UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Circuiti in logica combinatoria e ALU

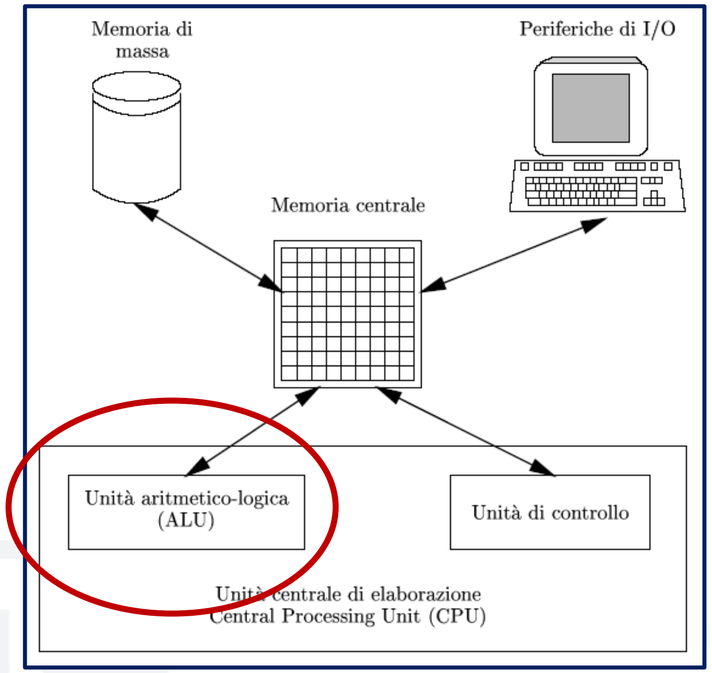
Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

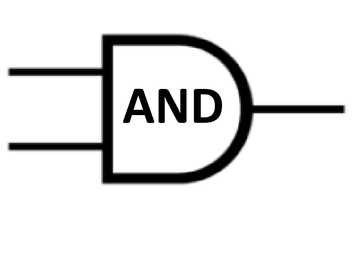
Trieste, 04 aprile 2025

RECAP

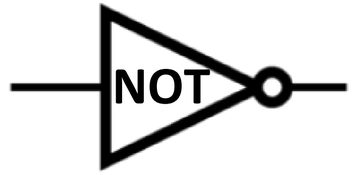
```
add $t0, $t1, $t2
and $t0, $t1, $t2
or $t0, $t1, $t2
xor $t0, $t1, $t2
sll $t0, $t1, 2
:
```



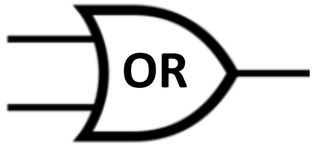
«Bastano» **CIRCUITI LOGICI COMBINATORI**, ovvero non ci serve memorizzare nulla (come nei circuiti logici sequenziali).



A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

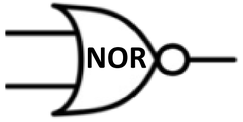


A	$\overline{A}$
0	1
1	0

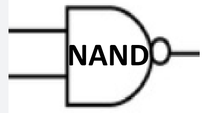


A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



A	B	$A + B$	A NOR B
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



A	B	$A \cdot B$	A NAND B
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Esercizio di *warm-up*

Scrivere l'espressione della funzione logica F = A $\oplus$ B

UNIVERSITÀ DEGLI STUDI DI TRIESTE

RECAP

Qualunque funzione logica può essere costruita usando le porte logiche AND, OR e NOT.

N°	Nome	Proprietà
1	**Proprietà associativa**	$a + (b + c) = (a + b) + c;\ a \cdot (b \cdot c) = (a \cdot b) \cdot c$
2	**Proprietà commutativa**	$a + b = b + a;\ a \cdot b = b \cdot a$
3	**Proprietà distributiva**	$a + (b \cdot c) = (a + b) \cdot (a + c)$ $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
4	**Assioma dell'annullamento**	$a \cdot 0 = 0;\quad a + 1 = 1$
5	**Assioma del Complemento**	$(a + \bar{a}) = 1;\quad (a \cdot \bar{a}) = 0$
6	**Assioma dell'idempotenza**	$a \cdot a = a;\quad a + a = a$
7	**Assioma doppia negazione**	$\bar{\bar{a}} = a$
8	**Teorema di DeMorgan**	$a + b = \overline{\bar{a} \cdot \bar{b}}\ ;\ a \cdot b = \overline{\bar{a} + \bar{b}}$
9	**Teorema dell'assorbimento**	**Se** $Y = a + ab$ **allora** $Y = a$
10	**Teorema del consenso**	**Se** $Y = ab + \bar{a}c + bc$ **allora** $Y = ab + \bar{a}c$

Il **Teorema dei Mintermini** afferma che **ogni funzione booleana può essere espressa come una somma (operazione OR) di mintermini**. Un **mintermine** è un prodotto (operazione AND) di tutte le variabili della funzione, ciascuna presente in forma diretta o negata, che rendono la funzione uguale a 1.

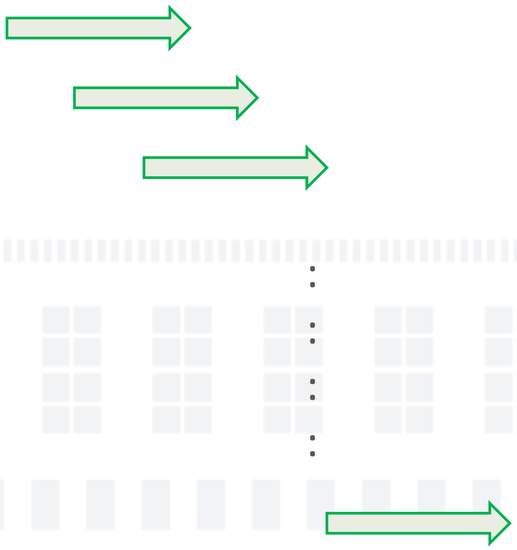
Il **Teorema dei Maxtermini** afferma che **ogni funzione booleana può essere espressa come un prodotto (operazione AND) di maxtermini**. Un **maxtermine** è una somma (operazione OR) di tutte le variabili della funzione, ciascuna presente in forma diretta o negata, che rendono la funzione uguale a 0.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

IL PRIMO CIRCUITO LOGICO

Full adder *(senza riporto)*

Inputs			Outputs	
A	B	C_{in}	S	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	



Ogni riga della tabella di verità genera un MINTERMINE.

A seconda di quale forma del *Teorema di Espressione Canonica (Mintermini/Maxtermini)* usiamo, dobbiamo formare i mintermini/maxtermini e poi combinarli tra loro in modo diverso.

