



# Architetture degli Elaboratori e delle Reti I

Laboratorio – linea 2 (G-Z)

5

## Moltiplicazione e ALU

- **Docente teoria:** prof. Federico Pedersini  
(<https://homes.di.unimi.it/pedersini/AE-INF.html>)
- **Docente laboratorio:** Matteo Re  
(<https://homes.di.unimi.it/re/arch1-lab-2015-2016.html>)
- **Sito laboratorio turno 2:**  
(<http://basilico.di.unimi.it/doku.php?id=pub:arch1-lab-2015-2016>)



# Architetture degli Elaboratori e delle Reti I

Laboratorio – linea 2 (G-Z)

5

## Moltiplicazione e ALU

- Richiami
- Moltiplicazione
- Arithmetic Logic Unit (ALU)



# Architetture degli Elaboratori e delle Reti I

Laboratorio – linea 2 (G-Z)

5

## Richiami

**Riferimenti:** slide teoria (da pg3, down, in poi)

[https://homes.di.unimi.it/pedersini/AER/AE1\\_15\\_L07-09.pdf](https://homes.di.unimi.it/pedersini/AER/AE1_15_L07-09.pdf)

In questa serie di slide iniziale vedremo un componente Logisim che utilizzeremo in questo laboratorio e realizzeremo degli esempi che richiedono la costruzione di sottocircuiti riutilizzabili.

Passeremo poi alla realizzazione di un circuito per la realizzazione della moltiplicazione (di numeri a 3 bit)

Realizzeremo un componente 1bit ALU

Utilizzeremo 1bit ALU per costruire una 8bit ALU



# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

## MULTIPLEXER (MUX):

Operatore di **selezione**

**IN:**     **n** linee di input (**data**)  
          **k** linee di controllo (**select**)

**OUT:**    **1** linea

Funzionamento:

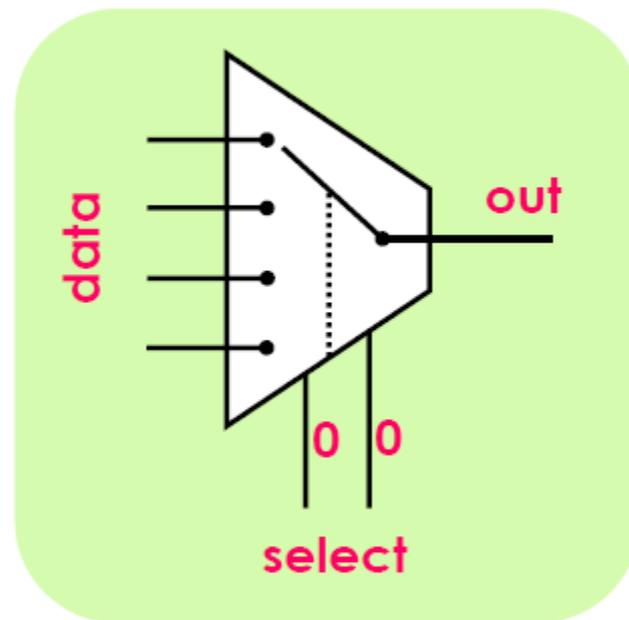
il valore fornito sulla linea di controllo viene connessa all'uscita la linea di ingresso selezionata.

Quante linee di selezione?

$$k = \text{ceil}(\log_2 n)$$

Linee di input:         **$n = 4$**

Linee di controllo:    **$k = \text{ceil}(\log_2 4) = 2$**





# Architetture degli Elaboratori e delle Reti I

5

Laboratorio – linea 2 (G-Z)

**Proprietà :**

## Select Bits.

Determina il numero di possibili valori in ingresso.  
Es. se vale 3 la possibilità di selezionare uno tra  $2^3$  valori.

**Include Enable?** Se impostato a No è sempre attivo (altrimenti ha un ingresso in più).

**Selection: Multiplexer**

Facing	East
Select Location	Bottom/Left
Select Bits	3
Data Bits	1
Disabled Output	Floating
Include Enable?	No



## Circuiti aritmetici

**Riferimenti:** slide teoria (da pg5, up, in poi)

[https://homes.di.unimi.it/pedersini/AER/AE1\\_15\\_L07-09.pdf](https://homes.di.unimi.it/pedersini/AER/AE1_15_L07-09.pdf)

Circuiti combinatori che realizzano operazioni aritmetiche su numeri binari.  
Vedremo **somma, moltiplicazione e ALU.**

Partiamo con la somma ...



# Architetture degli Elaboratori e delle Reti I

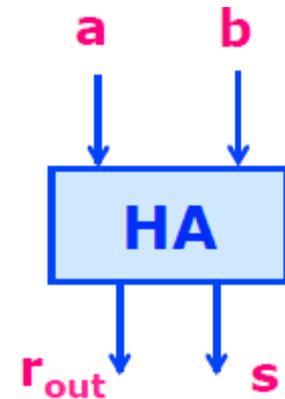
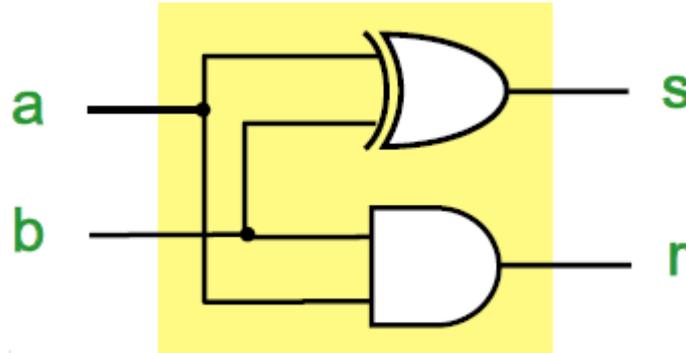
5

Laboratorio – linea 2 (G-Z)

## HALF ADDER :

- ❖ **Somma aritmetica tra 2 bit**
  - 2 ingressi: addendi: **a**, **b**
  - 2 uscite: somma: **s**  
riporto: **r**

**Note:**  
. Non ha riporto in ingresso.



**Obiettivo LAB05\_1 :** Realizzazione in Logisim e personalizzazione aspetto visivo del componente realizzato.



# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

## Step necessari :

- 1) Fare riferimento a schema circuito in slide teoria.
- 2) Per **ogni** porta impostare il numero di input a **2** e Gate size a **Narrow**.
- 3) Cambiare nome in Circuit Name (HA)
- 4) Creare un nuovo circuito nel progetto : **Project** → **Add circuit ...** → inserire il nome del nuovo circuito (es. main)

The screenshot shows the Logisim software interface. The main window displays a circuit diagram with two inputs, A and B, each represented by a square box containing a '0'. The circuit consists of two gates: an OR gate and an AND gate. The OR gate has two inputs connected to A and B, and its output is labeled 'S'. The AND gate also has two inputs connected to A and B, and its output is labeled 'r'. The circuit is connected to a power source (represented by a green circle with a '0') and a ground symbol (represented by a triangle with a circle). The properties window on the left shows the circuit name as 'main', the shared label as 'main', and the shared label font as 'SansSerif Plain 12'. The circuit name 'main' is highlighted with a red arrow.

NB: Ricordiamoci di impostare le label ...



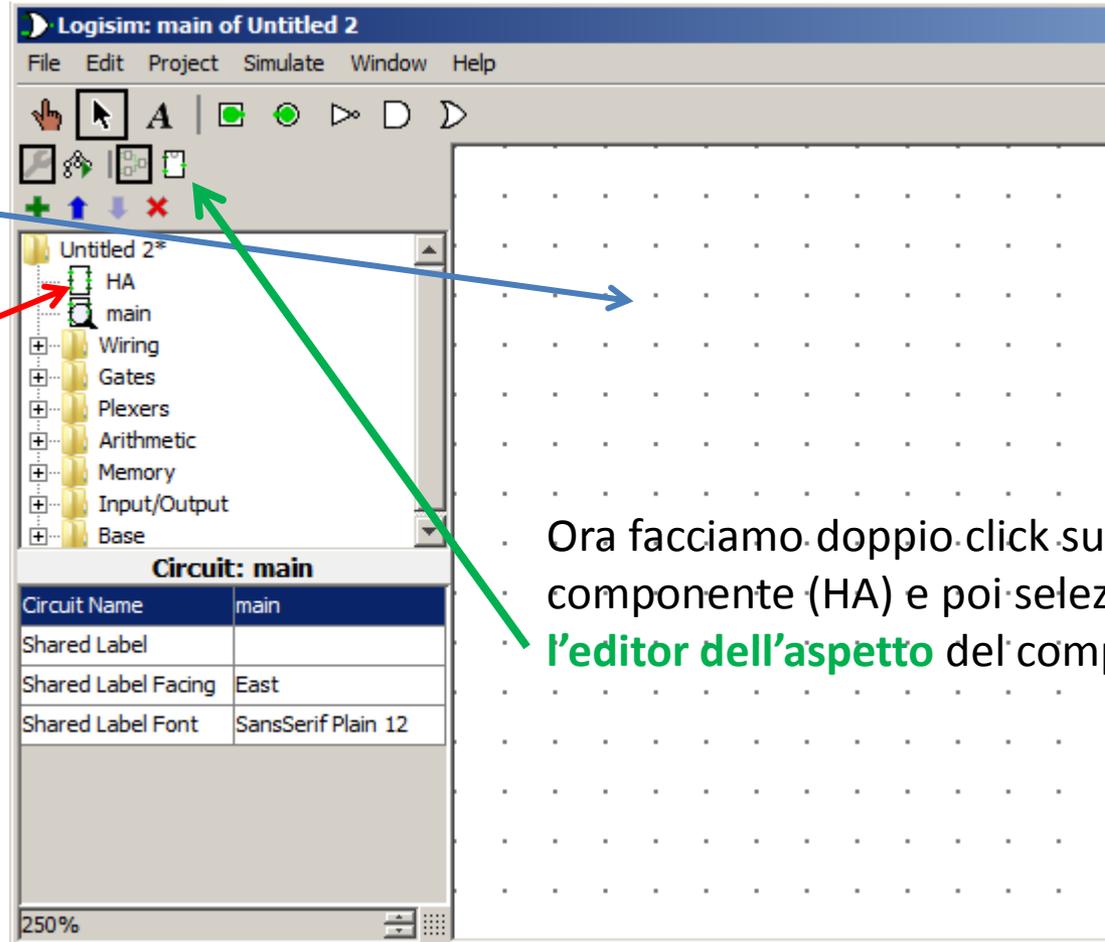
# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

## Cosa otteniamo:

- 1) Un circuito vuoto (quello appena creato)
- 2) Un nuovo componente nel progetto corrente



Ora facciamo doppio click sul nuovo componente (HA) e poi selezioniamo **l'editor dell'aspetto** del componente.



# Architetture degli Elaboratori e delle Reti I

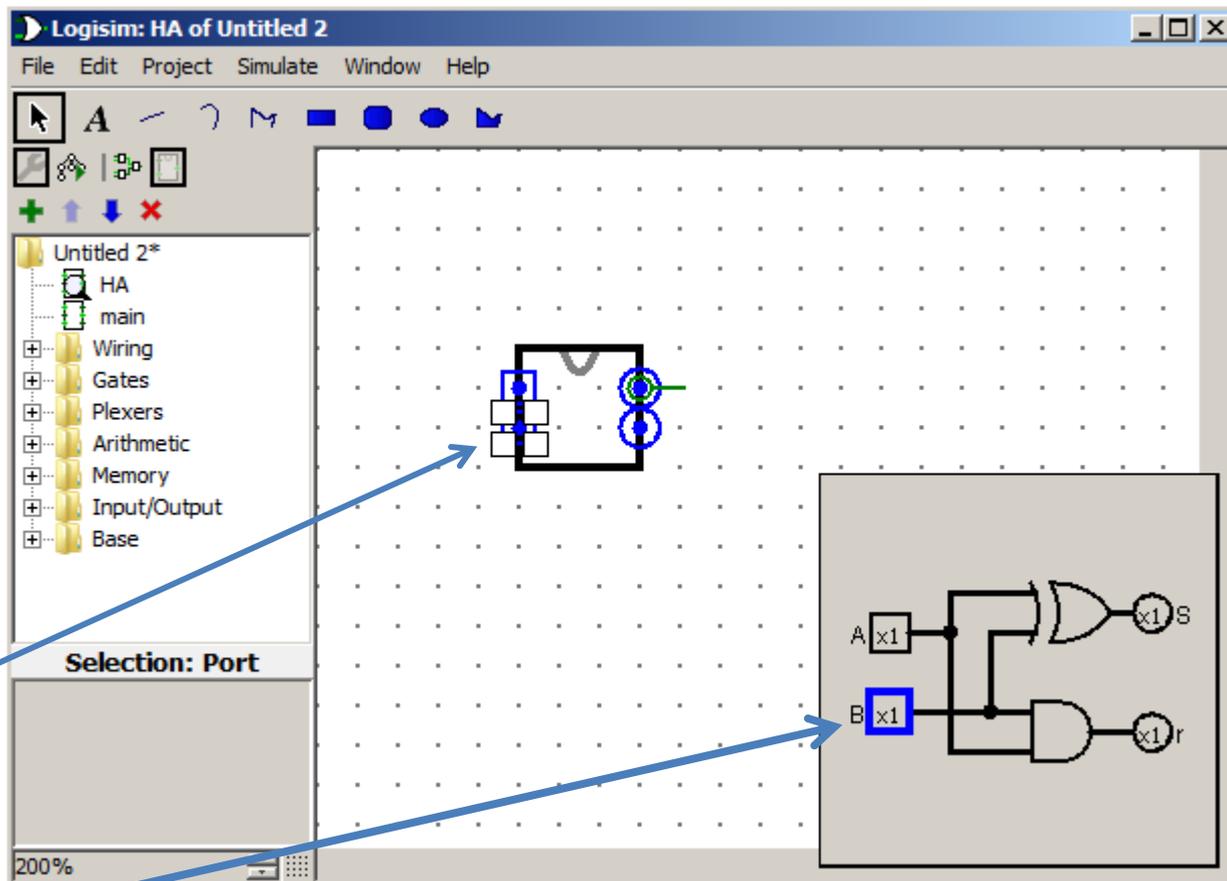
# 5

Laboratorio – linea 2 (G-Z)

**Nell'editor è possibile:**

- 1) Ridimensionare il quadrato
- 2) Spostare gli elementi lungo il perimetro del quadrato e cambiarne le proprietà

**NB:** Ogni volta che selezionate un sottocomponente compare uno schema del circuito in cui il componente è evidenziato.



