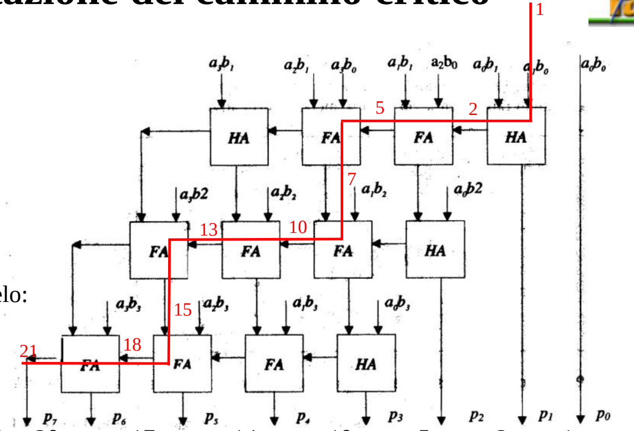
N sommatori per linea
M-1 righe

| | Numero linee | | Numero FA per linea | Numero HA per linea | Primo HA 1a linea | | Prodotti Parziali |
|---|---|---|---|---|---|---|---|
| $CO_{Tot}$ = | (M-1) | * | [(N-1) * 5 + | 1 * 2 ] | - 5 + 2 | + | M * 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.




Se N = M = 4 cammino critico totale = 21

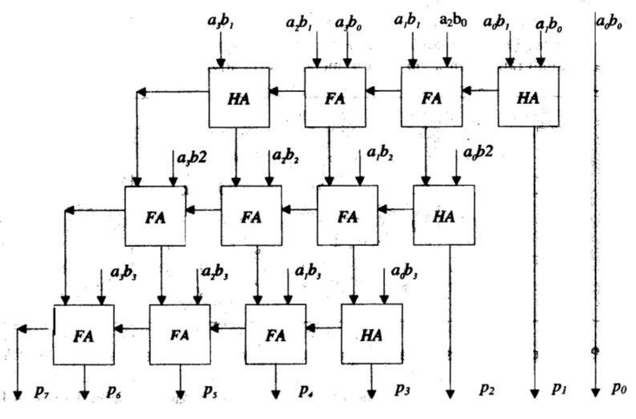
$$CC_{Tot} = 8 + (M-4)*(2+3) + 12 + 1$$



# 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 inferiore

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.

E’ costituita da circuiti combinatori. Utilizza i blocchi di base già visti.

Opera su parole (MIPS 32 bit).

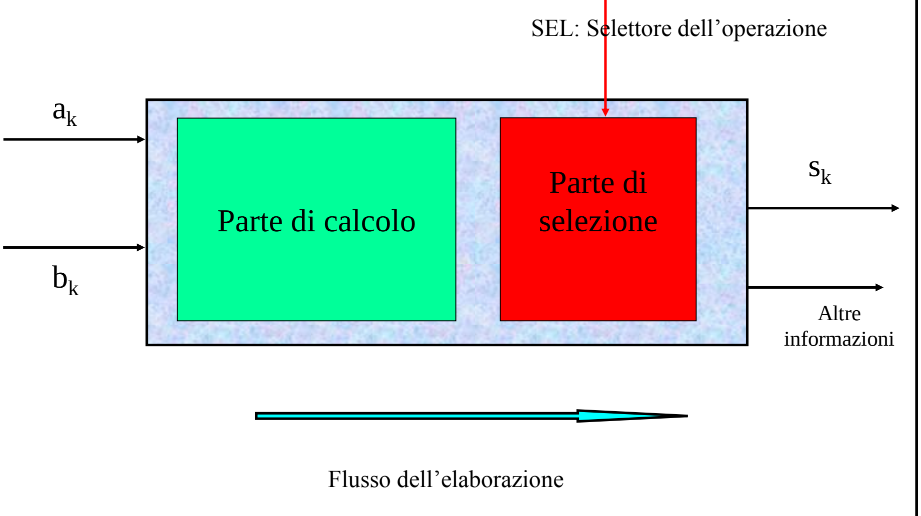
Le ALU non compaiono solamente nei micro-processori.

