



Corso di Architettura degli Elaboratori a.a. 2021/2022

Il livello logico digitale:
Algebra Booleana e
Circuiti logici digitali di base

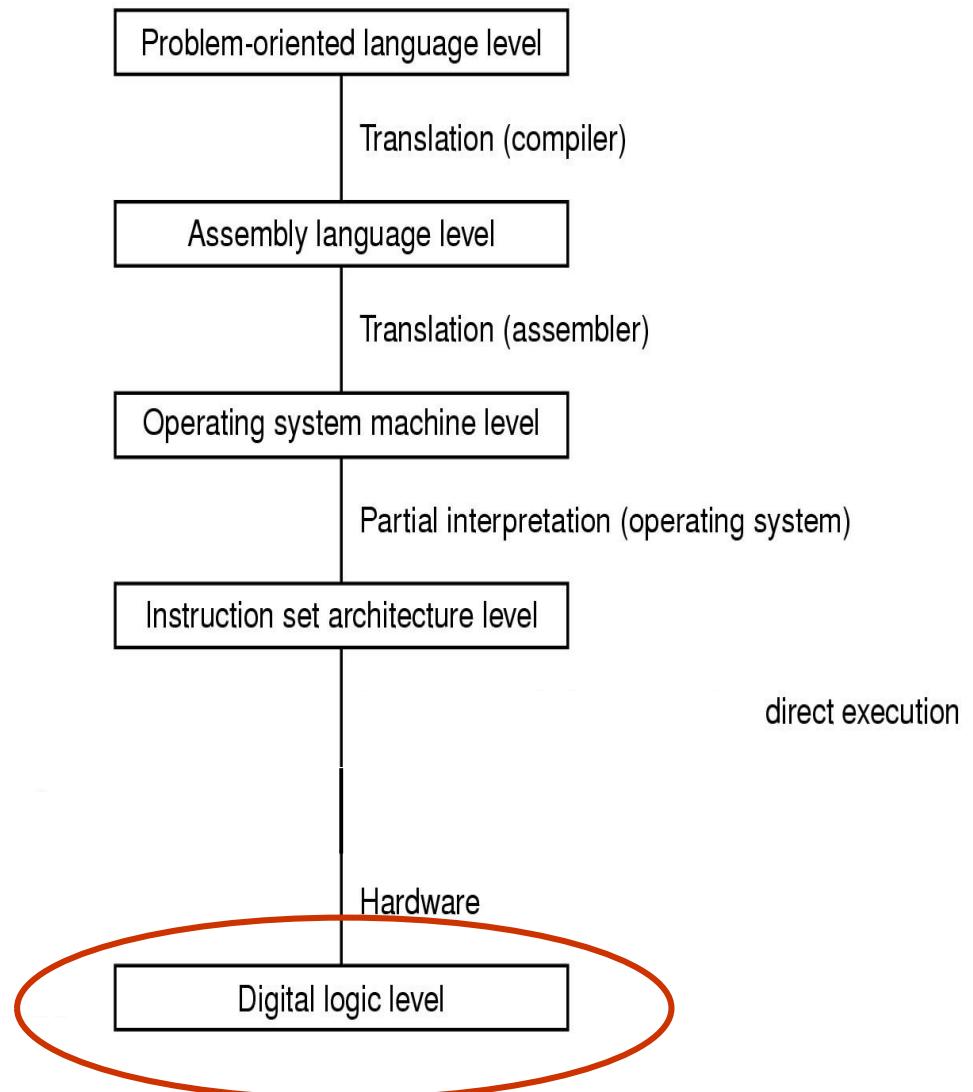
Livello della logica digitale

Livello della *Logico-Digitale*

Costituenti di base del computer:

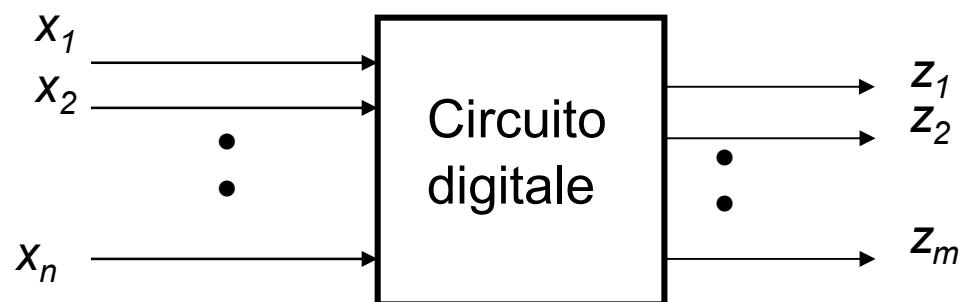
- porte
- registri
- memoria

Sotto questo livello ci sono i dispositivi (funzionamento interno delle porte: transistor)



Circuiti digitali

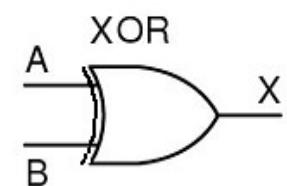
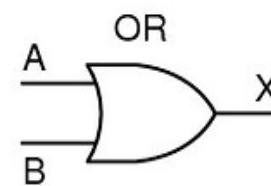
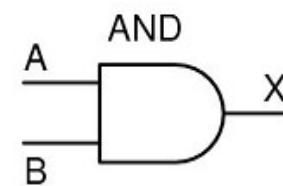
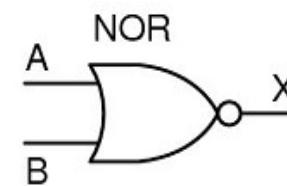
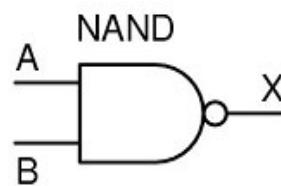
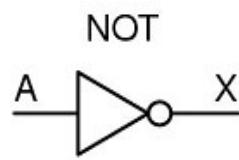
- Gli elementi di base con cui si sono costruiti i calcolatori si chiamano **circuiti digitali** (o reti digitali)
- Sono dispositivi che utilizzano **solo due valori logici**: 0 (segnale tra 0 e 1 volt) e 1 (segnale tra 2 e 5 volt).
 - I valori di tensione possono anche essere altri.
- Un circuito digitale trasforma segnali (binari) di **ingresso** x_1, x_2, \dots, x_n nei segnali (binari) di **uscita** z_1, z_2, \dots, z_m .



Porte logiche

- I circuiti sono detti **combinatori** quando l'uscita è funzione esclusivamente dell'ingresso; sono detti **sequenziali** quando l'uscita è funzione oltre che dell'ingresso anche di uno **stato**.
- Gli elementi primitivi dei circuiti digitali sono chiamati **porte logiche** e calcolano alcune funzioni di questi segnali a due valori.
- Questi dispositivi si basano sul fatto che si può far funzionare un **transistor** come un interruttore binario molto veloce.

Porte logiche



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

(f)

Diapositiva 5

DI0

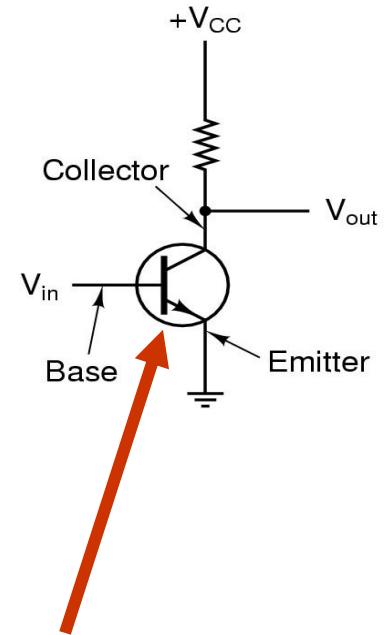
Per parlare delle porte prima del transitor

Drago Idilio; 2022-04-02T20:30:51.462

Transistor

Il transistor si comporta come un interruttore binario molto veloce:

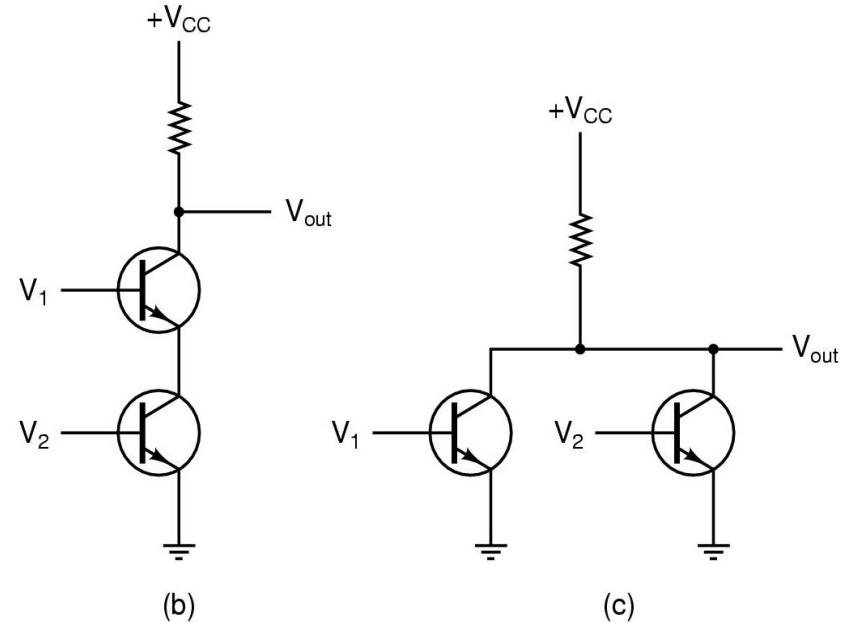
- quando V_{in} è basso, il transistor si disabilita (circuito aperto) e si comporta come “*una resistenza infinita*”, quindi V_{out} è alto; viceversa quando V_{in} è alto, il transistor si attiva e si comporta come “*un filo*” mettendo a terra V_{out}
- pochi **nanosecondi** per passare da uno stato all'altro: “alto” (tensione V_{CC}) 1 logico, “basso” (terra) 0 logico



Transistor bipolare

Porte logiche

- **Logica positiva** ($0 = \text{bassa tensione}$; $1 = \text{alta tensione}$)
 - (b) una **porta NAND**: due transistor collegati in serie
 - (c) una **porta NOR**: due transistor collegati in parallelo

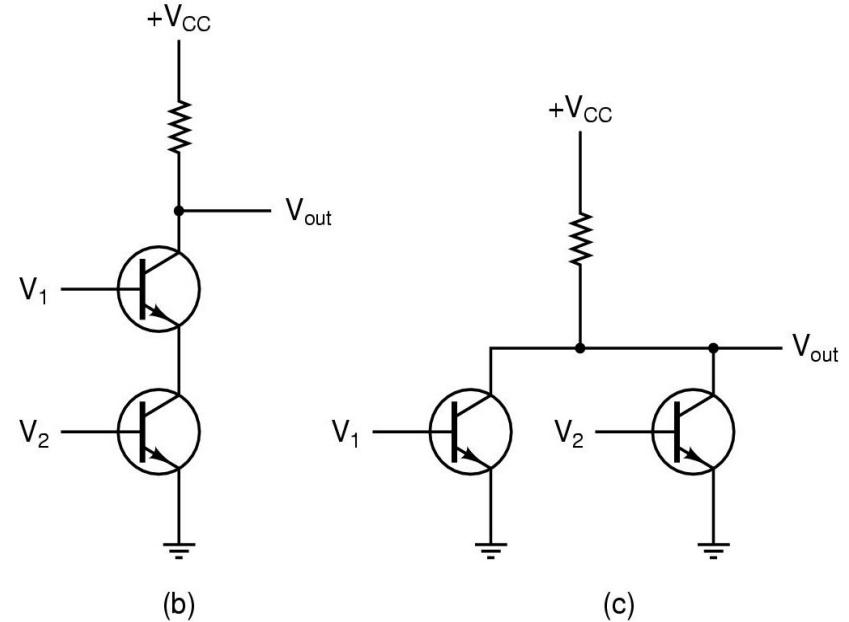


V_1	V_2	V_{out}	V_1	V_2	V_{out}
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0

(b) (c)

Porte logiche

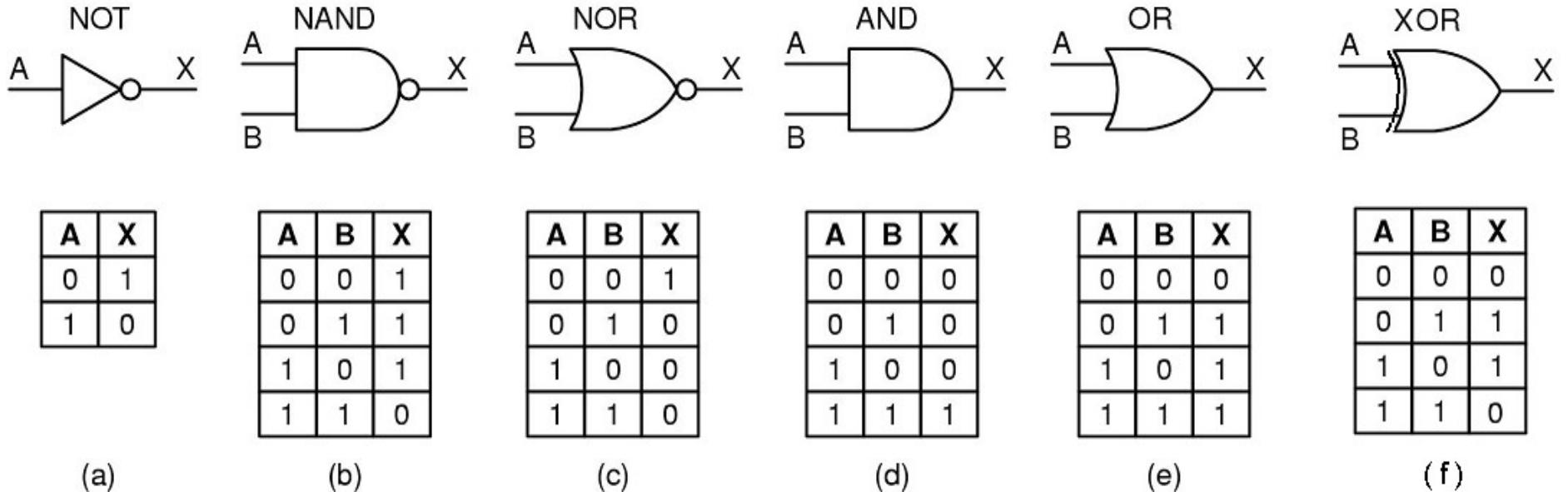
- **Logica negativa** ($0 = \text{alta tensione}$; $1 = \text{bassa tensione}$)
 - (b) una **porta NOR**: due transistor collegati in serie
 - (c) una **porta NAND**: due transistor collegati in parallelo



V_1	V_2	V_{out}	V_1	V_2	V_{out}
1	1	0	1	1	0
1	0	0	1	0	1
0	1	0	0	1	1
0	0	1	0	0	1

(b) (c)

Porte logiche



- NOT: un transistor (invertitore)
- NAND e NOR: due transistor
- AND e OR: tre transistor
(quelli del NAND e NOR, rispettivamente, più un invertitore)
- XOR: 8 transistor ($A \text{ XOR } B = (A \text{ OR } B) \text{ AND } (A \text{ NAND } B)$)

Algebra Booleana

(George Boole, 1815-1864)

- L'analisi e la **progettazione del comportamento** dei circuiti digitali si fonda sull'Algebra di Boole
 - **Analisi**: modo sintetico di descrivere le funzioni dei circuiti digitali
 - **Progettazione**: data una funzione del circuito digitale sviluppo di implementazione semplificata (o ottimizzata)
- Come ogni altra algebra si fa uso di variabili e di operazioni
 - Variabili logiche: possono assumere solo il valore 0 (FALSO) o 1 (VERO)
 - Operazioni logiche: AND, OR e NOT