E' un'ottima idea quella di inserire delle label all'interno della rappresentazione del componente (per riconoscere in modo più facile ingressi e uscite)



# Architetture degli Elaboratori e delle Reti I

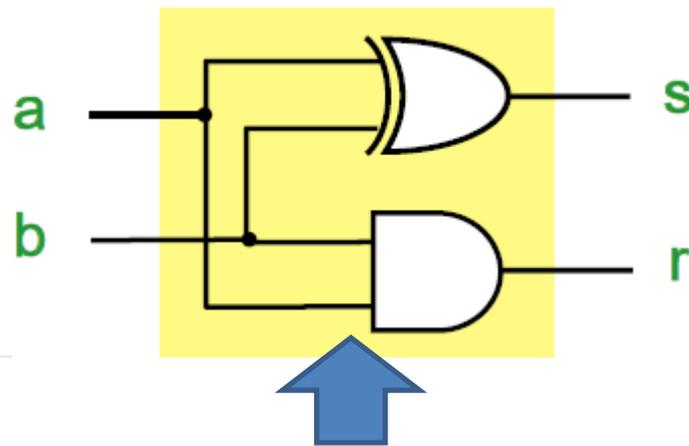
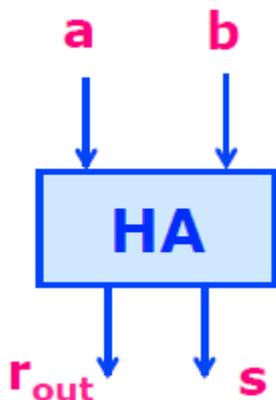
# 5

Laboratorio – linea 2 (G-Z)

L'aspetto non va definito a caso ...  
Esistono varie opzioni per «progettare» l'aspetto visuale di un componente personalizzato:

- 1) Seguite gli esempi presenti nelle slide di teoria
- 2) Progettate l'aspetto in modo che questo renda semplice l'utilizzo in circuiti di cui avete già lo schema

Opzioni (vedere slide teoria) :



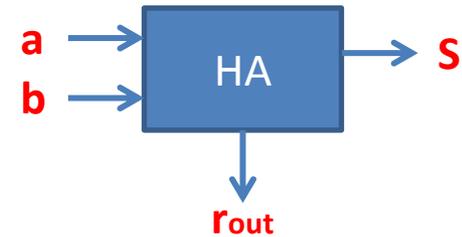
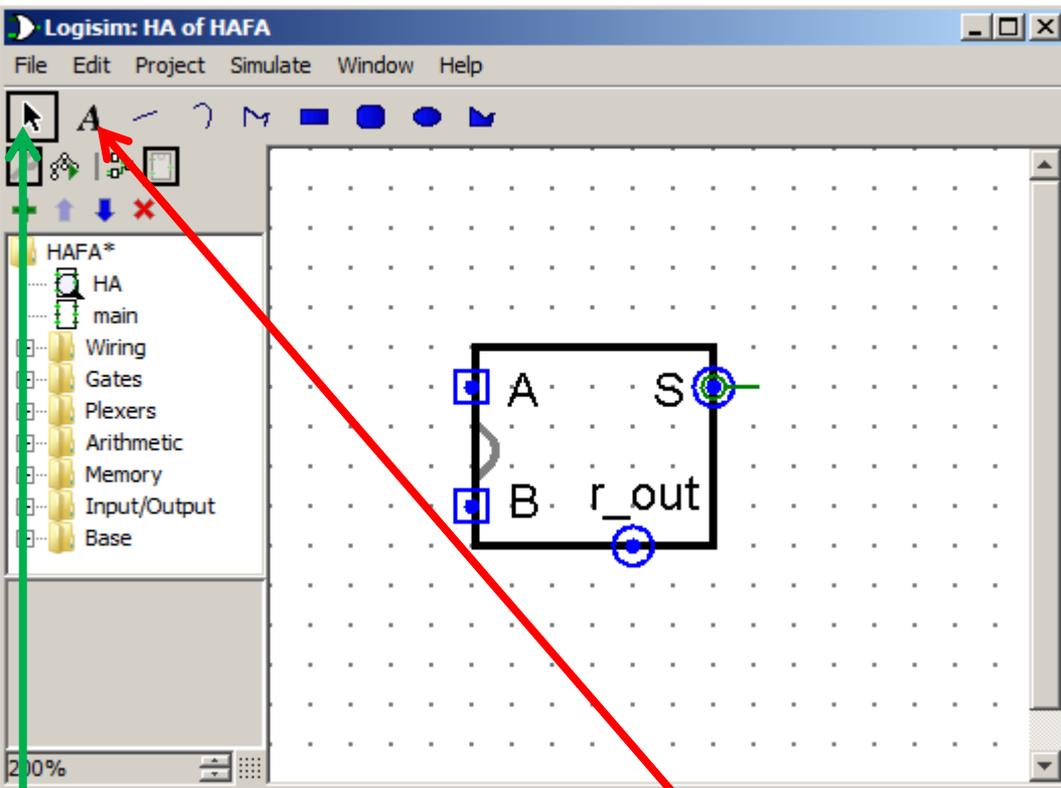
Noi realizzeremo una Rappresentazione simile a questo schema ma con l'uscita **r** in basso.



# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)



NB: Per inserire le etichette usare **l'apposito strumento** Inserire il testo e confermare (invio). **Selezionare e** riposizionare dove necessario.



# Architetture degli Elaboratori e delle Reti I

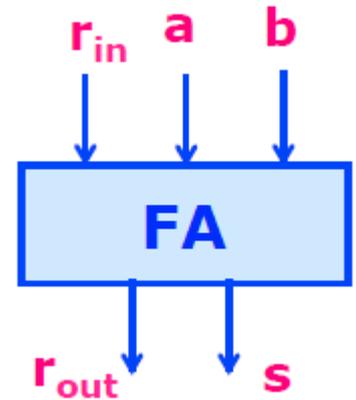
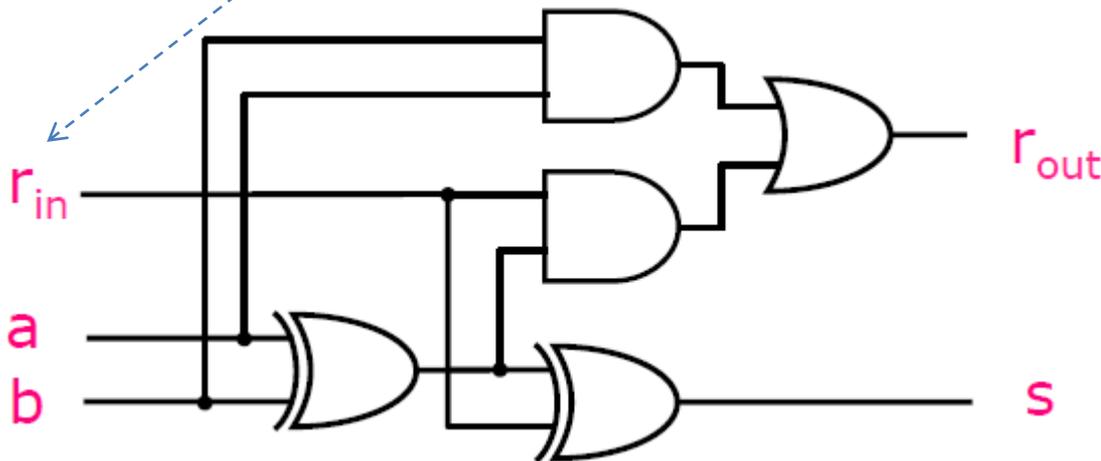
# 5

Laboratorio – linea 2 (G-Z)

## FULL ADDER :

Sommando numeri **di più bit**, c'è un problema:  
va gestito anche il **riporto in ingresso**

- 3 ingressi:  **$a, b, r_{IN}$**
- 2 uscite:  **$s, r_{OUT}$**



**Note:**  
. **C'è** riporto in ingresso.

**Obiettivo LAB05\_2 :** Realizzazione in Logisim e personalizzazione aspetto visivo del componente realizzato.



# Architetture degli Elaboratori e delle Reti I

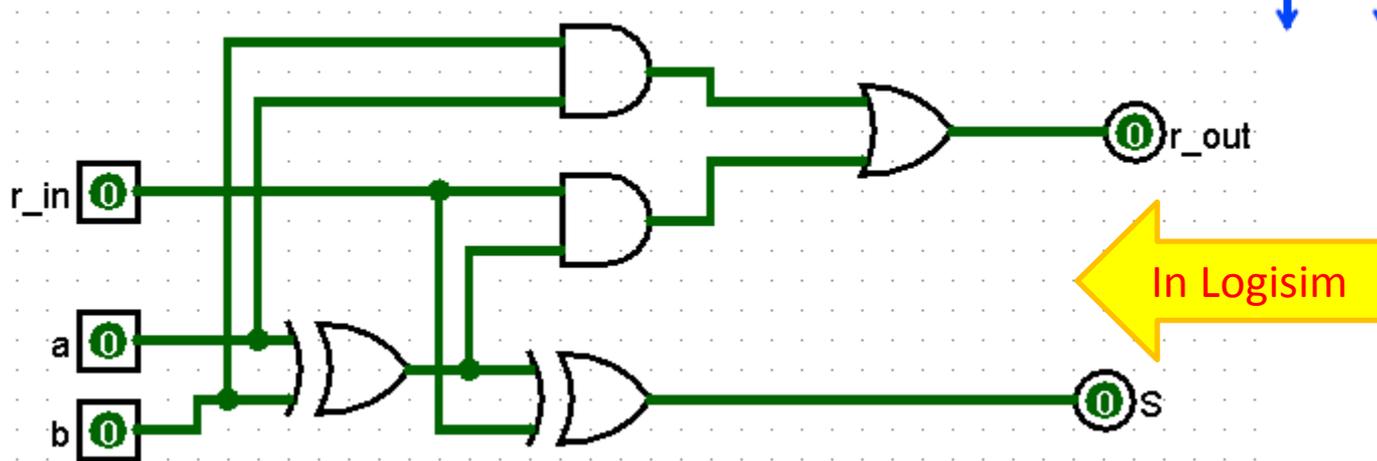
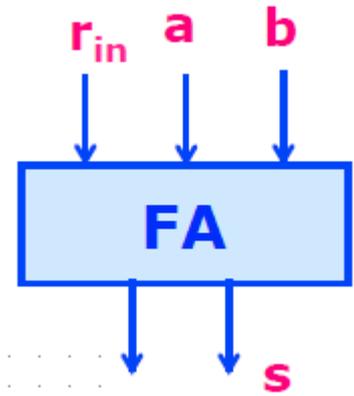
5

Laboratorio – linea 2 (G-Z)

## FULL ADDER :

Sommando numeri di più bit, c'è un problema:  
va gestito anche il riporto in ingresso

- 3 ingressi:  $a, b, r_{IN}$
- 2 uscite:  $s, r_{OUT}$



In Logisim

**NB:** Valgono le stesse regole di prima (impostare per ogni porta in modo corretto il numero di ingressi e settare Gate size a narrow). Impostare sempre il valore delle etichette. L'aspetto del componente come lo impostiamo?

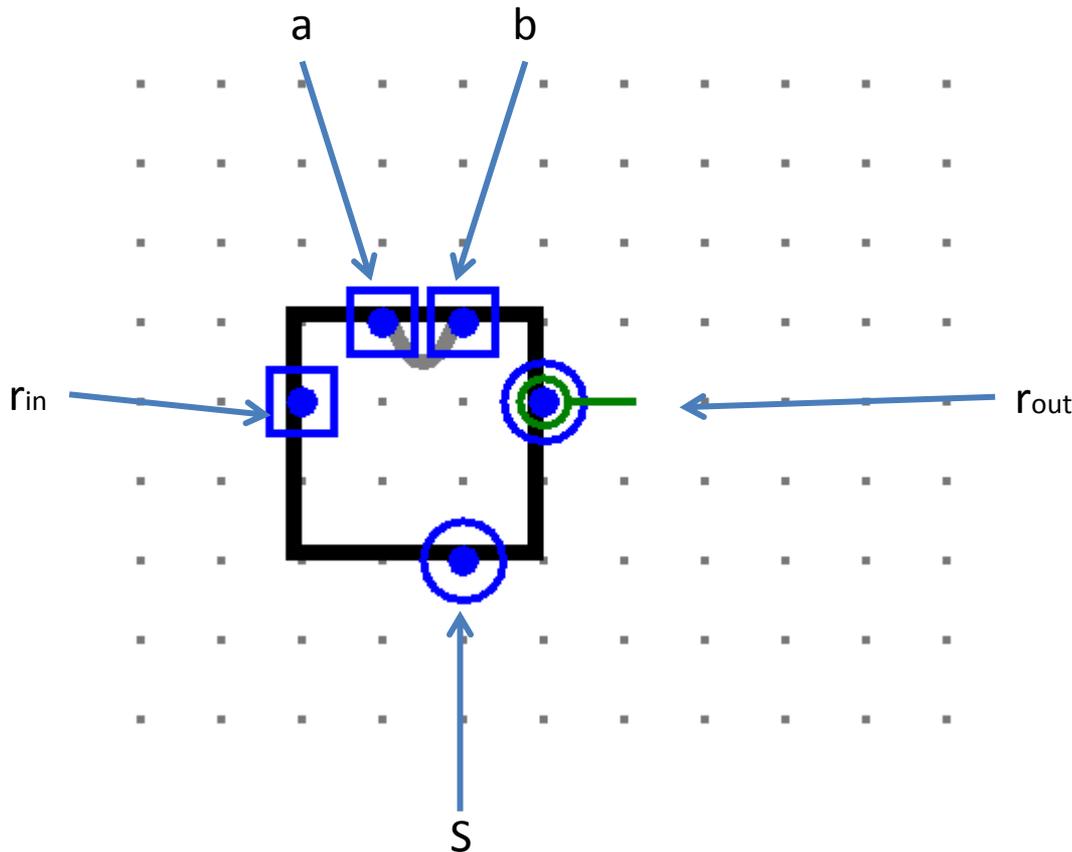


# Architetture degli Elaboratori e delle Reti I

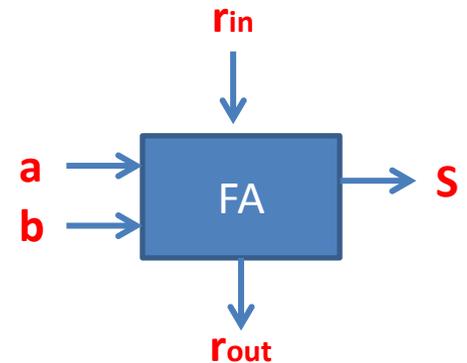
Laboratorio – linea 2 (G-Z)

5

**Esercizio 1:** Realizzare questa rappresentazione del componente



**Esercizio 2:** Realizzare questa rappresentazione del componente



**Esercizio 3:** Confrontate la visualizzazione del componente ottenuta in esercizio 2 con il componente (Arithmetic) **Adder** di Logisim. Cosa notate?<sup>15</sup>