(1) Nel caso della forma SOP, cerchiamo le righe **dove S=1**, formiamo i mintermini come prodotto delle variabili e poi li sommiamo.

(2) Nel caso della forma POS, cerchiamo le righe **dove S=0**, i maxtermini sono somma delle variabili e vengono moltiplicati tra loro.

DEGLI STUDI DI TRIESTE

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma SOP del Teorema.. (*continua*..)

Inputs			Outputs	
A	B	C_{in}	S	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

not(A) · not(B) · C

not(A) · B · not(C)

A · not(B) · not(C)

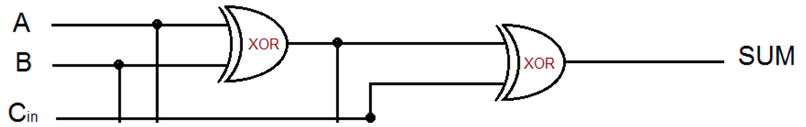
A · B · C

S = not(A) · not(B) · C + not(A) · B · not(C) + A · not(B) · not(C) + A · B · C

UNIVERSITÀ DEGLI STUDI DI TRIESTE

IL PRIMO CIRCUITO LOGICO

Full adder



Considerando la forma SOP del Teorema.. (*continua*..)

$$S = \text{not}(A) \cdot \text{not}(B) \cdot C + \text{not}(A) \cdot B \cdot \text{not}(C) + A \cdot \text{not}(B) \cdot \text{not}(C) + A \cdot B \cdot C$$

semplificazione (usando proprietà e assiomi dell'algebra booleana)

$$S = (\text{not}(A) \cdot \text{not}(B) + A \cdot B) \cdot C + (\text{not}(A) \cdot B + A \cdot \text{not}(B)) \cdot \text{not}(C)$$

pr. distrib. inversa

$$A \oplus B$$

$$\text{not}(A) \cdot \text{not}(B) + A \cdot B = \text{NOT}(A \oplus B)$$

$$S = \text{not}(A \oplus B) \cdot C + (A \oplus B) \cdot \text{not}(C)$$

not(X) **X**

A	B	A XOR B	NOT(A XOR B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

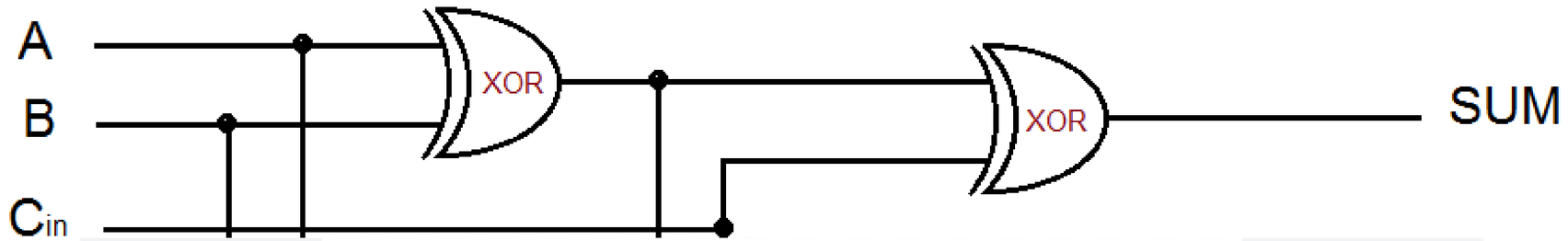
UNIVERSITÀ DEGLI STUDI DI TRIESTE

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma SOP del Teorema..

S = (A ⊕ B) ⊕ C



UNIVERSITÀ DEGLI STUDI DI TRIESTE

slide corretta!

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma POS del Teorema.. (*continua..*)

Inputs			Outputs
A	B	C_{in}	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A + B + C

A + not(B) + not(C)

not(A) + B + not(C)

not(A) + not(B) + C

S = (A + B + C) · (A + not(B) + not(C)) · (not(A) + B + not(C)) · (not(A) + not(B) + C)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

slide corretta!

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma POS del Teorema.. (*continua..*)

$S = (A + B + C) \cdot (A + \text{not}(B) + \text{not}(C)) \cdot (\text{not}(A) + B + \text{not}(C)) \cdot (\text{not}(A) + \text{not}(B) + C) =$

$= [A + (B + C) \cdot (\text{not}(B) + \text{not}(C)] \cdot [\text{not}(A) + (B + \text{not}(C)) \cdot (\text{not}(B) + C)] =$

$= [A + (B \cdot \text{not}(B) + C \cdot \text{not}(B) + B \cdot \text{not}(C) + C \cdot \text{not}(C)] \cdot [\text{not}(A) + (B \cdot \text{not}(B) + \text{not}(C) \cdot \text{not}(B) + B \cdot C + C \cdot \text{not}(C)] =$

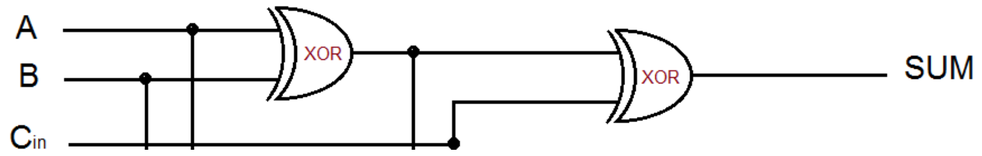
$= [A + C \cdot \text{not}(B) + B \cdot \text{not}(C)] \cdot [\text{not}(A) + \text{not}(C) \cdot \text{not}(B) + B \cdot C] =$

$B \oplus C$

$\text{not}(B \oplus C)$

$= A \cdot \text{not}(A) + (B \oplus C) \cdot \text{not}(A) + A \cdot \text{not}(B \oplus C) + (B \oplus C) \cdot \text{not}(B \oplus C) =$

$= \text{not}(A) \cdot (B \oplus C) + A \cdot \text{not}(B \oplus C) = A \oplus B \oplus C$



UNIVERSITÀ DEGLI STUDI DI TRIESTE

OSSERVAZIONI IMPORTANTI

- La tabella definisce una funzione logica unica
- **L’espressione canonica (SOP o POS) è unica**

- *La forma algebrica della funzione logica è unica:* ***NO!***
- *Ci possono essere circuiti equivalenti che implementano la stessa funzione logica.*

UNIVERSITÀ DEGLI STUDI DI TRIESTE

IL PRIMO CIRCUITO LOGICO

Full adder: circuito che realizza la SOMMA BINARIA tra due bit (con riporto).