# Problematiche di progetto

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

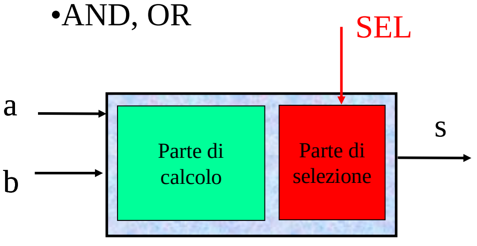


# Struttura a 2 livelli di una ALU



# Progettazione della ALU – 1 bit

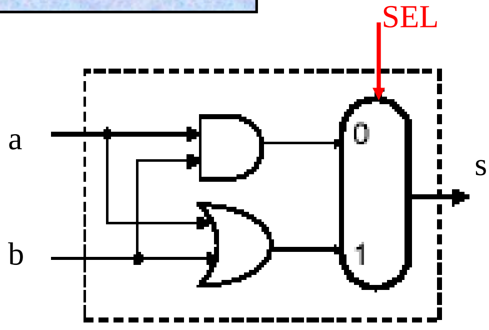
•AND, OR




| a b | a *AND* b | a *OR* b |
|---|---|---|
| 0 0 | 0 | 0 |
| 0 1 | 0 | 1 |
| 1 0 | 0 | 1 |
| 1 1 | 1 | 1 |

SEL = 0
s = AND(a,b)

SEL= 1
s = OR(a,b)

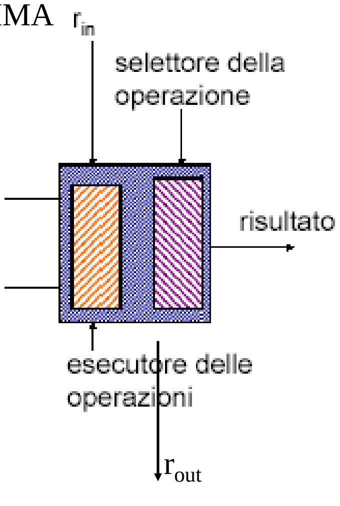


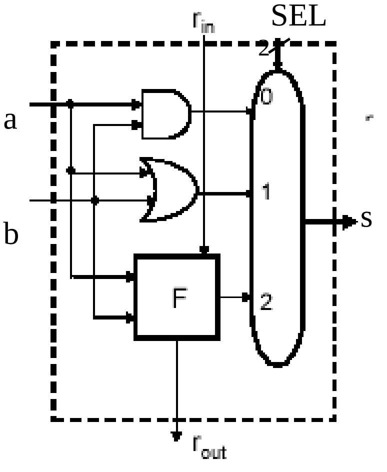

1 porta AND
1 porta OR
1 Mux



# La nuova struttura della ALU – 1 bit

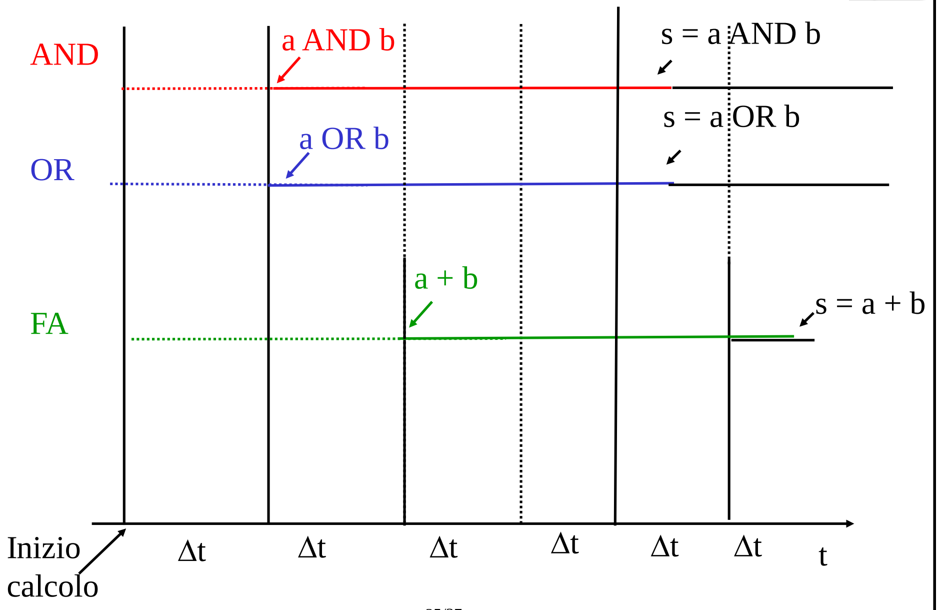
•AND
•OR
•SOMMA




Perchè SEL non viene messo in ingresso?