# Architetture degli Elaboratori e delle Reti I

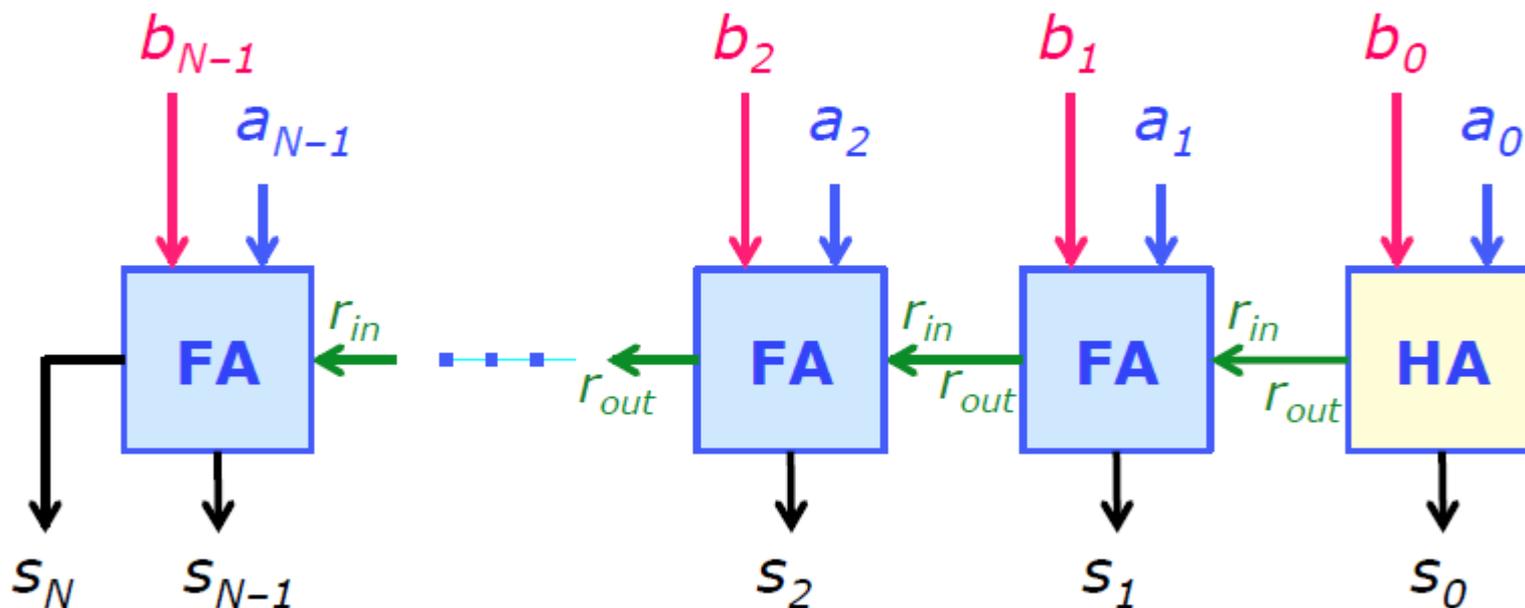
# 5

Laboratorio – linea 2 (G-Z)

## ❖ Sommatore a propagazione di riporto

**IN:** 2 parole di **N** bit

**OUT:** somma di **N+1** bit



*Cammino critico?* (HA=1 ; FA=3)

$$C = 3(N-1)+1 = 3N - 2$$



# Architetture degli Elaboratori e delle Reti I

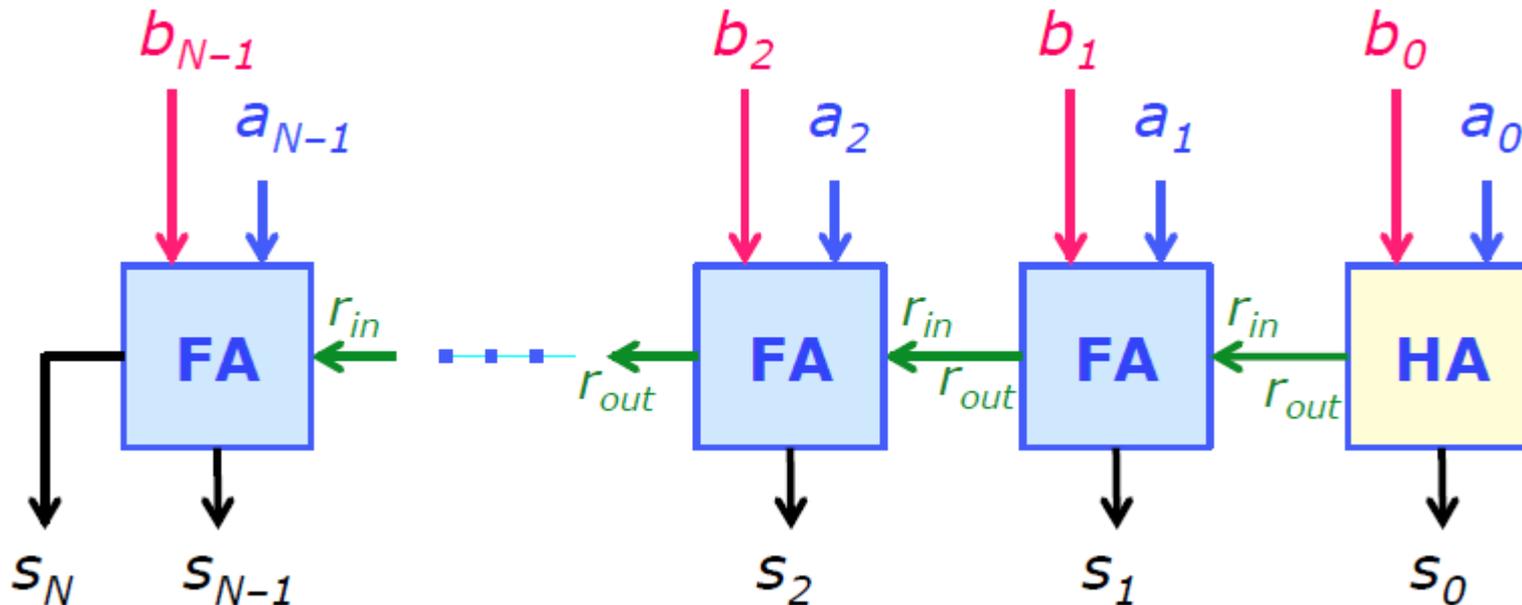
# 5

Laboratorio – linea 2 (G-Z)

## Sommatore a $n$ bit a propagazione di riporto:

Dati due numeri di  $n$  bit sommare le coppie di bit corrispondenti. Il primo componente è un half adder (a questo livello non serve gestire il riporto in ingresso) seguito da una serie di componenti full adder. **NB:** Nel risultato va messo anche il riporto in uscita dell'ultimo componente full adder.

**Esercizio 4:** Realizzare questo circuito in Logisim





# Architetture degli Elaboratori e delle Reti I

## 5

Laboratorio – linea 2 (G-Z)

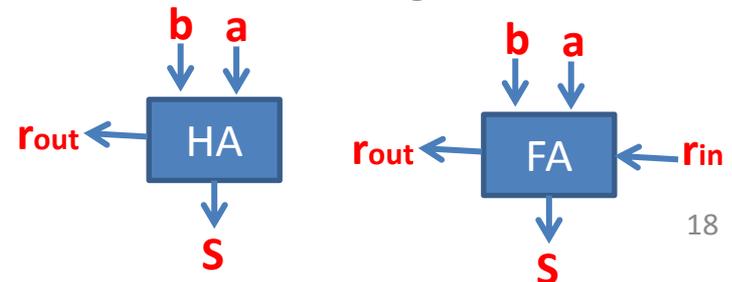
### Sommatore a $n$ bit a propagazione di riporto:

Dati due numeri di  $n$  bit sommare le coppie di bit corrispondenti. Il primo componente è un half adder (a questo livello non serve gestire il riporto in ingresso) seguito da una serie di componenti full adder. **NB:** Nel risultato va messo anche il riporto in uscita dell'ultimo componente full adder.

**Esercizio 4:** Realizzare in Logisim il sommatore a propagazione di riporto

Per risolvere l'esercizio ...

- E' necessario reimplementare HA e FA? **No** ... le logiche le avete già implementate.
- Potete utilizzare i componenti creati fino a questo momento? **Dipende**. Dovete verificare che le visualizzazioni dei componenti abbiano ingressi e uscite nelle posizioni corrette per poter realizzare il circuito **esattamente** come è mostrato nell'immagine della slide precedente.
- Le visualizzazioni dei componenti da usare sono:





# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria :**

Moltiplicando

$$\begin{array}{r} 11011 \times (27_{10}) \\ 111 = (7_{10}) \\ \hline 111111 \\ 11011 \\ 11011 - \\ 11011 - - \\ \hline 10111101 \quad (189_{10}) \end{array}$$

Moltiplicatore

Prodotto

❖ **Come fare il calcolo con circuiti logici ?**

- Possiamo scomporre l'operazione in **due stadi**:
- **Primo stadio: prodotti parziali**
  - ◆ si mette in AND ciascun bit del moltiplicatore con i bit corrispondenti del moltiplicando
- **Secondo stadio: somme**
  - ◆ si effettuano le somme (full adder) dei bit sulle righe contenenti i prodotti parziali



# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria** : la matrice dei prodotti parziali

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

In binario i prodotti parziali sono degli **AND**

Nel prossimo esercizio costruiremo in Logisim un moltiplicatore per **numeri a 3 bit**. Seguiremo **esattamente** lo stesso schema circuitale presente nelle slide di teoria. Una volta realizzato il moltiplicatore a 3 bit avrete, come esercizio, l'estensione del circuito in modo da permettere la moltiplicazione di numeri a 4 bit (sempre seguendo lo schema del circuito visto nelle lezioni di teoria).

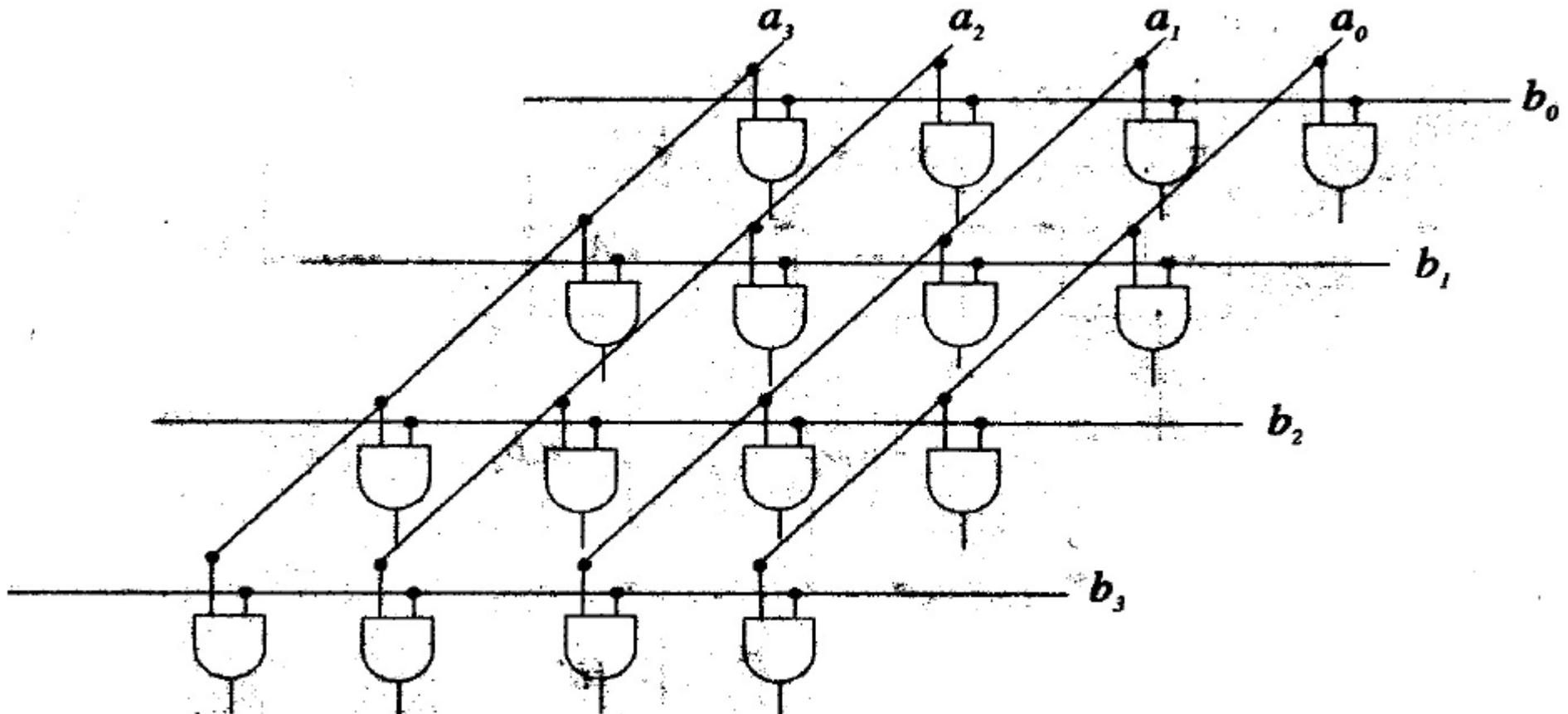


# Architetture degli Elaboratori e delle Reti I

5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria** : schema generale calcolo prodotti parziali