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
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

Esercizio

Scrivete l'espressione della funzione logica C_{out} e provate ad integrare il circuito di prima con le porte mancanti

UNIVERSITÀ DEGLI STUDI DI TRIESTE

IL PRIMO CIRCUITO LOGICO

Full adder: circuito che realizza la SOMMA BINARIA tra due bit (con riporto).

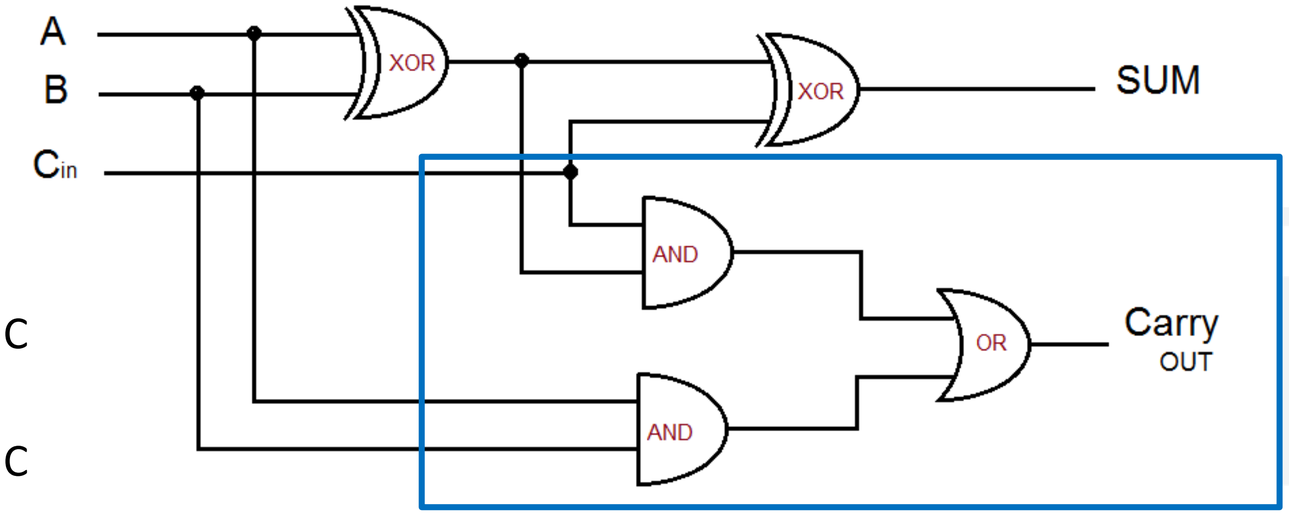
Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		1

not(A) · B · C

A · not(B) · C

A · B · not(C)

A · B · C



C_{out} = not(A) · B · C + A · not(B) · C + A · B · not(C) + A · B · C =

= (not(A) · B + A · not(B)) · C + A · B · (not(C) + C) =

= (**A** ⊕ **B**) · C + A · B · 1 = (**A** ⊕ **B**) · C + A · B

UNIVERSITÀ DEGLI STUDI DI TRIESTE

LOGICHE A DUE LIVELLI E PLA

Qualunque funzione logica può essere costruita usando porte AND, OR e NOT.

Possiamo creare logiche a due livelli:

- **Somma di prodotti**: somma logica (OR) di prodotti (AND)
- **Prodotto di somme**: prodotto (AND) di somme (OR)

SOP è più adatto per

- PLA (Programmable Logic Array): SOP è naturale perché le PLA implementano facilmente AND seguiti da OR
- ALU (Arithmetic Logic Unit) per realizzare somme, XOR, comparatori, flag di condizione
- Reti combinatorie generiche per sintetizzare direttamente la funzione a partire dai casi veri

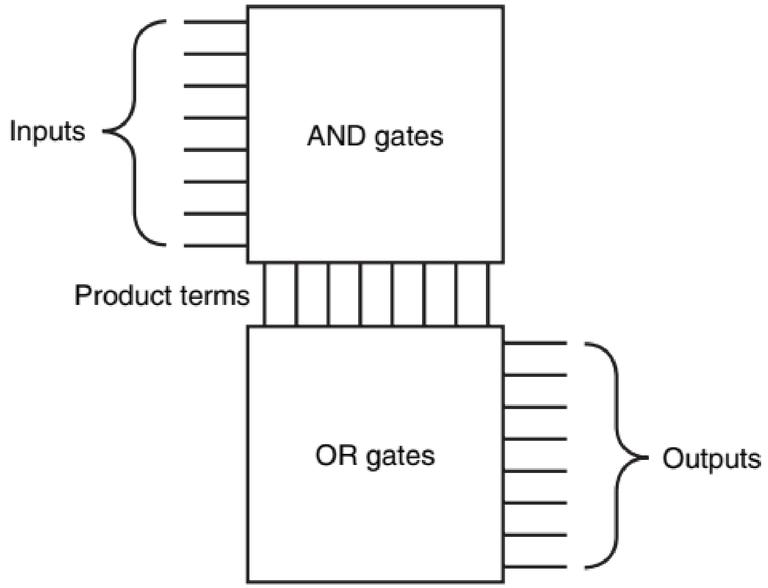
POS è più adatto per

- Selezione, controllo, decodifica per esprimere condizioni di disattivazione
- Multiplexer per il quale Le uscite sono 0 tranne che per una condizione → POS si adatta
- Decoder, demultiplexer: Ogni uscita è attiva solo per una configurazione specifica
- Controlli negativi: Quando è più comodo definire "quando non succede" qualcosa

UNIVERSITÀ DEGLI STUDI DI TRIESTE

PROGRAMMABLE LOGICAL ARRAY

Una **PLA** è un circuito logico programmabile che implementa **una o più funzioni booleane** nella forma Somma di Prodotti (SOP): $F = P_1 + P_2 + \cdots + P_n$
dove ciascun P_i è un prodotto logico (AND) di input o dei loro complementi.



- Un insieme di **input**
- I corrispondenti input complementati (mediante inverter) per poter gestire più uscite

- Una **logica** a due stage:
  Primo stage: un array di porte logiche AND (prodotto)
  Secondo stage: un array di porte logiche OR (somma)

Output: una stessa base di prodotti logici (AND) può essere combinata in modi diversi per produrre diverse funzioni logiche. Ogni uscita può combinare (OR) i prodotti in modo diverso.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

PLA

Esempio

Consideriamo la seguente tabella di verità.

Inputs			Outputs		
A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Costruire il PLA corrispondente come somme di prodotto

Come si fa?