Algebra Booleana

(George Boole, 1815-1864)

- Due valori costanti 0 e 1
- Operatore unario “**NOT**”:
 - \bar{A} : “not” A ($\neg A$)
- Operatori binari “**AND**” e “**OR**”:
 - AB: A “and” B
 - A + B: A “or” B
- Una qualunque combinazione di variabili o costanti booleane legate tra loro dagli operatori fondamentali è una **espressione logica**
- *Esempio:* $A\bar{B} + \bar{B}C$ (è vera solo quando $A = 1$ e $B = 0$ oppure $B = 1$ e $C = 0$)
- In assenza di parentesi AND ha precedenza su OR

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\bar{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Algebra Booleana

- Dimostriamola:

$$A+BC = (A+B)(A+C) \quad \text{TESI}$$

$$(A+B)(A+C) =$$

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\bar{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Diversa dall'algebra ordinaria

Algebra Booleana

- Dimostriamola:

$$A+BC = (A+B)(A+C) \quad \text{TESI}$$

$$\begin{aligned} & (A+B)(A+C) = \\ & = AA + AC + BA + BC = \\ & = A + AC + AB + BC = \\ & = A(1 + C + B) + BC = \\ & = A + BC \end{aligned}$$

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\bar{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Diversa dall'algebra ordinaria

Algebra Booleana

- Dimostriamola:

$$(\overline{A+B}) = \overline{AB}$$

TESI

$$(\overline{A+B})(A+B) = 0$$

se vale la tesi allora

$$(\overline{AB})(A+B) = 0$$

$$\overline{ABA} + \overline{ABB} = 0\overline{B} + \overline{A}0 = 0$$

CVD

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Una forma della legge di De Morgan

Algebra Booleana

- Dimostriamola:

$$\overline{(AB)} = \overline{A} + \overline{B}$$

TESI

$$\overline{(AB)}(AB) = 0$$

se vale la tesi allora

$$(\overline{A} + \overline{B})(AB) = 0$$

$$\overline{ABA} + \overline{ABB} = 0B + A0 = 0$$

CVD

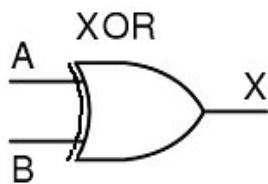
Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\overline{B}$

Una forma della legge di De Morgan

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)(\overline{AB}) = \overline{AB} + \overline{AB}$$



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

(f)

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)(\overline{AB}) = \overline{AB} + \overline{AB}$$

De Morgan

partiamo da

$$(A+B)(\overline{AB}) = (A+B)(\overline{A} + \overline{B})$$

De Morgan's law

$$\overline{AB} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A}\overline{B}$$

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)\overline{(AB)} = \overline{AB} + \overline{AB}$$

partiamo da

$$(A+B)\overline{(AB)} = (A+B)(\overline{A}+\overline{B}) = (A+B)(B+\overline{B})(\overline{A}+\overline{B})(A+\overline{A}) =$$

Sempre 1, per cui posso metterli in AND

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)\overline{(AB)} = \overline{AB} + \overline{AB}$$

partiamo da

$$\begin{aligned}(A+B)\overline{(AB)} &= (A+B)(\overline{A}+\overline{B}) = (A+B)(B+\overline{B})(\overline{A}+\overline{B})(A+\overline{A}) = \\ &= (AB + A\overline{B} + B\overline{B} + B\overline{B})(\overline{AA} + \overline{AA} + A\overline{B} + \overline{AB}) =\end{aligned}$$

Idempotenza ed AND con propria negazione

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)(\overline{AB}) = \overline{AB} + \overline{AB}$$

partiamo da

$$\begin{aligned}(A+B)(\overline{AB}) &= (A+B)(\overline{A}+\overline{B}) = (A+B)(B+\overline{B})(\overline{A}+\overline{B})(A+\overline{A}) = \\ &= (AB+A\overline{B}+B\overline{B}+B\overline{B})(\overline{AA}+\overline{AA}+\overline{AB}+\overline{AB}) = \\ &= (AB+A\overline{B}+B)(\overline{A}+\overline{AB}+\overline{AB})\end{aligned}$$

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)(\overline{AB}) = \overline{AB} + \overline{AB}$$

Sempre uguali a 1

partiamo da

$$\begin{aligned}(A+B)(\overline{AB}) &= (A+B)(\overline{A}+\overline{B}) = (A+B)(B+\overline{B})(\overline{A}+\overline{B})(A+\overline{A}) = \\ &= (AB+A\overline{B}+BB+B\overline{B})(\overline{AA}+\overline{AA}+\overline{AB}+\overline{AB}) = \\ &= (AB+A\overline{B}+B)(\overline{A}+\overline{AB}+\overline{AB}) = (\overline{AB}+B(A+1))(\overline{AB}+\overline{A}(1+\overline{B})) =\end{aligned}$$

Algebra Booleana

- Dimostrate le due equivalenti forme per lo XOR e calcolate numero di porte e di transistor:

$$(A+B)(\overline{AB}) = \overline{AB} + \overline{AB}$$

AND con propria negazione: sempre 0

partiamo da

$$\begin{aligned}(A+B)(\overline{AB}) &= (A+B)(\overline{A}+\overline{B}) = (A+B)(B+\overline{B})(\overline{A}+\overline{B})(A+\overline{A}) = \\&= (AB+A\overline{B}+BB+B\overline{B})(\overline{AA}+\overline{AA}+\overline{AB}+\overline{AB}) = \\&= (AB+A\overline{B}+B)(\overline{A}+\overline{AB}+\overline{AB}) = (\overline{AB}+B(A+1))(\overline{AB}+\overline{A}(1+\overline{B})) = \\&= (\overline{AB}+B)(\overline{AB}+\overline{A}) = \overline{ABA}\overline{B} + \overline{ABA}\overline{A} + \overline{BAB}\overline{B} + \overline{BAB}\overline{A} = \overline{AB} + \overline{AB}\end{aligned}$$

Funzioni Booleane

Una **funzione booleana** di n variabili x_1, x_2, \dots, x_n è una *relazione* che associa un valore booleano a ciascuna delle 2^n configurazioni possibili delle n variabili:

$$y = f(x_1, x_2, \dots, x_n)$$

Una funzione booleana può essere espressa in varie forme:

- Tabelle di verità
- Formule algebriche
- Mappe di Karnaugh
- Binary Decision Diagrams (BDDs)

Algebra Booleana

- **Tabelle di verità:** descrivono completamente il valore di una funzione Booleana attraverso tutte le combinazioni di input; n input corrispondono a 2^n combinazioni (righe)

- È **finito** l'insieme delle funzioni Booleane di n input: 2^{2^n} funzioni
(es., se $n = 2$ allora 16 funzioni diverse)

- **Forma canonica:** righe ordinate per valori crescenti degli ingressi interpretando i valori delle variabili di ingresso come cifre di una codifica binaria

n variabili di input				variabili di output (valore funzione)
...	
0	0			0
0	0			1
				...
1	1			1

Equivalenza con tabelle di verità

Due, o più, diverse espressioni algebriche sono equivalenti se hanno identiche tabelle di verità

n variabili di input					
		...			
0	0		0		
0	0		1		
		...			
1	1		1		

Espressione 1 Espressione 2

Algebra Booleana: forme canoniche

- **Formula normale disgiuntiva (FND):**
 - Sommatoria (OR) di termini ciascuno dei quali è una produttoria (AND) di **letterali** costituiti da nomi di variabili di ingresso o da negazioni dei nomi di variabili di ingresso.
 - È **minimale** quando, applicando le proprietà algebriche di equivalenza non è possibile ottenere una FND equivalente contenente un numero di letterali inferiore
- **Formula normale congiuntiva (FNC)**
 - Concetto “duale” del precedente ossia è una produttoria (AND) di termini ciascuno dei quali è una sommatoria (OR) di **letterali** costituiti da nomi di variabili di ingresso o da negazioni di nomi di variabili di ingresso

Algebra Booleana

- *Funzione di maggioranza su tre input:* restituisce 1 se la maggioranza degli input è 1, 0 altrimenti
- La funzione produce 1 nella quarta, sesta, settima e ottava riga
- La funzione M è 1 nelle righe: $\bar{A}BC$, $A\bar{B}C$, $A\bar{B}\bar{C}$, ABC
- $M = \bar{A}BC + A\bar{B}C + A\bar{B}\bar{C} + ABC$
- Forma normale disgiuntiva

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Se mettessimo in OR anche i casi dove M vale 0 non cambierebbe nulla

Algebra Booleana

- La funzione produce 0 nella prima, seconda, terza e quinta riga
- La funzione M è 0 nelle righe:

$$M = (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$$

- Forma normale congiuntiva

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Se mettessimo in AND anche i casi dove M vale 1 non cambierebbe nulla



Da formula algebrica a tabella di verità

- Elencare tutte le possibili configurazioni delle n variabili di ingresso
- Per ogni configurazione, valutare i valori di uscita delle funzioni elementari **NOT**, **AND** e **OR** che compongono l'espressione
- Assumendo l'espressione iniziale una **FND** (FNC), l'uscita della funzione **OR** (AND) rappresenta il valore da inserire nella corrispondente riga della tabella che si sta costruendo

Tabella di verità \longleftrightarrow Formula algebrica

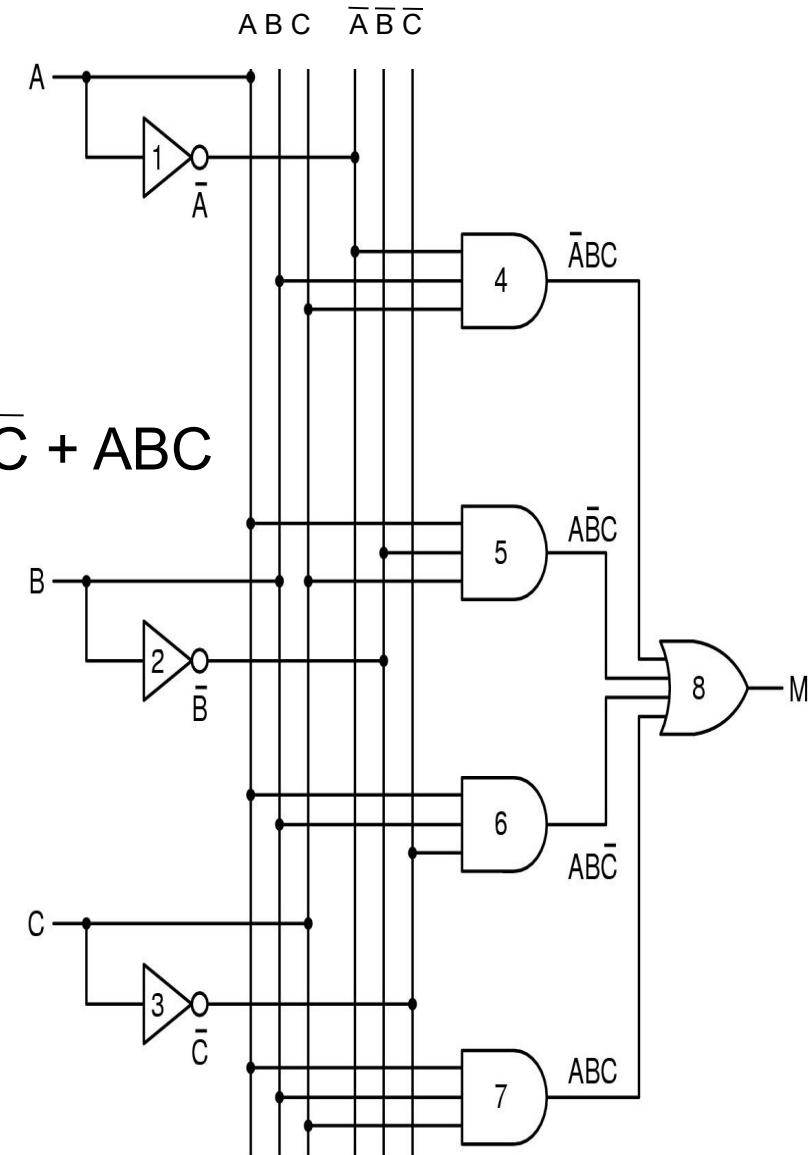
Da tabella di verità a formula algebrica

1. Scrivere la tabella di verità per la funzione
2. Disporre gli invertitori per generare il complemento di ogni input
3. Introdurre una porta AND per ogni termine con un 1 nella colonna dei risultati
4. Collegare le porte AND agli input appropriati
5. Inviare l'output di tutte le porte AND in una porta OR

Convenzione: l'incrocio tra due linee non implica alcuna connessione a meno che non sia presente il simbolo • nel punto di intersezione

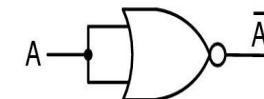
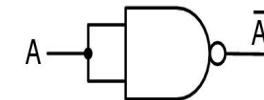
$$\bar{A}\bar{B}C + A\bar{B}C + A\bar{B}\bar{C} + ABC$$

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

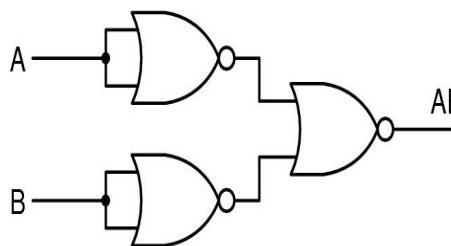
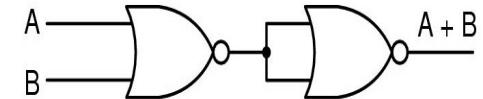
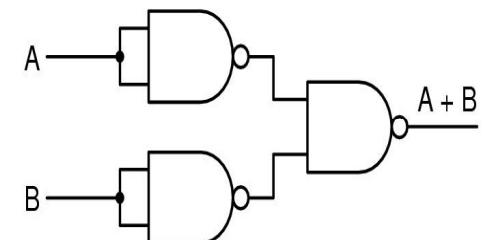
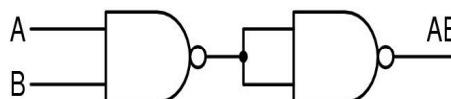


Implementazione di circuiti

- Sostituire le porte con più input con dei circuiti equivalenti che usano porte a due input
- Convertire il circuito in un solo tipo di porta (per convenienza)
- NAND e NOR sono porte **complete**
- ***Usate le identità per dimostrare l'equivalenza dei circuiti fatti solo con NAND e NOR***
- Nota: in generale non si ottiene il circuito ottimale (per numero di porte impiegate)



(a)