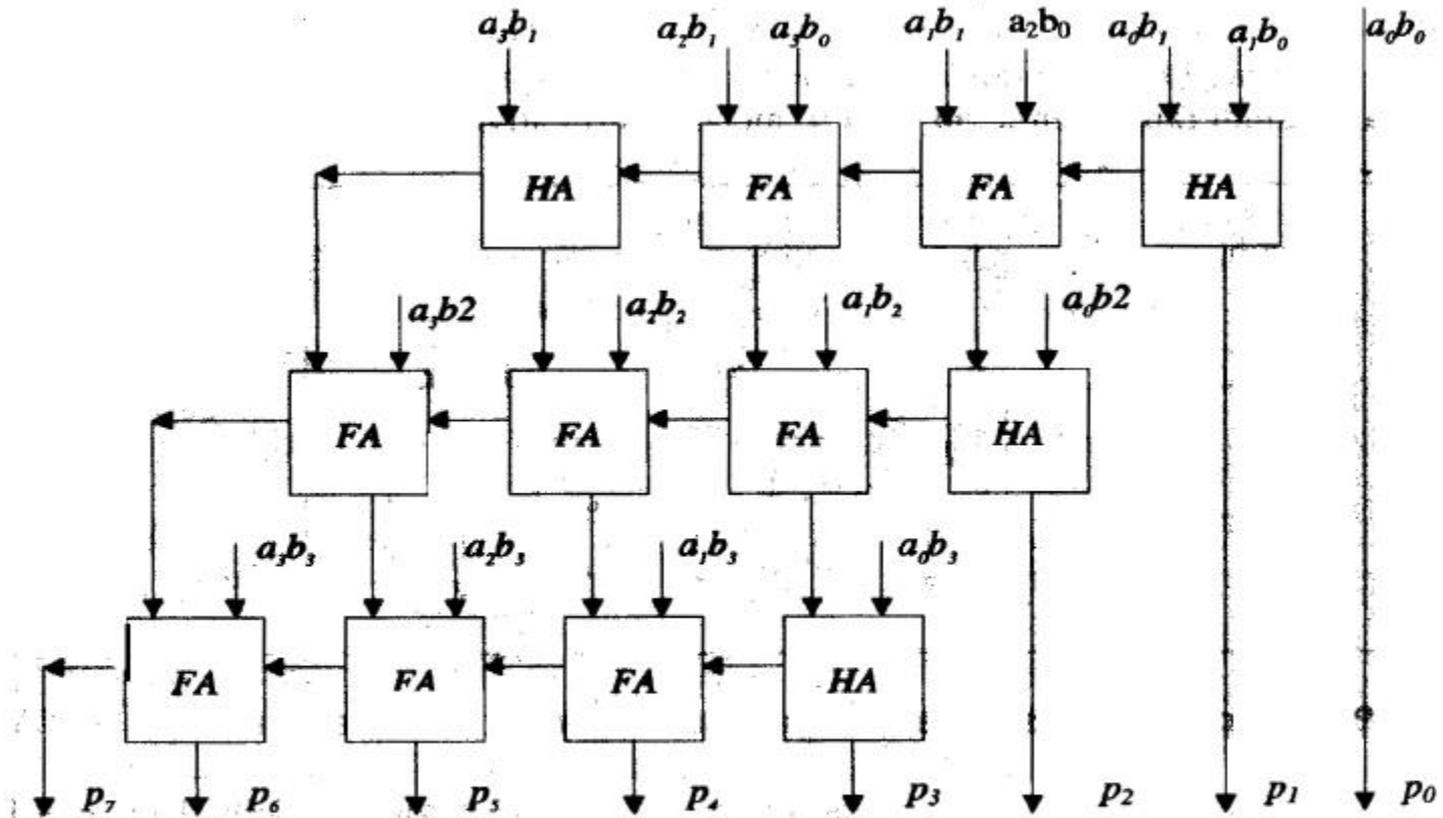


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria** : schema generale somma prodotti parziali



A e B su: **N bit** → P su: **2N bit**



# Architetture degli Elaboratori e delle Reti I

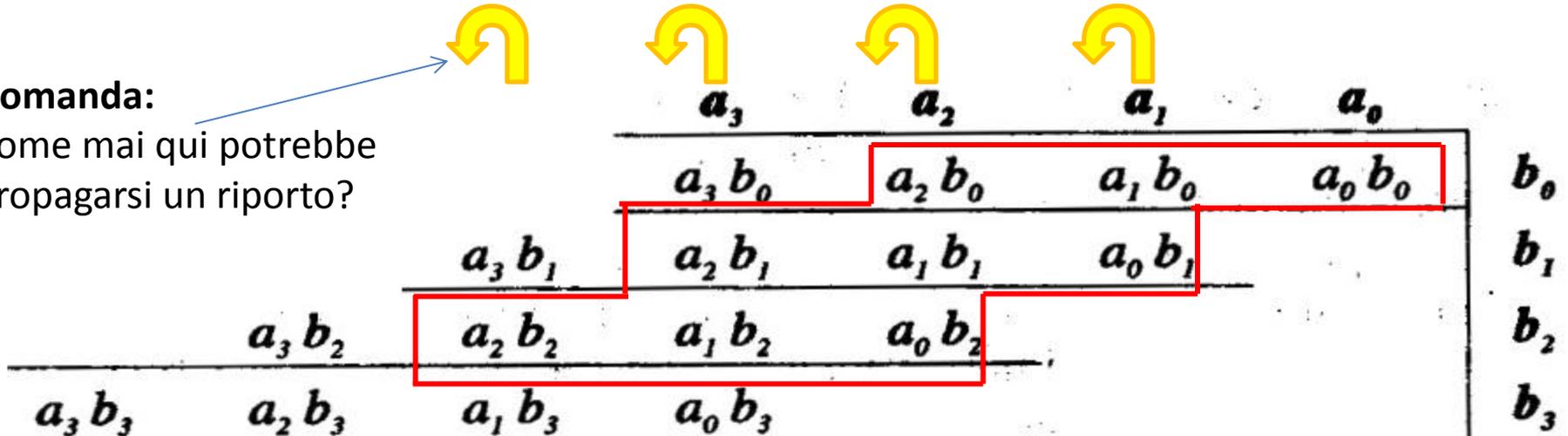
# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria** : torniamo alla matrice dei prodotti parziali  
(e vediamo cosa succede nel caso del prodotto di due numeri di 3 bit)

**Domanda:**

Come mai qui potrebbe propagarsi un riporto?



Le frecce gialle indicano punti in cui potrebbero propagarsi dei riporti

**Note:**

$p_0$  deriva direttamente dal prodotto  $a_0 b_0$  .  $p_1$  è la somma di  $a_1 b_0$  e di  $a_0 b_1$  . Non c'è necessità di considerare riporto in ingresso (ma potrebbe esserci riporto in uscita). Quindi è corretto utilizzare un Half Adder.

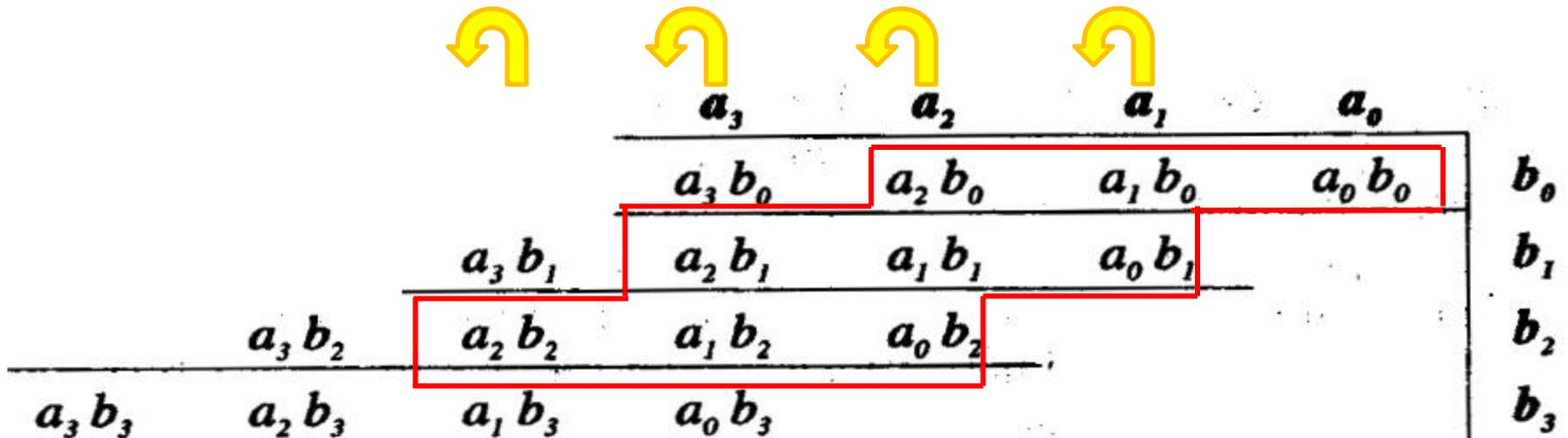


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria** : torniamo alla matrice dei prodotti parziali  
(e vediamo cosa succede nel caso del prodotto di due numeri di 3 bit)



Per aumentare l'aderenza della soluzione proposta con le slide di teoria per la parte del circuito che si occupa delle somme dei prodotti parziali verranno utilizzati dei componenti HA e Adder (sappiamo da Esercizio 3, slide 14 che il componente Adder di Logisim è un Full Adder).

**NB:** Invece che sviluppare il circuito dall'alto verso il basso la soluzione propone uno sviluppo da sinistra a destra.

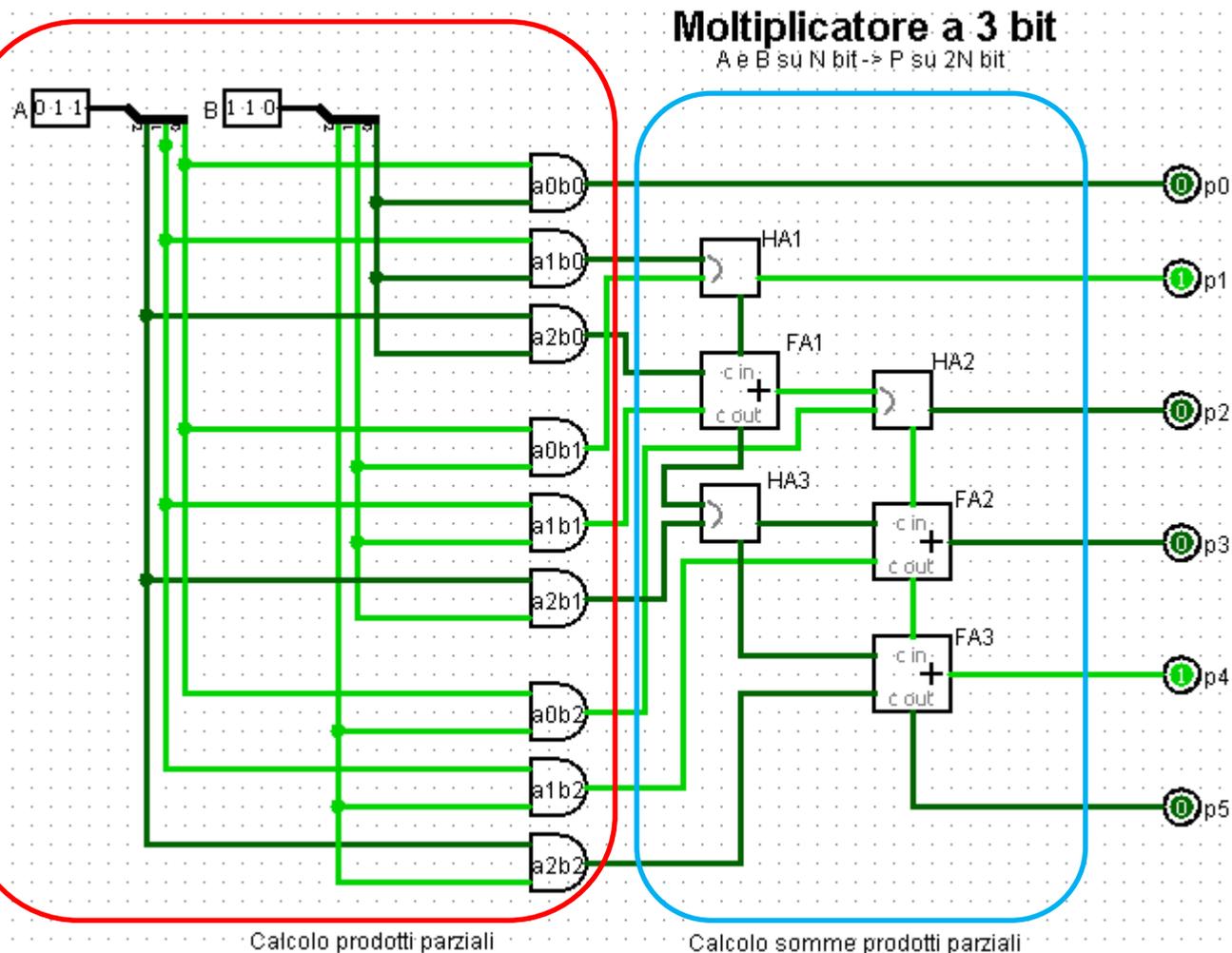


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



Come fare il calcolo con circuiti logici ?

- Possiamo scomporre l'operazione in due stadi:
- Primo stadio: prodotti parziali
  - ✦ si mette in AND ciascun bit del moltiplicatore con i bit corrispondenti del moltiplicando
- Secondo stadio: somme
  - ✦ si effettuano le somme (full adder) dei bit sulle righe contenenti i prodotti parziali

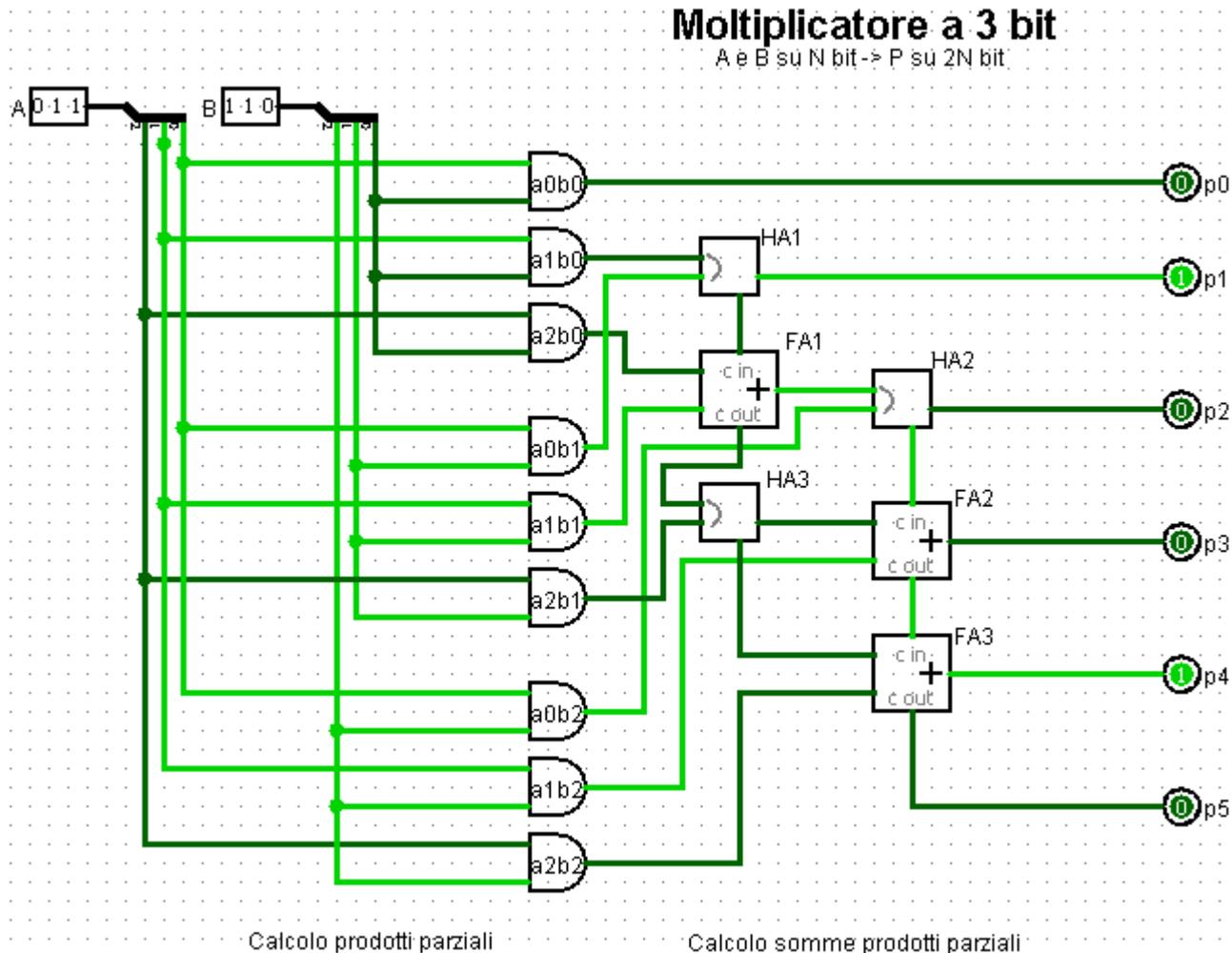


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



Concentriamoci sulla parte delle somme ...