Si costruisce la SOP per ogni colonna. Si tratta ogni colonna come una funzione logica indipendente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

PLA

Esempio

Inputs			
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

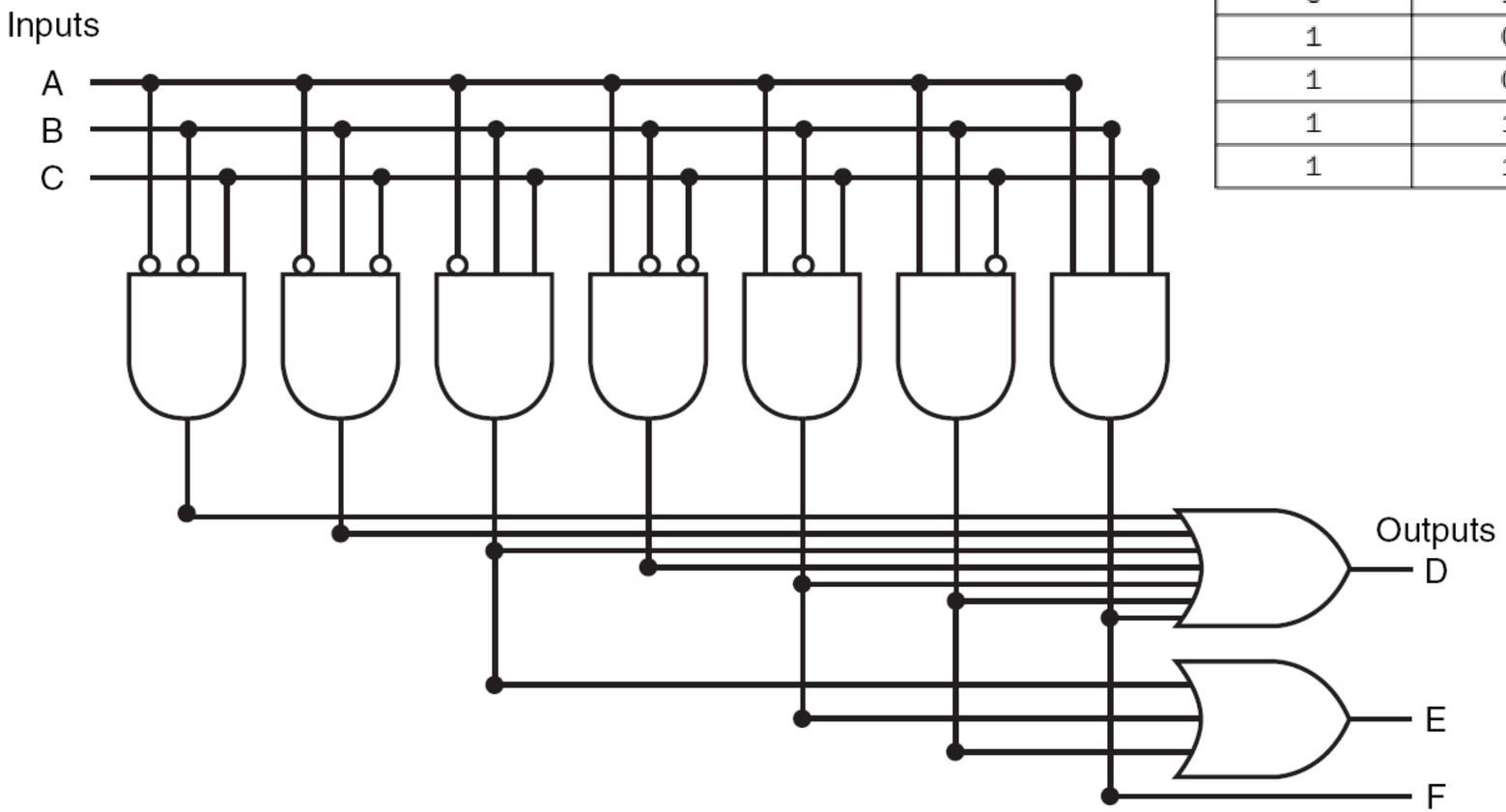


FIGURE C.3.4 The PLA for implementing the logic function described in the example.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

DECODER (1/2)

Un decoder è un componente elettronico caratterizzato dall'avere **n ingressi e 2^n uscite**.

*Lo scopo del decoder è di **impostare allo stato alto l'uscita corrispondente alla conversione in base 10** della codifica binaria a n bit ricevuta **in input** (e di impostare allo stato basso tutte le altre).*

In pratica:

- gli n input sono interpretati come un numero unsigned
- se questo numero rappresenta il numero i, allora
  - solo il bit in output di indice i (i=0,1,...,2^{n-1}) verrà posto ad 1
  - tutti gli altri verranno posti a 0

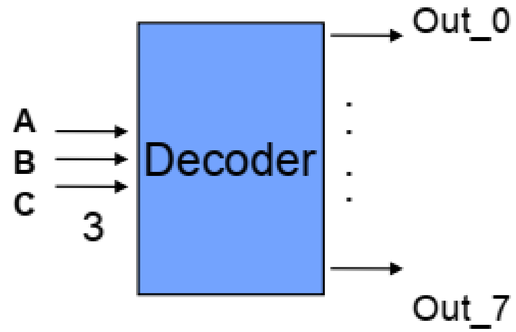
UNIVERSITÀ DEGLI STUDI DI TRIESTE

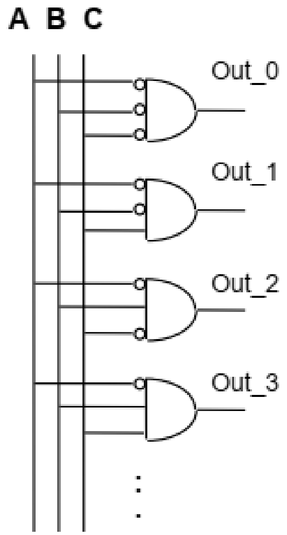
DECODER (2/2)

2^n uscite, solo 1 valore è attivo per ogni combinazione di input.