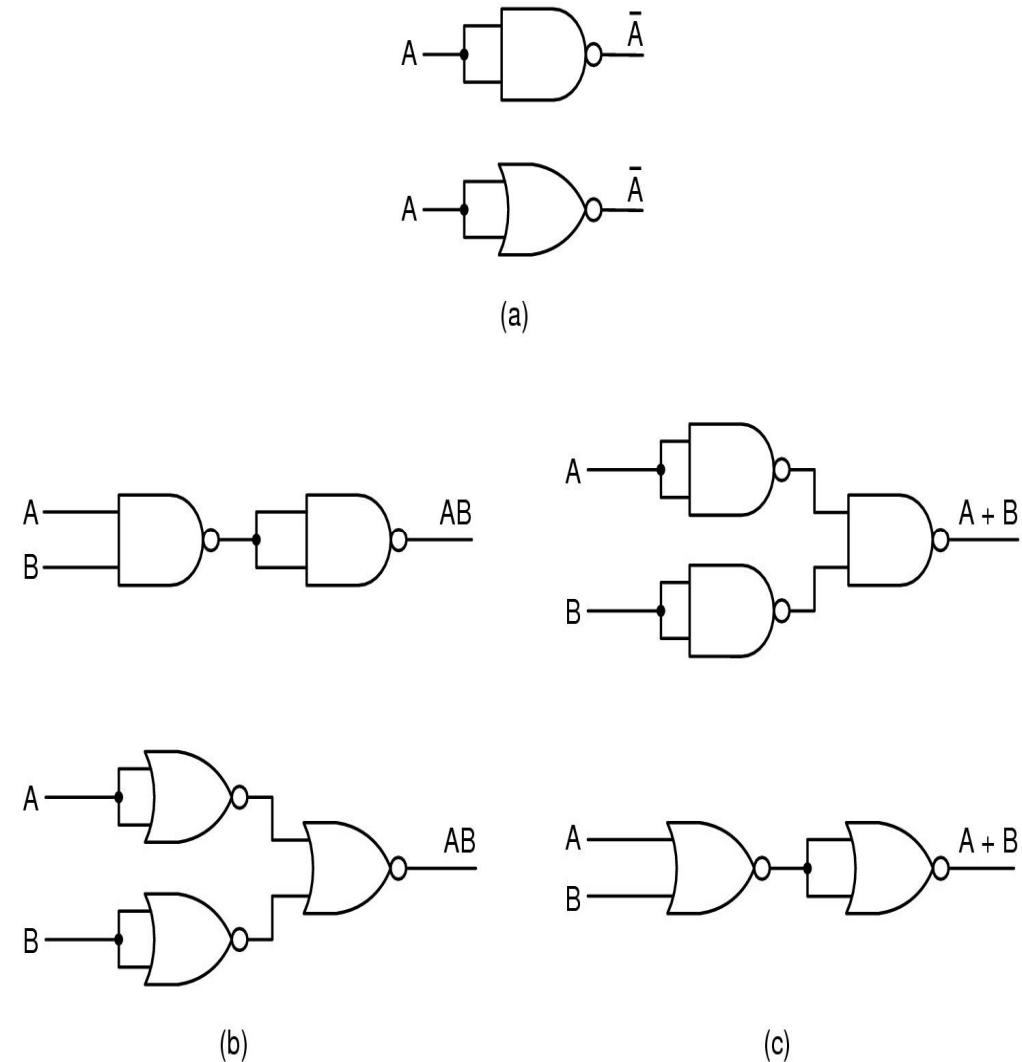


(b)

(c)

Implementazione di circuiti

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\bar{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$



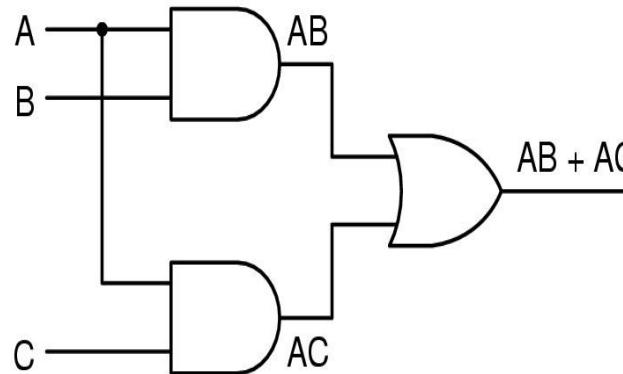
Diapositiva 32

DI0 Intenzionalmente bianco per dimostrare le relazioni, se avanza tempo

Drago Idilio; 2022-04-02T20:32:31.873

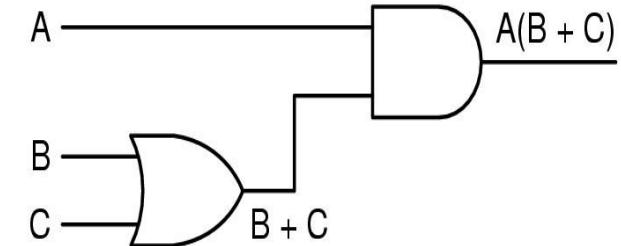
Implementazione di circuiti

- Equivalenza di circuiti:
esistono più circuiti che
realizzano la stessa
funzione booleana
- È importante trovare
quella più semplice nel
senso del **minor numero
di porte**
- Usare a tal fine le
proprietà dell'algebra
Booleana



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

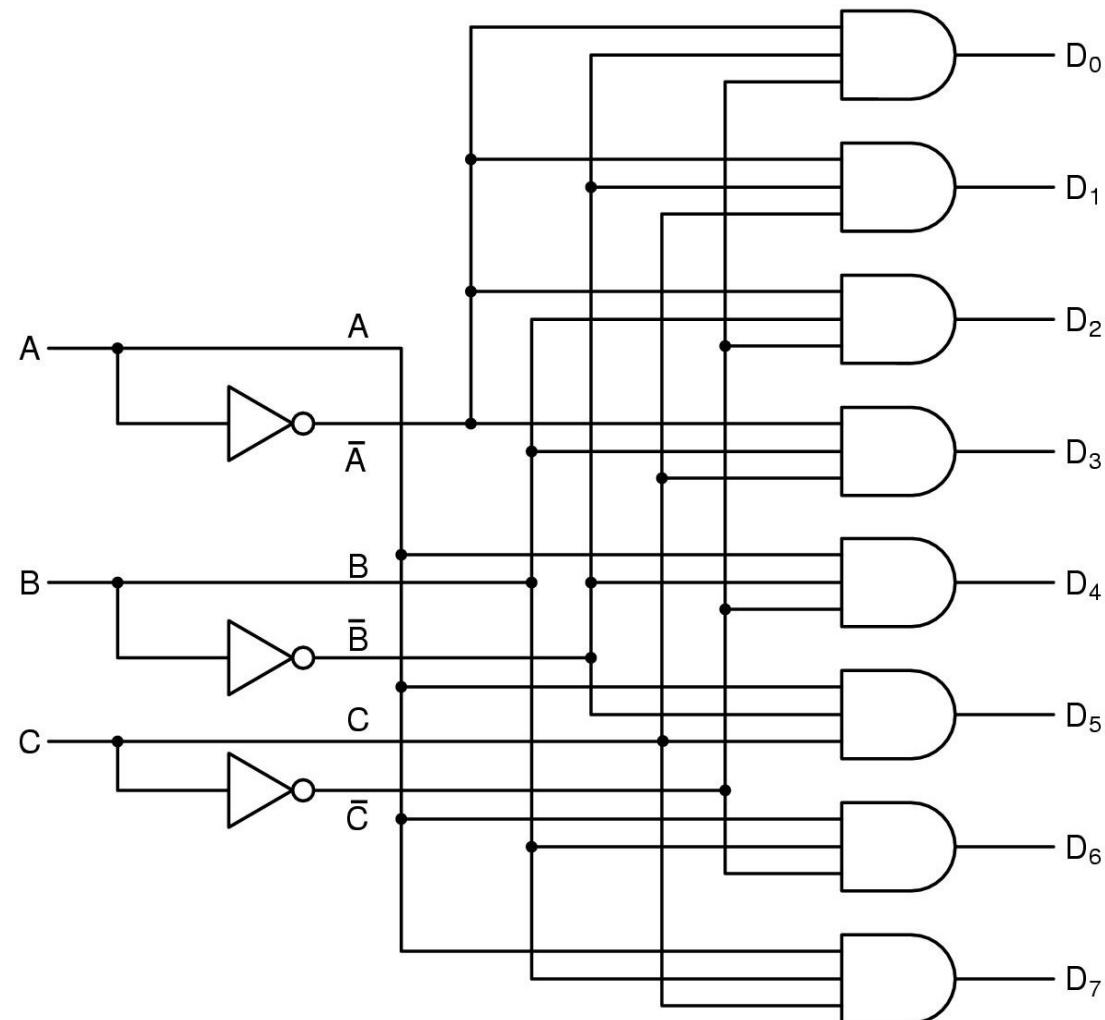
(b)

Circuiti di base

- Circuiti integrati (IC) o **chip**
- I chip si possono classificare in:
 - SSI (*Small Scale Integrated*): da 1 a 10 transistor
 - MSI (*Medium Scale Integrated*): da 10 a 100 transistor
 - LSI (*Large Scale Integrated*): da 100 a 100.000 transistor
 - VLSI (*Very Large Scale Integrated*): più di 100.000 transistor
 - ULSI (*Ultra Large Scale Integrated*): fino a 10 milioni di transistor

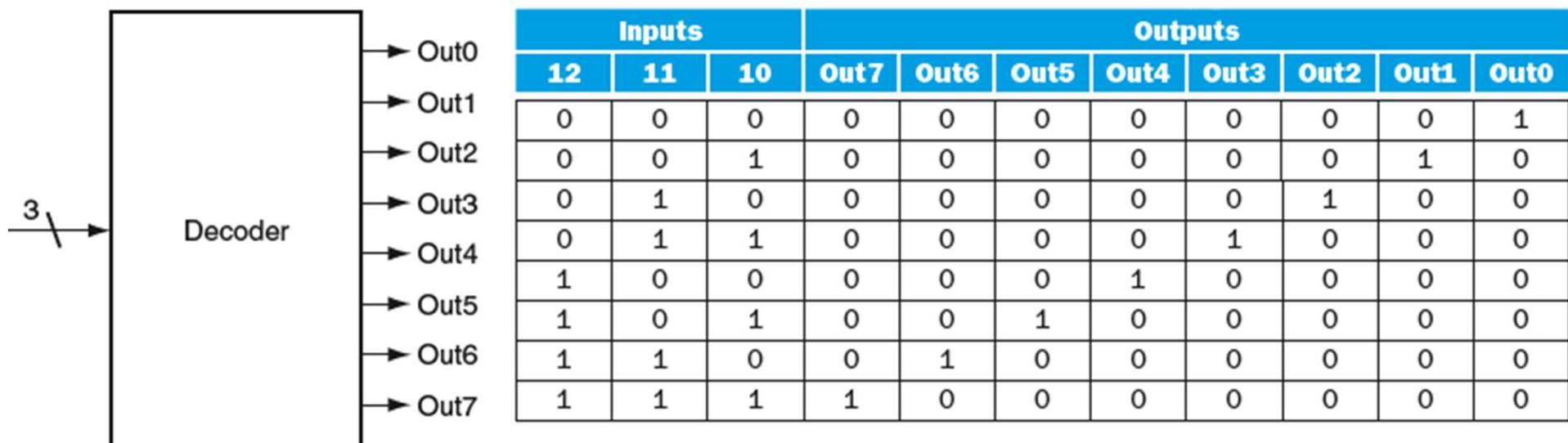
Circuiti combinatori

- Circuito combinatorio:
l'output viene
**determinato solo dagli
input**
- Convenzione: l'incrocio
tra due linee non implica
alcuna connessione a
meno che non sia
presente il simbolo • nel
punto di intersezione



Circuiti combinatori: decoder

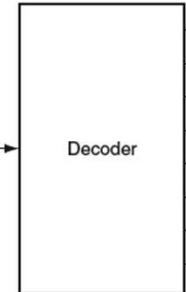
- **Decoder:** prende un numero di n bit come ingresso e lo usa per selezionare (mettere a 1, asserire) una delle 2^n linee di uscita
- Può essere utilizzato per attivare una certa componente (vedi ALU più avanti), oppure un banco di memoria, ecc.
- Interpretiamo gli ingressi A B C (o I₂ I₁ I₀) come le cifre di un numero in base 2 con A (I₂) quella più significativa



a. A 3-bit decoder

b. The truth table for a 3-bit decoder

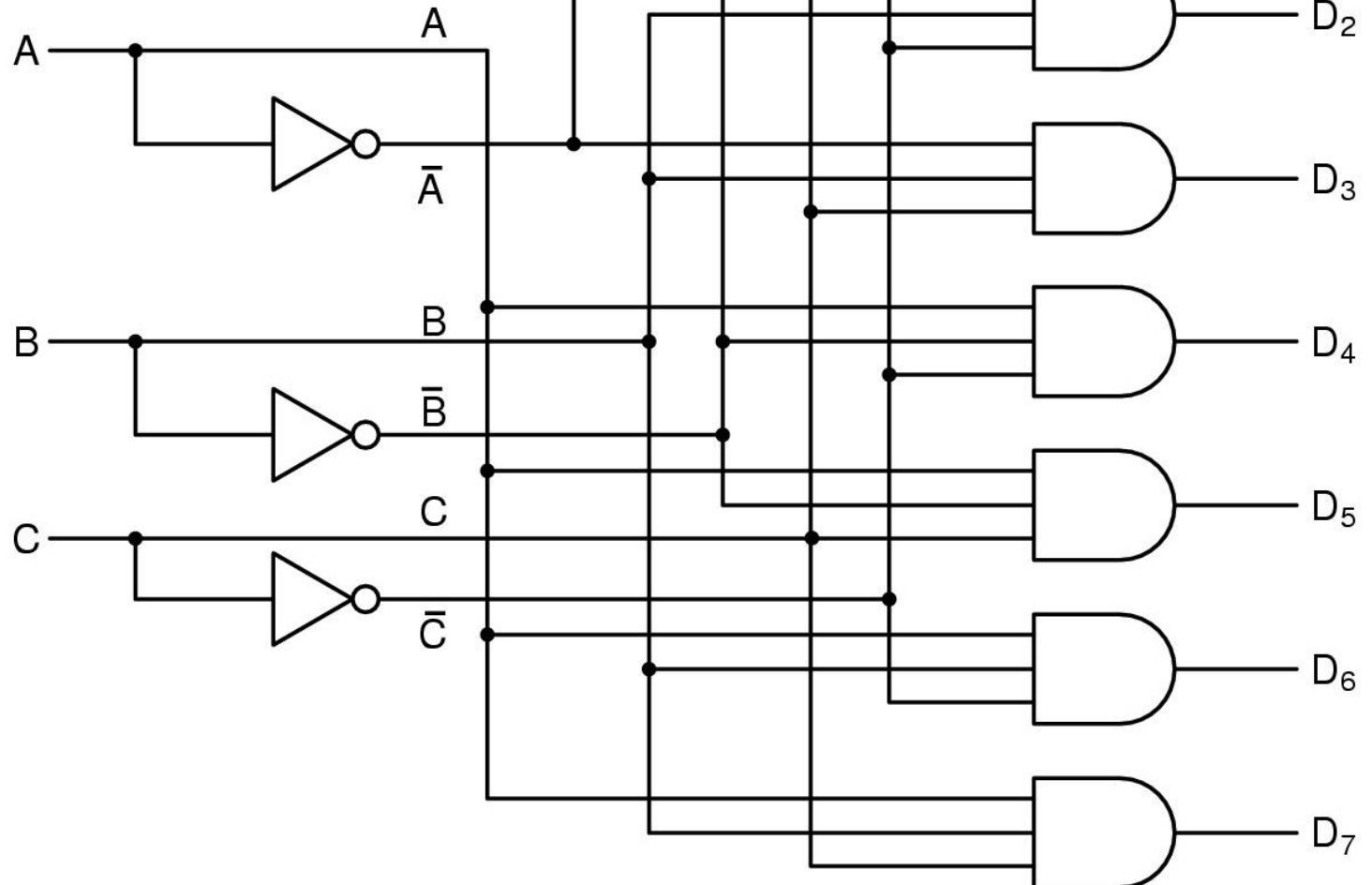
Circuiti combinatori: decoder



a. A 3-bit decoder

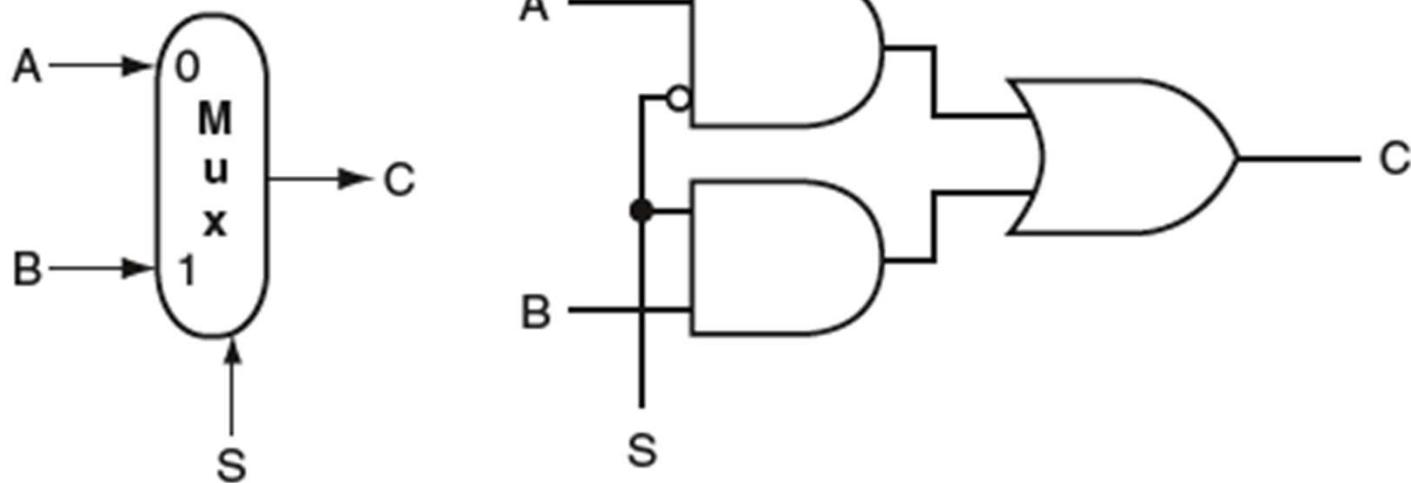
Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