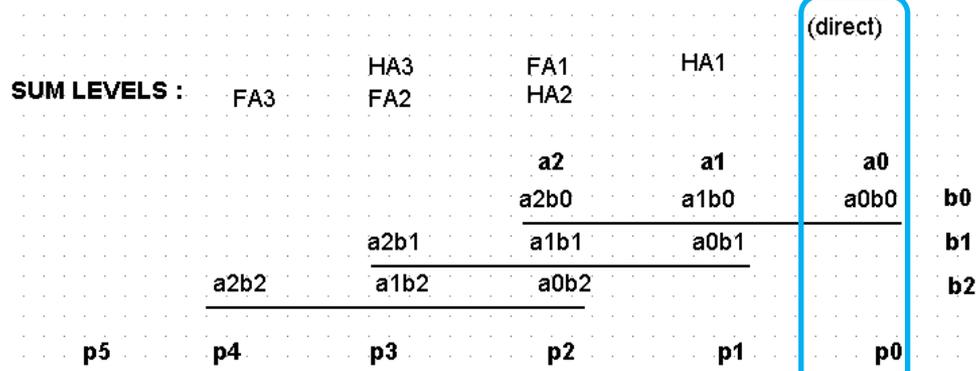
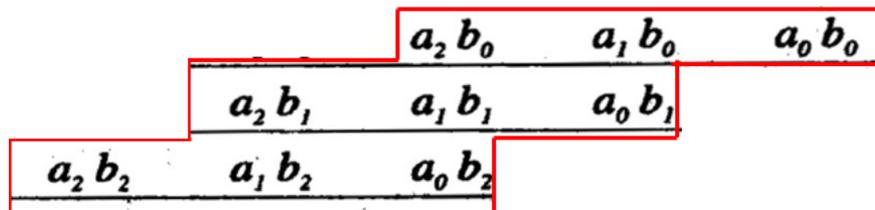
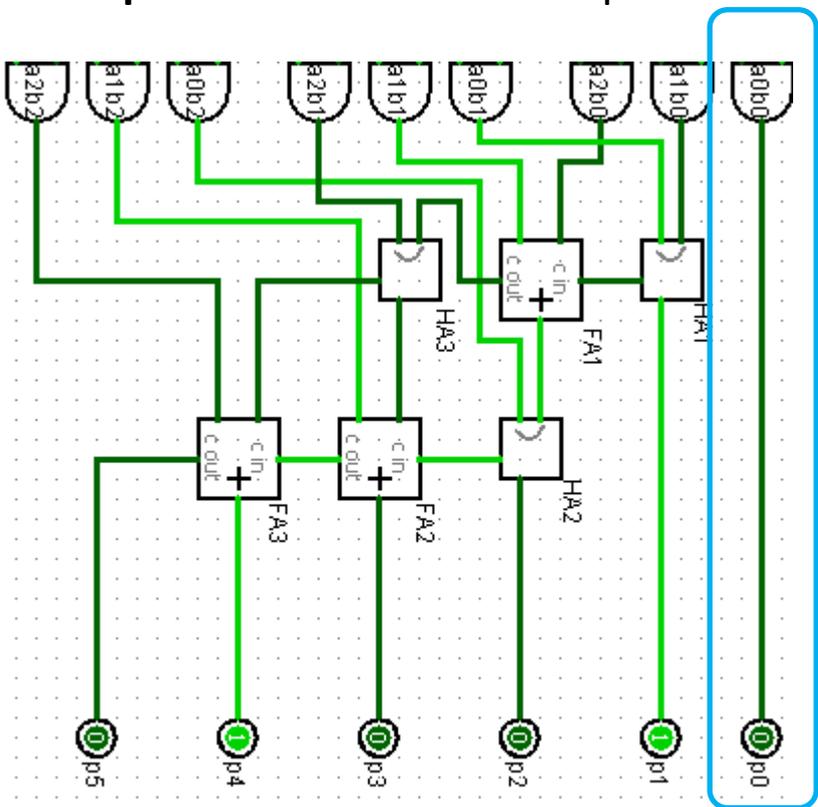


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



$$\begin{array}{r}
 A \quad 0 \ 1 \ 1 \ x \\
 B \quad 1 \ 1 \ 0 \ = \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 \quad \quad 0 \ 0 \ 0 \ + \\
 \quad \quad 0 \ 1 \ 1 \ - \\
 \hline
 1 \ 1 \\
 \quad \quad 0 \ 1 \ 1 \ 0 \ + \\
 \quad \quad 0 \ 1 \ 1 \ - \\
 \hline
 0 \ 1 \ 0 \ 0 \ 1 \ 0 \\
 p5 \ p4 \ p3 \ p2 \ p1 \ p0
 \end{array}$$

Concentriamoci sulla parte delle somme ...



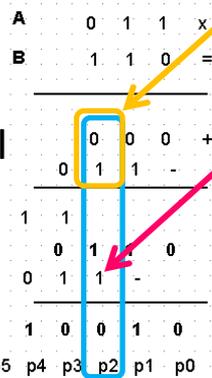
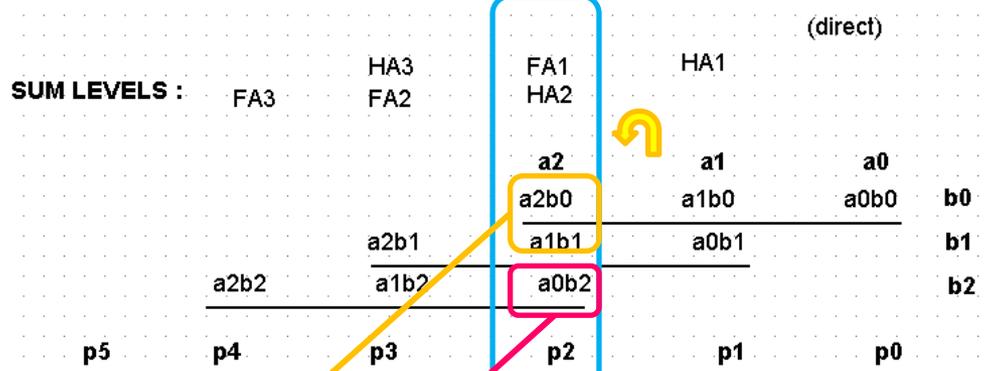
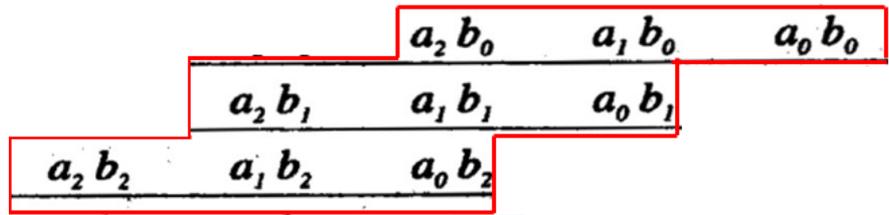
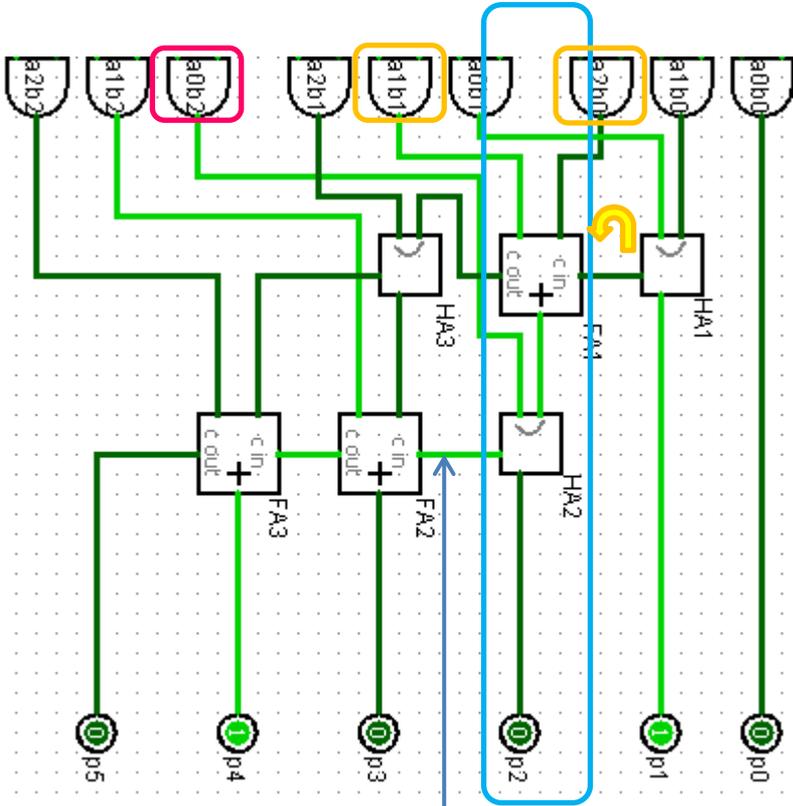


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



Dobbiamo gestire possibile riporto e somma di 2 valori ( $a_2b_0$  e  $a_1b_1$ ) → **FA1**

Inoltre dobbiamo gestire ulteriore valore ( $a_0b_2$ ) e il **risultato** della somma precedente → **HA2** (2 ingressi)

Notare che qui un **riporto** c'è ma è al livello II ... viene propagato da **HA2**

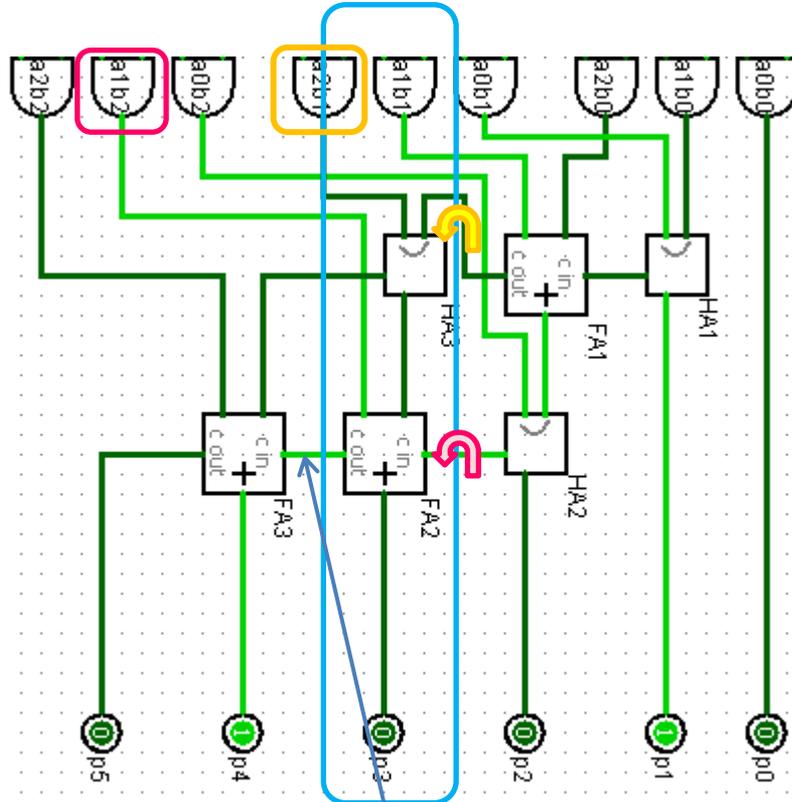


# Architetture degli Elaboratori e delle Reti I

# 5

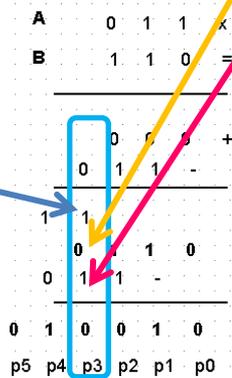
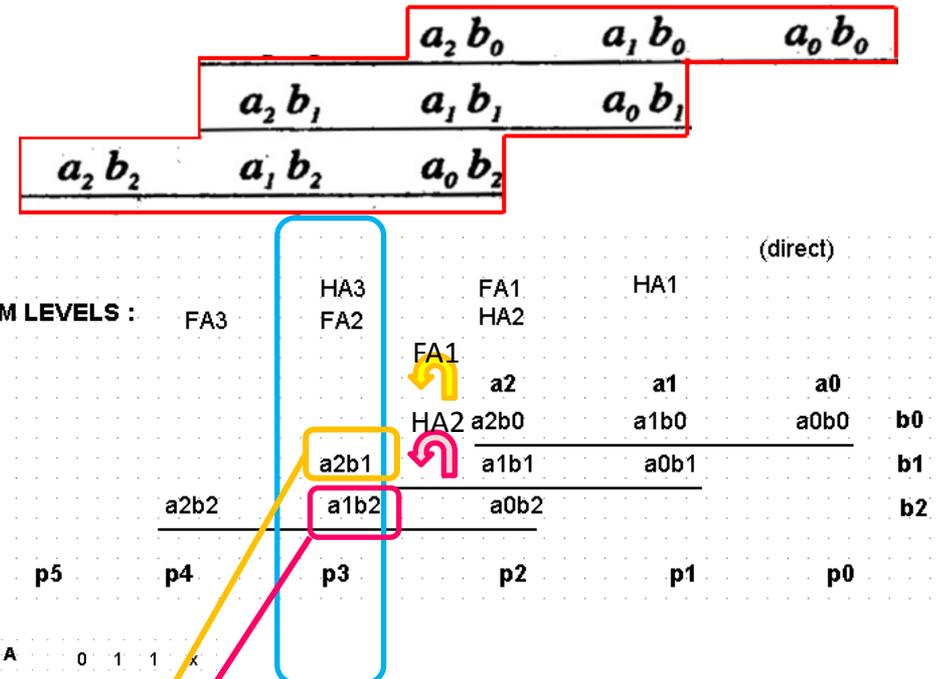
Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



Riporto di HA2 ...