Quindi:

- **l'ingresso seleziona una delle uscite**;
- l'uscita selezionata ha valore 1 tutte le altre 0





A	B	C	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

C'è un solo 1 per ogni riga

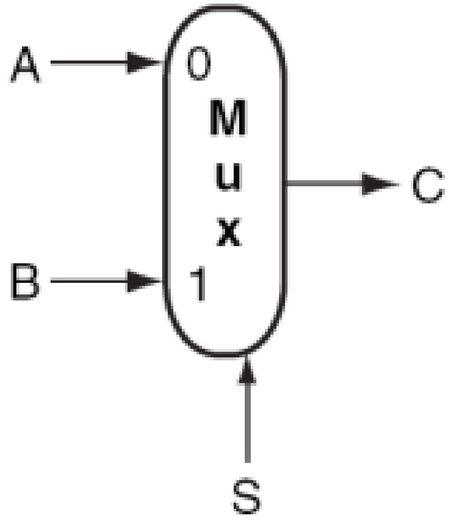
Con 3 bit possiamo selezionare UNA di $2^3 = 8$ uscite

UNIVERSITÀ DEGLI STUDI DI TRIESTE

MULTIPLEXER

Un multiplexer, detto anche selettore, è un componente elettronico caratterizzato da:

- 2^n entrate principali
- n entrate di controllo (selettore)
- 1 uscita



Il valore del selettore determina quale input diviene output.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

MULTIPLEXER

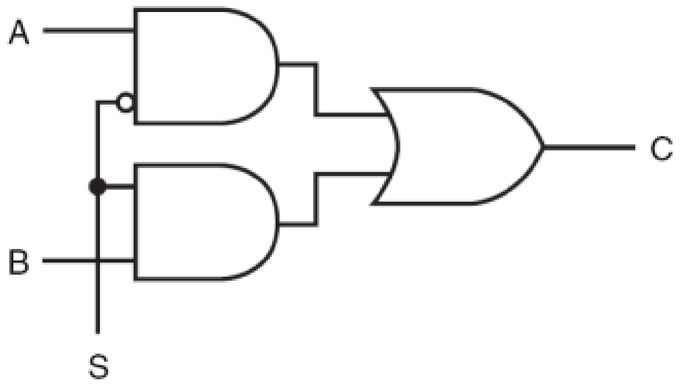
Esempio 1

Funzione logica

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

Qual è la tabella di verità di questo circuito?

Circuito logico equivalente



S	A	B	C
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

S ha la funzione di segnale di controllo

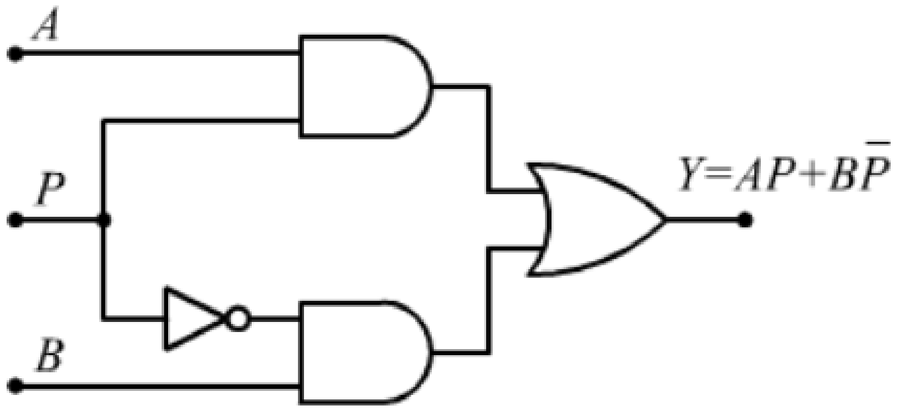
UNIVERSITÀ DEGLI STUDI DI TRIESTE

MULTIPLEXER

Esempio 2

Nota. Questo circuito è equivalente al precedente, ma il controllo dato dal segnale P è invertito rispetto a prima. In questo caso, se P=1, viene abilitato l'ingresso A.

