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-2017-2018.html)

Moltiplicazione e ALU

- Richiami
- Moltiplicazione
- Arithmetic Logic Unit (ALU)

Richiami

Riferimenti: slide teoria

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

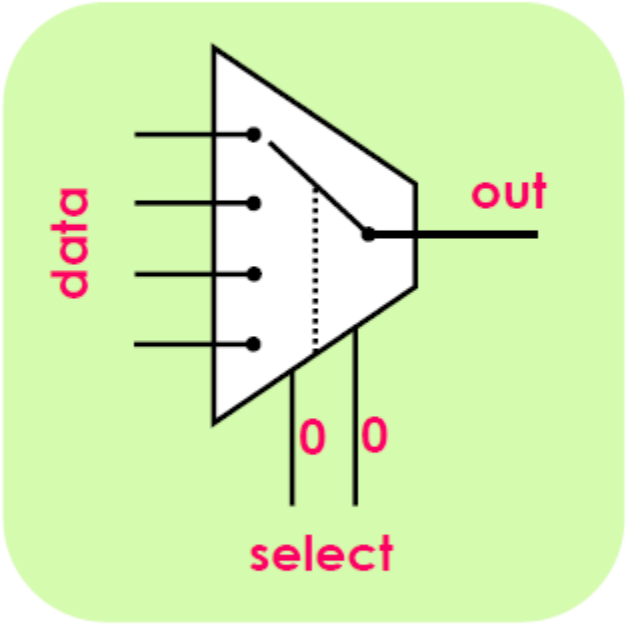
Utilizzeremo 1bit ALU per costruire una 8bit ALU

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?

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

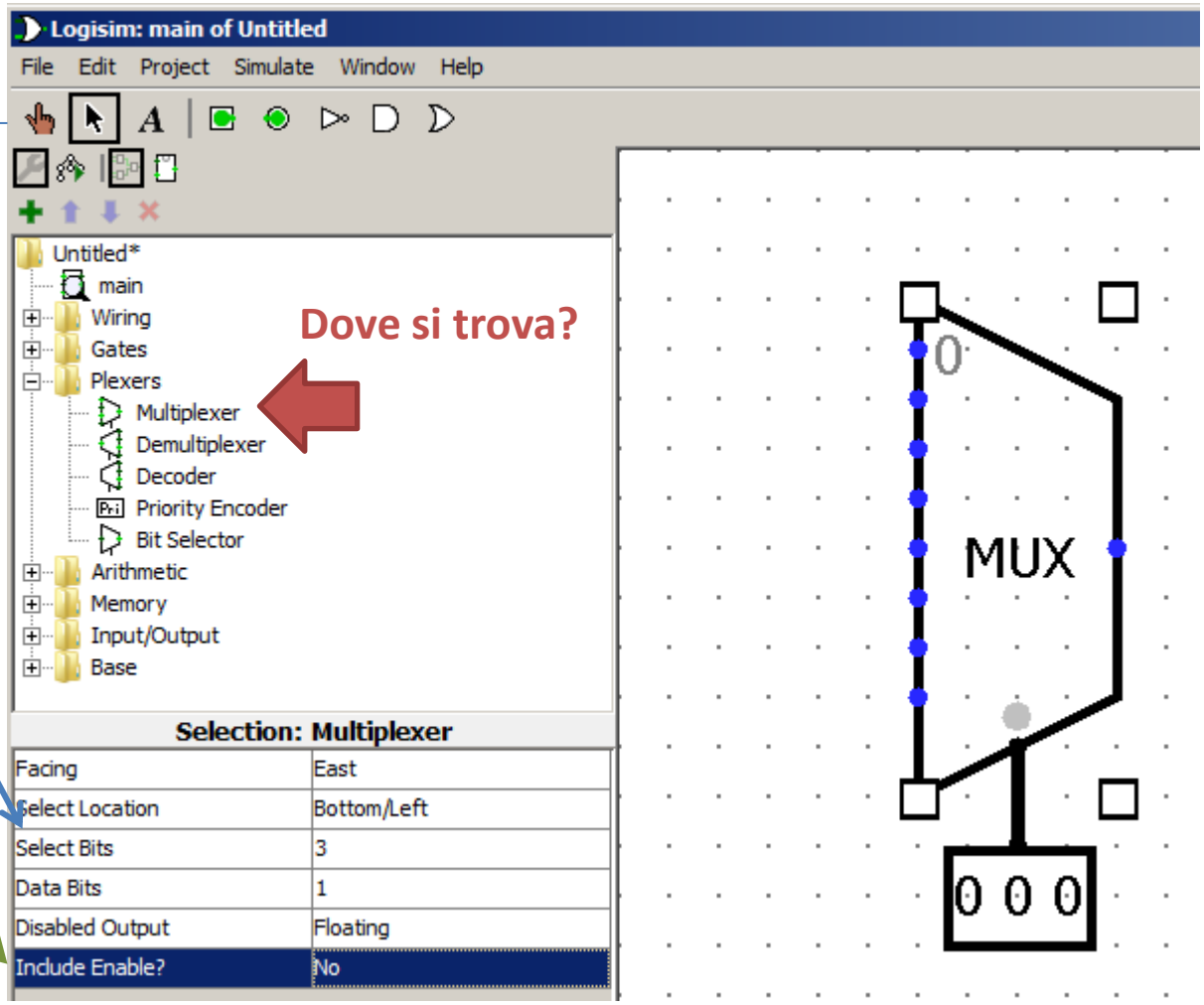
Linee di input: $n = 4$

Linee di controllo: $k = ceil(\ log_2 4\ ) = 2$

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).



Circuiti aritmetici

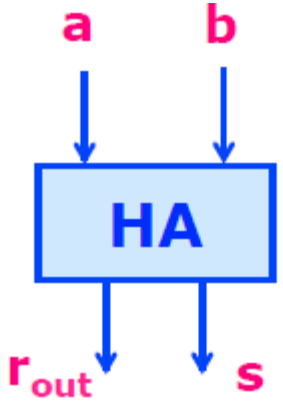
Riferimenti: slide teoria

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

Partiamo con la somma …

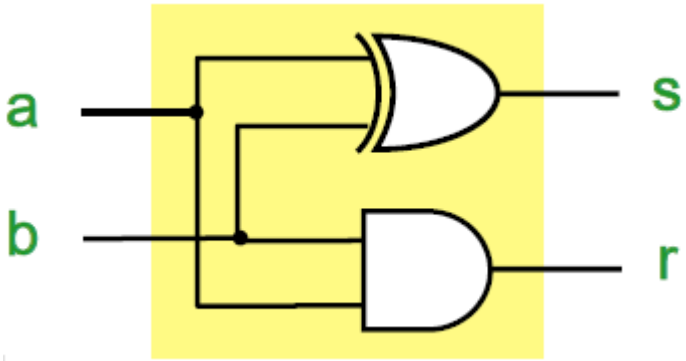
HALF ADDER :

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



Note:

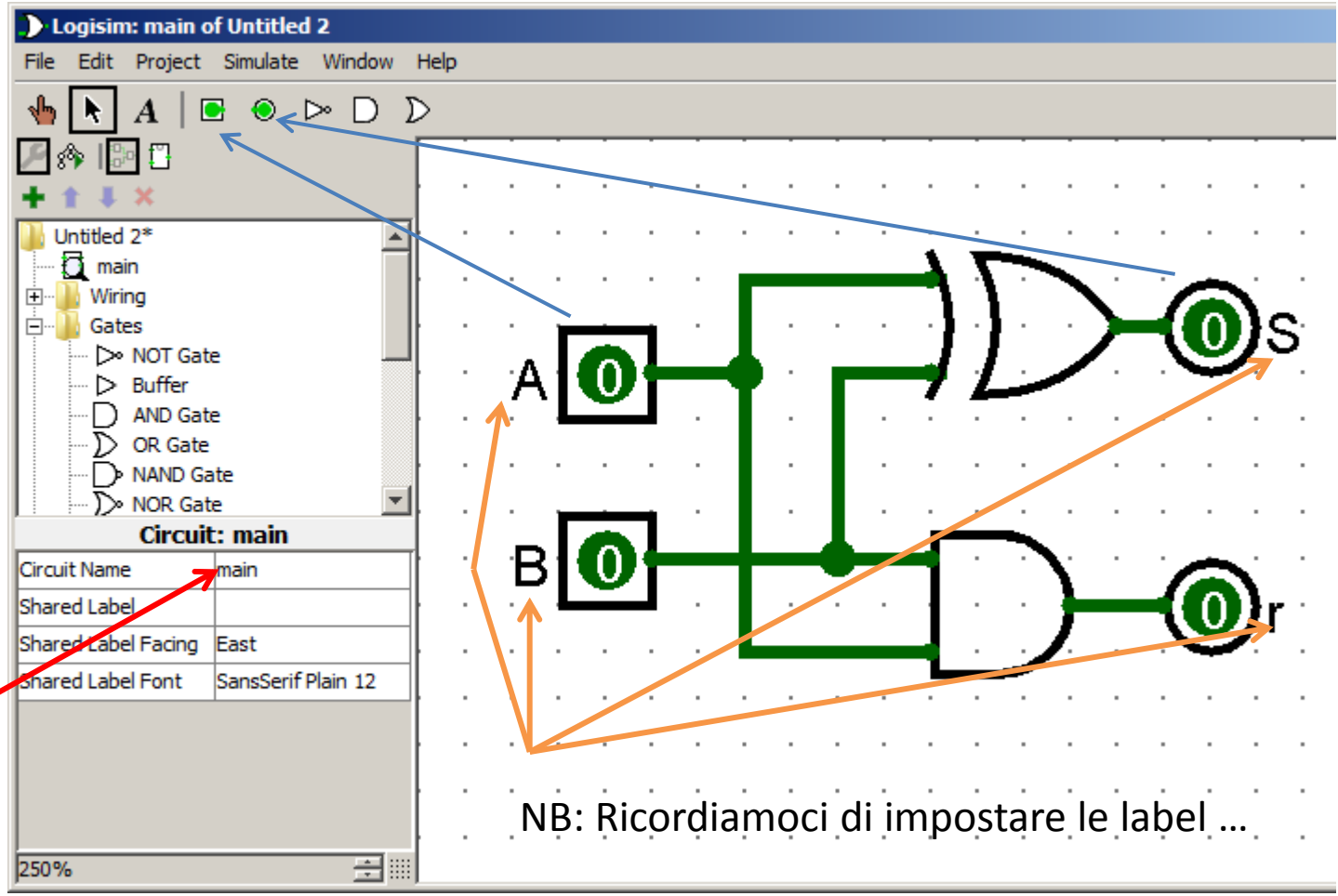
. Non ha riporto in ingresso.



Obiettivo LAB05_1 : Realizzazione in Logisim e personalizzazione aspetto visivo del componente realizzato.

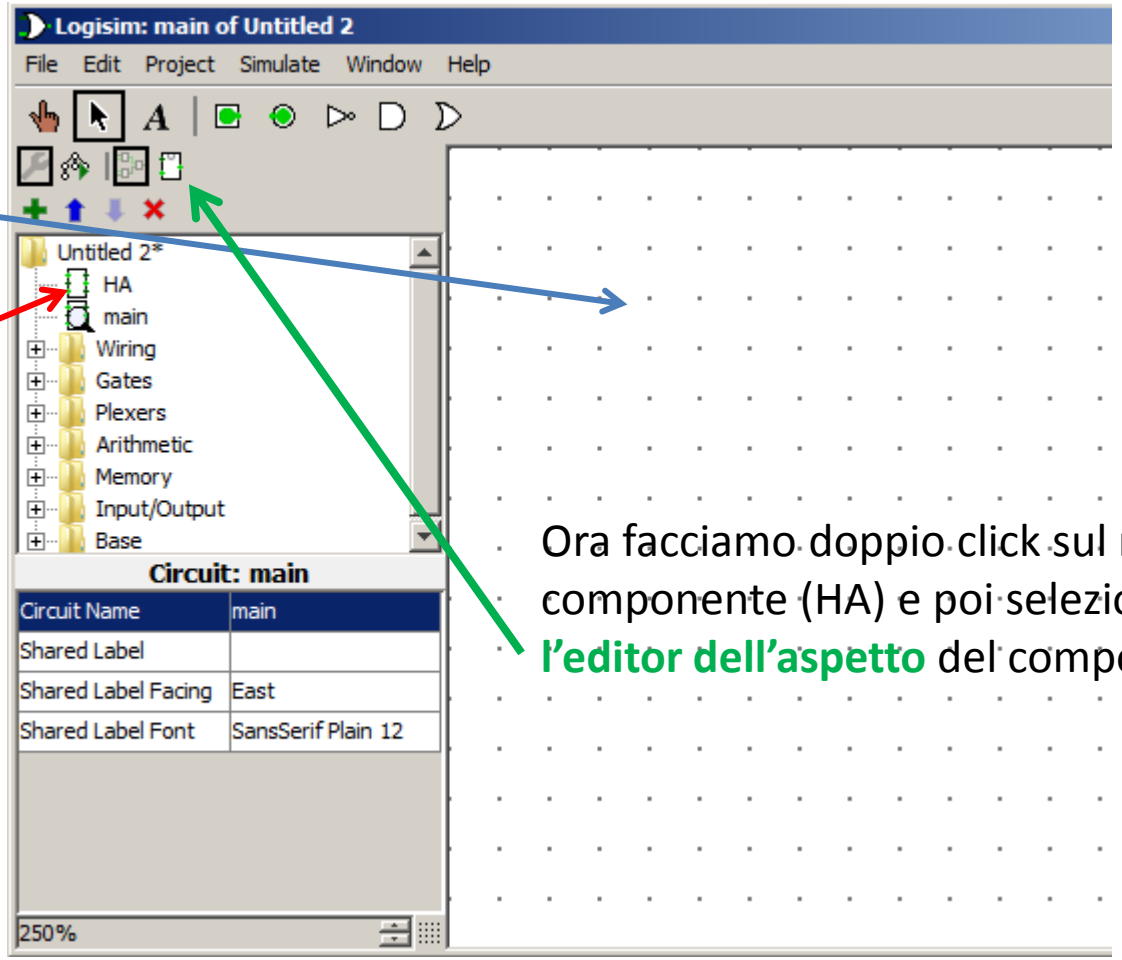
Step necessari :

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)



Cosa otteniamo:

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

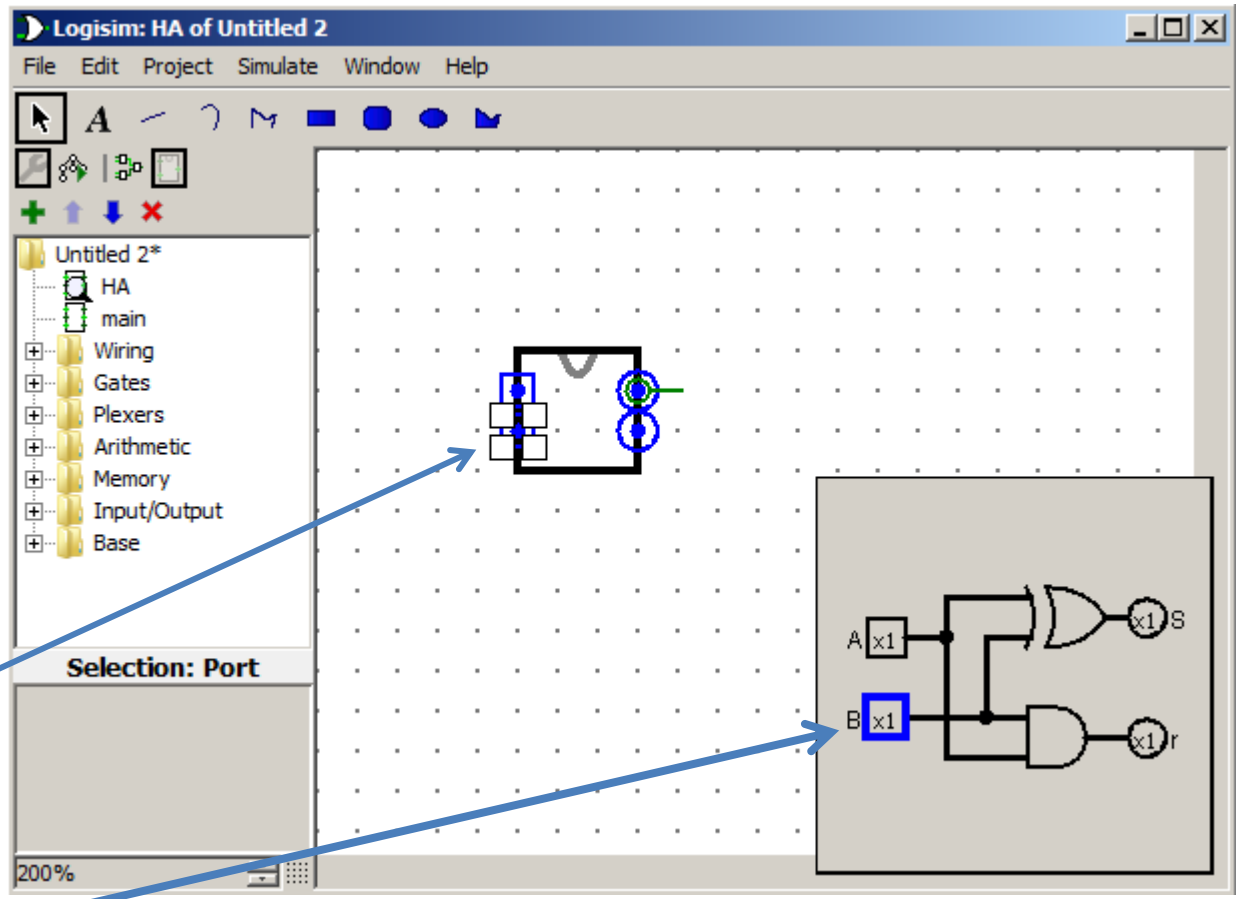


Ora facciamo doppio click sul nuovo componente (HA) e poi selezioniamo **l'editor dell'aspetto** del componente.