**NB:** Anche qui ho **riporto**. E anche qui viene Propagato a livello II (da FA2)



Dobbiamo gestire possibile riporto (da FA1), somma di **2 valori** (a2b1 e a1b2) e possibile **riporto** da HA2. Un FA non basta! Sommiamo a2b1 e riporto FA1 usando un HA (HA3)

Sommiamo Il risultato di livello I , a1b2 e il riporto di HA2 usando un FA (FA2)

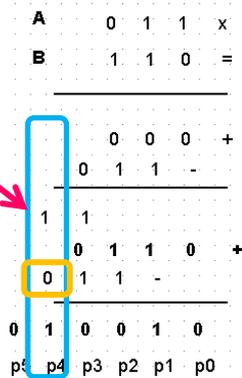
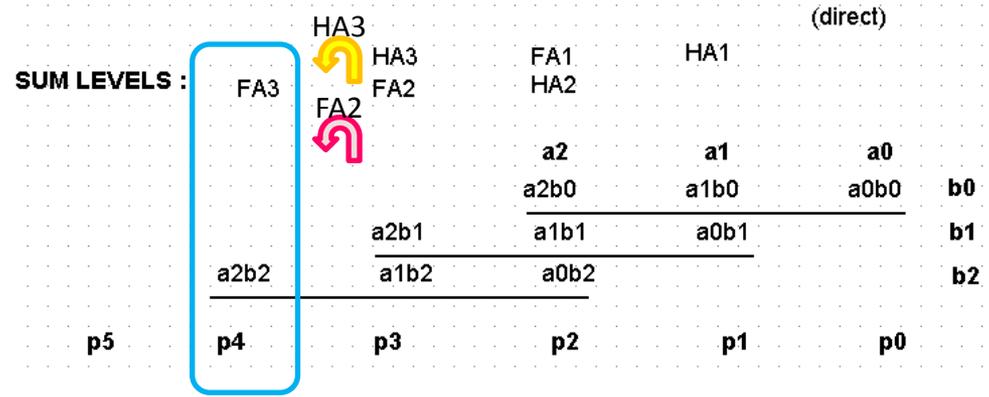
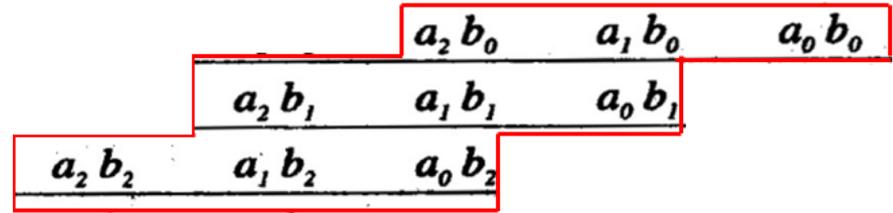
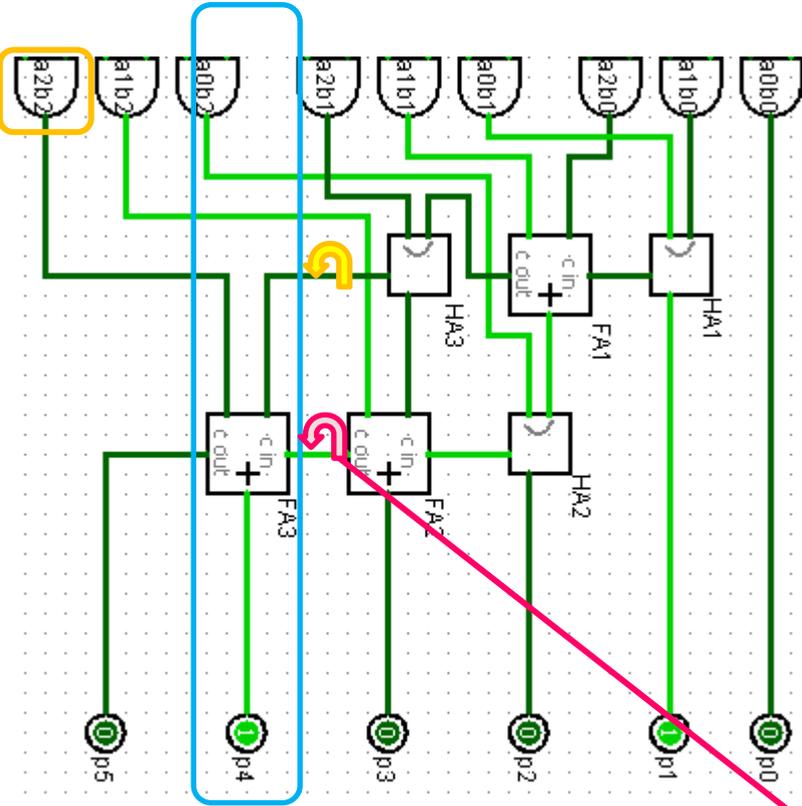


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



Dobbiamo gestire possibile riporto (da HA3 level I), somma di **1 valore** (a2b2) e possibile **riporto** da Level II FA2. Un FA basta!

Sommiamo **a2b1** e **riporto HA3** e, come r\_in in sommiamo riporto FA2.

**NB:** Il **riporto** di questo FA3 fornisce l'ultimo bit del prodotto risultante.

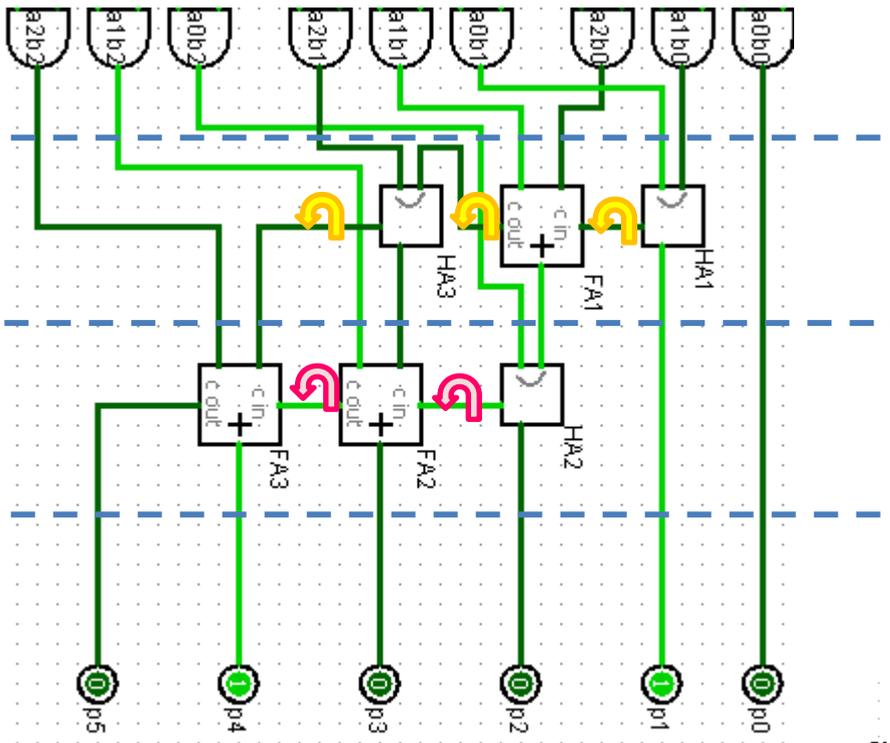


# Architetture degli Elaboratori e delle Reti I

# 5

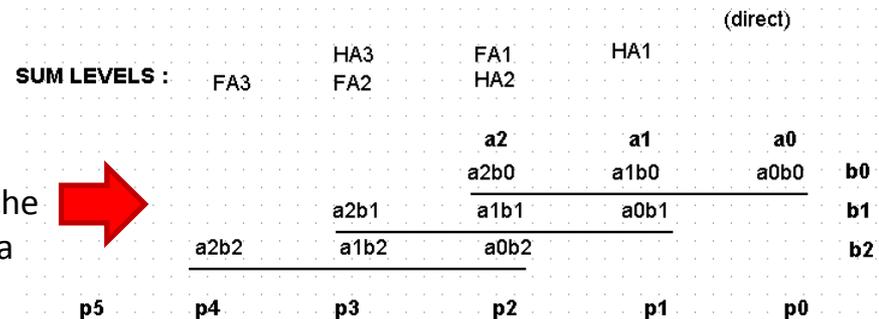
Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



## OSSERVAZIONI :

- Prodotto di due numeri di N bit genera risultato composto da 2N bit
- I riporti si propagano in modo coerente tra i livelli (level I / level II)
- La scelta per l'utilizzo di HA o FA dipende da quanti elementi (addendi / riporti) devo considerare in un dato step del calcolo (per step intendo calcolo delle cifre del prodotto risultante: p0, p1, ..., p2N-1)



Per capire la logica del moltiplicatore è necessario considerare ad ogni step la **geometria** delle operazioni che stiamo svolgendo : quanti elementi considerare → scelta HA/FA.



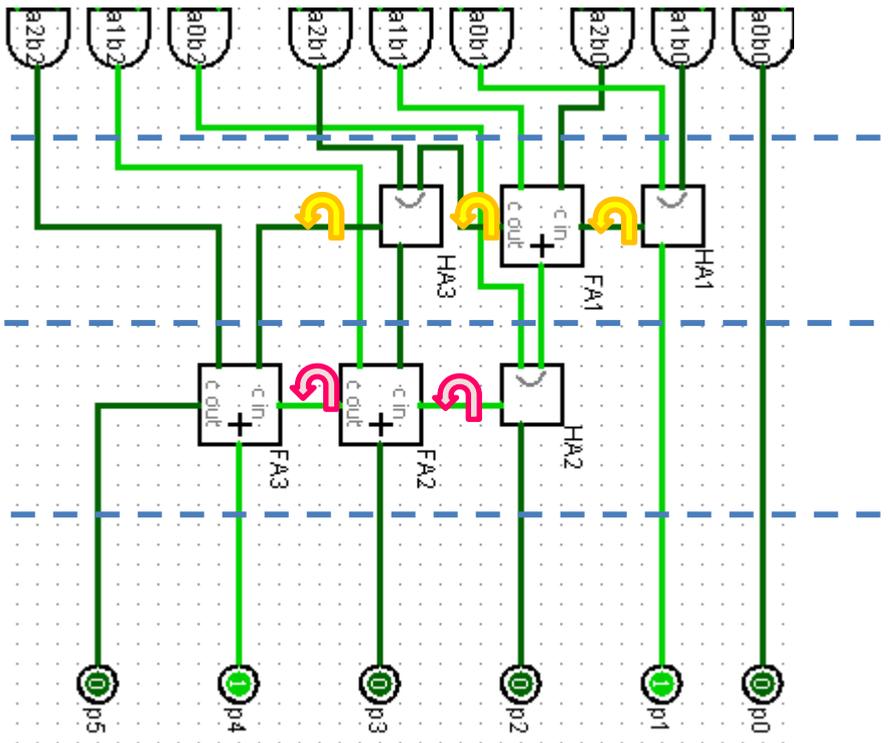


# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Moltiplicazione binaria:** Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



**Esercizio 5:** Estendere il moltiplicatore di numeri a 3 bit in modo da ottenere un moltiplicatore di numeri a 4 bit.

Seguire fedelmente lo schema circuitale disponibile in [slide 21](#) per la parte di somma dei prodotti parziali.

Scegliete voi se sviluppare il circuito in senso verticale o orizzontale.



# Architetture degli Elaboratori e delle Reti I

# 5

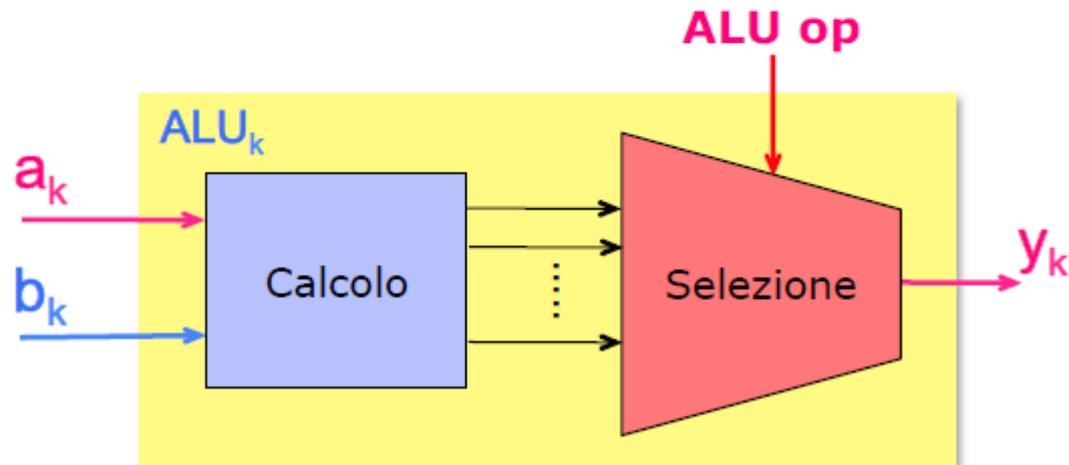
Laboratorio – linea 2 (G-Z)

## Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU

### Struttura **ALU** elementare:

- ❖ **Ingressi:** Operandi:  $a_k, b_k$   
Riporto in ingresso:  $r_{in}$   
Selettore operazione:  $ALUop$
- ❖ **Uscite:** Risultato:  $y_k$   
Riporto in uscita:  $r_{out}$





# Architetture degli Elaboratori e delle Reti I

5

Laboratorio – linea 2 (G-Z)

## Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU

Operazioni logiche: **AND/OR**

➤ Selezionabile

❖ Componenti:

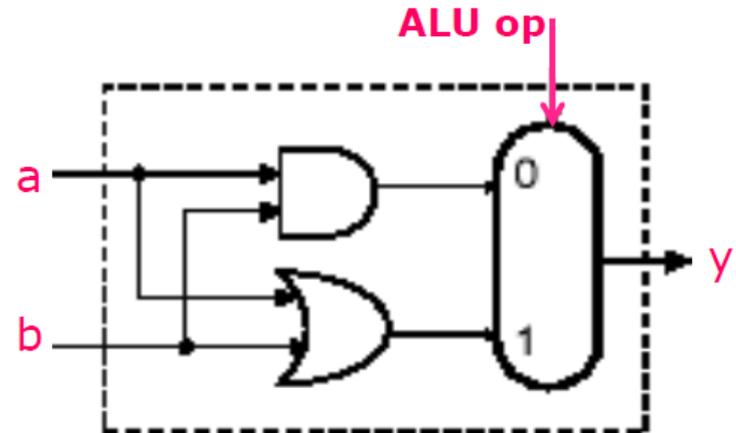
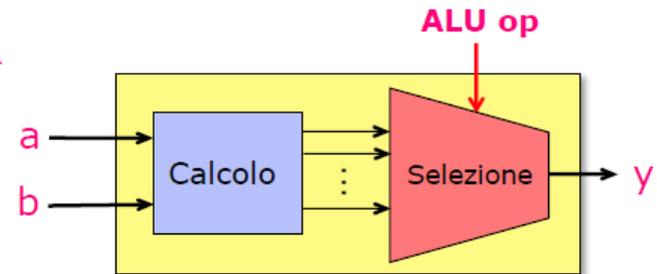
- 1 porta AND
- 1 porta OR
- 1 Multiplexer (MUX)