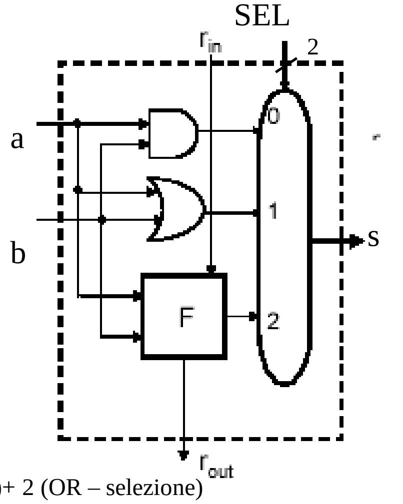
# I cammini critici all'interno della ALU






# Valutazione ALU a 1 bit

- AND
- OR
- SOMMA




Complessità 1° livello (calcolo): 5+2 = 7
Complessità 2° livello (mux): 3*1+(3+2*2) = 10 (Decoder + AND (semaforo) + OR (congiunzione))
Complessità totale: 7+10 = 17

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

# Sommario

ALU ad 1 bit

ALU a 32 bit

Comparazione, Overflow, Test di uguaglianza

Tecnologie di costruzione di una ALU

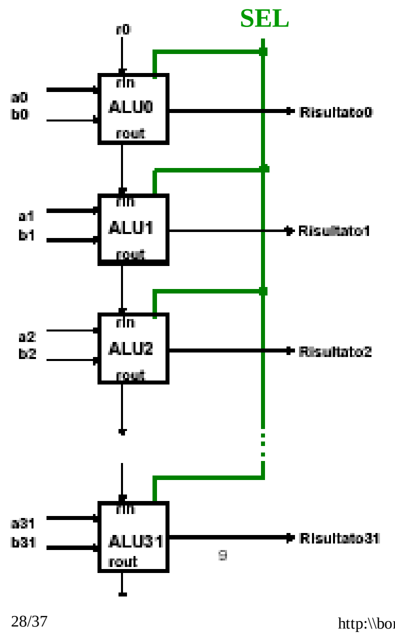


# ALU a 32 bit

Come collegare le ALU ad 1 bit?

Flusso di calcolo

Perchè non si può parallelizzare?

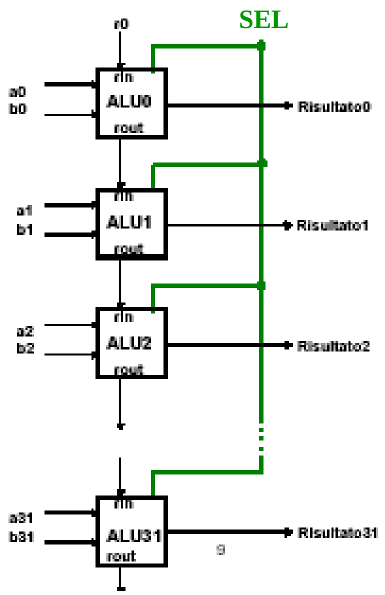


# Valutazione ALU a 32 bit

Complessità: 17 x 32 = 544 porte logiche

Cammino critico: 3 x 31 (propagazione riporti) + 2 ($s_{31}$) + 1 (semaforo ultimo mux) + 2 (congiunzione ultimo mux) = 98 porte logiche

*per 4 operazioni*






# 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 |

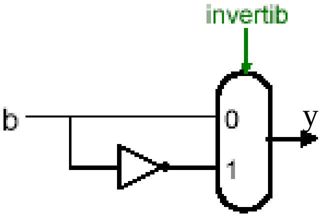
**Posso quindi scrivere: $-b = \bar{b} + 1$**

# Sottrazione

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

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

a)



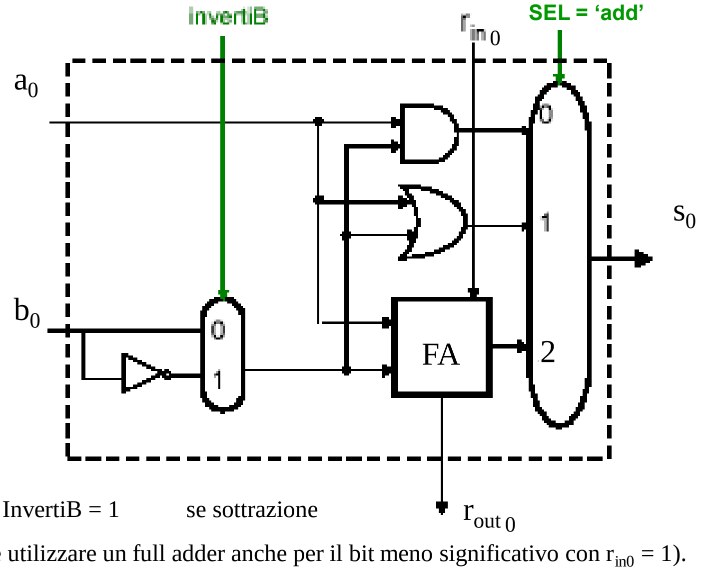

Iff invertib
y = !b

Aggiunge 2 porte logiche al cammino critico.

b) Da dove prendo la costante 1?