b. The truth table for a 3-bit decoder



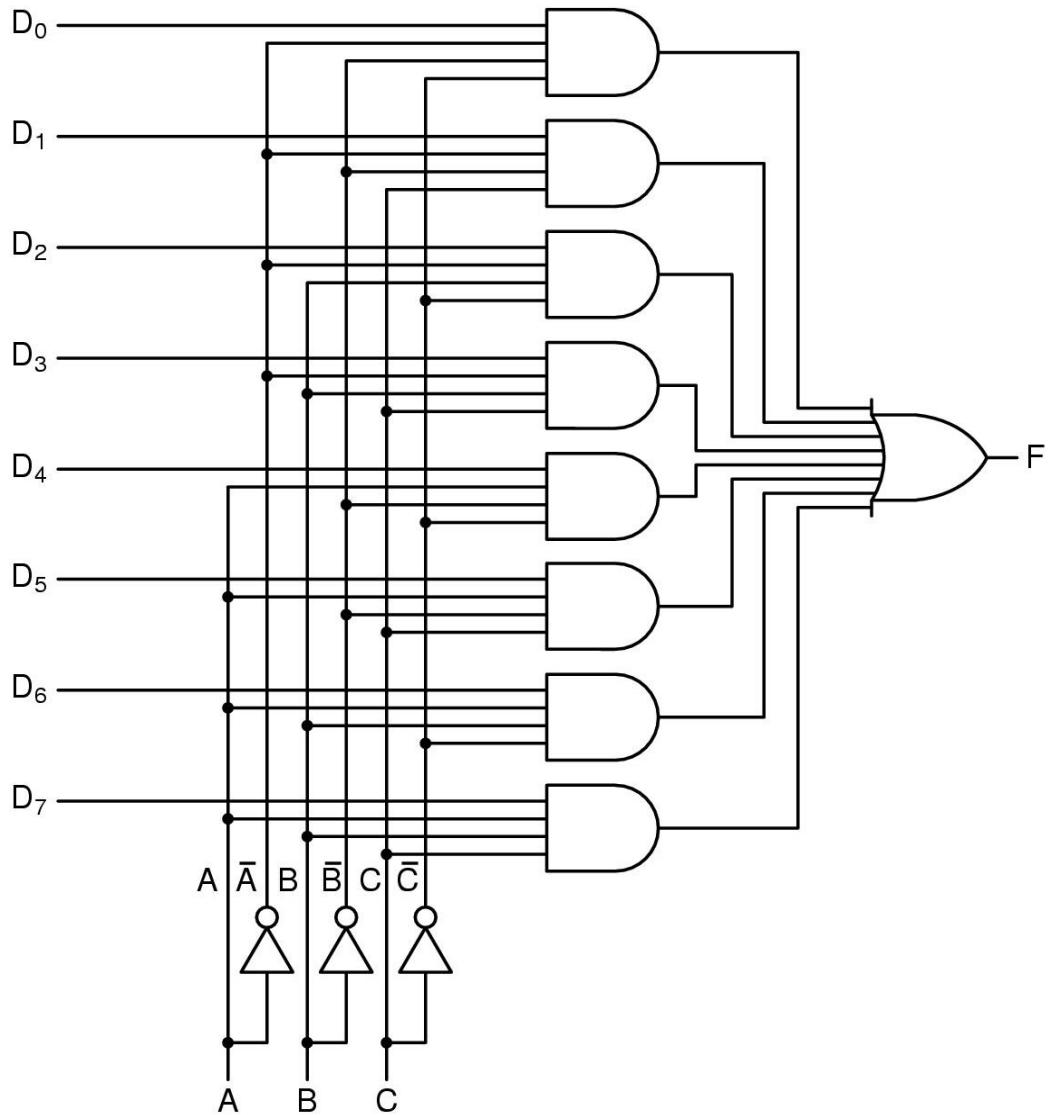
Circuiti combinatori: multiplexer

- **Multiplexer (o selettore):**
2 ingressi, 1 uscita e 1 ingresso di controllo
- La linea di controllo determina quale dei 2 ingressi deve essere selezionato per essere inviato all'uscita
- Esempio con 2 ingressi (A e B), un uscita (C) ed un ingresso di controllo (S)

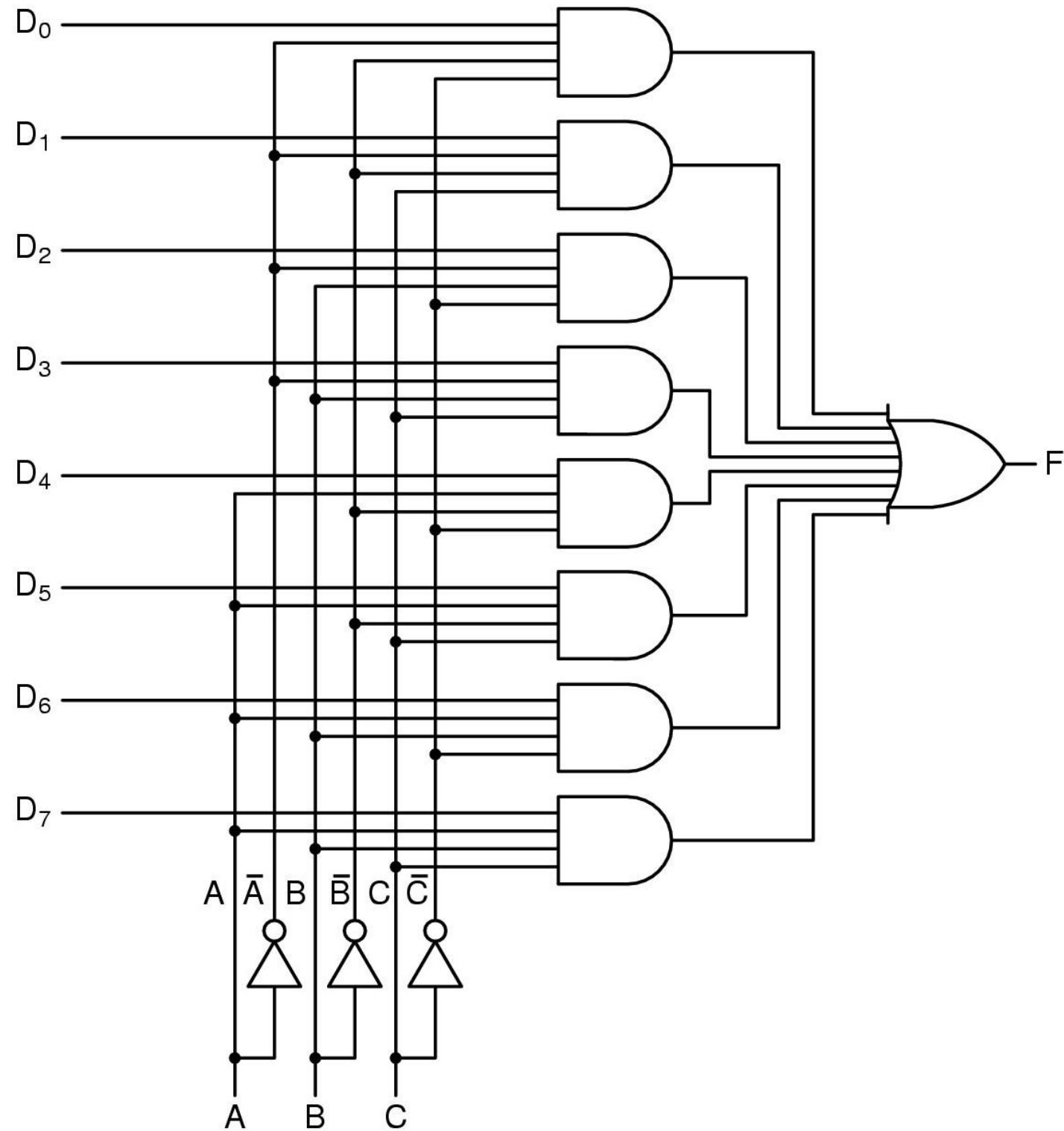


Circuiti combinatori: multiplexer

- **Multiplexer:** In generale, possiamo avere 2^n ingressi, 1 uscita e n ingressi di controllo
- Le linee di controllo determinano quale dei 2^n ingressi deve essere selezionato per essere inviato all'uscita
- Esempio con 8 ingressi ($D_0,.. D_7$), un uscita (F) e tre ingressi di controllo (A, B e C)



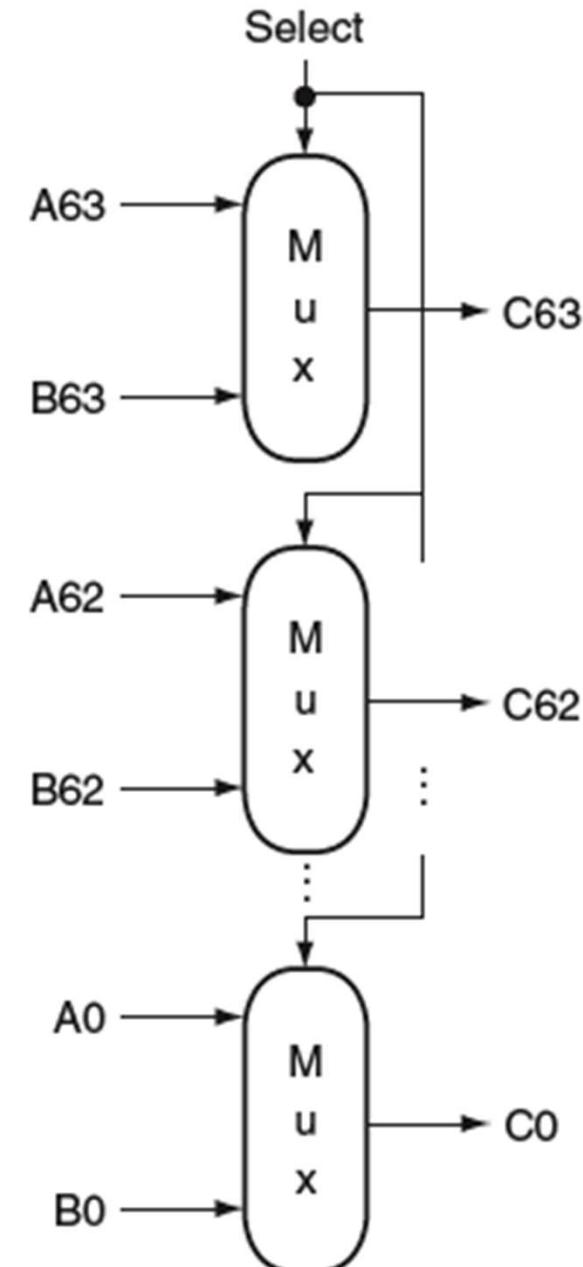
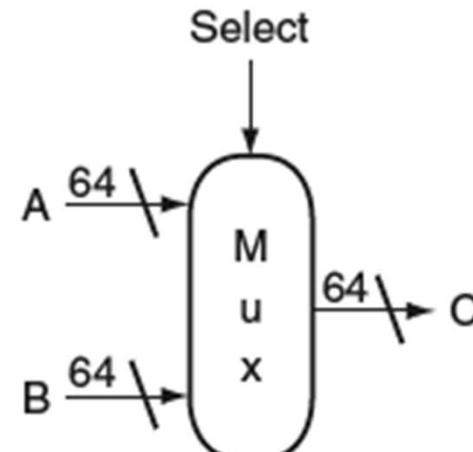
Multiplexer



Un multiplexer è composto da un decoder dove ogni AND accoglie un ingresso e l'OR finale

Circuiti combinatori: multiplexer

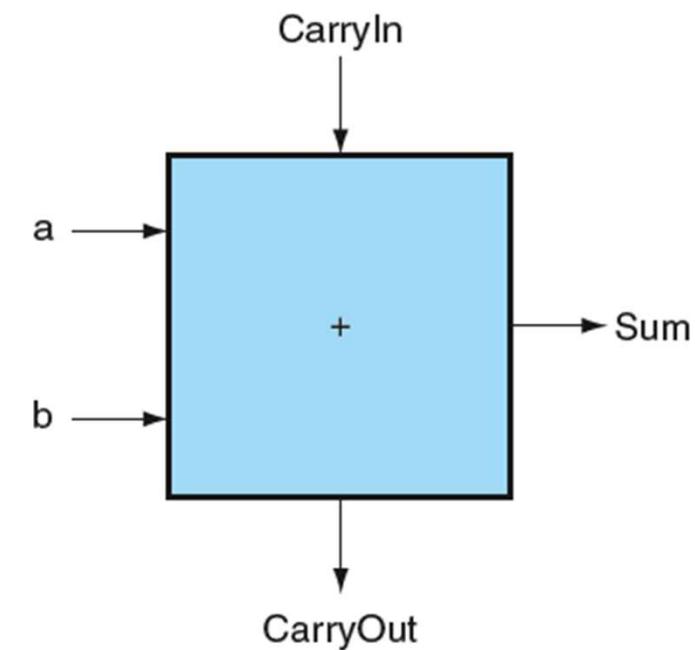
- **Multiplexer (o selettore):**
2 ingressi da 64 linee ciascuno, 1 uscita da 64 linee e 1 ingresso di controllo
- La linea di controllo determina quale dei 2 ingressi da 64 linee deve essere selezionato per essere inviato all'uscita da 64 linee
- Esempio con 2 ingressi (A e B), un uscita (C) ed un ingresso di controllo (Select)



Circuiti numerici: addizionatori

- riceve in ingresso due bit (cifre in base 2) da sommare, **a** e **b** nello schema
- riceve in ingresso un bit (cifra in base 2) di riporto, **CarryIn** nello schema
- restituisce un bit (cifra in base 2) in uscita che rappresenta il risultato, **Sum** nello schema
- restituisce un bit (cifra in base 2) in uscita che rappresenta il riporto, **CarryOut** nello schema