NB: ➔

Selezione:

$ALUop = 0 \rightarrow y = \mathbf{AND}(a,b)$

$ALUop = 1 \rightarrow y = \mathbf{OR}(a,b)$





# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

## Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU

Operazioni aritmetiche: **SOMMA**

❖ **FA: gestione dei riporti (in/out)**

- 3 ingressi:  **$a, b, r_{IN}$**
- 2 uscite:  **$s, r_{OUT}$**





# Architetture degli Elaboratori e delle Reti I

5

Laboratorio – linea 2 (G-Z)

Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU

Operazioni: **OR, AND, somma**

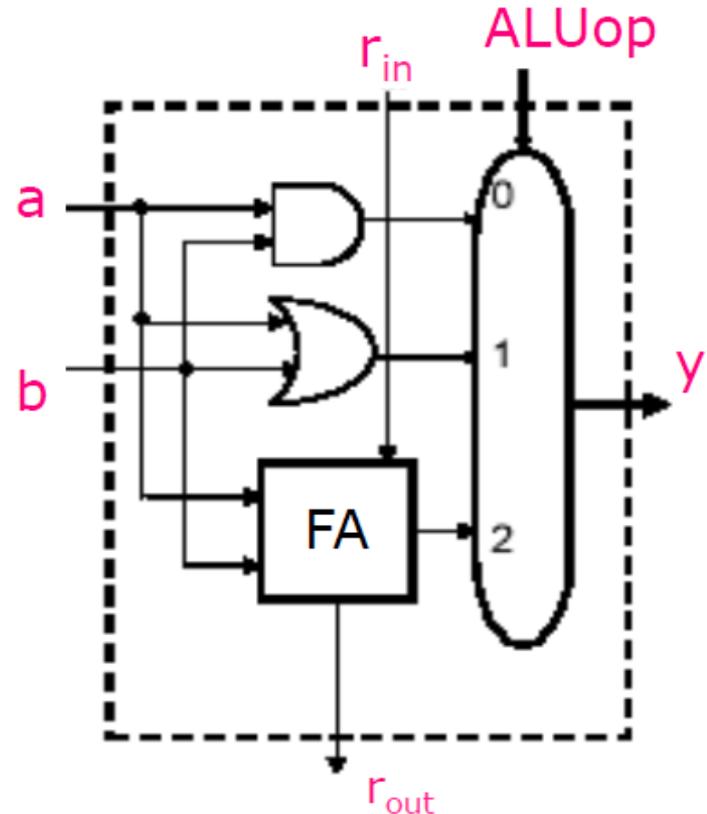
Selezione:

❖ **ALUop** a 2 bit

**00: AND:**  $s = a \text{ and } b$

**01: OR:**  $s = a \text{ or } b$

**10: +:**  $s/r_{out} = a + b + r_{in}$





# Architetture degli Elaboratori e delle Reti I

5

Laboratorio – linea 2 (G-Z)

## Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU: **Obiettivi laboratorio**

Costruzione di un componente 1bit ALU che consenta di effettuare le seguenti operazioni:

- AND
- OR
- XOR
- NOR
- ADD/SUB
- (e che fornisca anche la possibilità di considerare l'operando **less** )

Impostare l'aspetto del componente risultante in modo che sia facile realizzare Nbit ALU (cosa che vedremo negli esercizi successivi)



# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

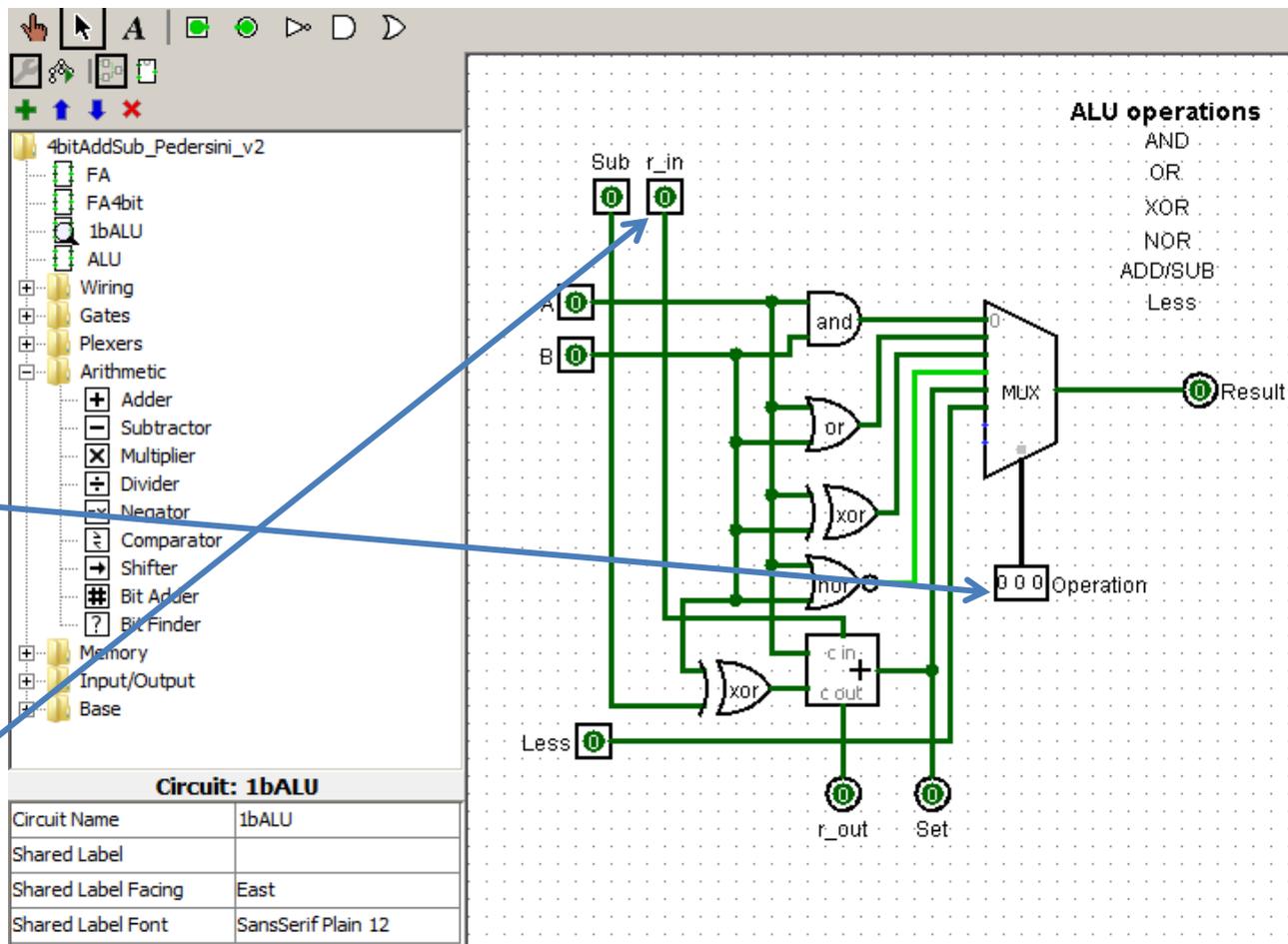
## Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU: **realizzazione in Logisim**

### Note:

Selettore operazione (del MUX) Deve poter selezionare tra 6 possibili input. servono almeno **3 bit**.

Se vogliamo eseguire la Sottrazione la porta XOR più vicina a FA fornisce **C1** ma È necessario impostare **r\_in** a **1**.



Provate a verificare il comportamento di 1bit ALU cambiando i valori in ingresso e il valore del selettore dell'operazione



# Architetture degli Elaboratori e delle Reti I

5

Laboratorio – linea 2 (G-Z)

## Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU: **realizzazione in Logisim**

### Aspetto componente:

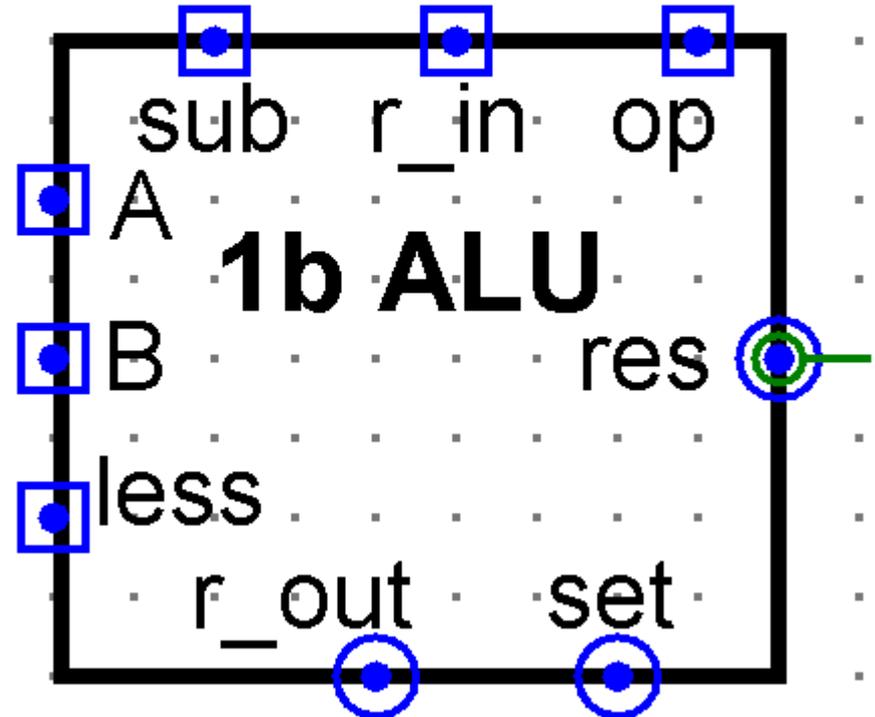
E' di fondamentale importanza che l'aspetto del componente sia funzionale al suo utilizzo.

E' quindi necessario:

Decidere come impostare le posizioni dei sottocomponenti per poter usare 1bit ALU in una Nbit ALU

Etichettare **OGNI** elemento

**NB:** Usiamo questa rappresentazione poiché vorremmo facilitare uno sviluppo verticale per la Nbit ALU ... (guardate posizione r\_in e r\_out)





# Architetture degli Elaboratori e delle Reti I

# 5

Laboratorio – linea 2 (G-Z)