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.

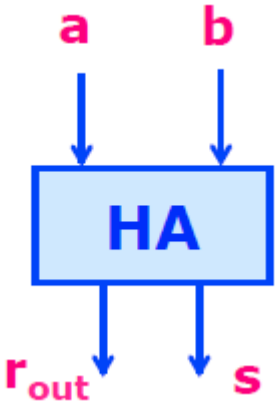


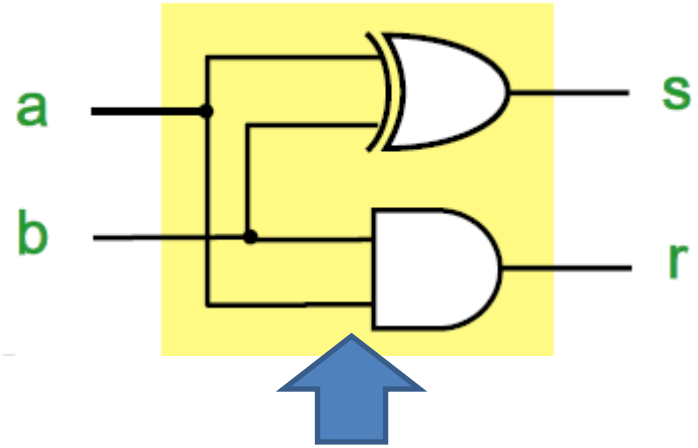
E' un'ottima idea quella di inserire delle label all'interno della rappresentazione del componente (per riconoscere in modo più facile ingressi e uscite)

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

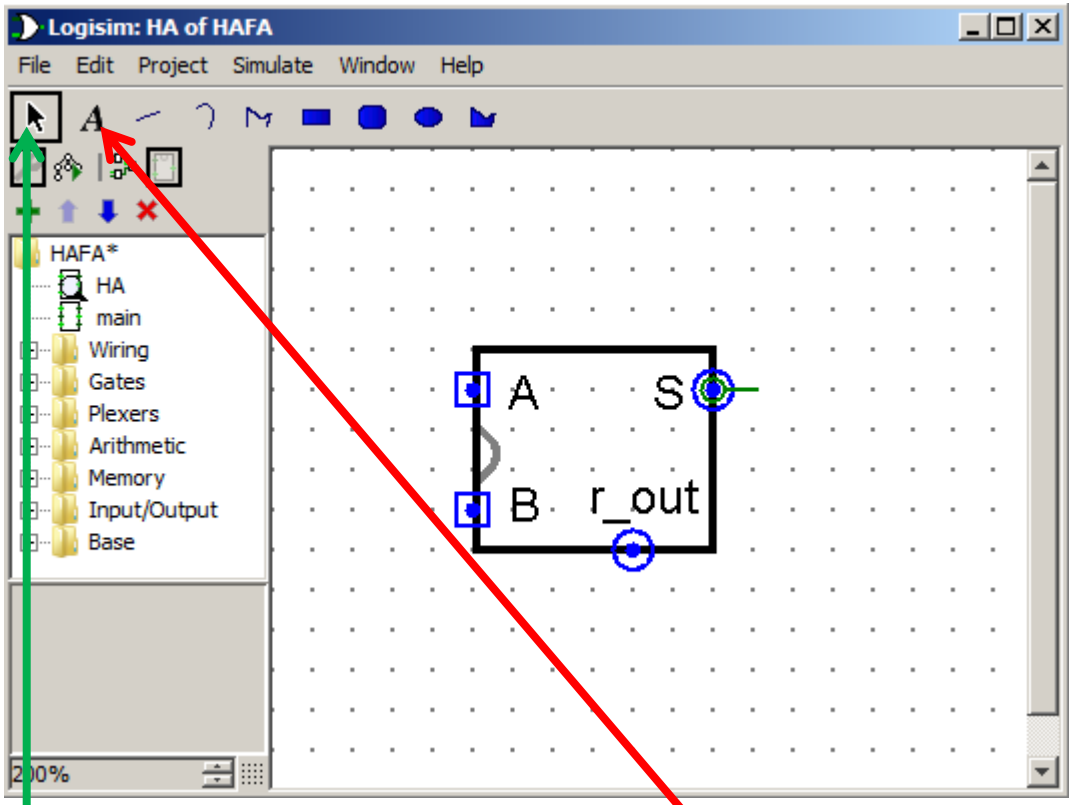
1) Seguite gli esempi presenti nelle slide di teoria
2) Progettate l'aspetto in modo che questo <u>renda semplice</u> l'utilizzo in circuiti di cui avete già lo schema

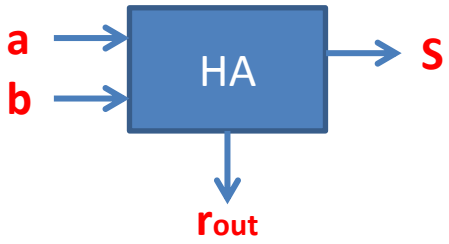
Opzioni (vedere slide teoria) :





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



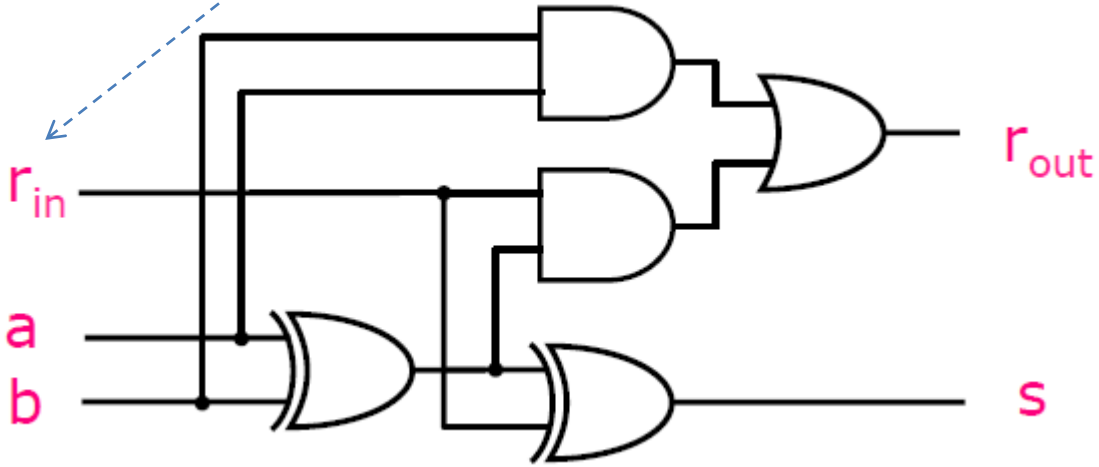


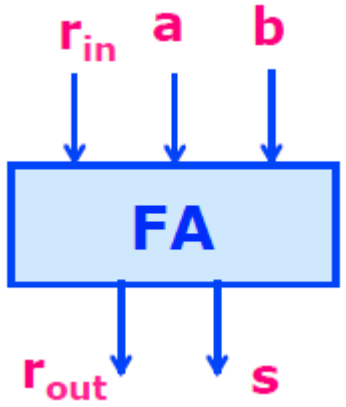
NB: Per inserire le etichette usare **l'apposito strumento**
Inserire il testo e confermare (invio). **Selezionare** e riposizionare dove necessario.

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:

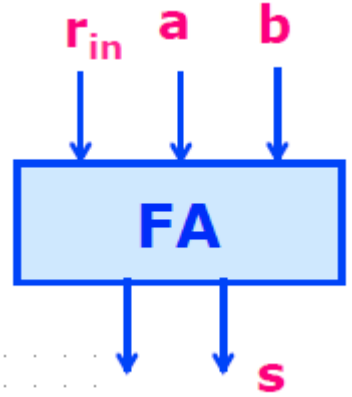
. **C'è** riporto in ingresso.

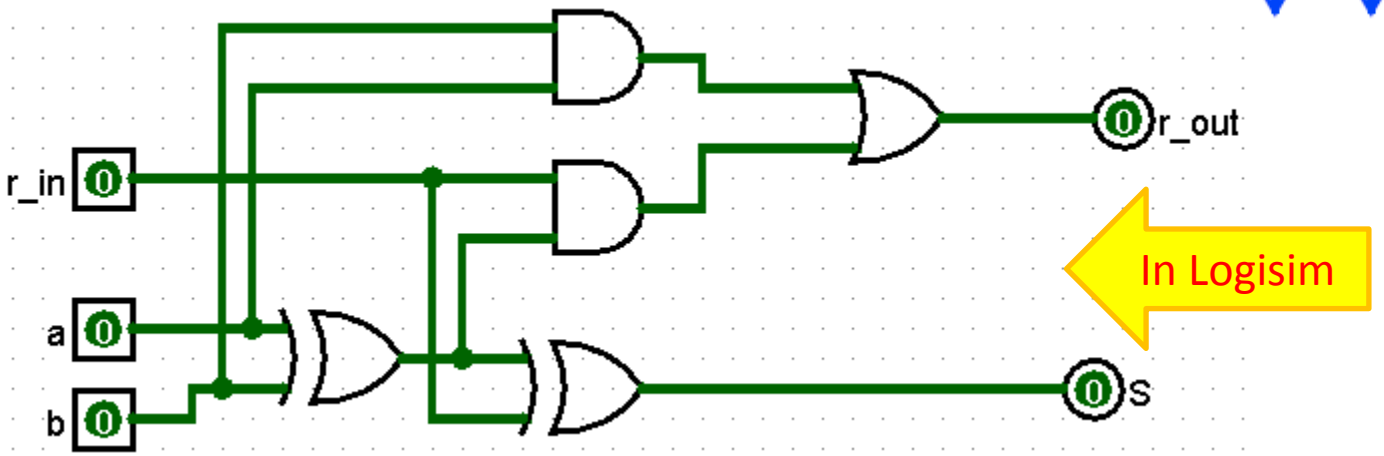
Obiettivo LAB05_2 : Realizzazione in Logisim e personalizzazione aspetto visivo del componente realizzato.

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}**





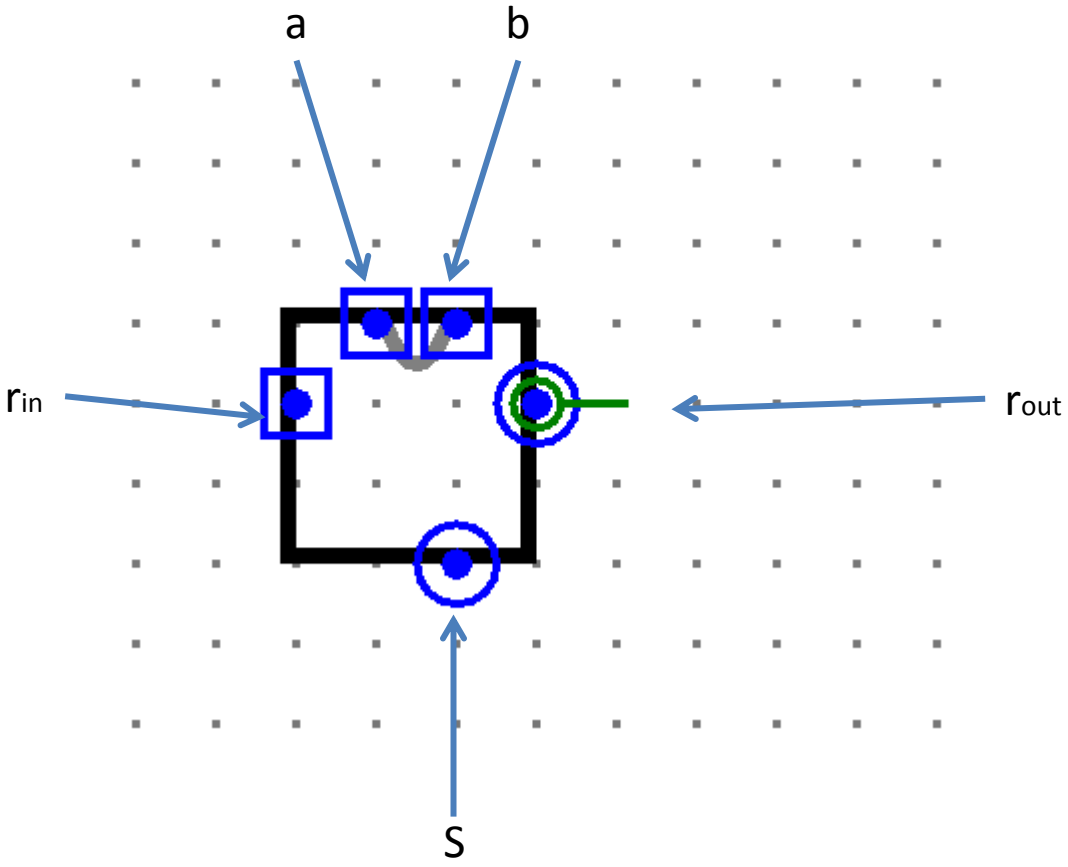
NB: Valgono le stesse regole di prima (impostare per ogni porta in modo corretto il numero di ingressi e settare Gate size a narrow). Impostare sempre il valore delle etichette. L'aspetto del componente come lo impostiamo?