Inputs			Outputs	
a	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



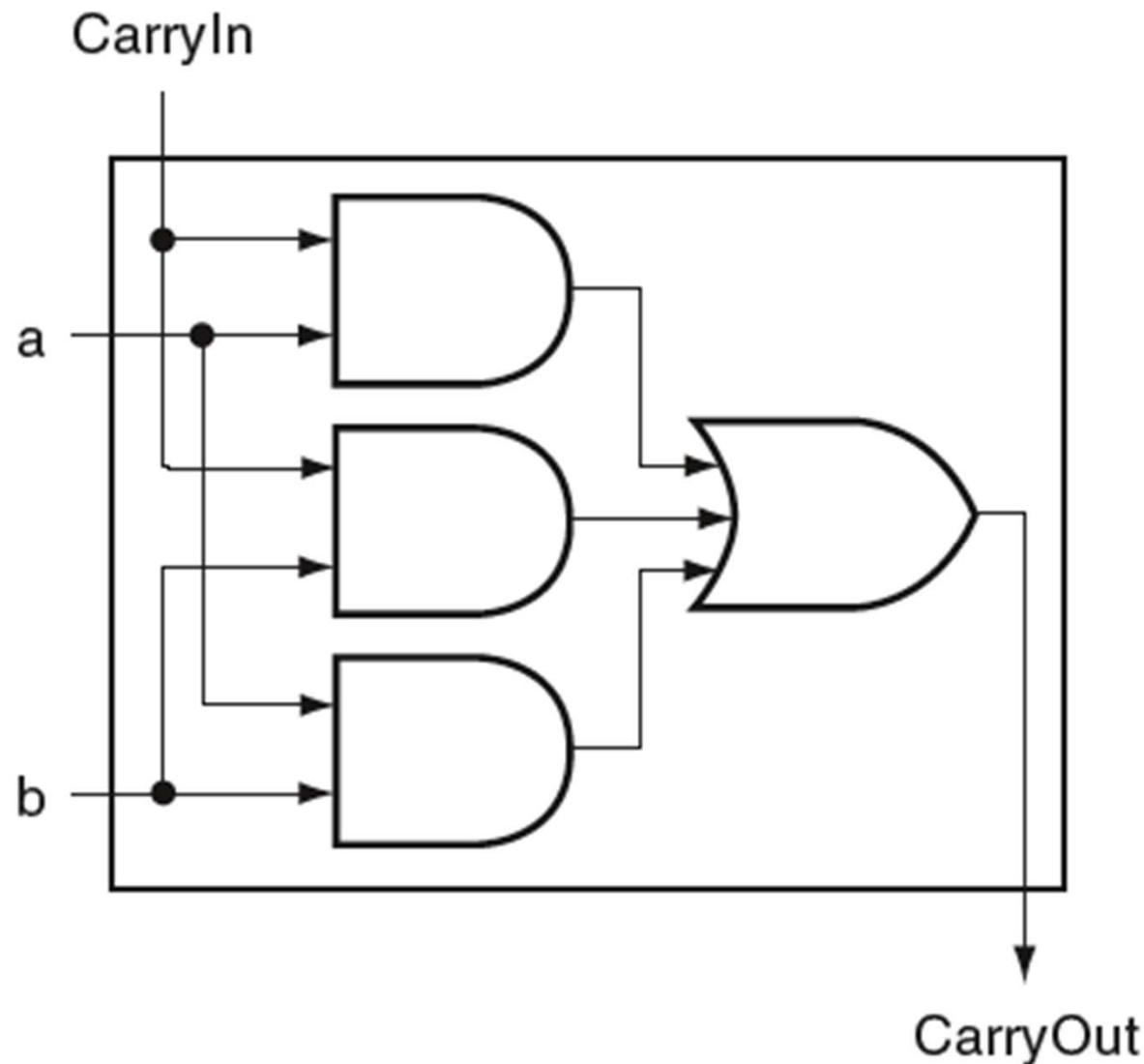
Circuiti numerici: addizionatori

$$C_{Out} = (b \ CIn) + (a \ C_{In}) + (a \ b)$$

Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_{\text{two}}$
0	0	1	0	1	$0 + 0 + 1 = 01_{\text{two}}$
0	1	0	0	1	$0 + 1 + 0 = 01_{\text{two}}$
0	1	1	1	0	$0 + 1 + 1 = 10_{\text{two}}$
1	0	0	0	1	$1 + 0 + 0 = 01_{\text{two}}$
1	0	1	1	0	$1 + 0 + 1 = 10_{\text{two}}$
1	1	0	1	0	$1 + 1 + 0 = 10_{\text{two}}$
1	1	1	1	1	$1 + 1 + 1 = 11_{\text{two}}$

Circuiti numerici: addizionatori

$$C_{Out} = (b \ CIn) + (a \ C_{In}) + (a \ b)$$



Circuiti numerici: addizionatori

$$\text{Sum} = (a \cdot \bar{b} \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot \bar{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

$$Sum = (a \text{ } XOR \text{ } b) \text{ } XOR \text{ } C_{In}$$

Inputs			Outputs	
a	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Circuiti numerici: addizionatori

$$C_{Out} = (\bar{a} b CIn) + (a \bar{b} C_{In}) + (a b \overline{C_{In}}) + (a b C_{In})$$

$$C_{Out} = (\bar{a} b CIn) + (a \bar{b} CIn) + a b$$

$$C_{Out} = C_{In}(\bar{a} b + a \bar{b}) + a b$$

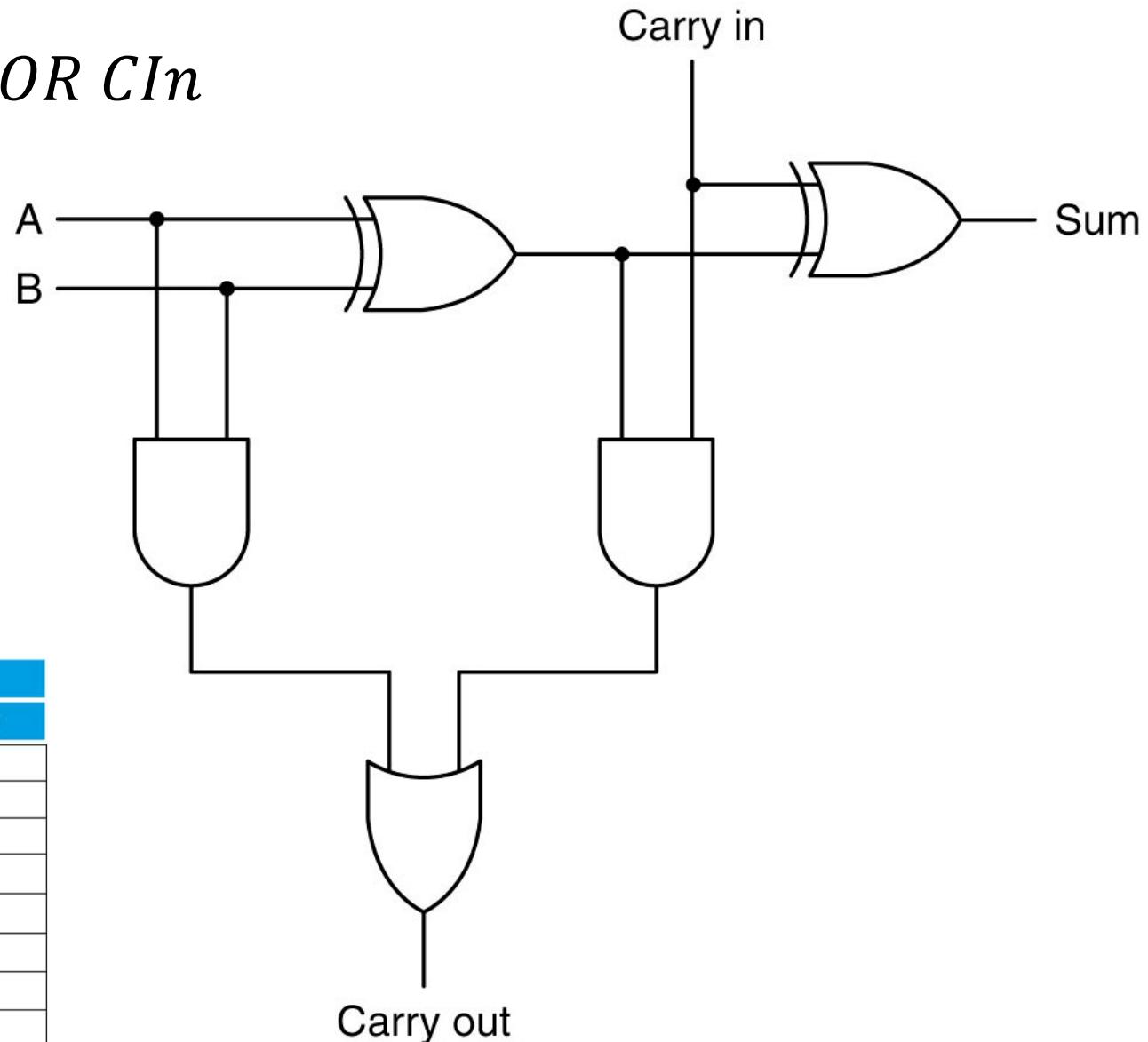
$$C_{Out} = C_{In}(a \text{ } XOR \text{ } b) + a b$$

Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_{\text{two}}$
0	0	1	0	1	$0 + 0 + 1 = 01_{\text{two}}$
0	1	0	0	1	$0 + 1 + 0 = 01_{\text{two}}$
0	1	1	1	0	$0 + 1 + 1 = 10_{\text{two}}$
1	0	0	0	1	$1 + 0 + 0 = 01_{\text{two}}$
1	0	1	1	0	$1 + 0 + 1 = 10_{\text{two}}$
1	1	0	1	0	$1 + 1 + 0 = 10_{\text{two}}$
1	1	1	1	1	$1 + 1 + 1 = 11_{\text{two}}$

Circuiti numerici: addizionatori con XOR

$$C_{Out} = CIn(a \text{ } XOR \text{ } b) + ab$$

$$Sum = (a \text{ } XOR \text{ } b) \text{ } XOR \text{ } CIn$$



Inputs			Outputs	
a	b	Carryin	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

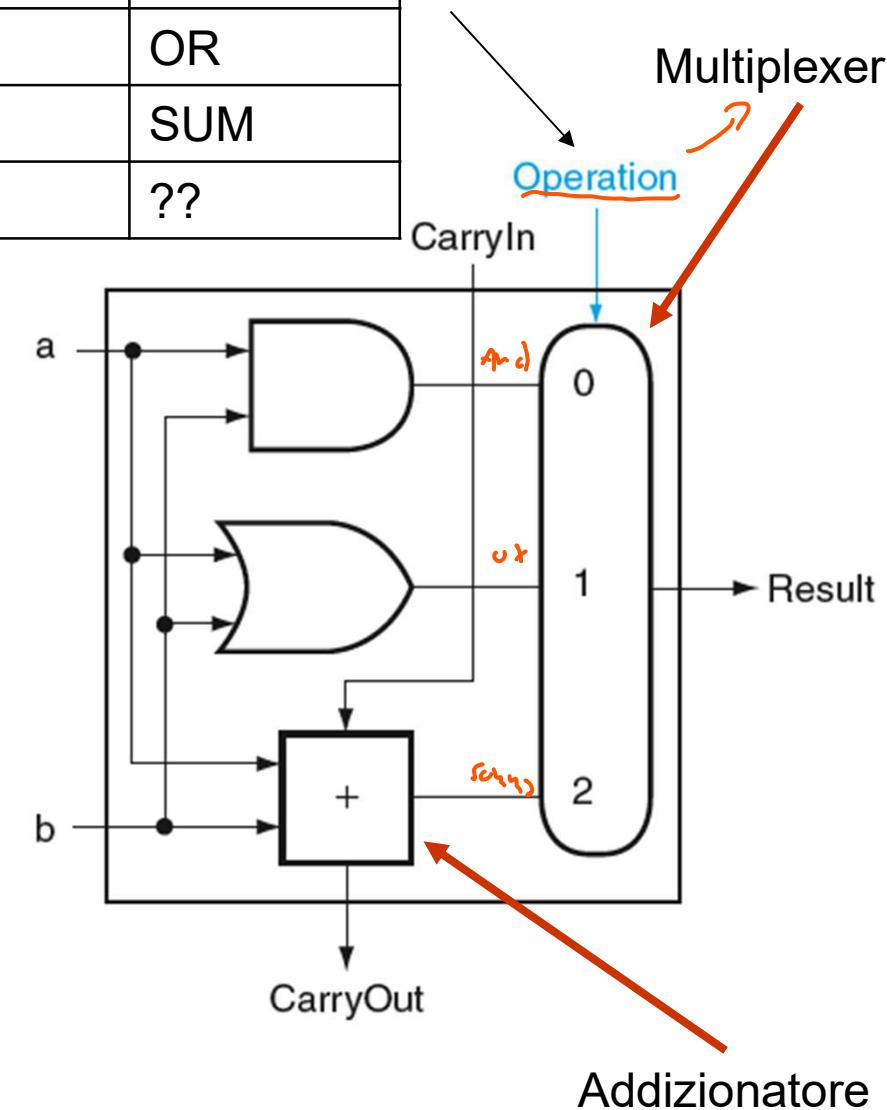
non si tolgono le

nuova lezione

RISC-V: ALU ad 1 bit

- Dati a e b è in grado di calcolare:
 - $\text{Result} = a \text{ AND } b$
 - $\text{Result} = a \text{ OR } b$
 - $\text{Result} = a + b$ (somma)
- Un ingresso di controllo (**Operation**) seleziona l'operazione desiderata. Dovendo selezionare 3 possibili ingressi in realtà abbiamo 2 linee **Operation** di controllo.
- Un terzo ingresso fornisce in **CarryIn** per la somma
- Una seconda uscita rappresenta il **CarryOut**

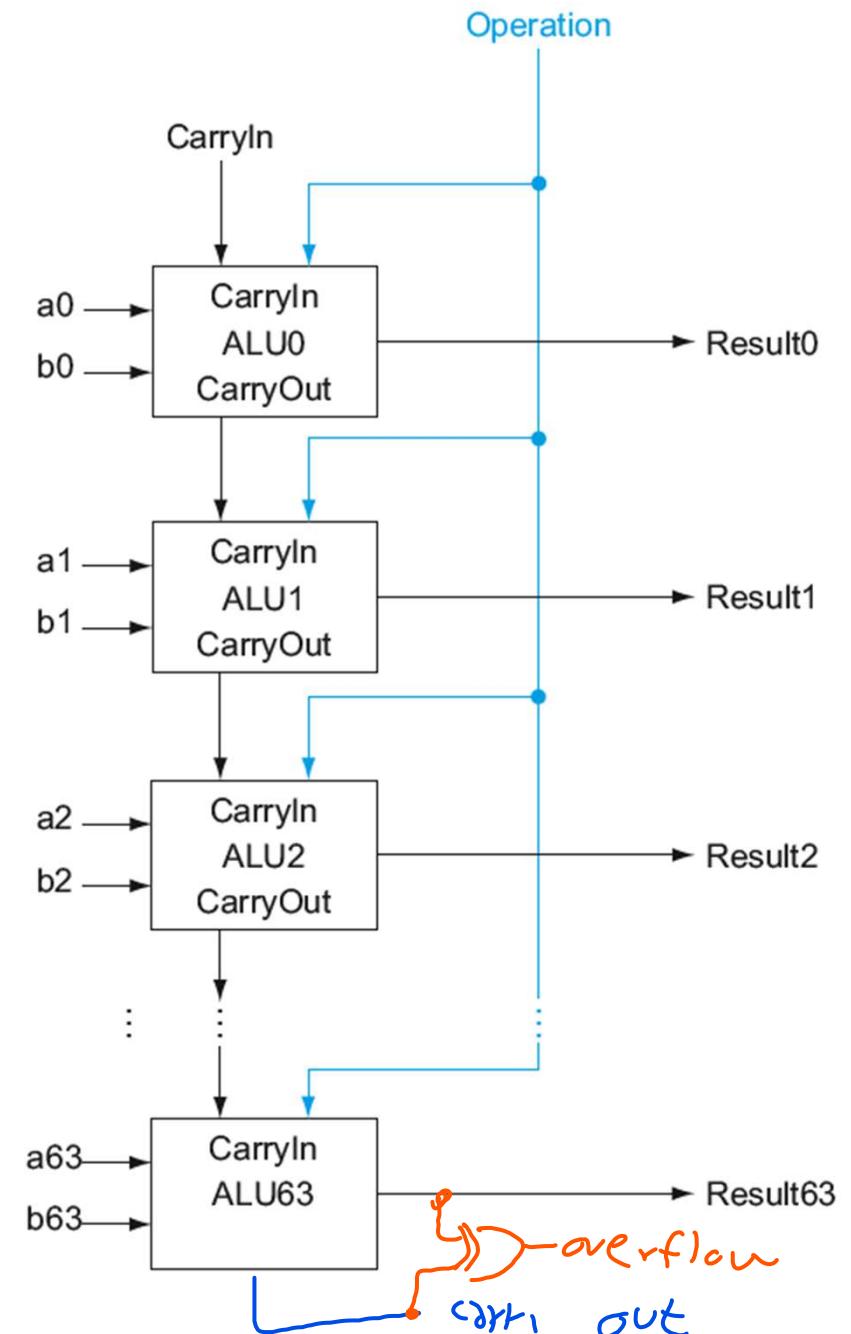
Operation	Function
00	AND
01	OR
10	SUM
11	??