**Arithmetic Logic Unit (ALU) : Struttura modulare ALU**

**Nbit ALU: realizzazione in Logisim (8bit ALU)**

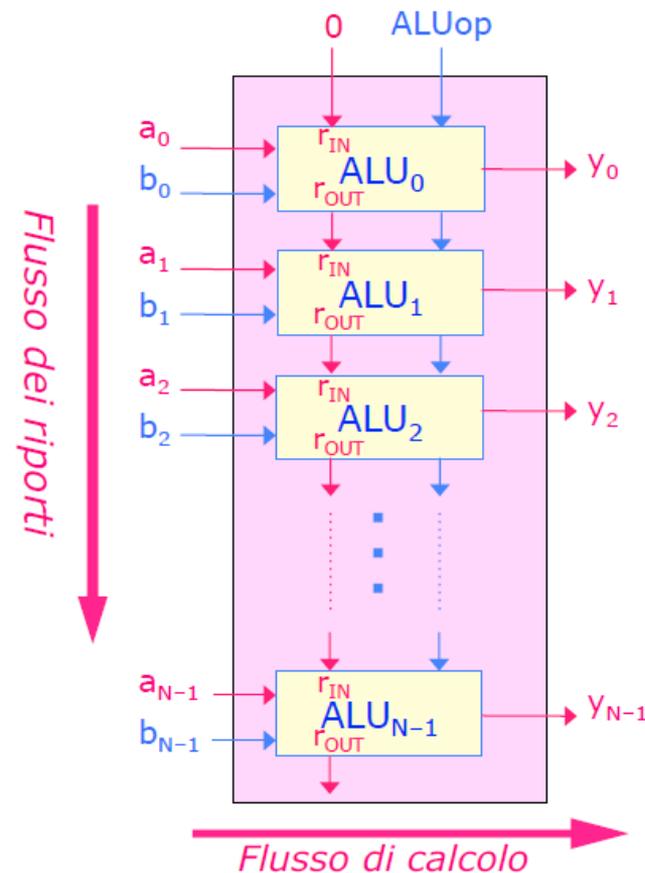
## **ALU a N-bit**

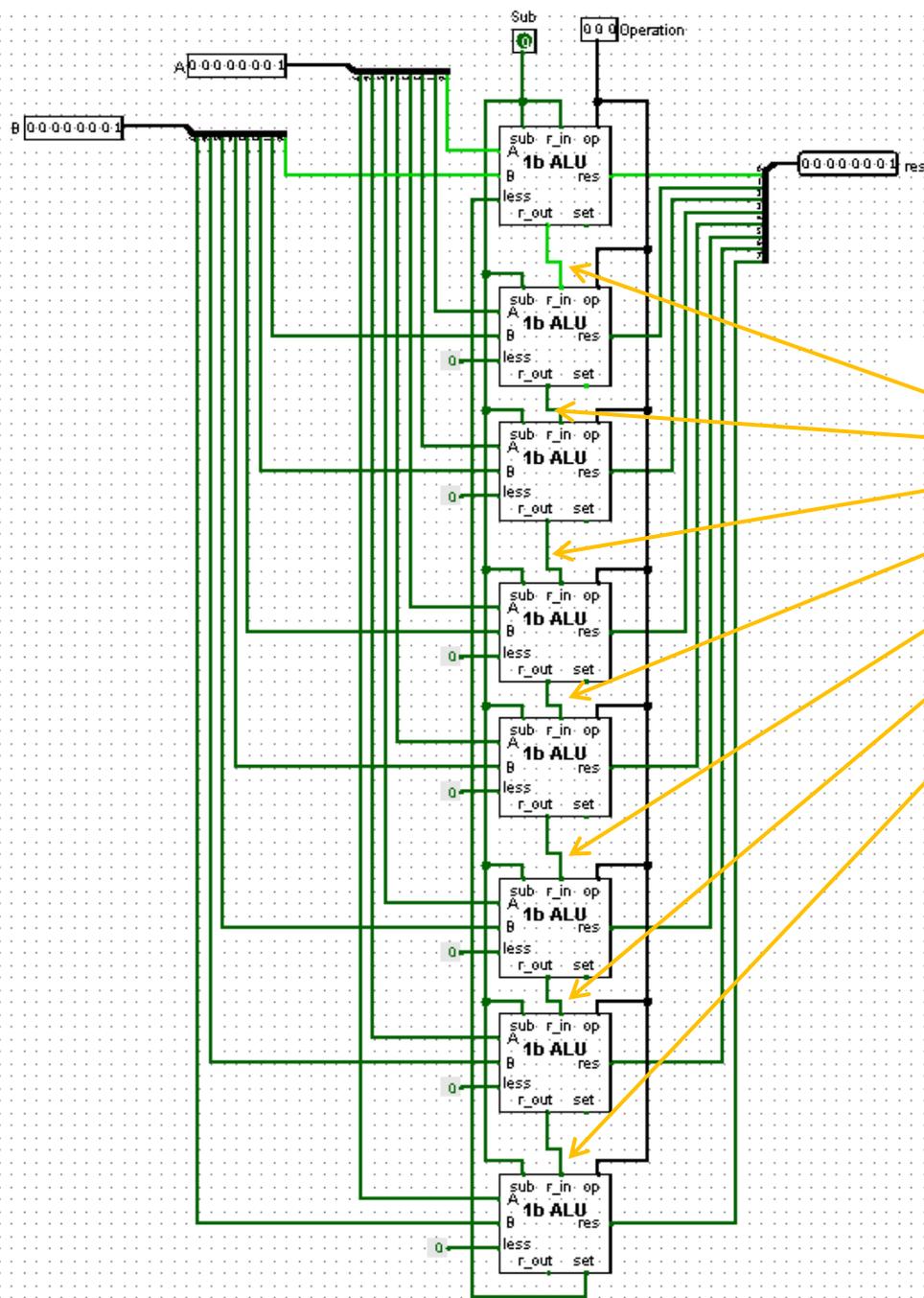
- ❖ Come collegare N ALU a 1 bit per ottenere una ALU a N bit?

## **ALU a N bit:**

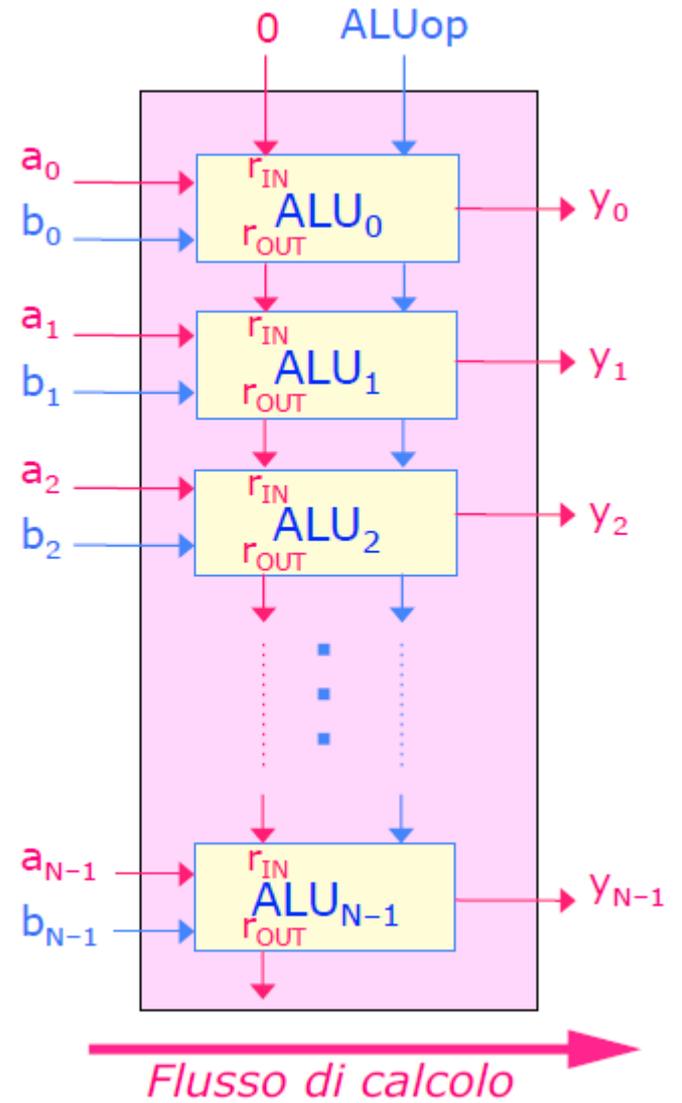
**N ALU in parallelo + propagazione dei riporti**

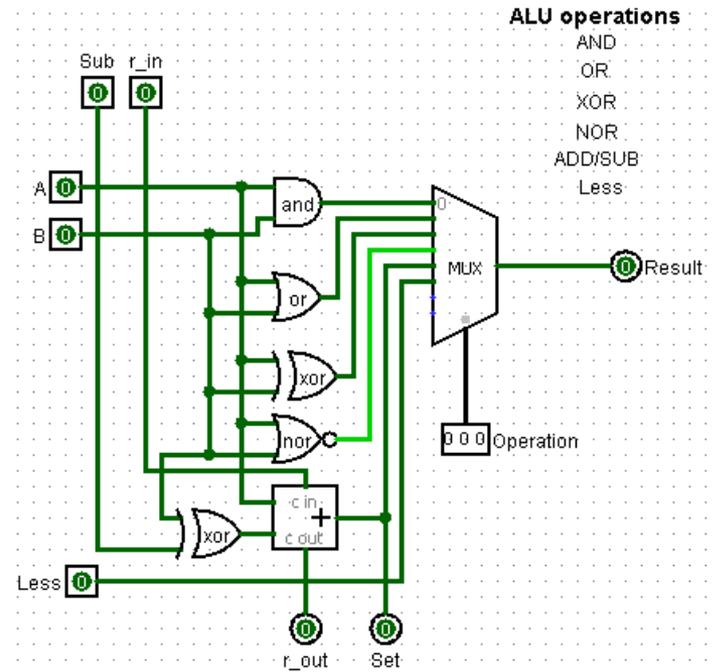
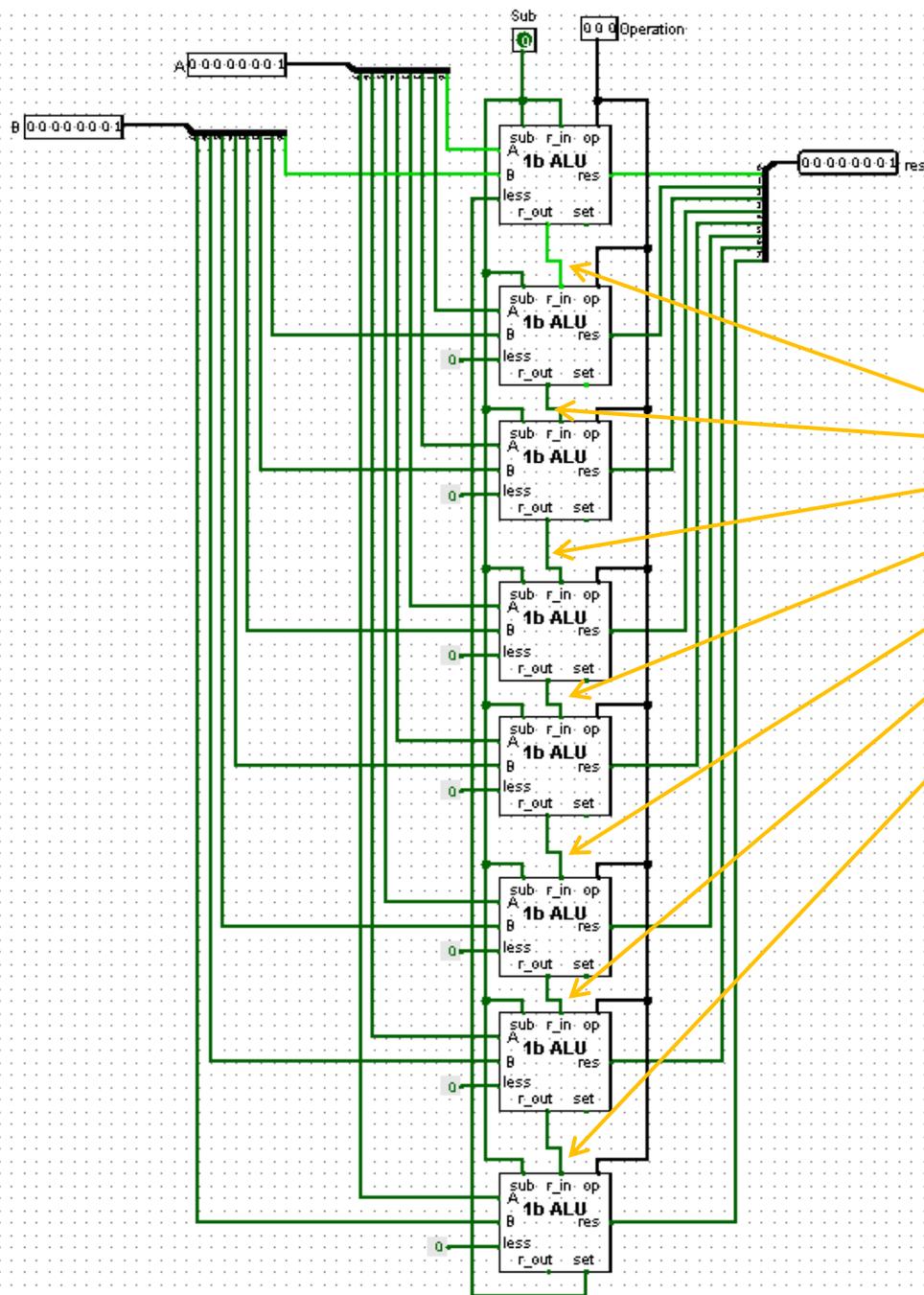
- ❖ Problema: il cammino critico si allunga → limite alla velocità di calcolo



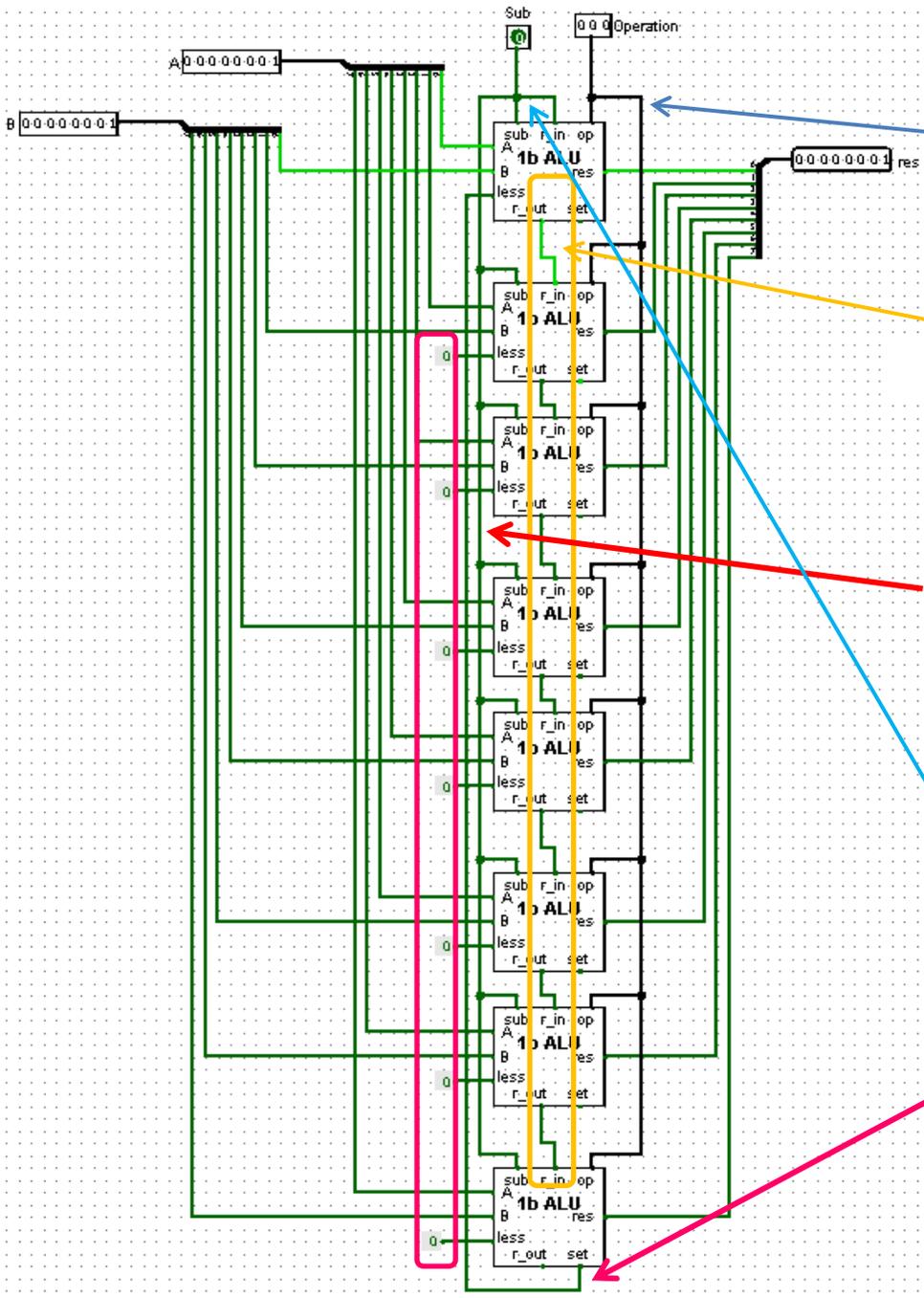


*Flusso dei riporti*





Flusso di calcolo  
(1bit ALU)



**Note collegamento tra 1bit ALU:**

Il valore di Operation (**3bit** input) va «comunicato» a tutte le 1bit ALU

Tutti i valori **r\_in** dalla seconda 1bit ALU in poi vengono ottenuto **da r\_out** della 1bit ALU precedente (questo determina il flusso dei riporti)

Il valore del bit dell'input **Sub** va «comunicato» in cascata a tutte le 1bit ALU

Se, quando l'operazione selezionata è **ADD/SUB**, vogliamo eseguire la sottrazione possiamo inviare il valore del bit **Sub** in **r\_in** della **prima** 1bit ALU

Il valore dell'uscita **set** dell'**ultima** 1bit ALU va «comunicato» all'ingresso **less** della prima 1bit ALU