Architetture degli Elaboratori e delle Reti I

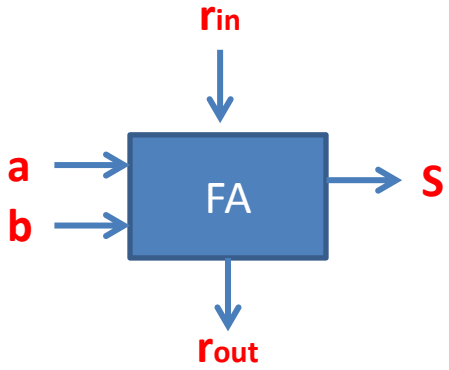
Laboratorio – linea 2 (G-Z)

5

Esercizio 1: Realizzare questa rappresentazione del componente

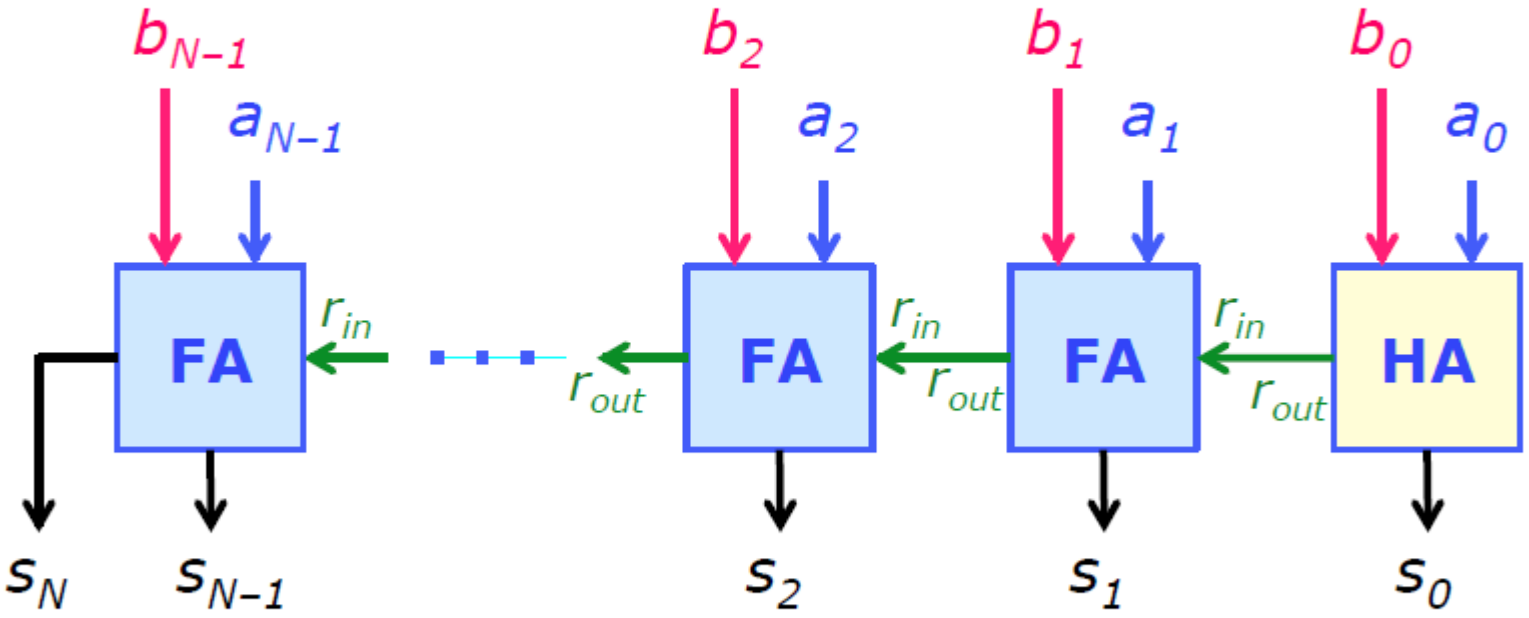


Esercizio 2: Realizzare questa rappresentazione del componente



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

- Sommatore a propagazione di riporto
  - **IN**: **2** parole di **N** bit
  - **OUT**: somma di **N+1** bit



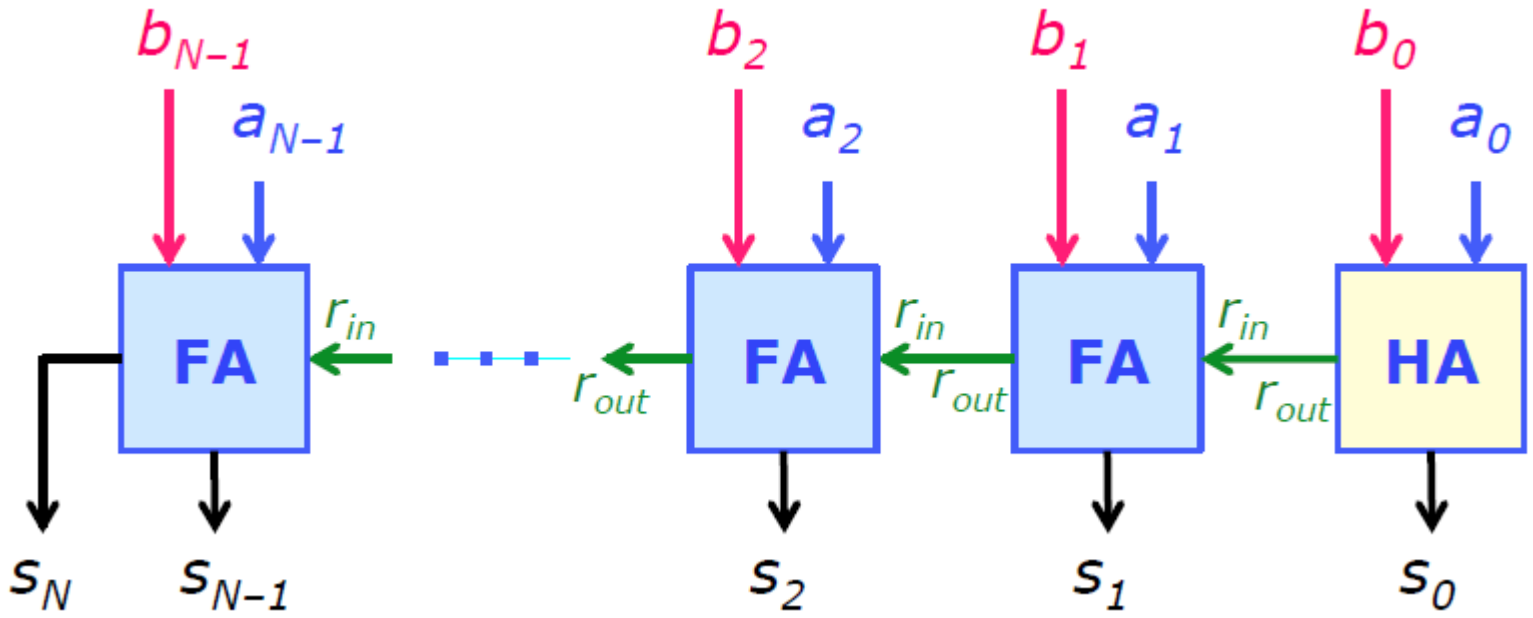
Cammino critico? (HA=1 ; FA=3)

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

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



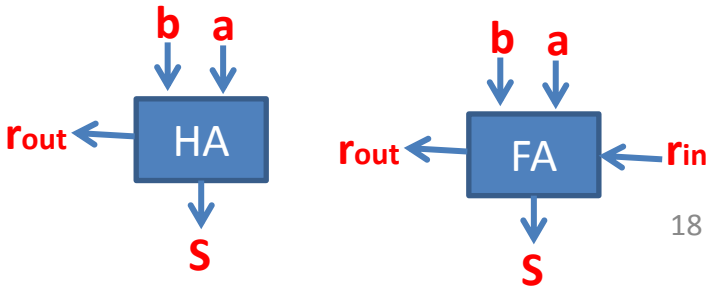
Sommatore a n bit a propagazione di riporto:

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.

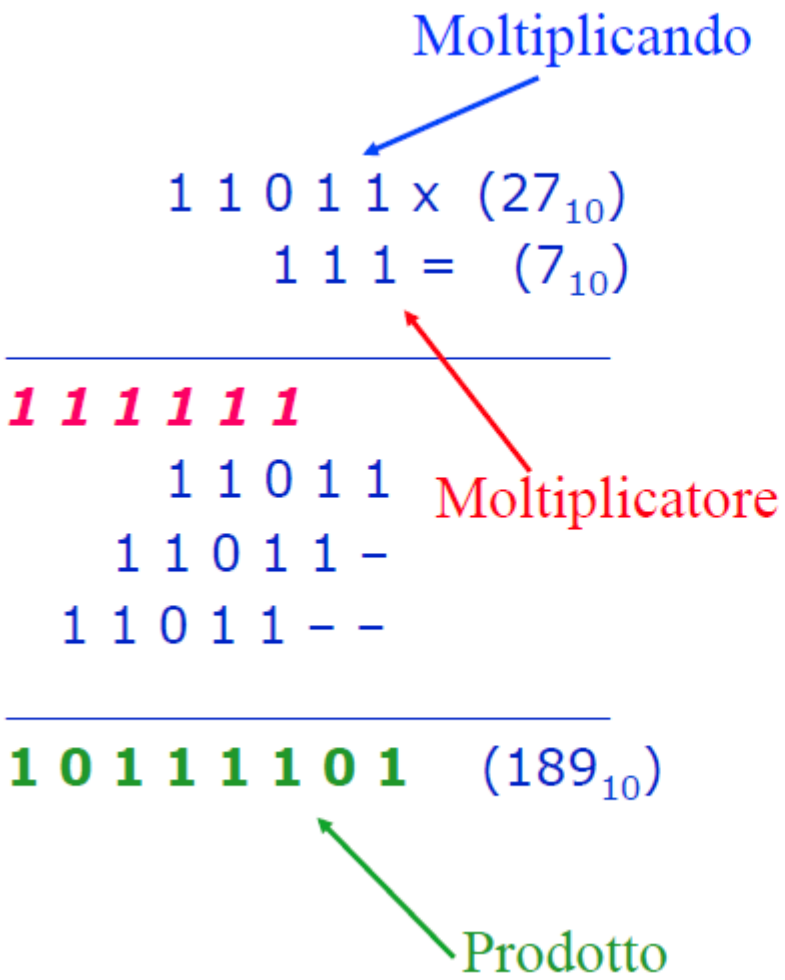
Esercizio 4: Realizzare in Logisim il sommatore a propagazione di riporto

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:

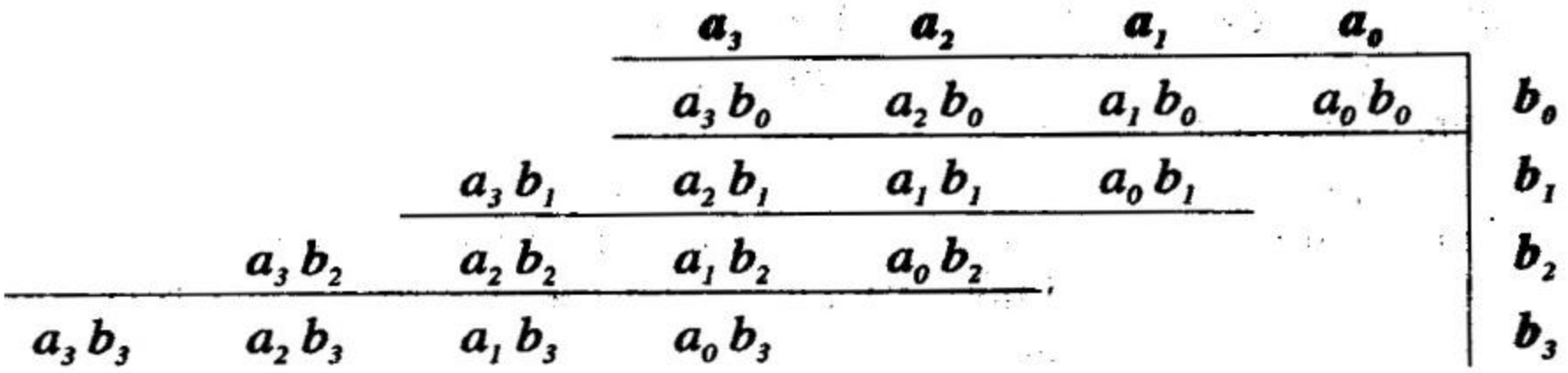


Moltiplicazione binaria :



- 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

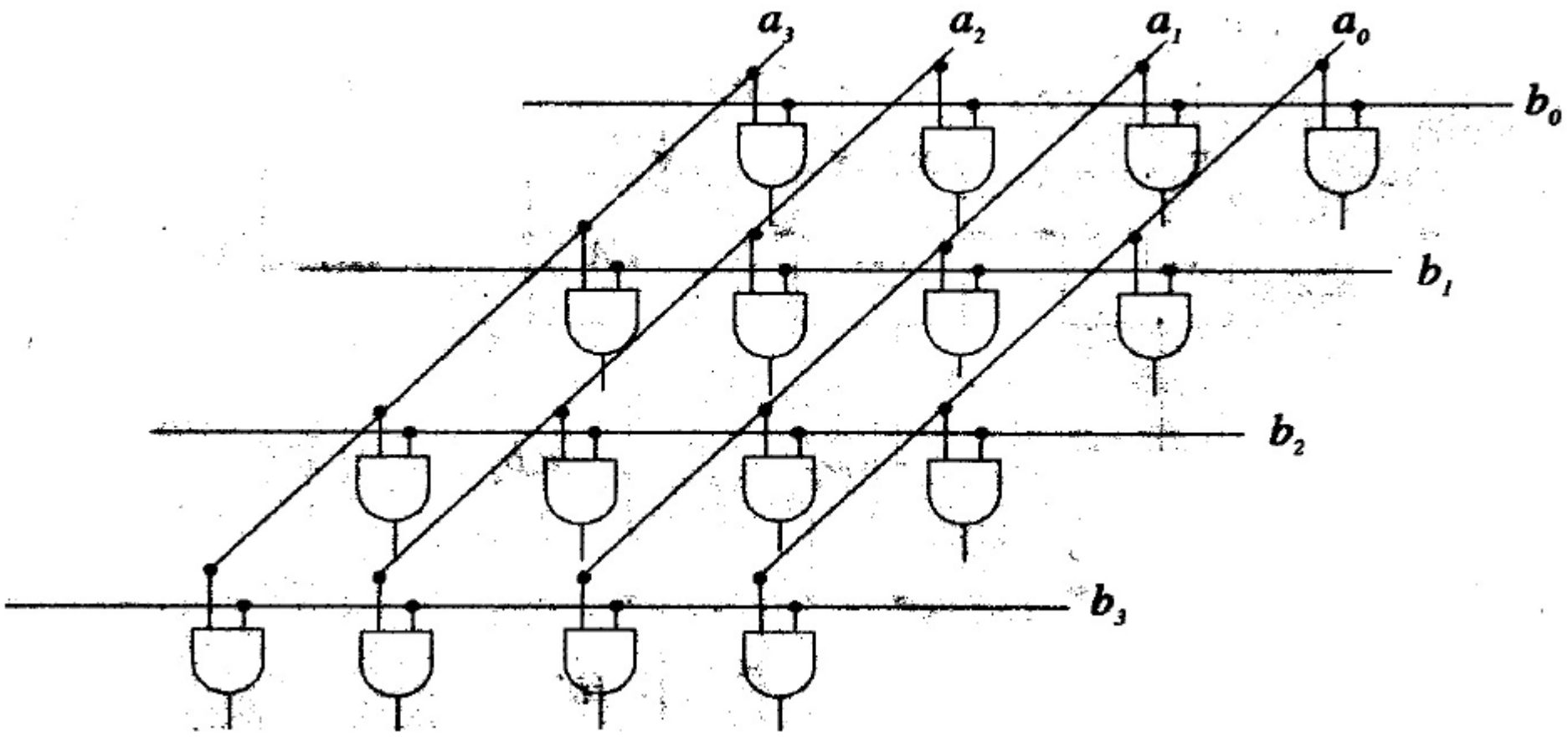
Moltiplicazione binaria : la matrice dei prodotti parziali