Circuito logico equivalente



A=0, B=0, P=0

A=0 AND P=0 >> 0
B=0 AND notP=1 >>0

0 OR 0 >> 0

A=1, B=0, P=0

A=1 AND P=0 >> 0
B=0 AND notP=1 >>0

0 OR 0 >> 0

A=1, B=1, P=0

A=1 AND P=0 >> 0
B=1 AND notP=1 >>1

0 OR 1>> 1

A=0, B=1, P=0

A=0 AND P=0 >> 0
B=1 AND notP=1 >>1

0 OR 1 >> 1

UNIVERSITÀ DEGLI STUDI DI TRIESTE

MULTIPLEXER

Esempio 2

Circuito logico equivalente

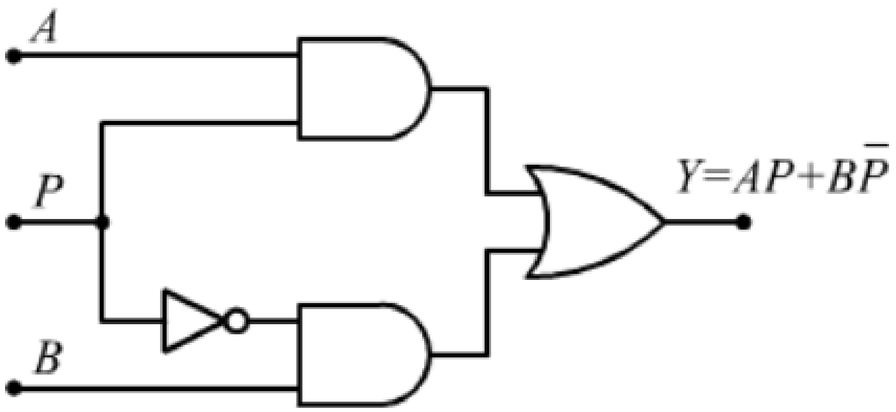


Tabella di verità

P	Y
0	B
1	A

A=0, B=0, P=1

A=0 AND P=1 >> 0
B=0 AND notP=0 >>0

0 OR 0 >> 0

A=1, B=0, P=1

A=1 AND P=1 >> 0
B=0 AND notP=0 >>0

1 OR 0 >> 1

A=1, B=1, P=1

A=1 AND P=1 >> 1
B=1 AND notP=0 >>0

0 OR 1>> 1

A=0, B=1, P=1

A=0 AND P=1 >> 0
B=1 AND notP=0 >>1

0 OR 1 >> 1

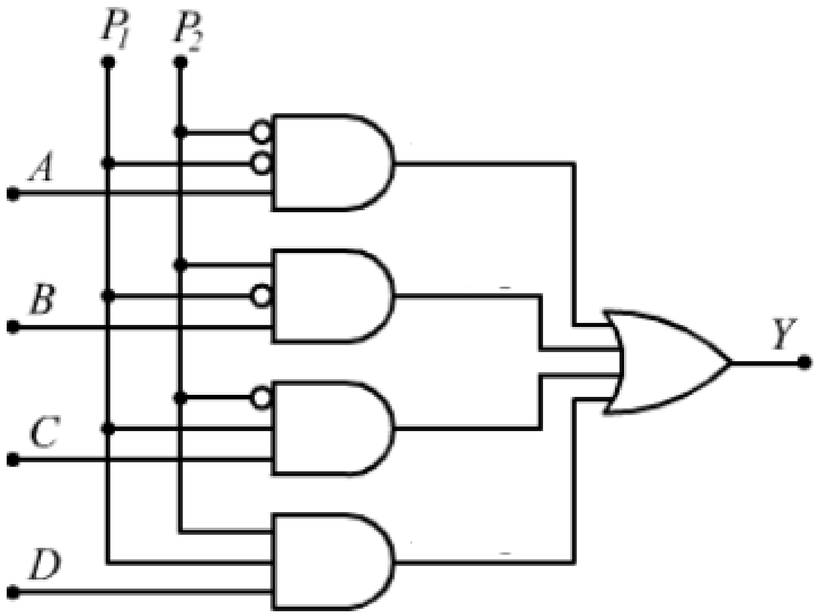
UNIVERSITÀ DEGLI STUDI DI TRIESTE

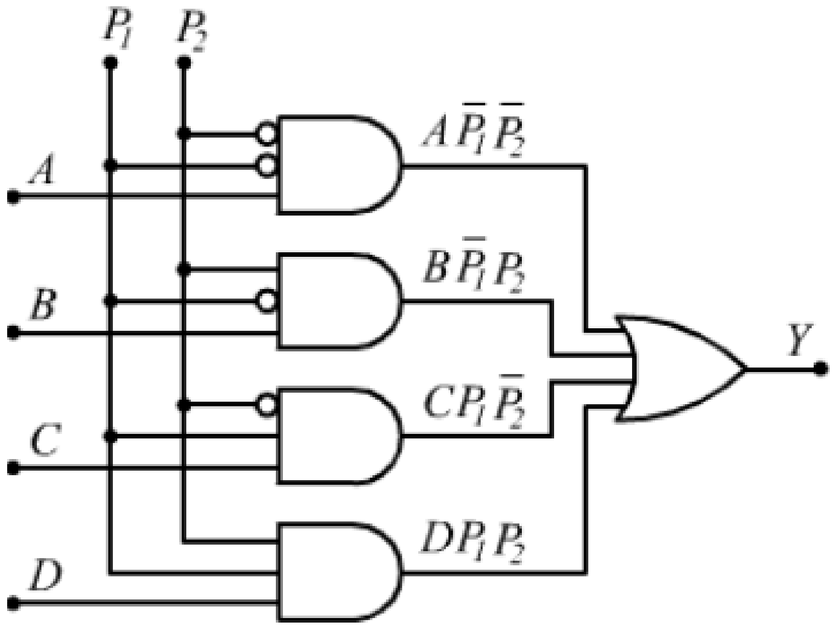
MULTIPLEXER

Esempio 3

Circuito logico

Quale funzione logica viene implementata??





$$Y = A\bar{P}_1\bar{P}_2 + B\bar{P}_1P_2 + CP_1\bar{P}_2 + DP_1P_2$$

UNIVERSITÀ DEGLI STUDI DI TRIESTE

MULTIPLEXER

Esempio 3

$$Y = A\bar{P}_1\bar{P}_2 + B\bar{P}_1P_2 + CP_1\bar{P}_2 + DP_1P_2$$

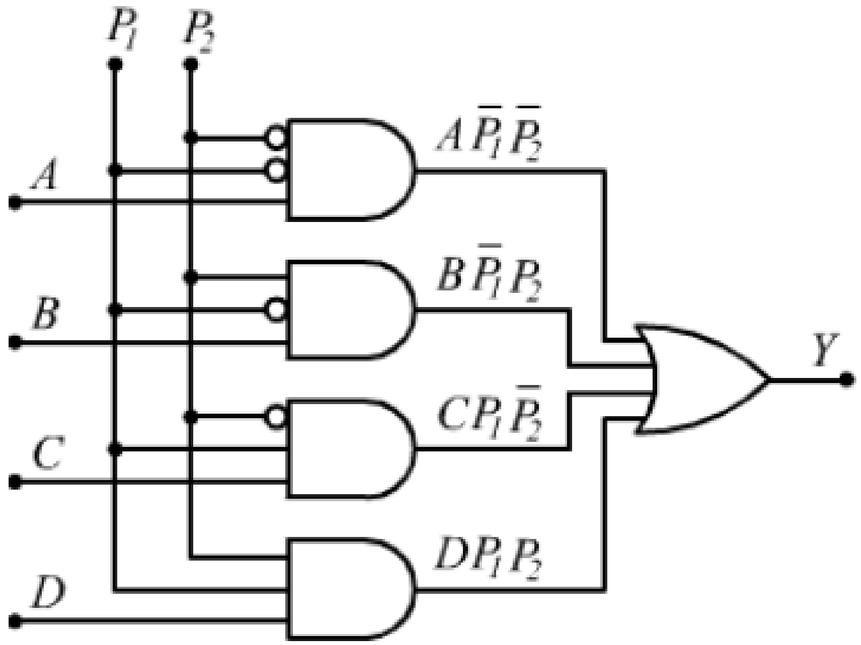
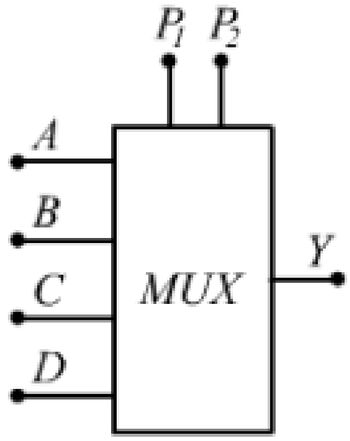


Tabella di verità

P_1	P_2	Y
0	0	A
0	1	B
1	0	C
1	1	D



Circuito con **2 segnali di controllo**

UNIVERSITÀ DEGLI STUDI DI TRIESTE

ARITHMETIC LOGIC UNIT (ALU)

L’Aritmethic Logic Unit:

- E’ la parte del processore che svolge le operazioni aritmetico-logiche
- E’ un insieme di circuiti combinatori che implementa:
  - Operazioni aritmetiche: es somma e sottrazione
  - Operazioni logiche: es AND e OR