RISC-V: ALU a 64 bit

- Dati **a** e **b** su 64 bit è in grado di calcolare:
 - Result = a AND b (bit a bit)**
 - Result = a OR b (bit a bit)**
 - Result = a + b (somma su 64 cifre)**
- Un ingresso di controllo (**Operation**) seleziona l'operazione desiderata.
Dovendo selezionare 3 possibili ingressi in realtà abbiamo 2 linee **Operation** di controllo.

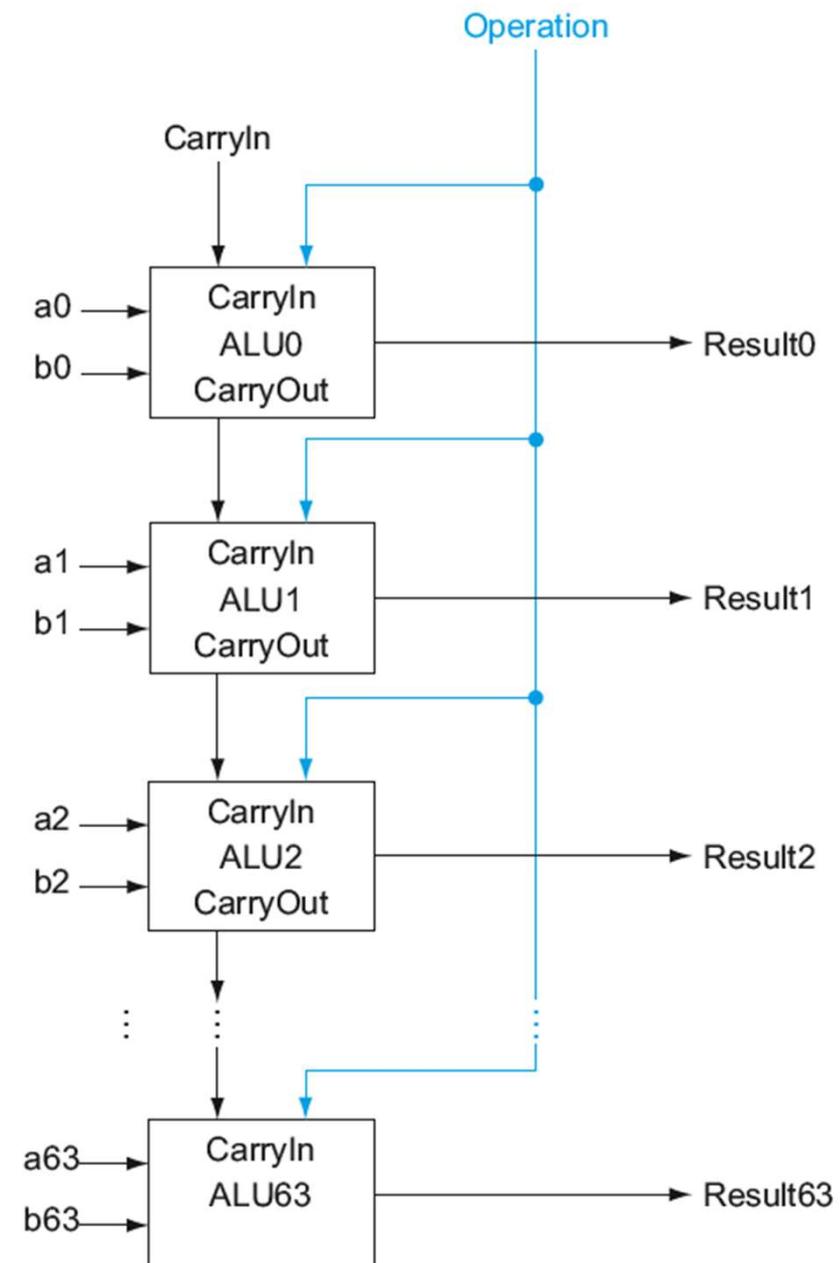
Operation	Function
00	AND
01	OR
10	SUM
11	??



RISC-V: ALU a 64 bit

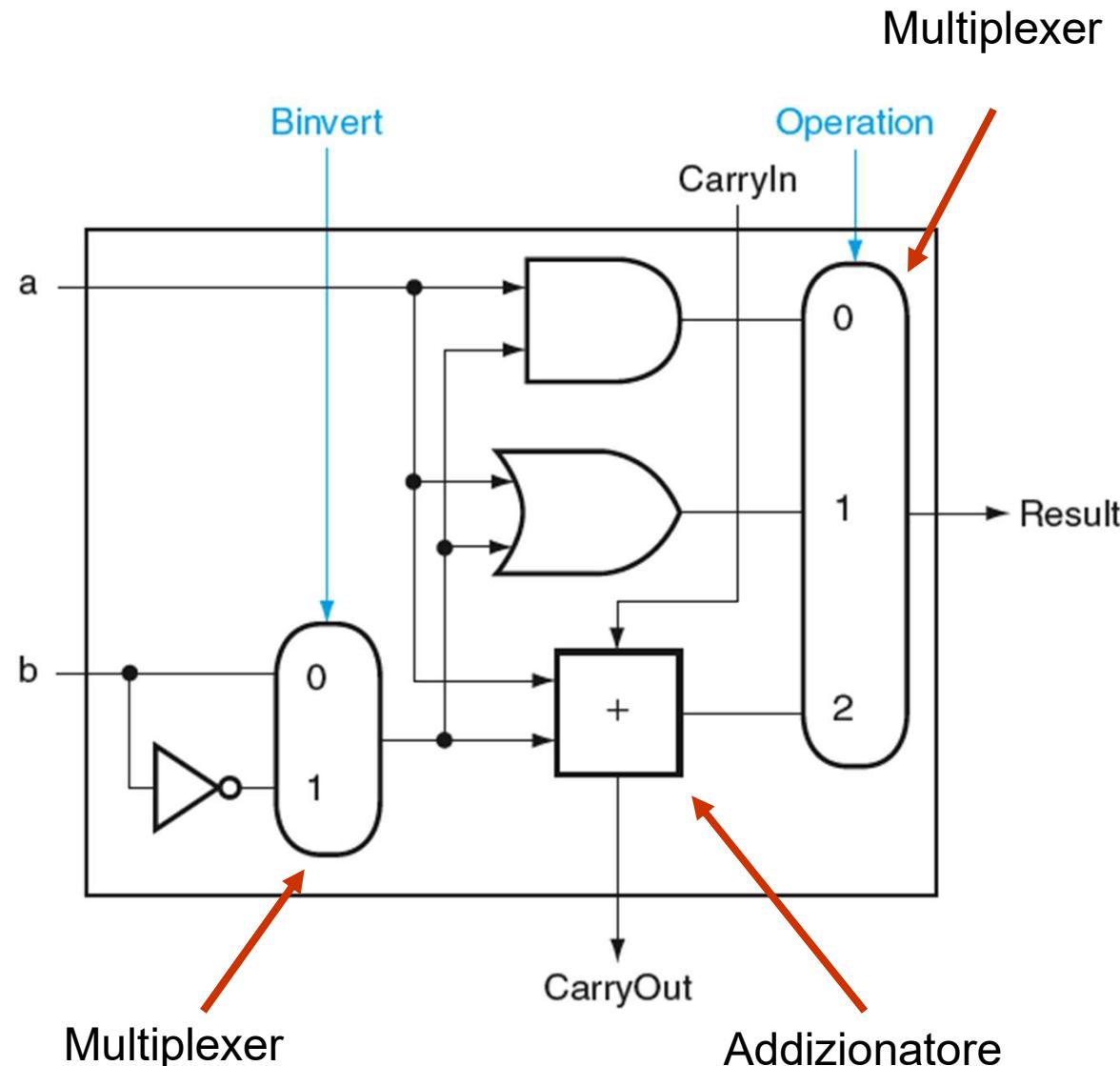
- Una ALU per operandi **a** e **b** ad n bit si ottiene utilizzando n ALU ad 1 bit
- Ogni ALU esegue l'operazione sulla coppia di bit degli operandi **a** e **b** nella stessa posizione (bit slice)
- Per sommare due operandi **a** e **b** di n bit il **CarryOut** dell'ALU per il bit in posizione i diventa il **CarryIn** del bit in posizione $i + 1$
- Il **CarryIn** del bit meno significativo (a_0 e b_0) può essere usato come segnale di incremento per calcolare **$a + b + 1$** (servirà per le sottrazioni)

Operation	Function
00	AND
01	OR
10	SUM
11	??



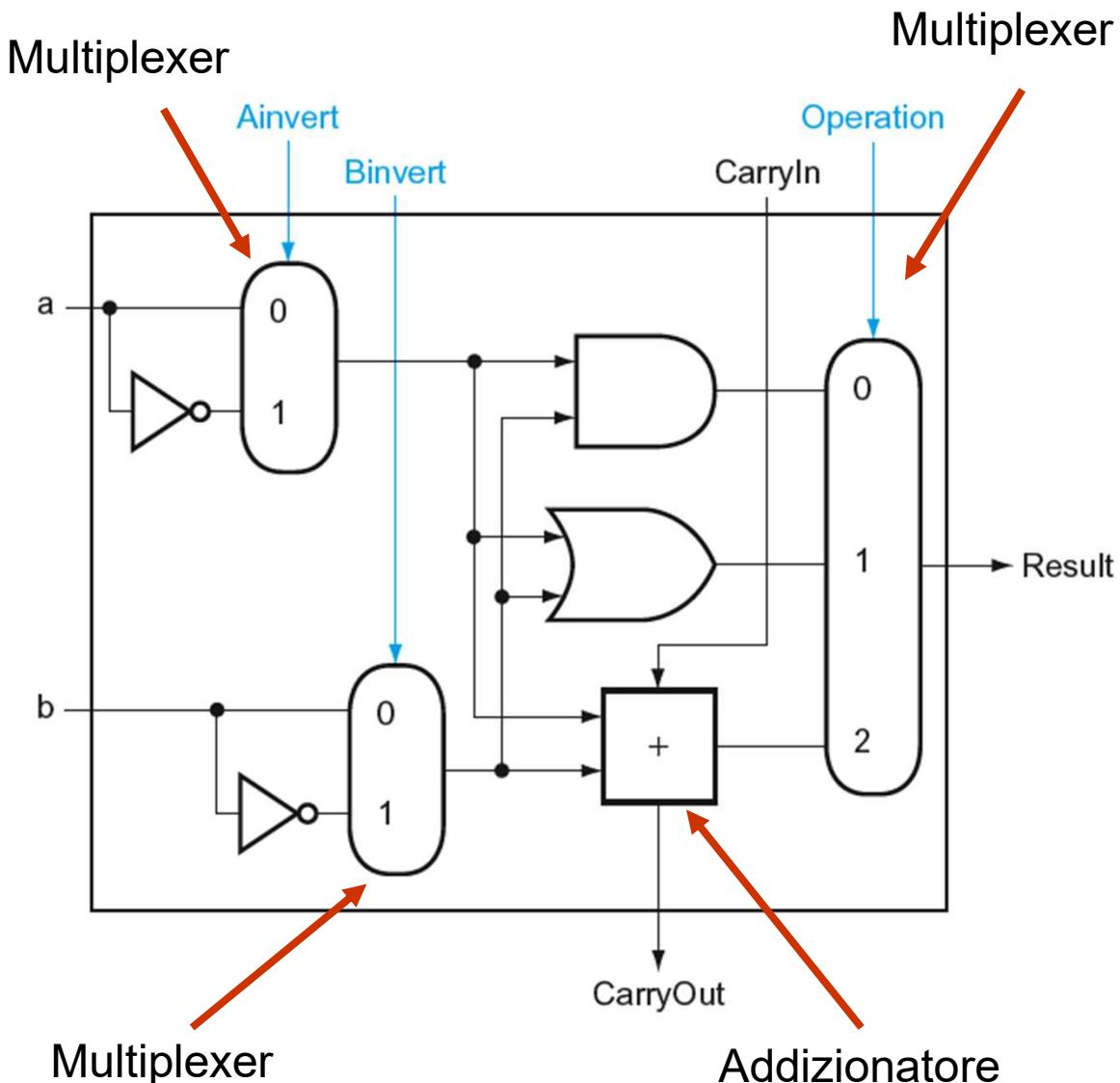
RISC-V: ALU ad 1 bit (sub)

- Dati a e b è in grado di calcolare le funzioni precedenti con b o \bar{b}
- Un ingresso di controllo aggiuntivo (**Binvert**) seleziona l'operazione desiderata
- Serve, ad esempio, per le sottrazioni ($a - b$)
 - Ricordare il complemento a 2
 - Con l'uso di \bar{b}
 - CarryIn = 1** per il bit meno significativo



RISC-V: ALU ad 1 bit (nor)

- Dati **a** e **b** è in grado di calcolare le funzioni precedenti con **$a \text{ o } \bar{a}$**
- Un ingresso di controllo aggiuntivo (**Ainvert**) seleziona l'operazione desiderata
- Serve, ad esempio, per funzioni **NOR: $(a \text{ OR } b)$** con l'uso di **$(\bar{a} \text{ AND } \bar{b})$**