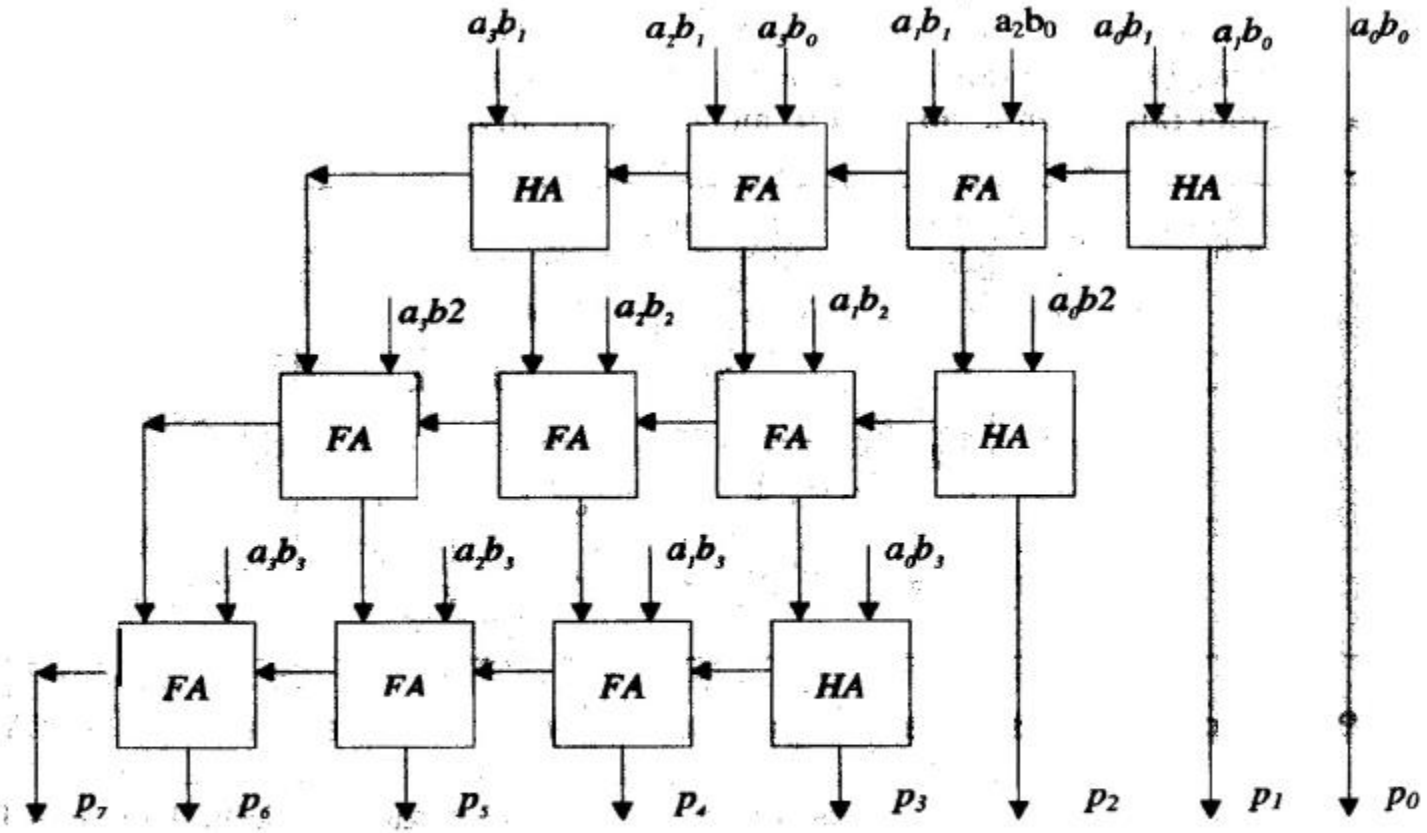
In binario i prodotti parziali sono degli **AND**

Nel prossimo esercizio costruiremo in Logisim un moltiplicatore per **numeri a 3 bit**. Seguiremo **esattamente** lo stesso schema curcuitale 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).

Moltiplicazione binaria : schema generale calcolo prodotti parziali



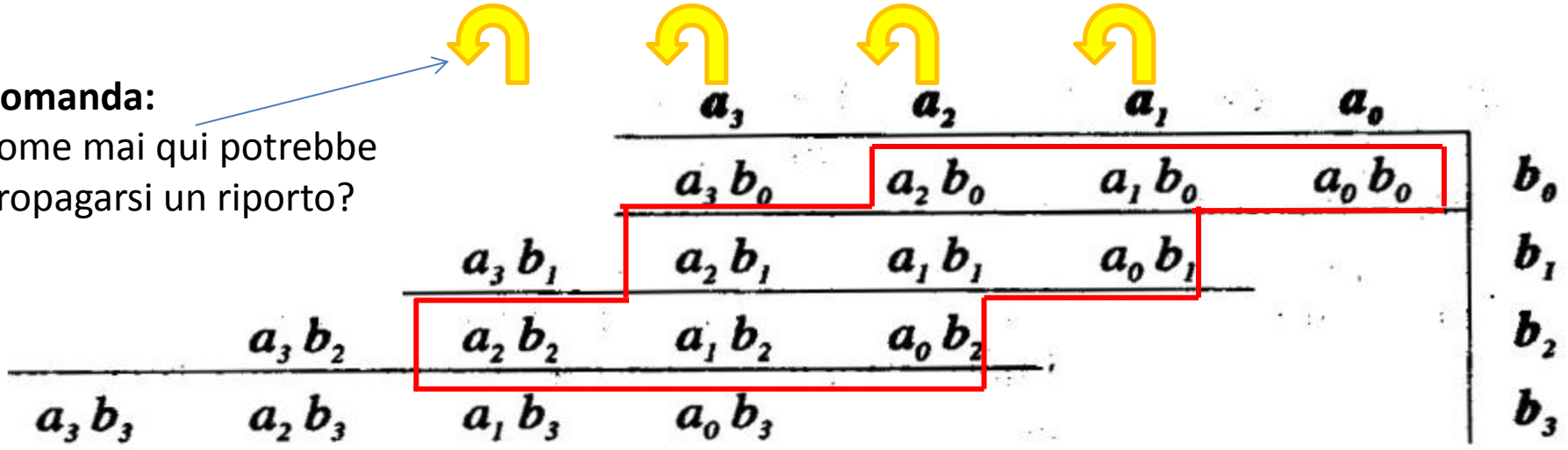
Moltiplicazione binaria : schema generale somma prodotti parziali



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

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?

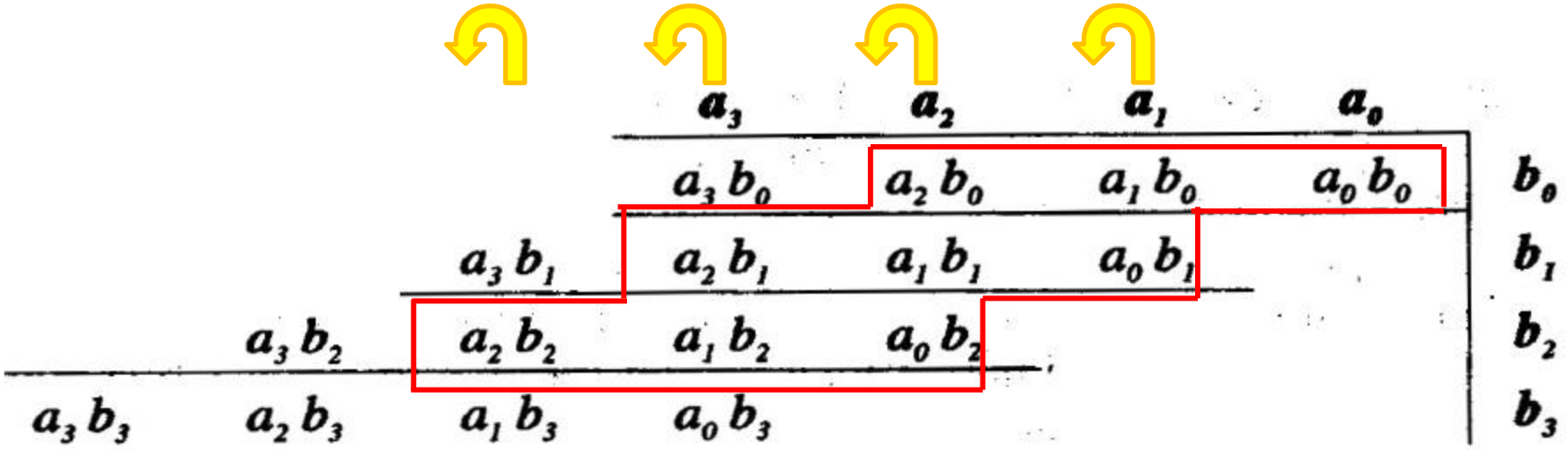


Le freccie gialle indicano punti in cui potrebbero propagarsi dei riporti

Note:

p0 deriva direttamente dal prodotto a0b0 . p1 è la somma di a1b0 e di a0b1 . Non c'è necessità di considerare riporto in ingresso (ma potrebbe esserci riporto in uscita). Quindi è corretto utilizzare un Half Adder.

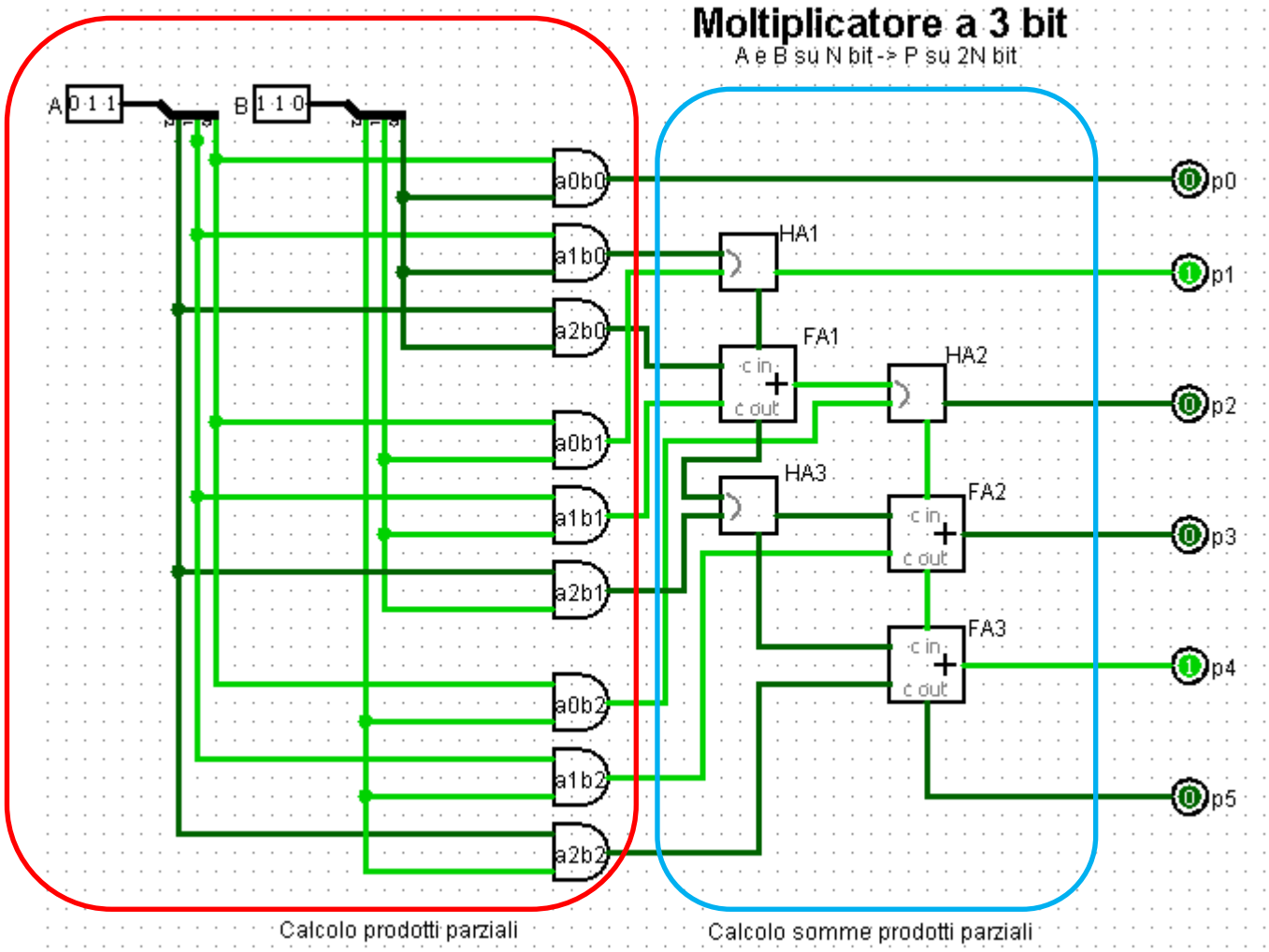
Moltiplicazione binaria : torniamo alla matrice dei prodotti parziali
(e vediamo cosa succede nel caso del prodotto di due numeri di 3 bit)



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.

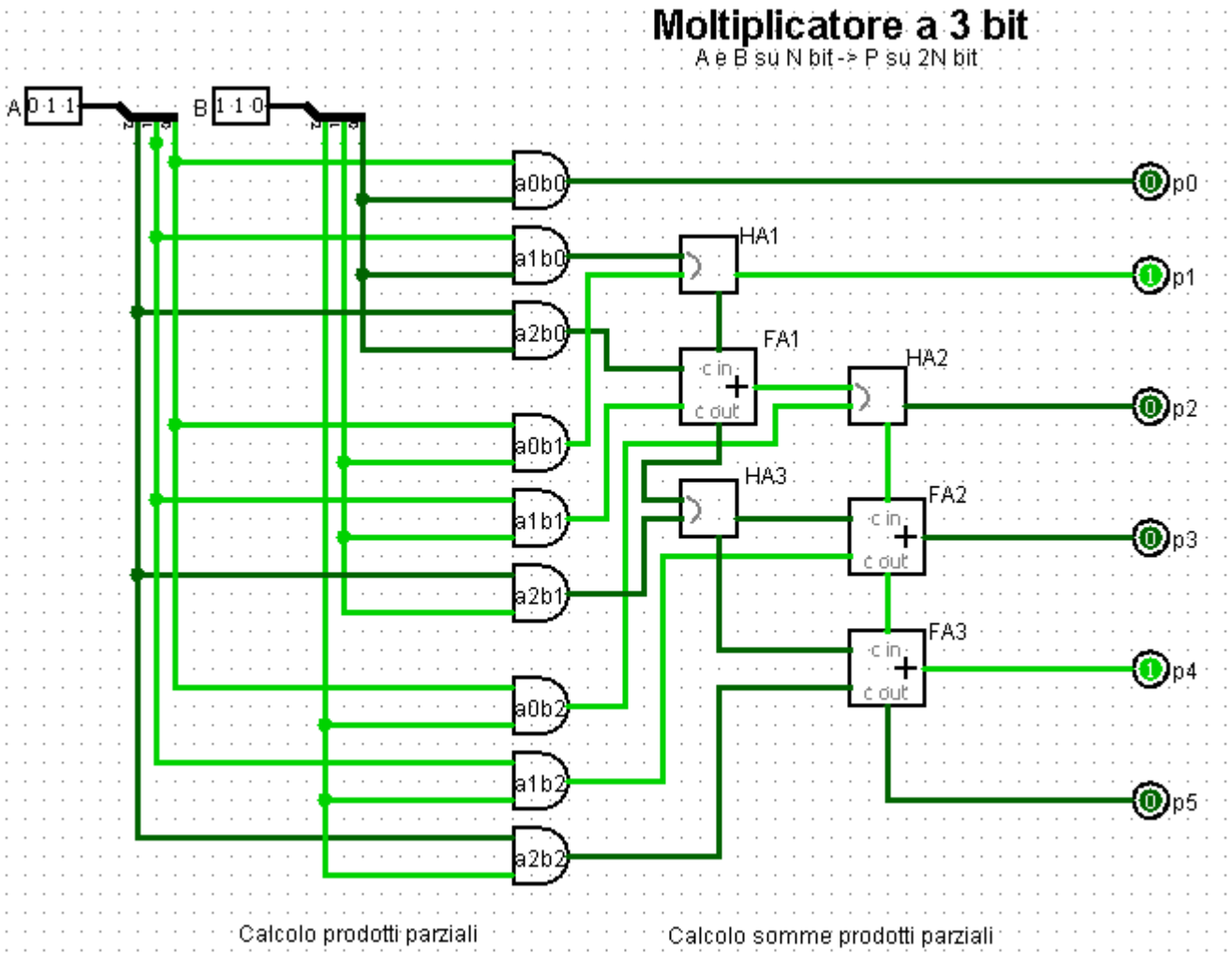
Moltiplicazione binaria: Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



Come fare il calcolo con circuiti logici ?