UNIVERSITÀ DEGLI STUDI DI TRIESTE

BLOCCHI DI BASE PER COSTRUIRE L'ALU

1. AND gate (c = a · b)

a	b	c = a · b
0	0	0
0	1	0
1	0	0
1	1	1

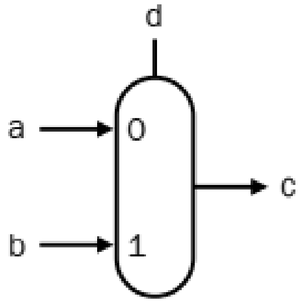
2. OR gate (c = a + b)

a	b	c = a + b
0	0	0
0	1	1
1	0	1
1	1	1

3. Inverter (c = ā)

a	c = ā
0	1
1	0

4. Multiplexor
(if d = = 0, c = a;
else c = b)

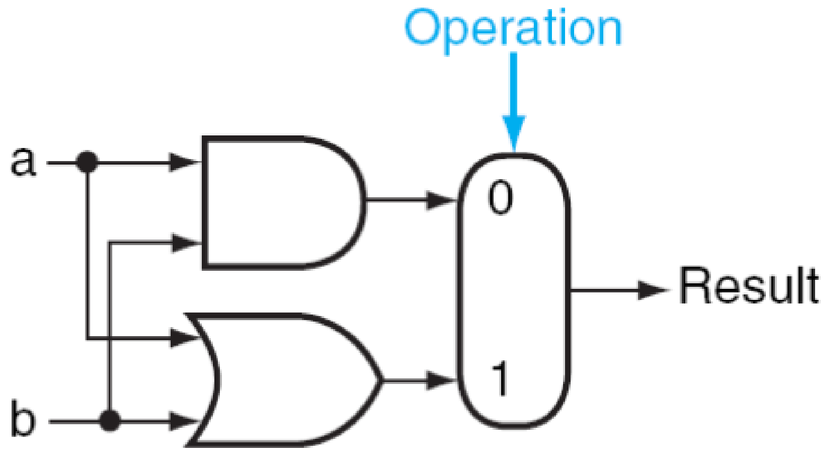


d	c
0	a
1	b

UNIVERSITA DEGLI STUDI DI TRIESTE

ALU A 1 BIT: OPERAZIONI LOGICHE

ALU su 1 bit che implementa AND e OR



UNIVERSITÀ DEGLI STUDI DI TRIESTE

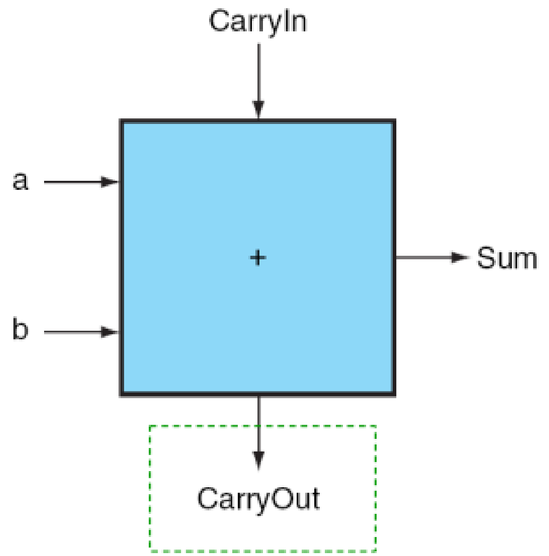
ALU A 1 BIT: OPERAZIONI ARITMETICHE

Addizione

```
0000000000000000 0001  1 1 1 1  0 0 1 1  0 1 0 0      carry

0000000000000000 0000  1 0 1 1  1 1 0 0  1 0 1 1  +
0000000000000000 0000  0 1 1 0  1 0 0 1  1 0 1 0  =
-----------------------------------------------------

0000000000000000 0001  0 0 1 0  0 1 1 0  0 1 0 1
```

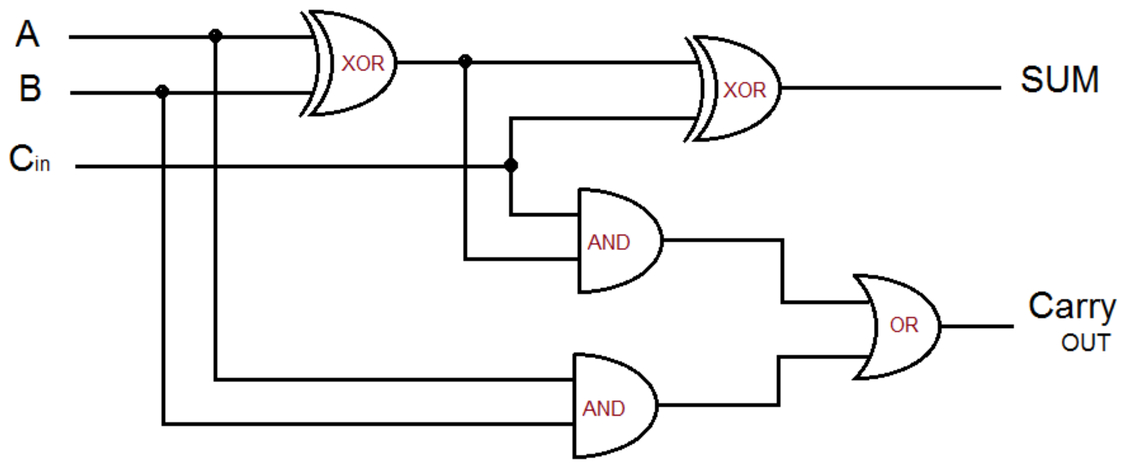


UNIVERSITÀ DEGLI STUDI DI TRIESTE

ALU A 1 BIT: OPERAZIONI ARITMETICHE

Addizione (carry, sum)