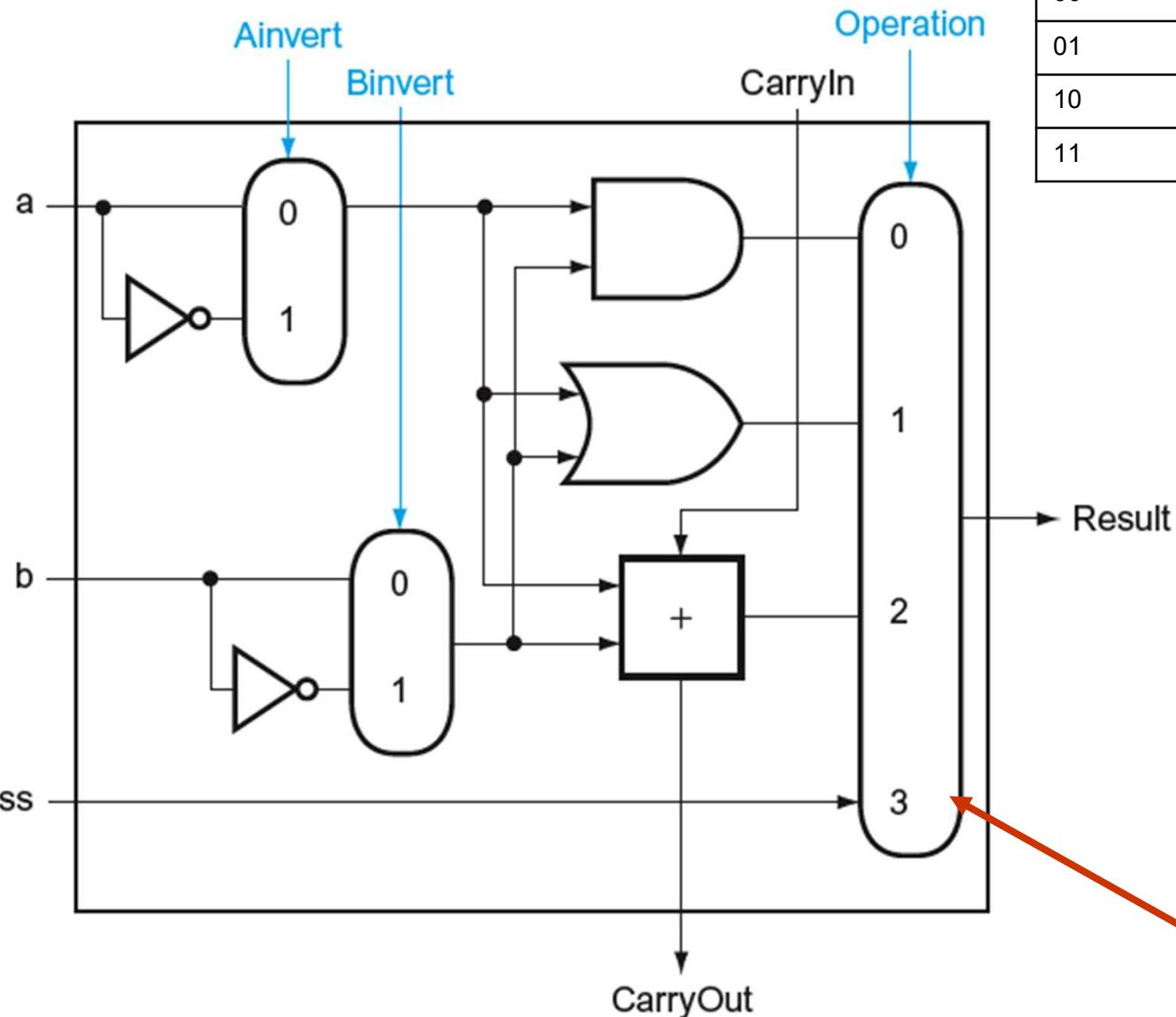
RISC-V: ALU per slt

- L'istruzione Set on Less Than (slt) restituisce 1 se $rs1 < rs2$ e 0 altrimenti
- Il valore di tutti i bit in uscita tranne quello meno significativo deve essere 0
- Il valore del bit meno significativo dipende dal confronto
- Idea: **usare la sottrazione**
 - ($a - b < 0$) implica $a < b$
- Per vedere se ($a - b$) è negativo verifico se il **bit più significativo vale 1**

RISC-V: ALU per slt

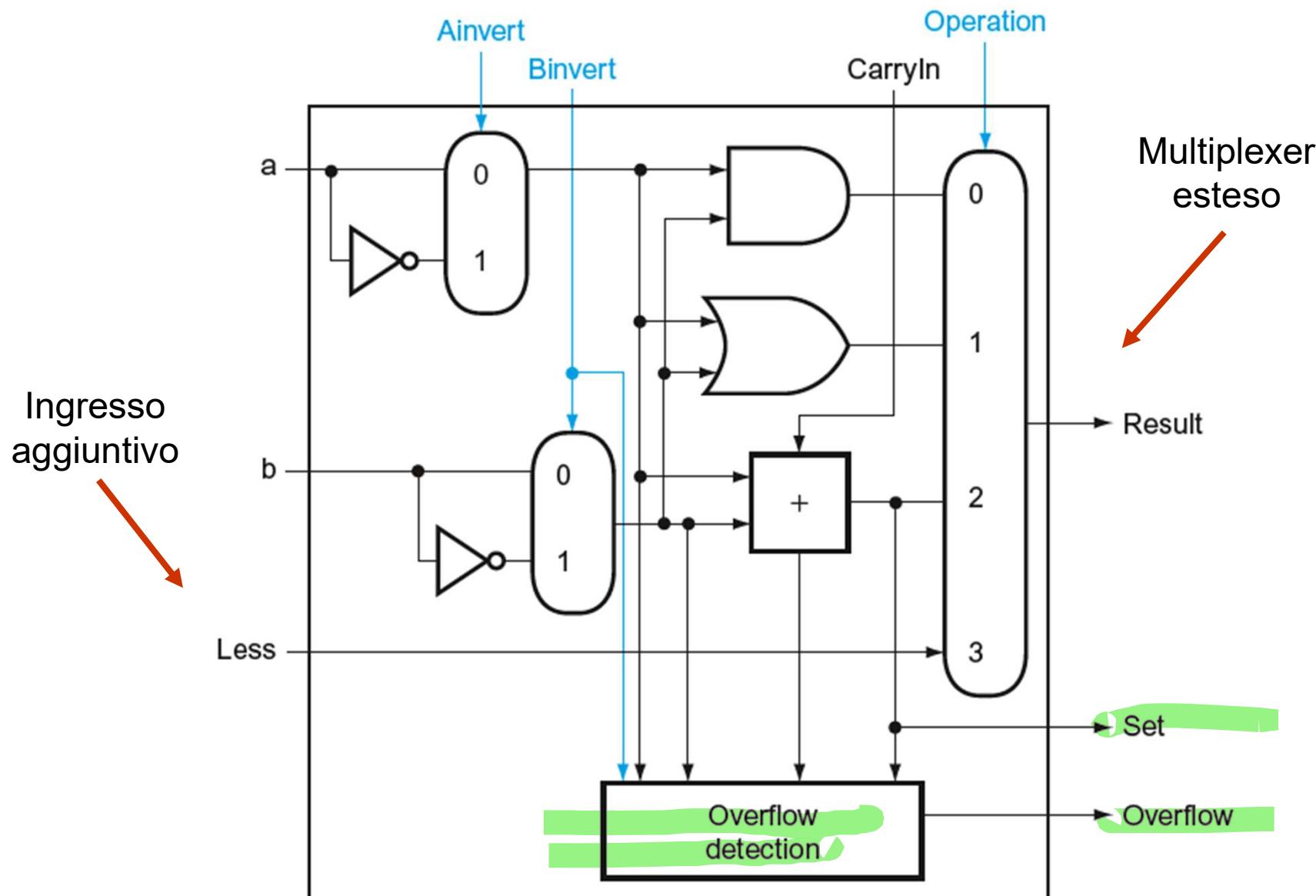
- ALU estesa per ognuno dei 63 bit meno significativi degli operandi

Operation	Function
00	AND
01	OR
10	SUM
11	LESS



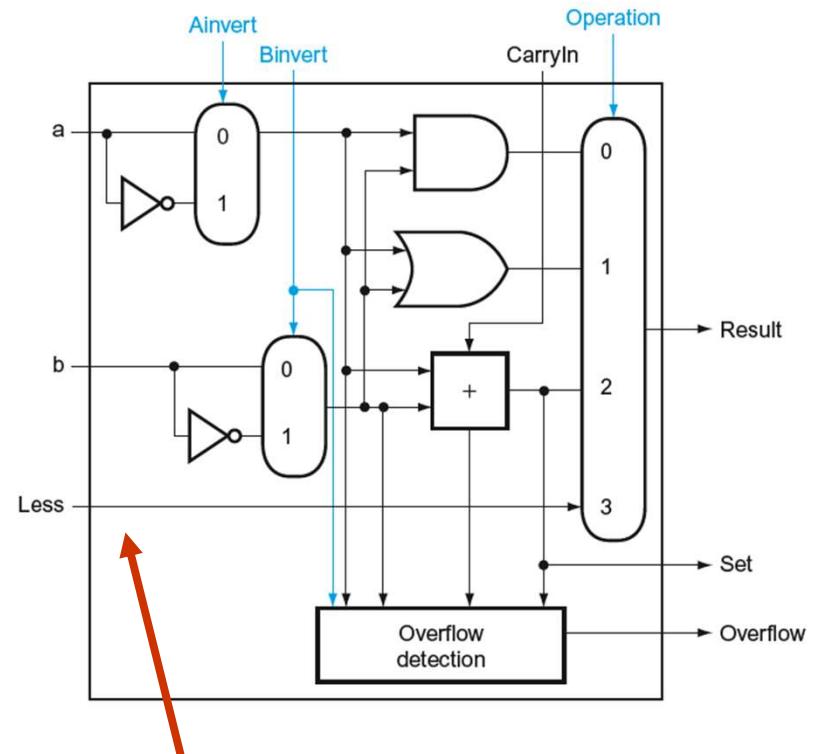
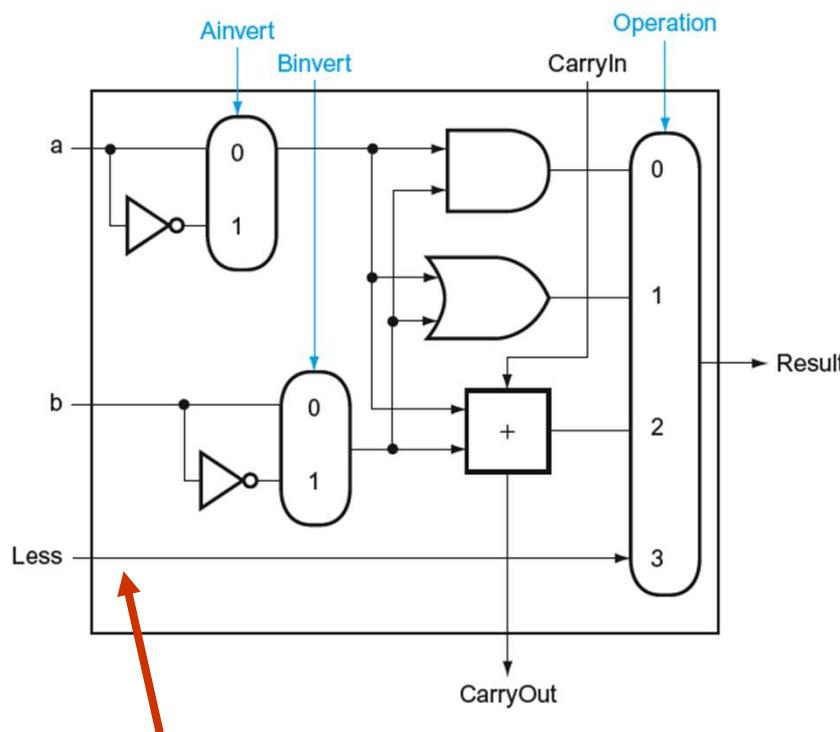
RISC-V: ALU per slt

- ALU estesa per il 64-esimo bit, quello più significativo degli operandi



RISC-V: ALU per slt

- L'ingresso Less dell'ALU per i bit da 1 a 63 deve essere uguale a 0
- L'ingresso Less dell'ALU per il bit 0 (il meno significativo) deve essere uguale al bit di segno del risultato della differenza $a - b$
 - ovvero il bit della somma dell'ALU del bit 63
- Per questo l'ALU per il bit 63 ha un'uscita Set per il risultato dell'addizionatore