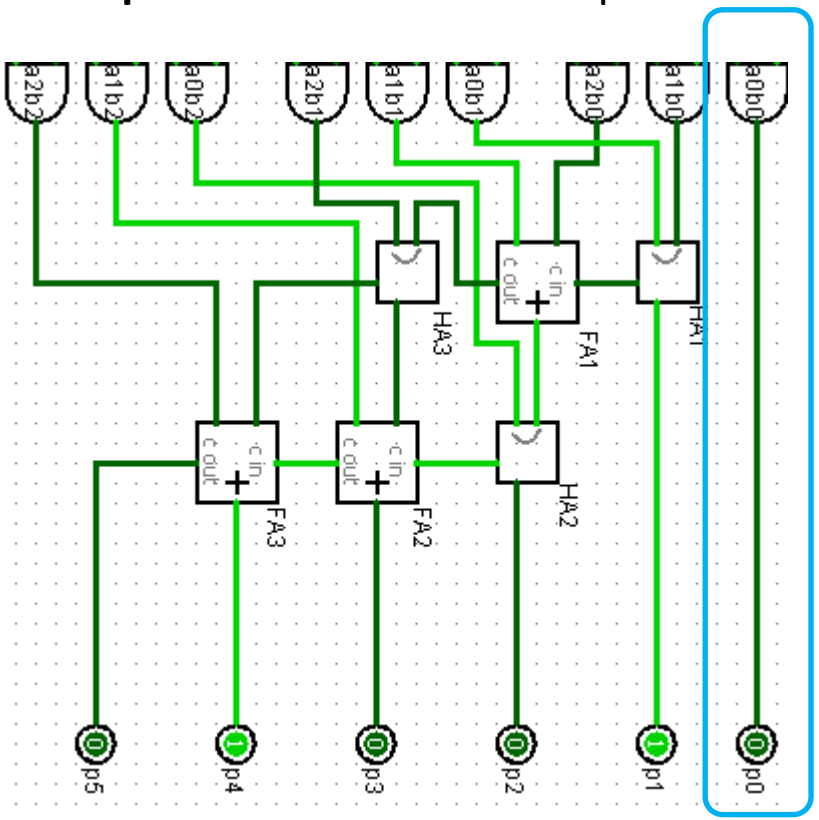
- Possiamo scomporre l'operazione in due stadi:
- Primo stadio: prodotti parziali
  - si mette in AND ciascun bit del moltiplicatore con i bit corrispondenti del moltiplicando
- Secondo stadio: somme
  - si effettuano le somme (full adder) dei bit sulle righe contenenti i prodotti parziali

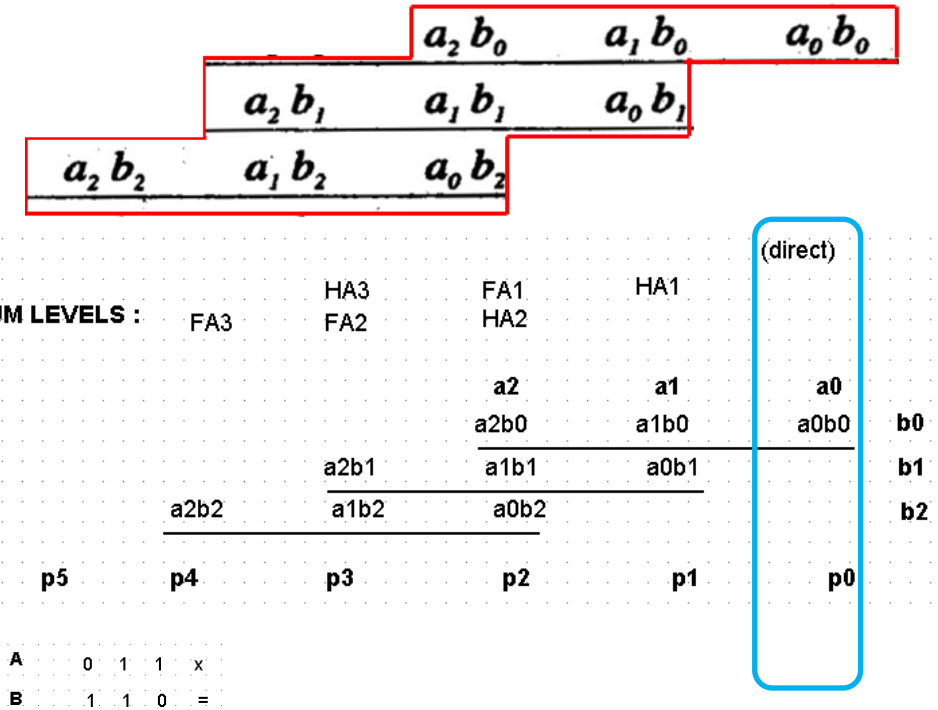
Moltiplicazione binaria: Moltiplicatore numeri a 3 bit in Logisim. Soluzione.

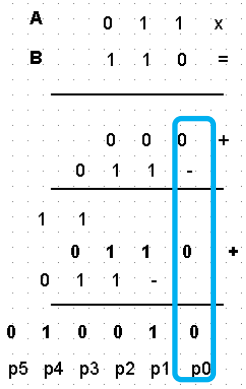


Concentriamoci sulla parte delle somme …

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

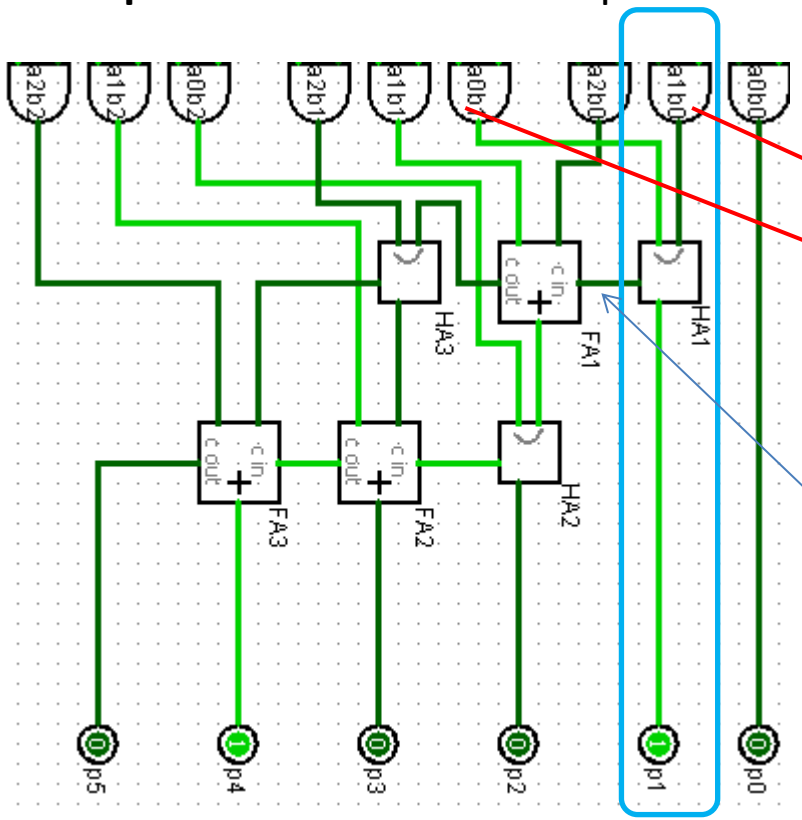


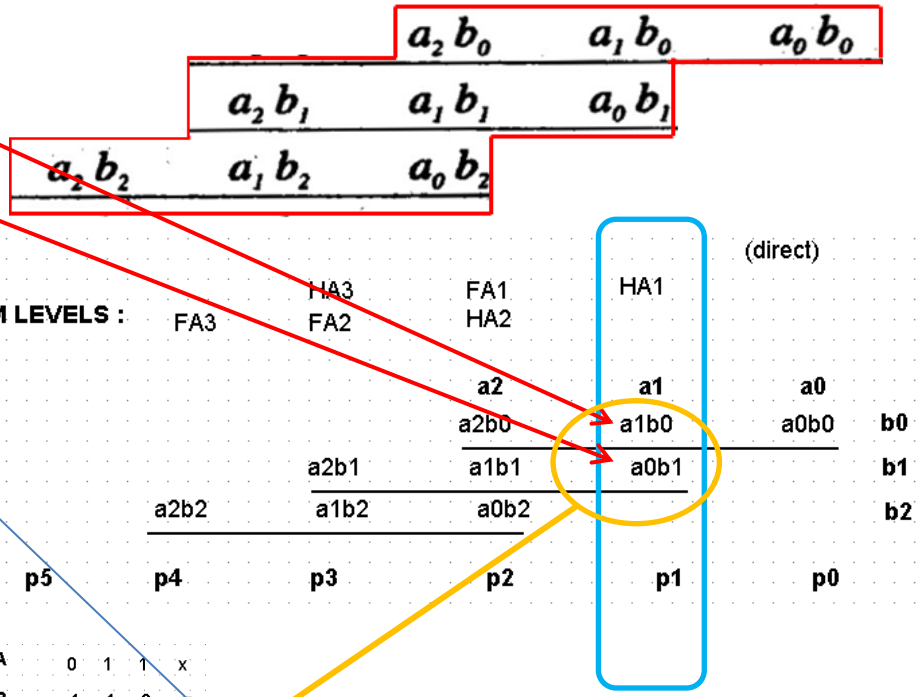


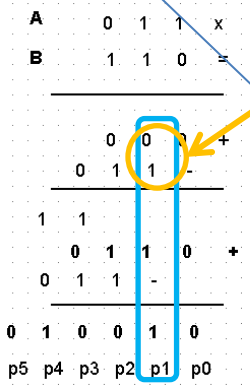


Concentriamoci sulla parte delle somme …

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



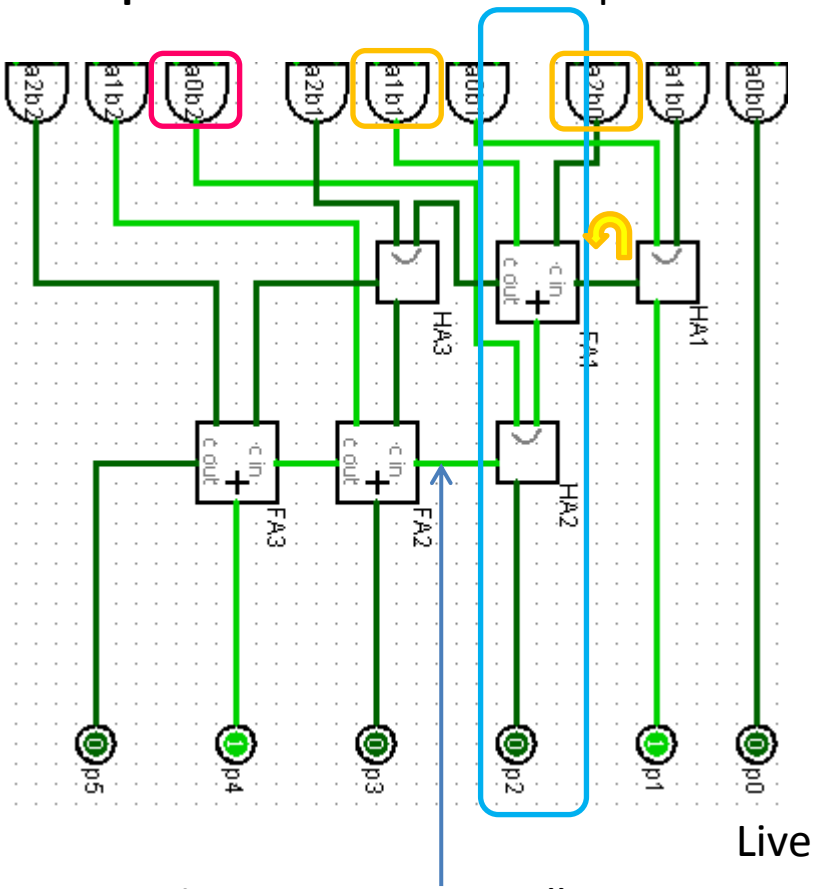




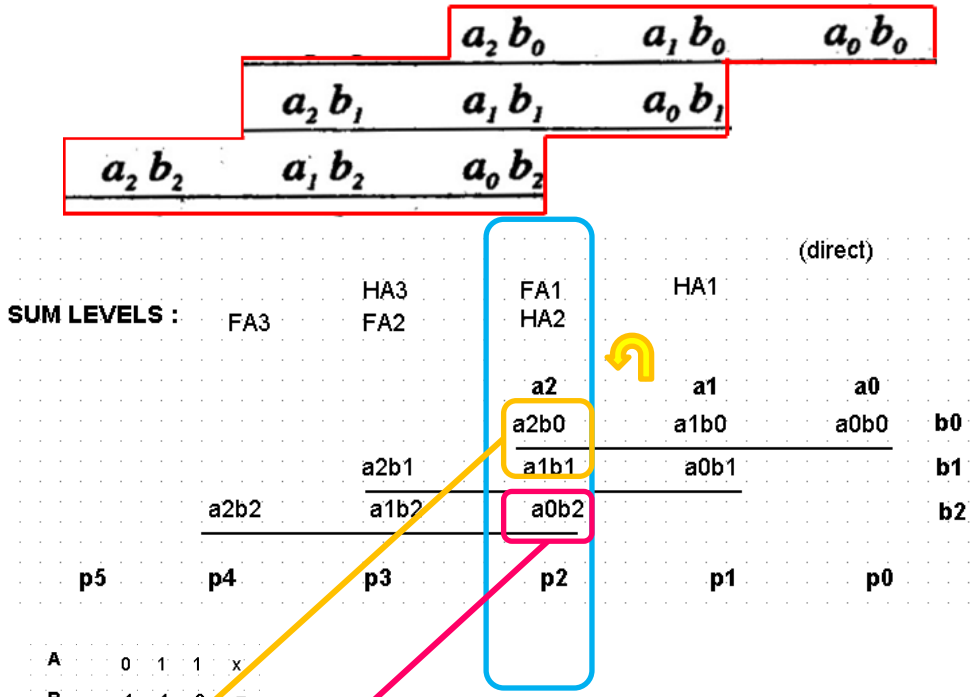
Concentriamoci sulla parte delle somme …