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



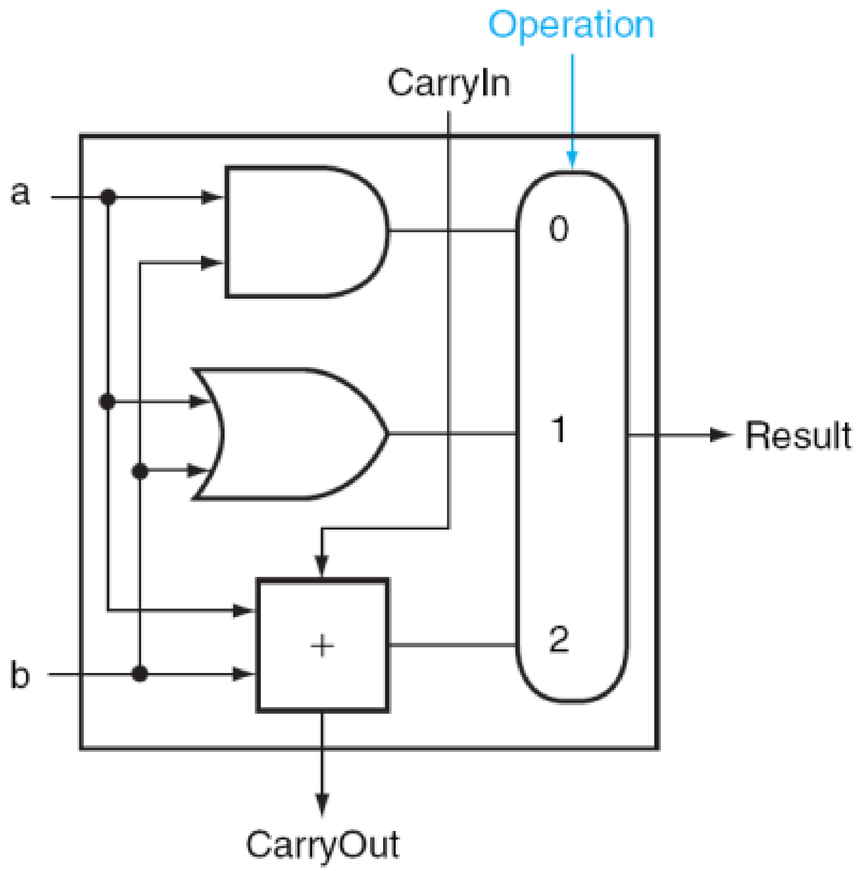
S = (A $\oplus$ B) $\oplus$ C

$\mathbf{C_{out}}$ = (A $\oplus$ B) · C + A · B

UNIVERSITÀ DEGLI STUDI DI TRIESTE

ALU A 1 BIT: SCELTA DI DIVERSE OPERAZIONI

ALU a 1 bit che può eseguire SOMMA, AND oppure OR

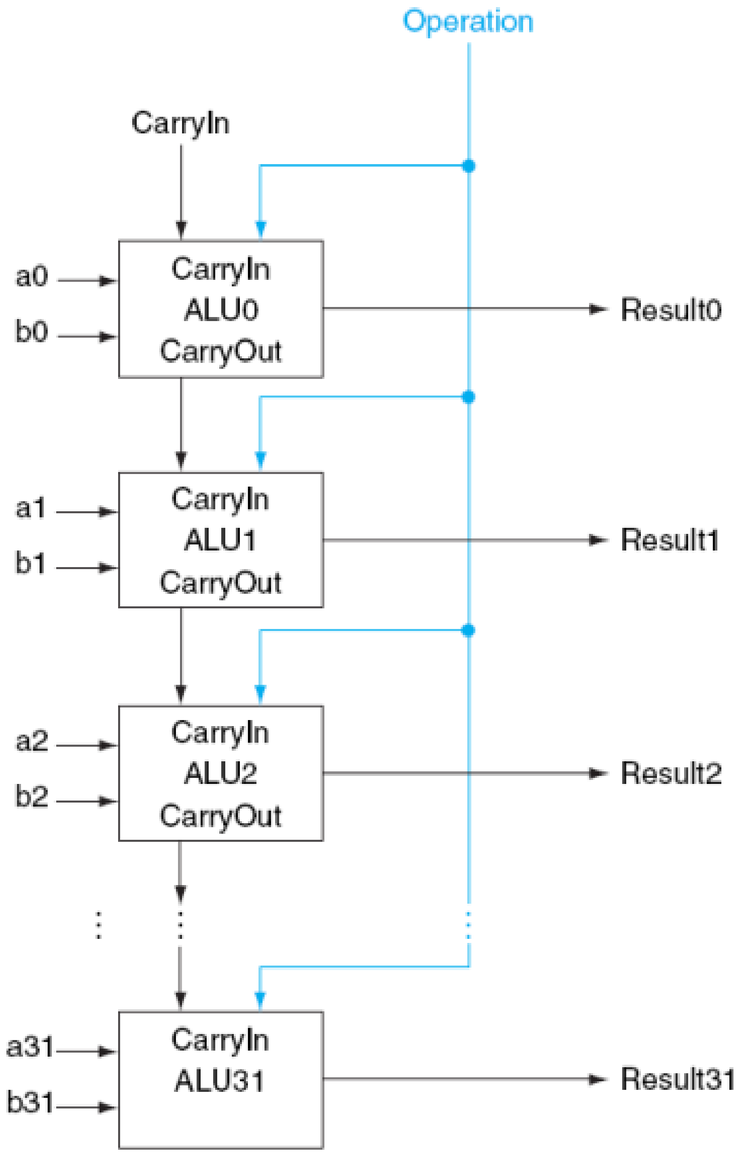


UNIVERSITÀ DEGLI STUDI DI TRIESTE

ALU A **32 BIT**

Connessione di 1-bit ALU adiacenti

RIPPLE CARRY



UNIVERSITÀ DEGLI STUDI DI TRIESTE

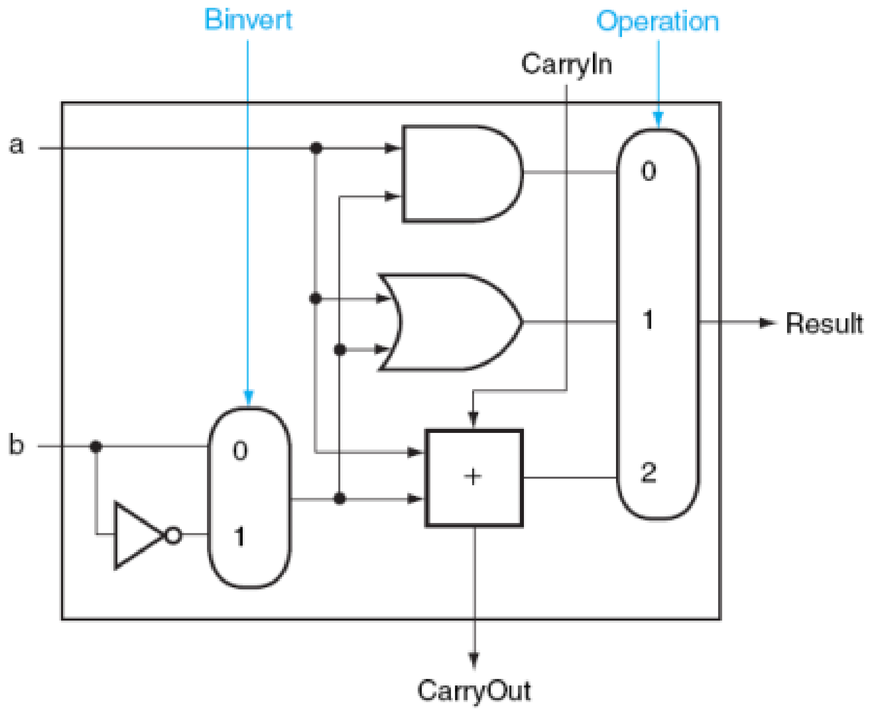
ALU A 1 BIT: SCELTA DI DIVERSE OPERAZIONI

Cosa succede se settiamo CarryIn a 1?

Possiamo sommare a+b+1

Se neghiamo b, possiamo ottenere una sottrazione (in CA2)

$$a - b = a + (\bar{b} + 1)$$



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Materiale per la lezione

- Appendice B, Patterson & Hennessy

Prossima lezione: MARTEDì 8 ***aprile, h.14:00****,* ***aula*** Aula 4A – Edificio D