# Sottrazione - $ALU_0$

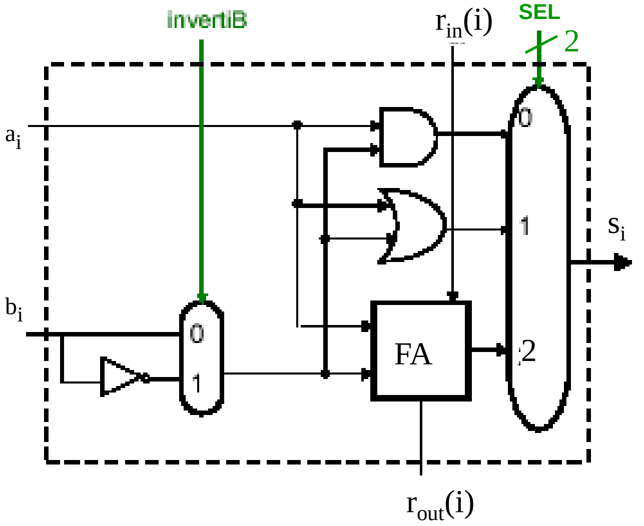



$r_{in}(0)$ = InvertiB = 1 se sottrazione

(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.

# Sottrazione - $ALU_i$

- AND
- OR
- SOMMA
- SOTTRAZIONE




$r_{in}(i) = r_{out}(i-1)$ $i = 1,2,3,....31$ $i \neq 0$
InvertiB = 1 se sottrazione



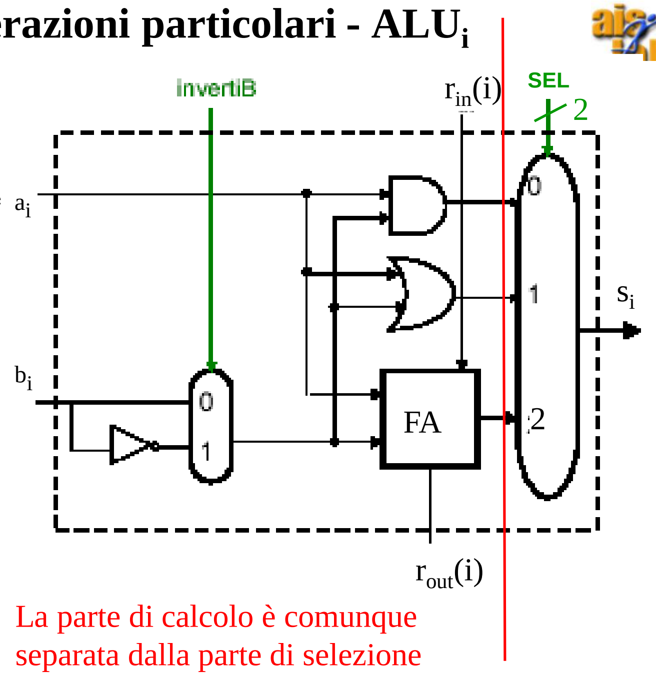
# Operazioni particolari - $ALU_i$

E' possibile programmare questa ALU per eseguire

a AND !b

oppure:

a OR !b




InvertiB = 1
SEL = AND, OR

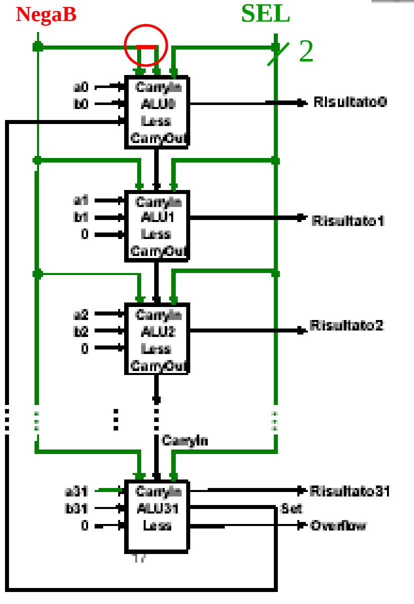
La parte di calcolo è comunque separata dalla parte di selezione

# 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