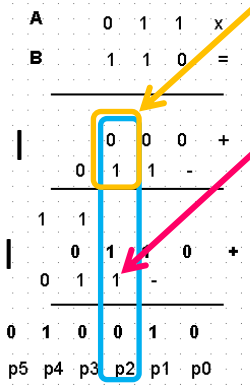
NB: nessun riporto…

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



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





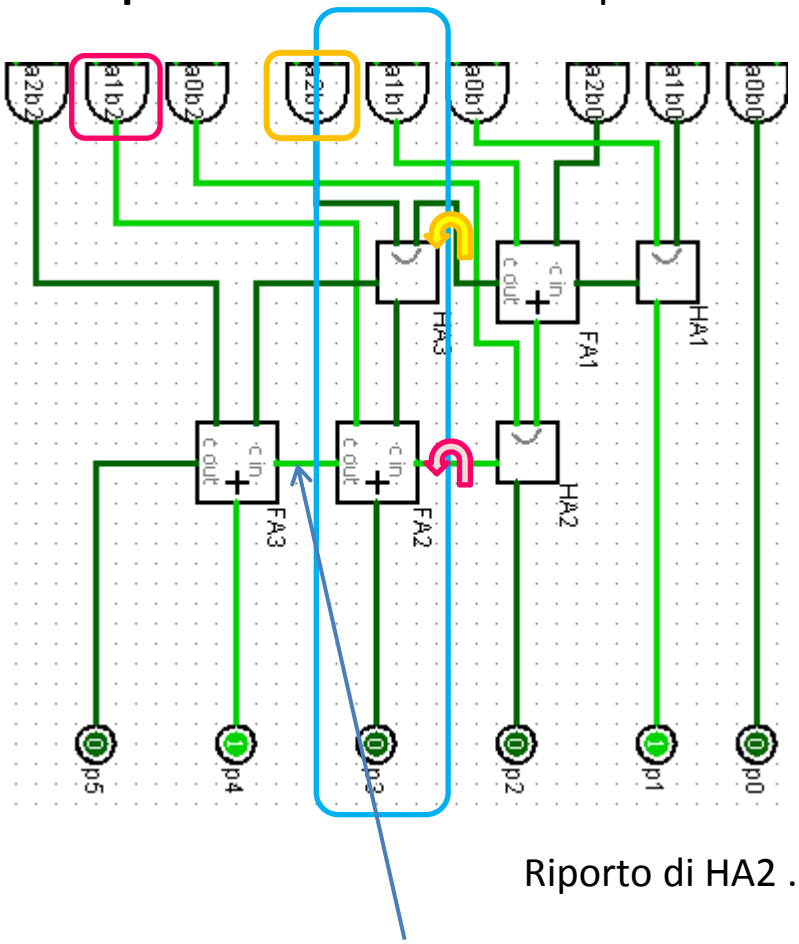
Livello I

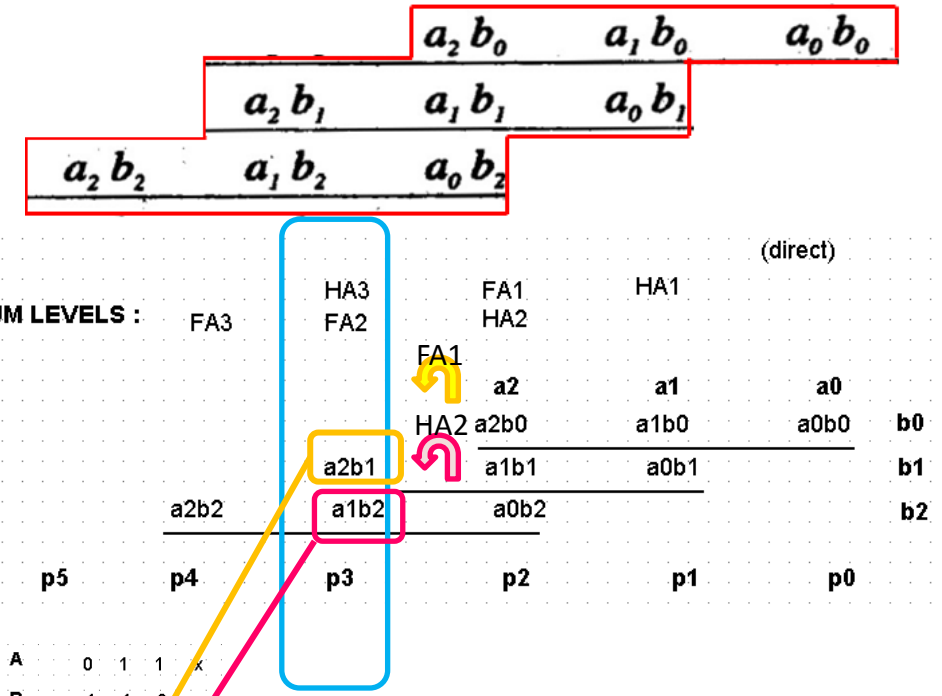
Livello II

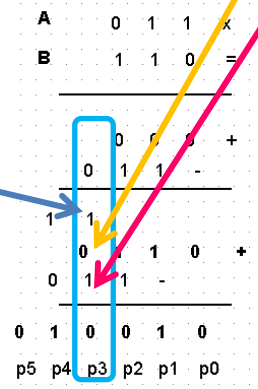
Dobbiamo gestire possibile riporto e somma di 2 valori (a2b0 e a1b1) → **FA1**

Inoltre dobbiamo gestire ulteriore valore (a0b2) **e** il **risultato** della somma precedente → **HA2** (**2** ingressi)

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







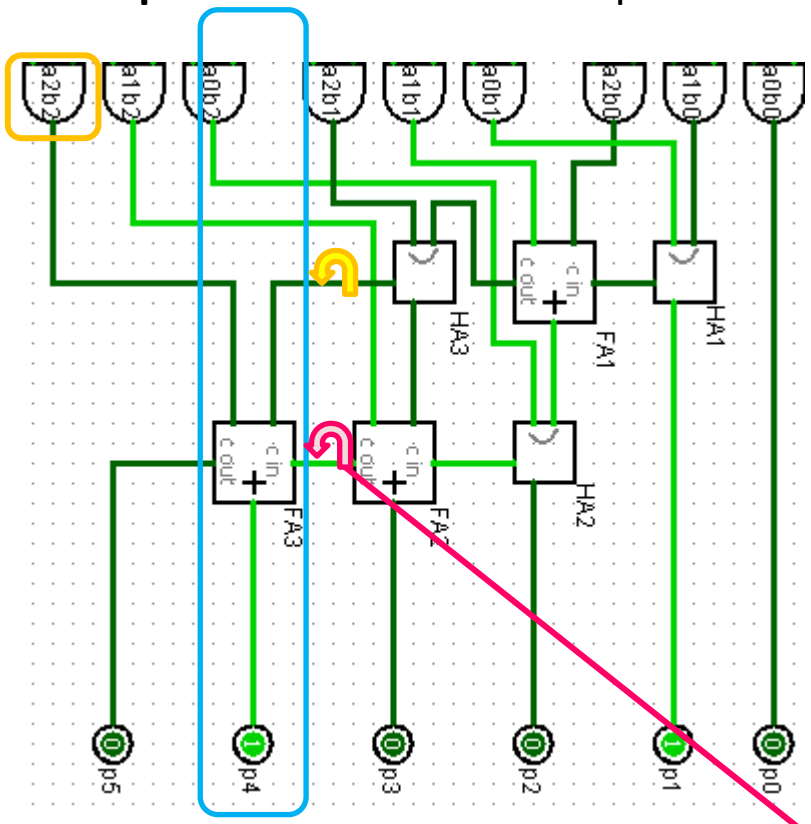
Riporto di HA2 …

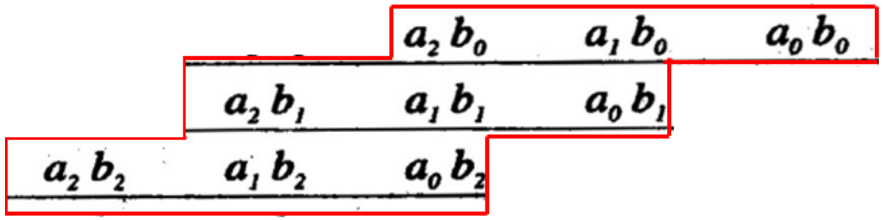
NB: Anche qui ho **riporto**. E anche qui viene Propagato a livello II (da FA2)

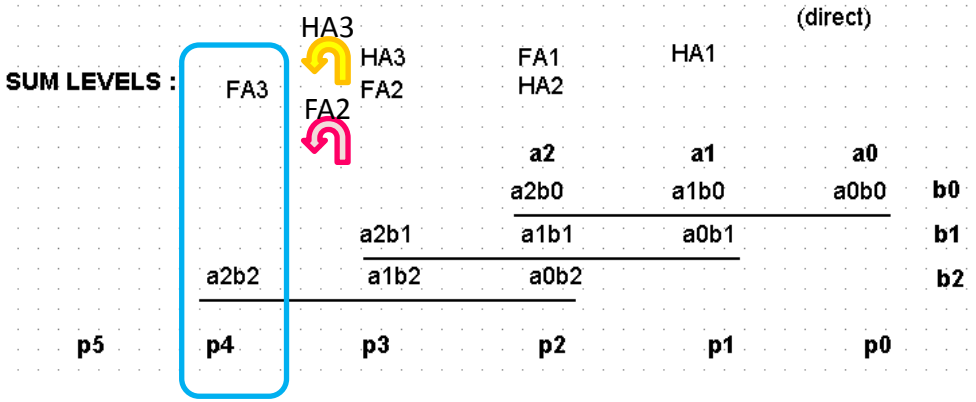
Dobbiamo gestire <u>possibile **riporto** (da FA1),</u> somma di **2 valori** (a2b1 e a1b2) e possibile **riporto** da HA2. <u>Un FA non basta</u>! Sommiamo **a2b1** e **riporto FA1** usando un HA (HA3)

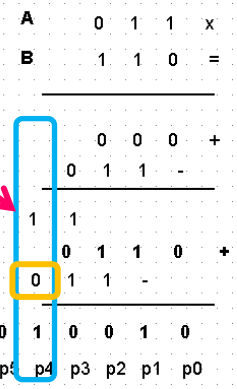
Sommiamo Il risultato di livello I , a1b2 e il riporto di HA2 usando un FA (FA2)

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



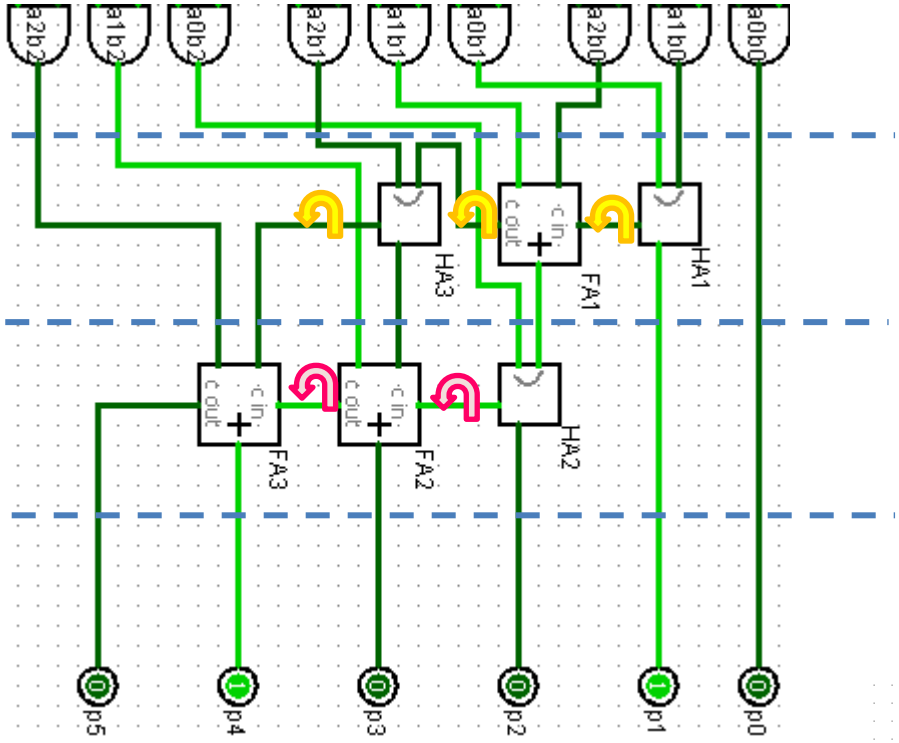






Dobbiamo gestire possibile **riporto** (da HA3 level I), somma di **1 valore** (a2b2) **e** possibile **riporto** da Level II FA2. Un FA basta!
Sommiamo **a2b2** e **riporto HA3** e, come r_in sommiamo riporto FA2.
NB: Il **riporto** di questo FA3 fornisce l'ultimo bit del prodotto risultante.

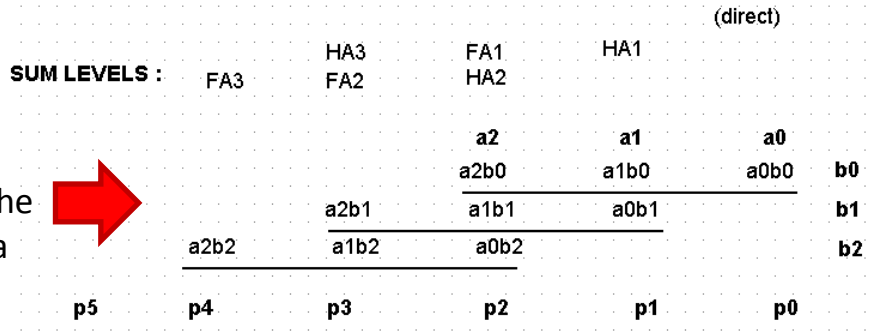
Moltiplicazione binaria: Moltiplicatore numeri a 3 bit in Logisim. Soluzione.



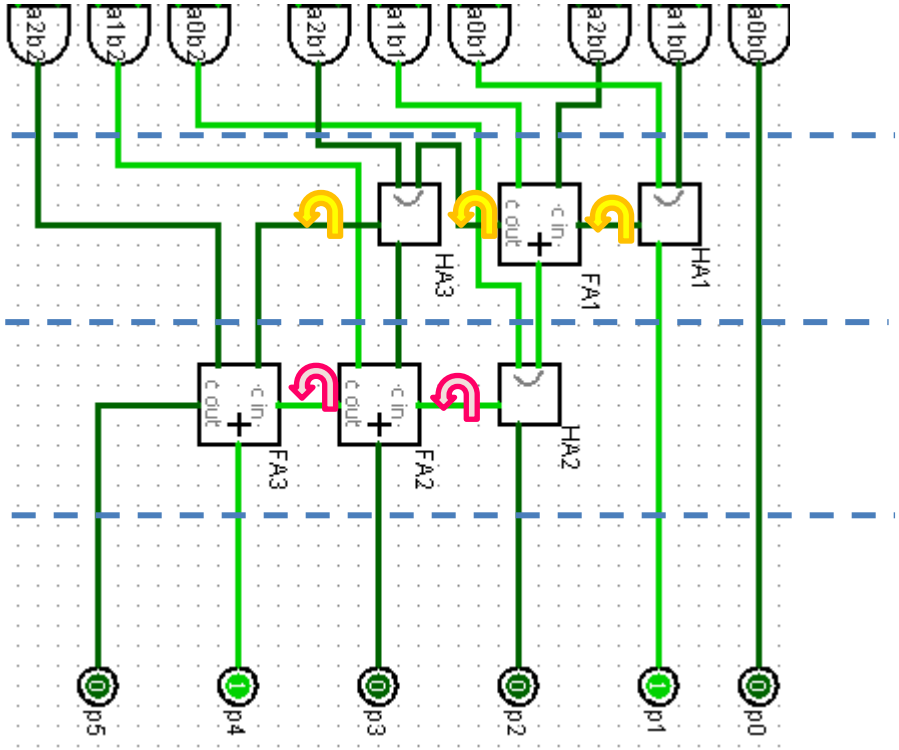
OSSERVAZIONI :

- 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.