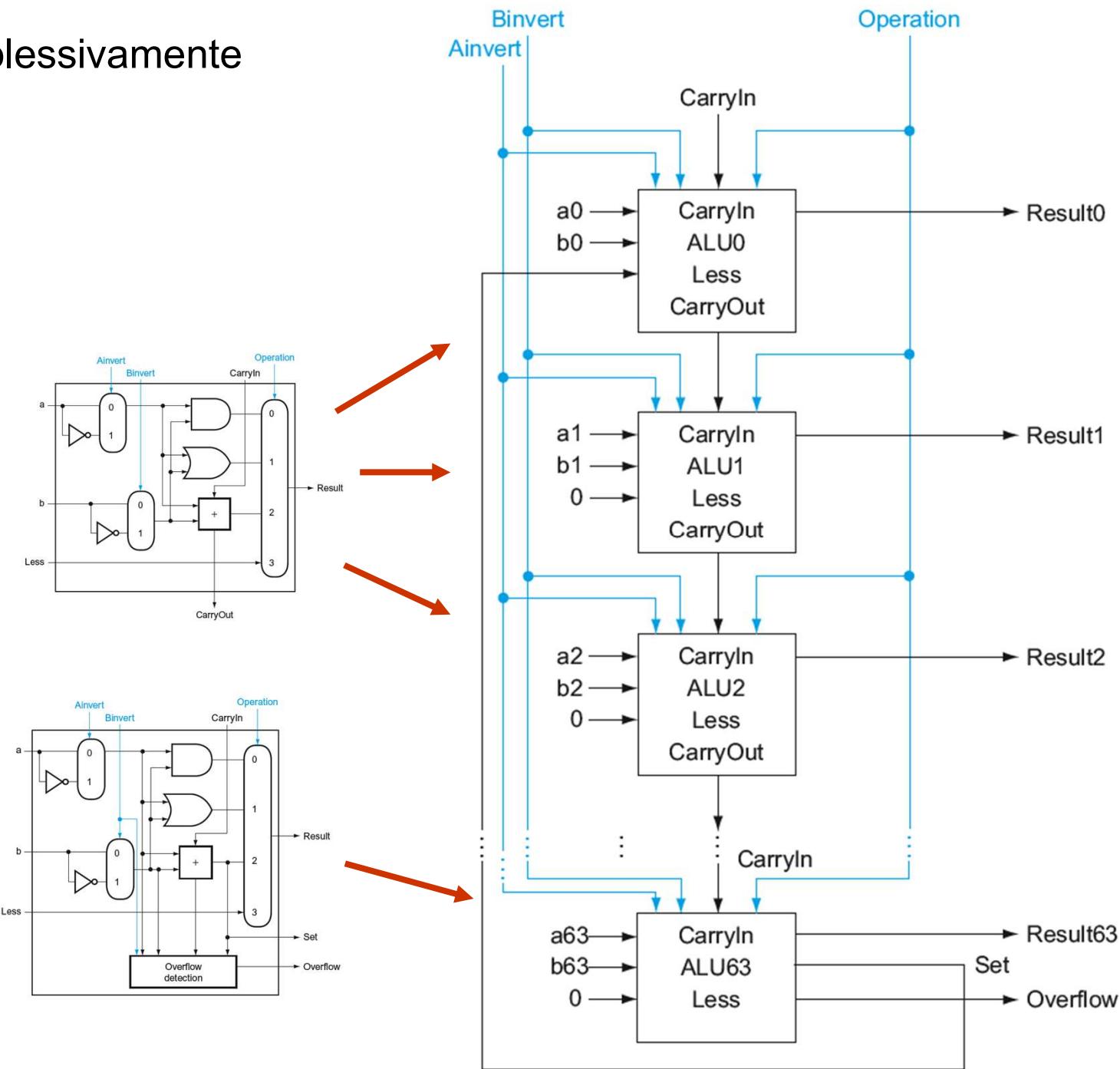


- Ingresso a 0 per tutti tranne che per il bit meno significativo

- Ingresso a 0

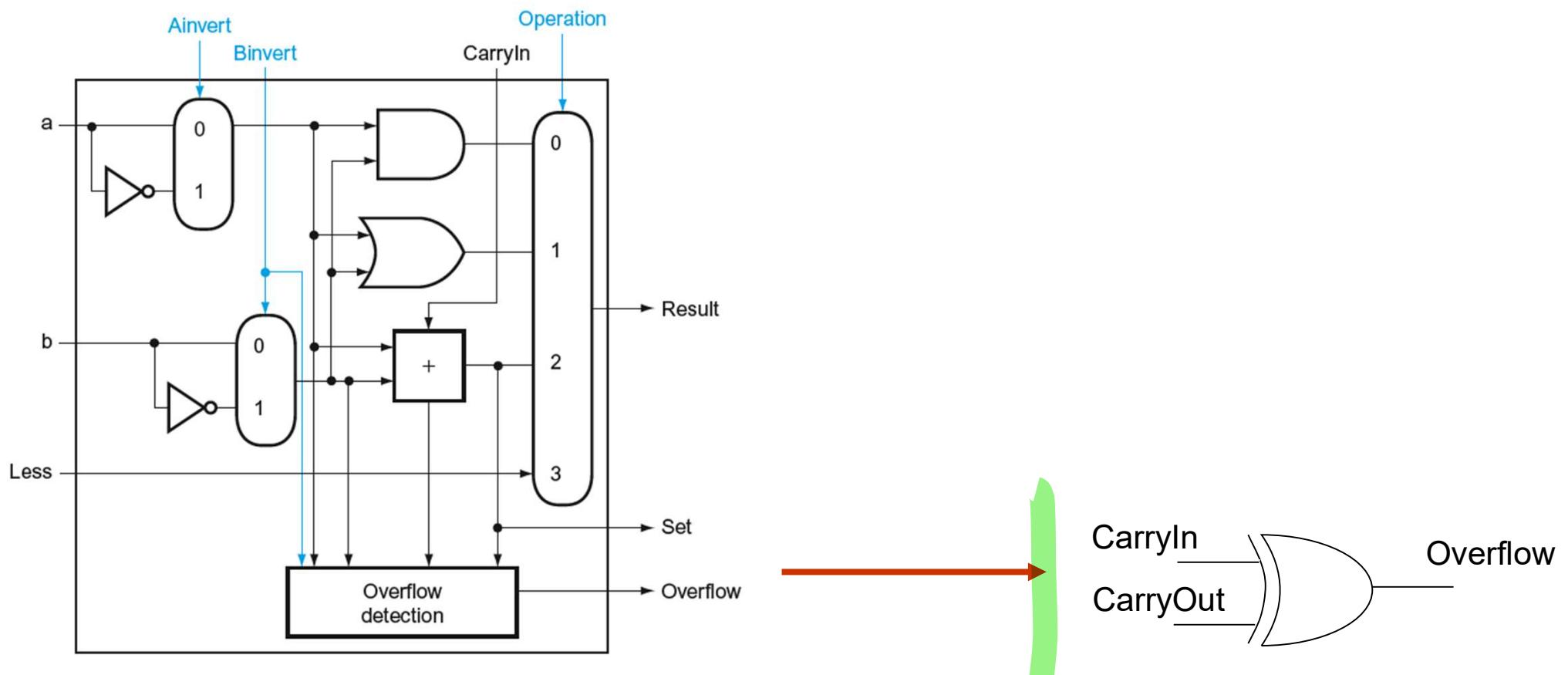
RISC-V: ALU per slt

- Complessivamente



RISC-V: overflow

- Figura: Nella somma o differenza di due interi con segno in complemento a due, abbiamo overflow se i due operandi hanno lo stesso segno ma il risultato ha segno opposto.
- Ricordate la teoria del complemento a 2? Si ha overflow quando il CarryIn ed il CarryOut del bit più significativo sono discordi.



Controllo operazioni dell'ALU

- Ogni volta che vogliamo che l'ALU esegua sottrazioni dobbiamo asserire (porre a 1) sia CarryIn sia Binvert

Operazione	Funzione	Binvert	CarryIn0
0	AND	0	0
1	OR	0	0
2	ADD	0	0
2	SUB	1	1
3	SLT	1	1

CarryIn0 = Binvert

Usiamo una sola linea chiamata **Bnegate**

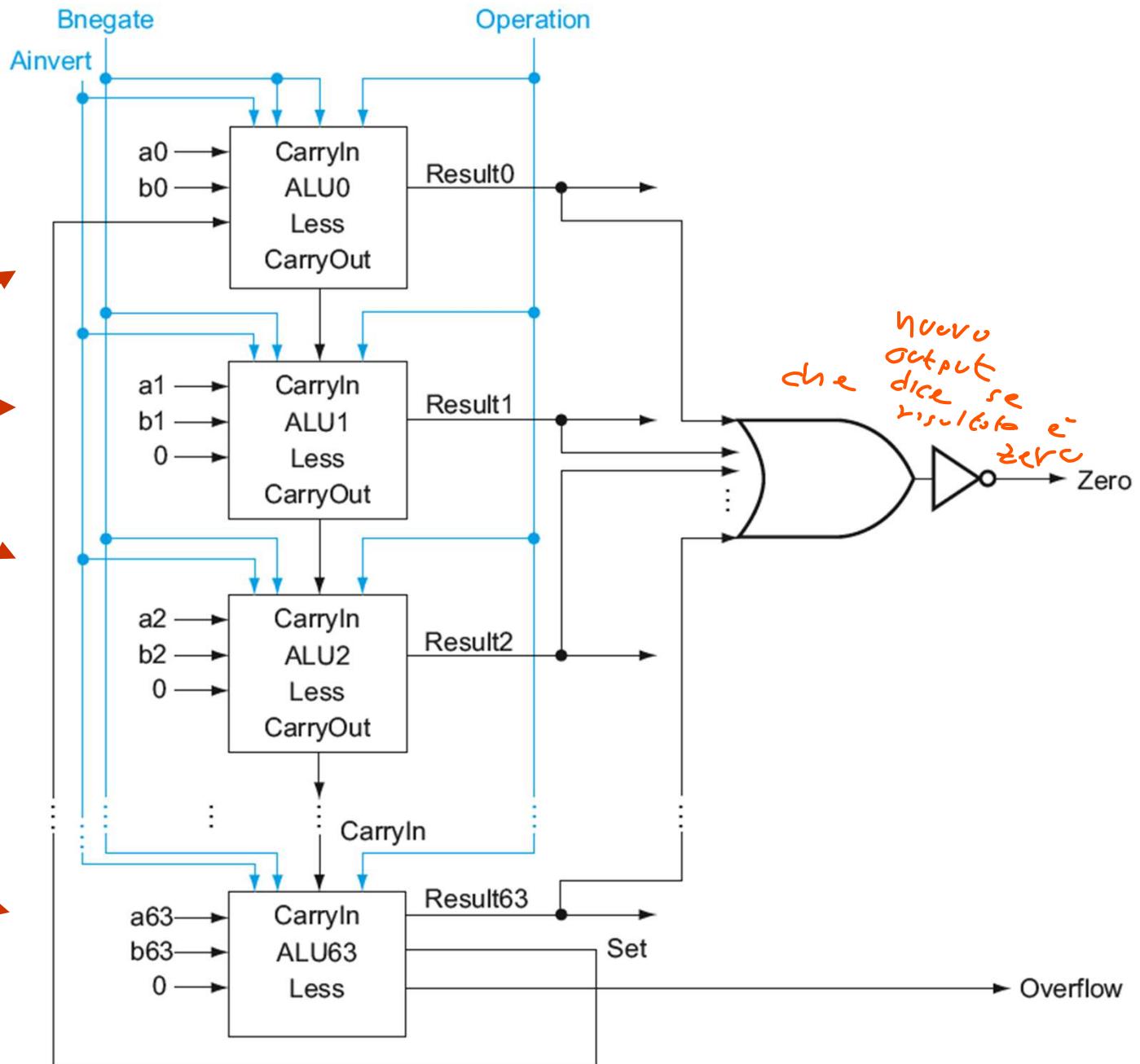
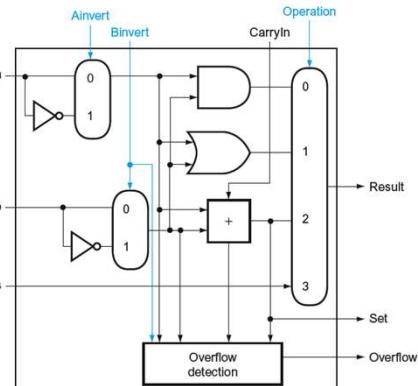
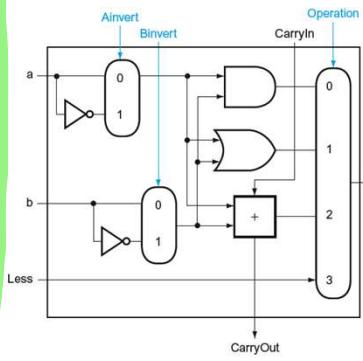
RISC-V: ALU per beq

- L'istruzione Branch if Equal (beq) realizza un salto se due registri sono uguali
- Si realizza il salto eseguendo $(a - b)$ e controllando se il risultato è uguale a 0
- L'ALU fornisce il valore 1 quando tutti i bit del risultato sono a zero, in quanto:

$$(a - b) == 0 \rightarrow a == b$$

“Zero” vale 1 quando il risultato $(a - b)$ è zero!

RISC-V: ALU per beq



Controllo operazioni dell'ALU

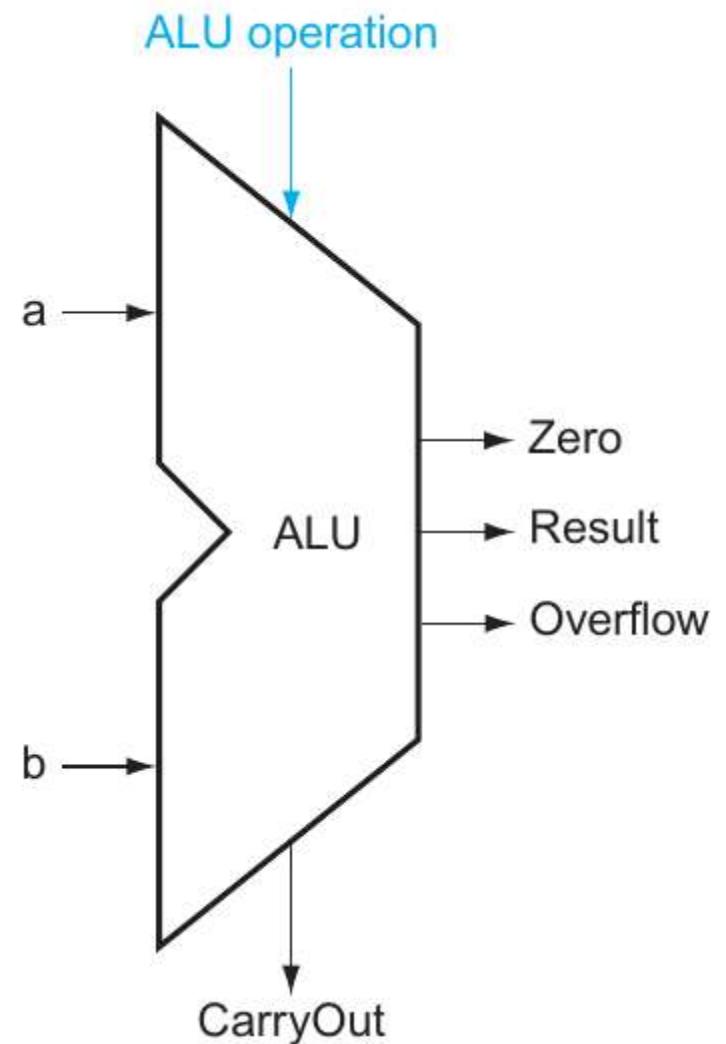
- Per il controllo dell'ALU, possiamo pensare alla combinazione di
 - 1 bit per l'ingresso Ainvert
 - 1 bit per l'ingresso Bnegate
 - 2 bit per gli ingressi Operation
- come a linee di controllo a 4 bit per fare in modo che l'ALU esegua la somma, la sottrazione, l'AND, l'OR, la NOR o la slt

Ainvert	Bnegate	Operation	ALU control lines	Function
0	0	00	0000	AND
0	0	01	0001	OR
0	0	10	0010	add
0	1	10	0110	subtract
0	1	11	0111	set less than
1	1	00	1100	NOR

RISC-V: ALU

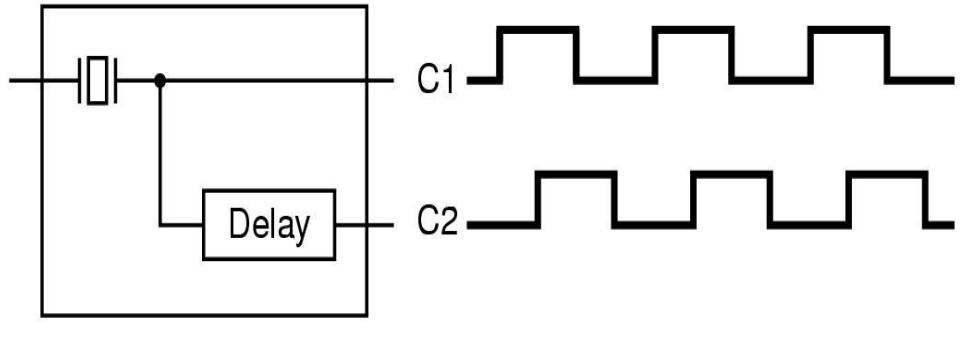
- Il simbolo comunemente usato per rappresentare una **ALU**
- Anche comunemente usato per indicare un **addizionatore**, quindi è prassi indicare **esplicitamente** con una scritta quale di questi due componenti si intende.

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

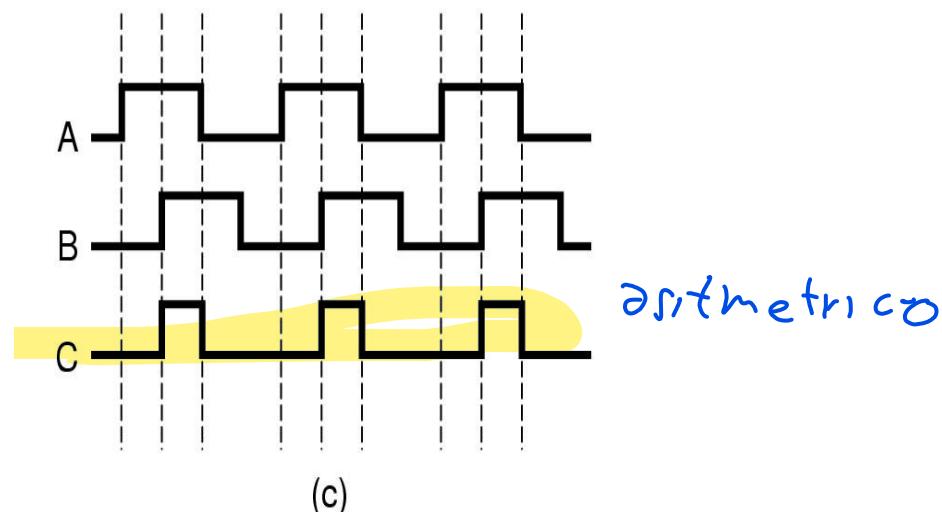


Clock

- **Clock**: un circuito che emette una serie di impulsi con una specifica larghezza e intermittenza
- **Tempo di ciclo di clock**: intervallo fra i fronti corrispondenti di due impulsi consecutivi
- Fronte di salita di C1
fronte di discesa di C1
fronte di salita di C2
fronte di discesa di C2



(a) (b)



500 MHz = 2 nsec di tempo
di ciclo di clock

Clock

Tempo di ciclo di clock

Il tempo si misura nei calcolatori in sottomultipli di secondo:

- **1 ms** (millisecondo) = 1×10^{-3} sec.
- **1 μ s** (microsecondo) = 1×10^{-6} sec.
- **1 ns** (nanosecondo) = 1×10^{-9} sec.

$$\text{Frequenza} = 1/\text{Tempo di ciclo}$$

Frequenza di clock

La frequenza specifica il numero di periodi di clock per unità di tempo (ovvero per secondo).

La frequenza si calcola come inverso del tempo di ciclo di clock. L'unità di misura è **l'Hertz**, di cui si utilizzano per i calcolatori, i multipli:

$$1 \text{ KHz} \text{ (KiloHertz)} = 1 \times 10^3 \text{ Hertz}$$

$$1 \text{ MHz} \text{ (MegaHertz)} = 1 \times 10^6 \text{ Hertz}$$

$$1 \text{ GHz} \text{ (GigaHertz)} = 1 \times 10^9 \text{ Hertz}$$