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



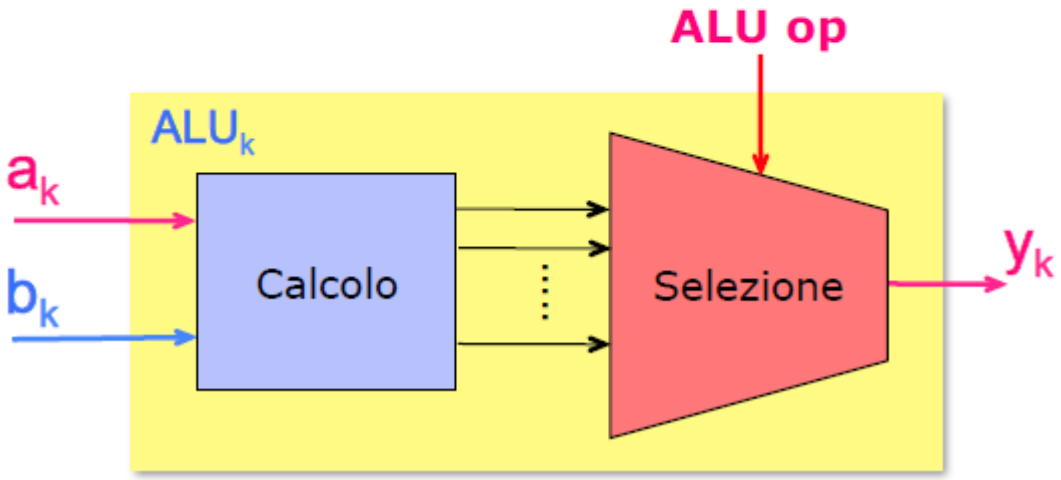
Esercizio 5: Estendere il moltiplicatore di numeri a 3 bit in modo da ottenere un moltiplicatore di numeri a 4 bit.
Seguire fedelmente lo schema circuitale disponibile in slide 21 per la parte di somma dei prodotti parziali.
Scegliete voi se sviluppare il circuito in senso verticale o orizzontale.

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}



Arithmetic Logic Unit (ALU) : Struttura modulare ALU

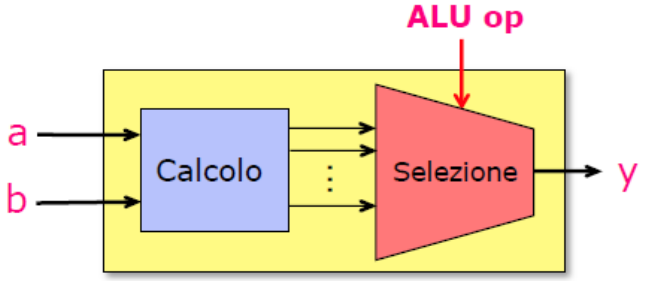
1bit ALU

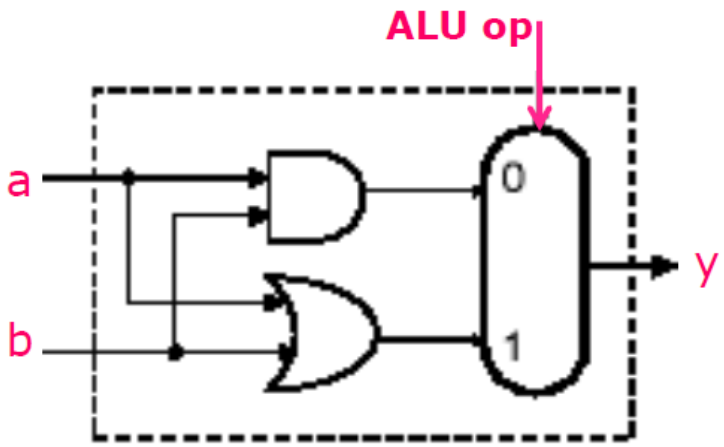
Operazioni logiche: **AND/OR**

- Selezionabile

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

NB:





Selezione:

ALUop = 0 → y = **AND**(a,b)

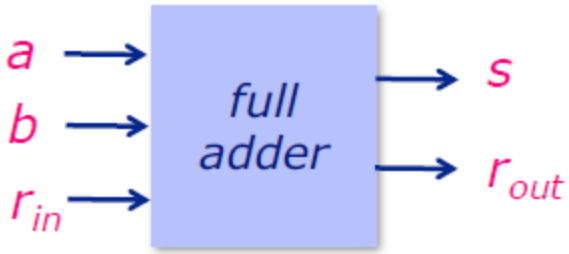
ALUop = 1 → y = **OR**(a,b)

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}**



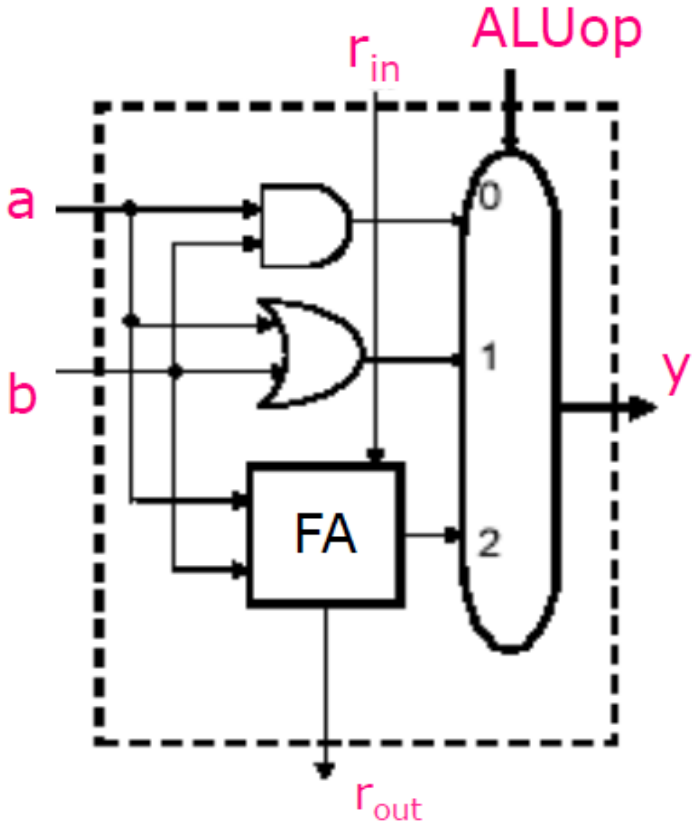
Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU

Operazioni: **OR, AND, somma**

Selezione:

- **ALUop** a 2 bit
  - **00: AND: s = a and b**
  - **01: OR: s = a or b**
  - **10: +: $s/r_{out} = a + b + r_{in}$**



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**)

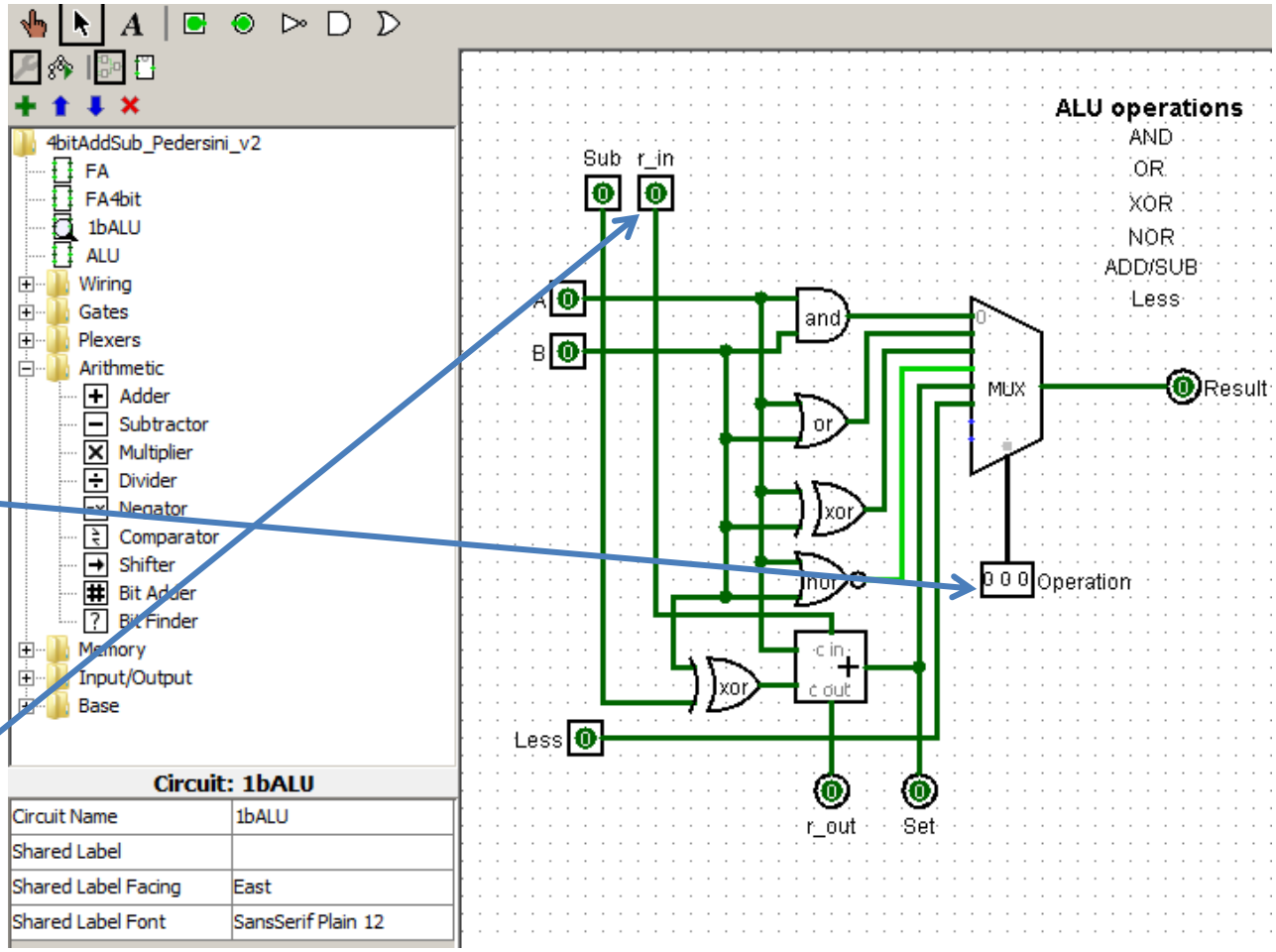
Impostare l'aspetto del componente risultante in modo che sia facile realizzare Nbit ALU (cosa che vedremo negli esercizi successivi)

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

Arithmetic Logic Unit (ALU) : Struttura modulare ALU

1bit ALU: **realizzazione in Logisim**

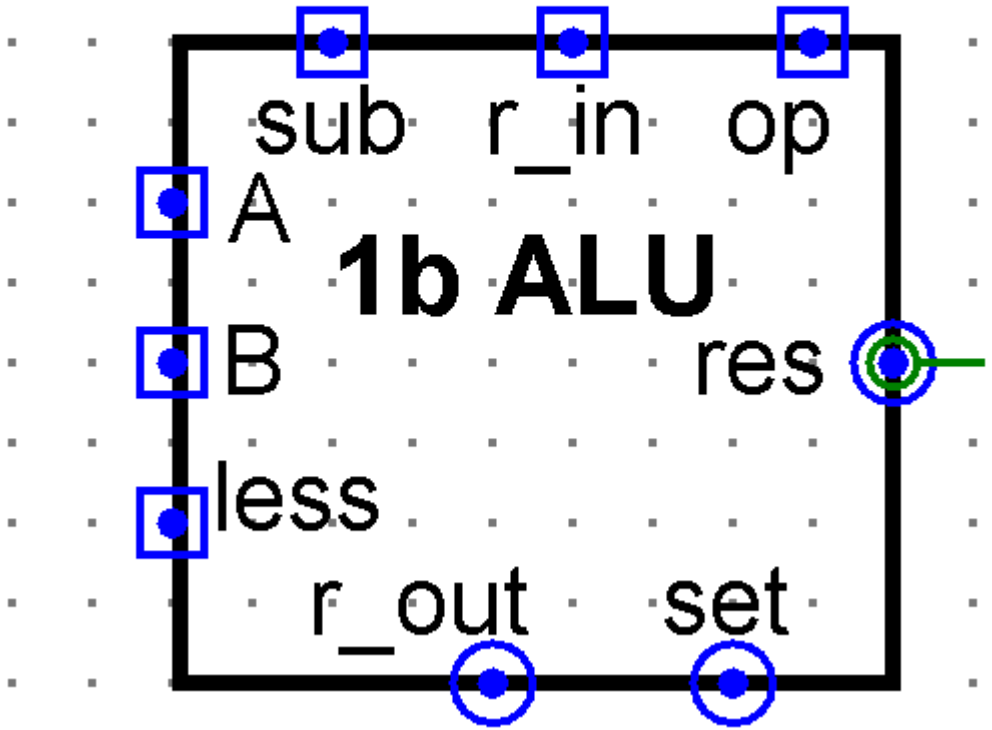
Aspetto componente:
E' di fondamentale importanza che l'aspetto del componente sia funzionale al suo utilizzo.

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)



Arithmetic Logic Unit (ALU) : Struttura modulare ALU

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

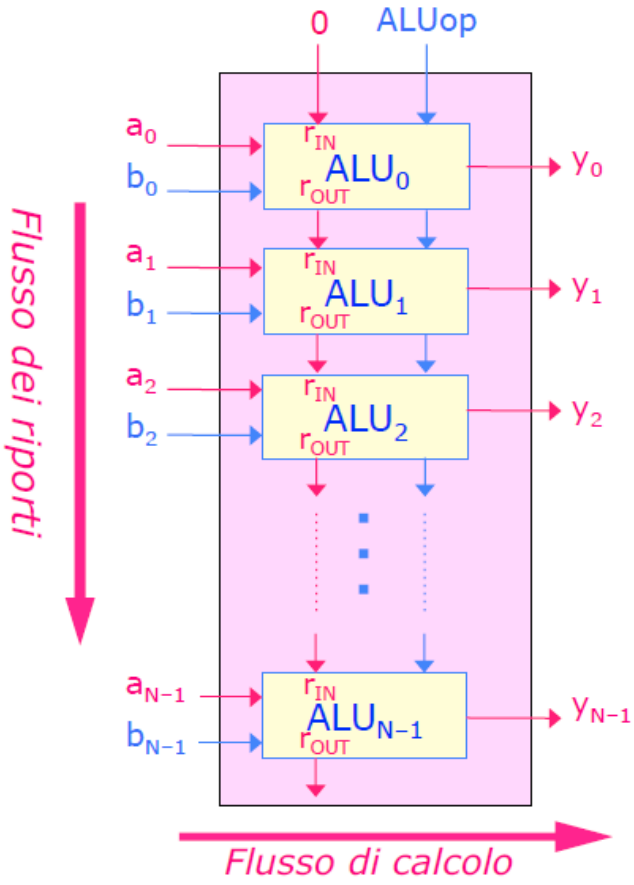
ALU a N-bit

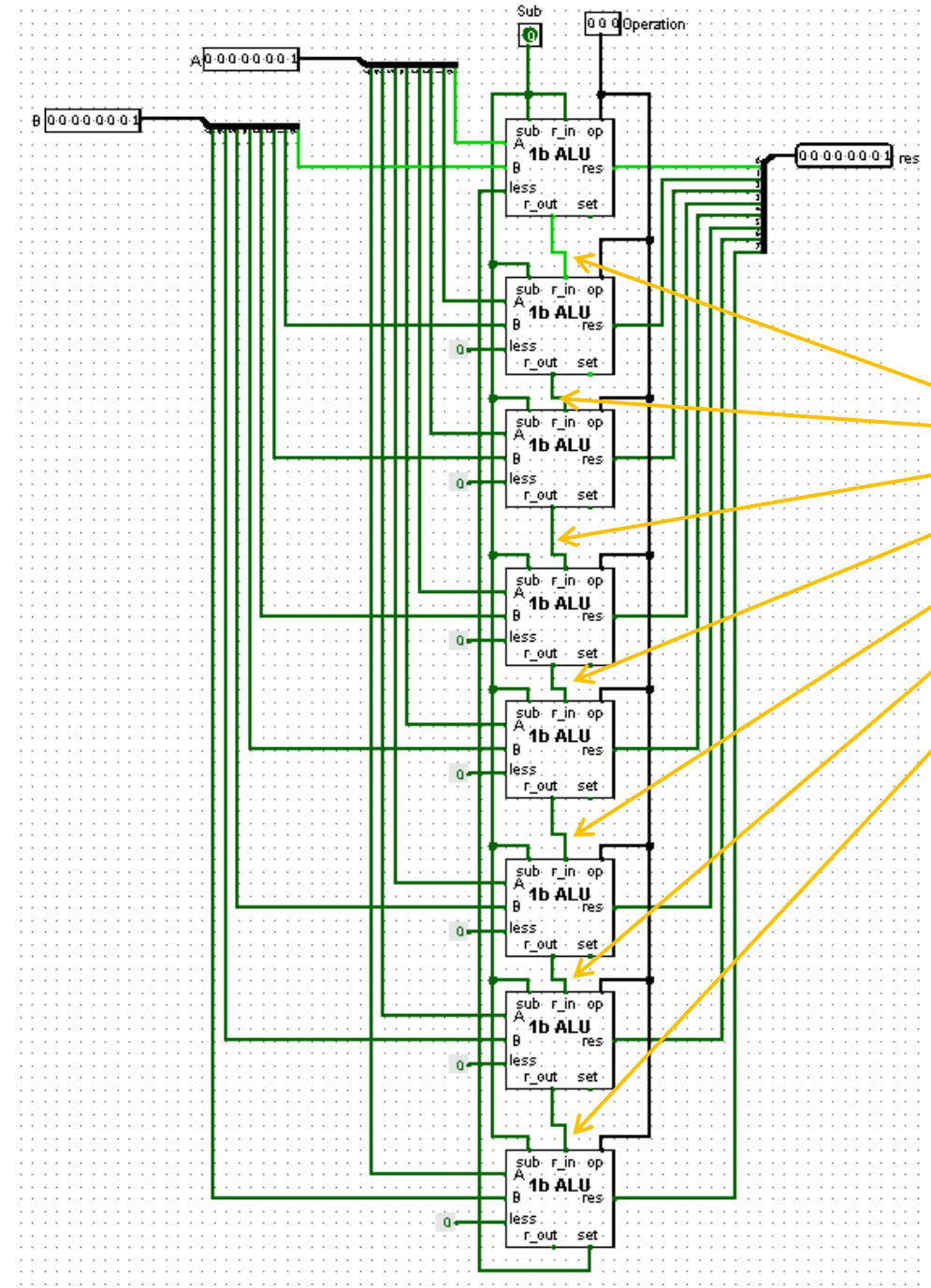
- Come collegare N ALU a 1 bit per ottenere una ALU a N bit?

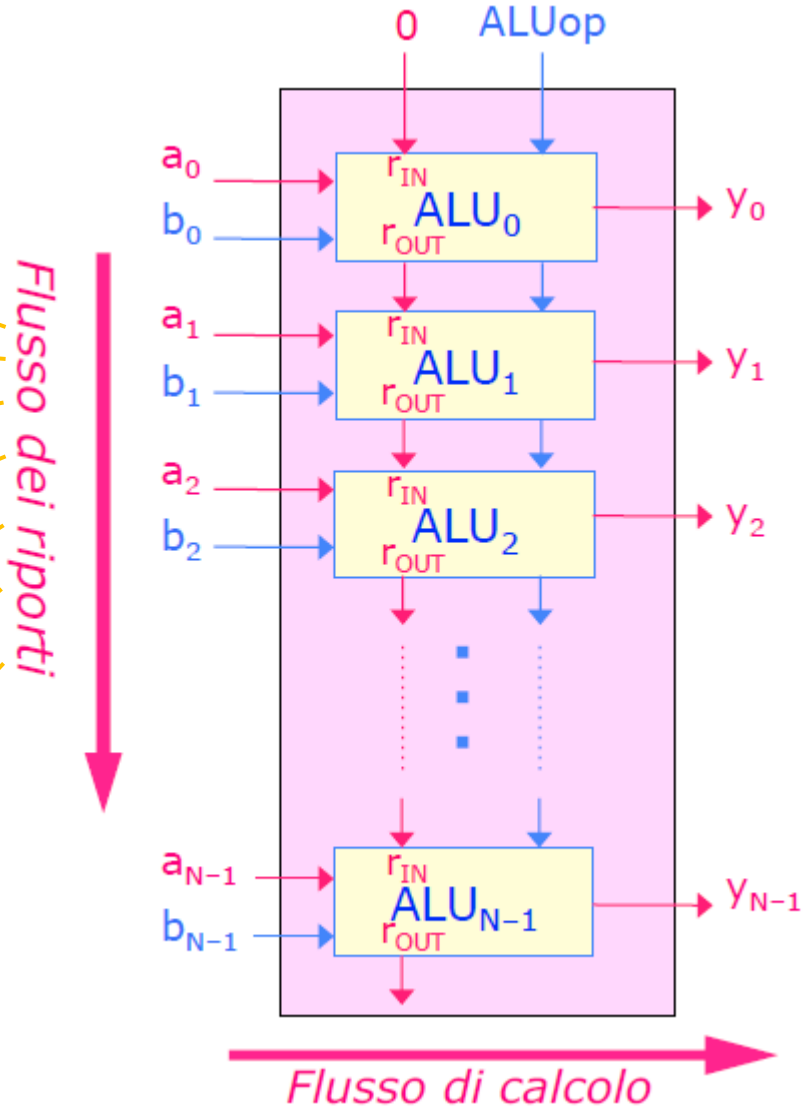
ALU a N bit:

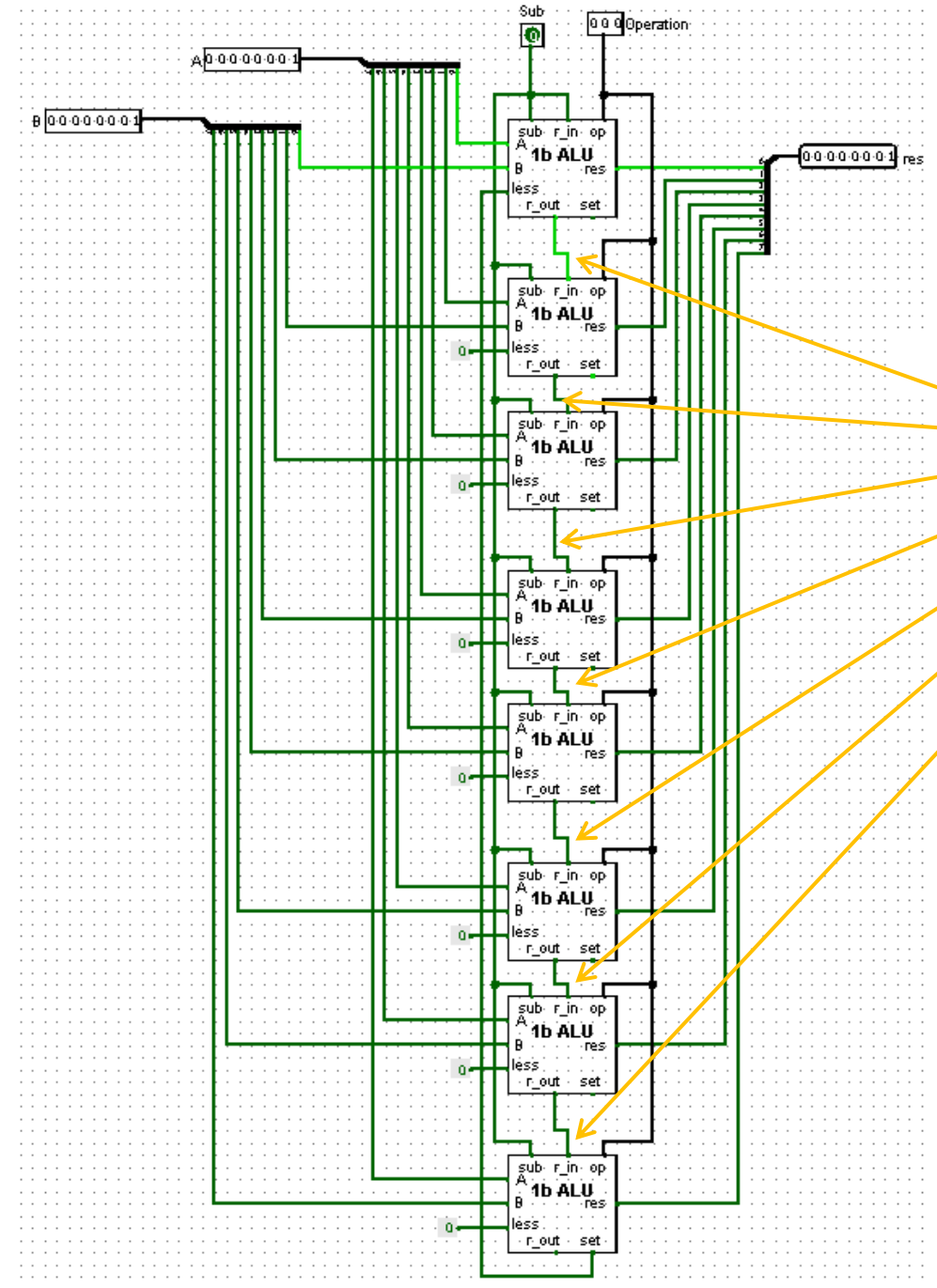
N ALU in parallelo + propagazione dei riporti

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

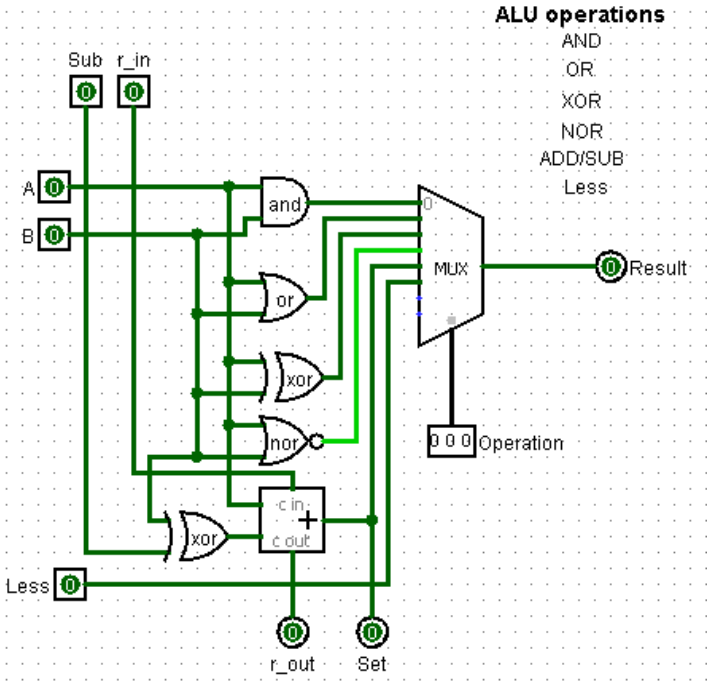



Sub
Operation
A
B
res
sub r_in op
1b ALU
less
r_out set

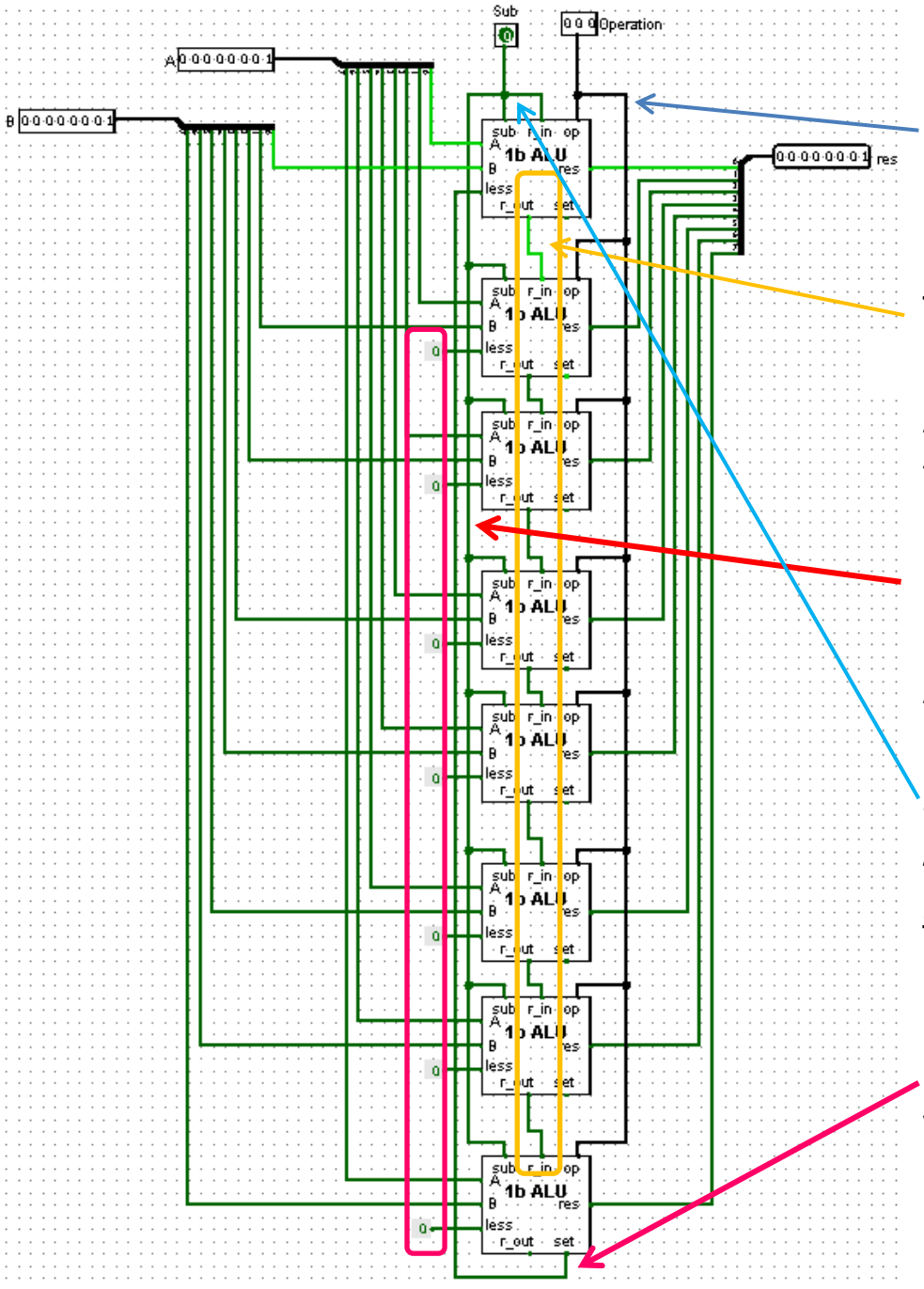

0
ALUop
a_0
b_0
r_{IN}
ALU_0
r_{OUT}
y_0
a_1
b_1
ALU_1
y_1
a_2
b_2
ALU_2
y_2
a_{N-1}
b_{N-1}
ALU_{N-1}
y_{N-1}
Flusso dei riporti
Flusso di calcolo



Flusso dei riporti



Flusso di calcolo
(1bit ALU)



Note collegamento tra 1bit ALU:

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

Tutti i valori **r_in** dalla <u>seconda</u> 1bit ALU in poi vengono ottenuto **da r_out** della 1bit ALU <u>precedente</u> (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 <u>sottrazione</u> 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