

Corso di
Architettura degli Elaboratori
a.a. 2018/2019

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

“Mentre ENIAC è dotato di 18.000 tubi a vuoto e pesa 30 tonnellate, i computer del futuro potranno avere 1000 tubi e pesare, forse, solo mezza tonnellata”

RIVISTA POPULAR MECHANICS, 1949

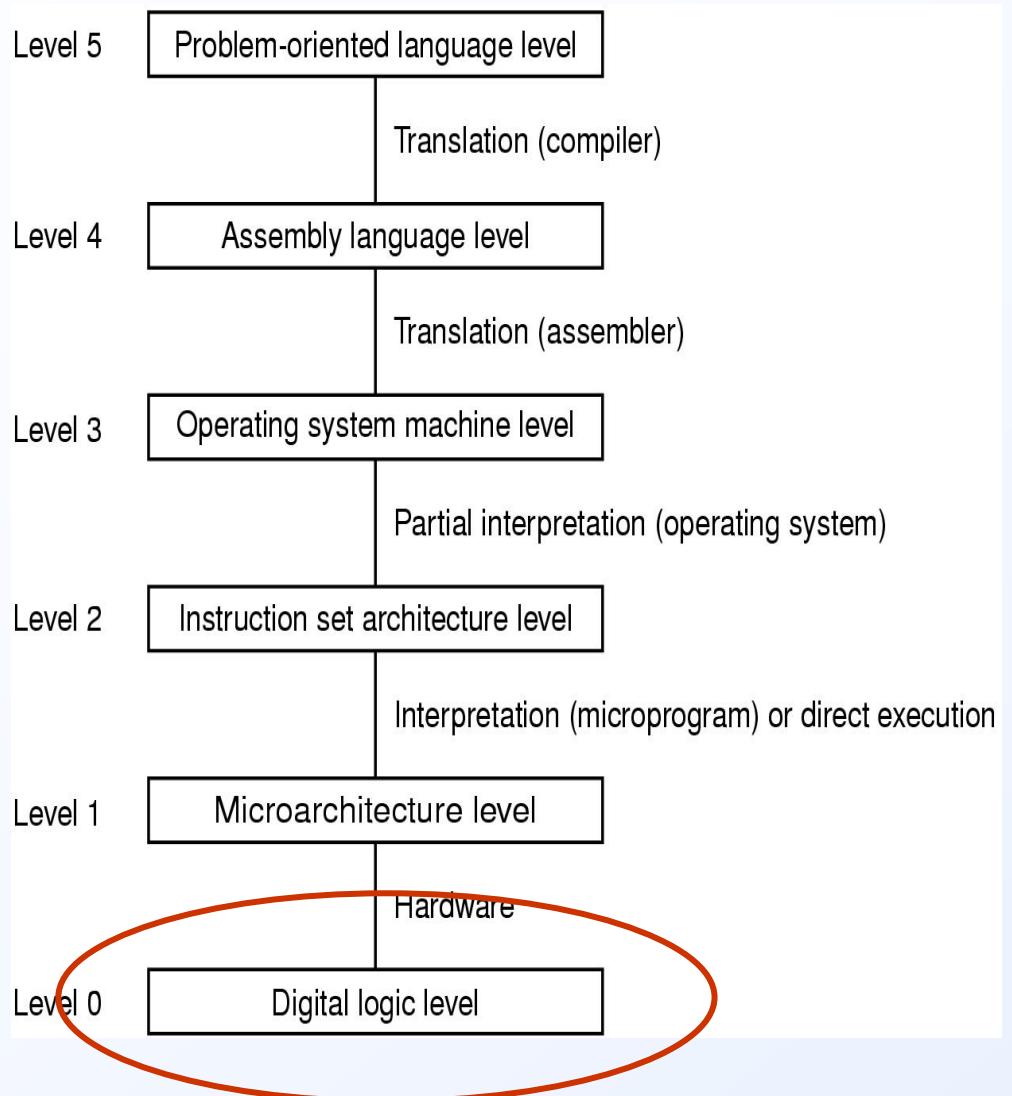
Livello 0: logico-digitale

Livello 0: *Logico-Digitale*

Costituenti di base del computer:

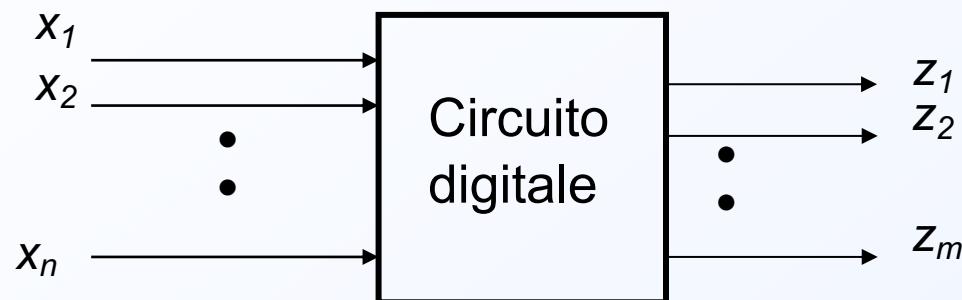
- porte
- registri
- memoria

Sotto il livello 0 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) e 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 .



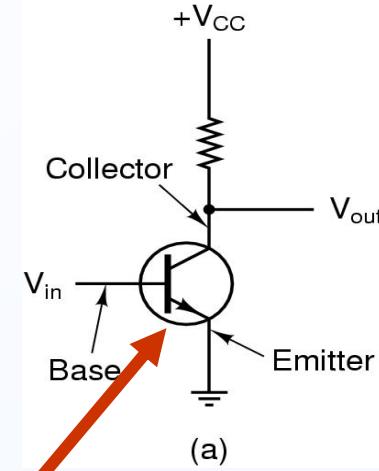
Porte logiche

- 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.

Transistor

Il transistor si comporta come un interruttore binario molto veloce:

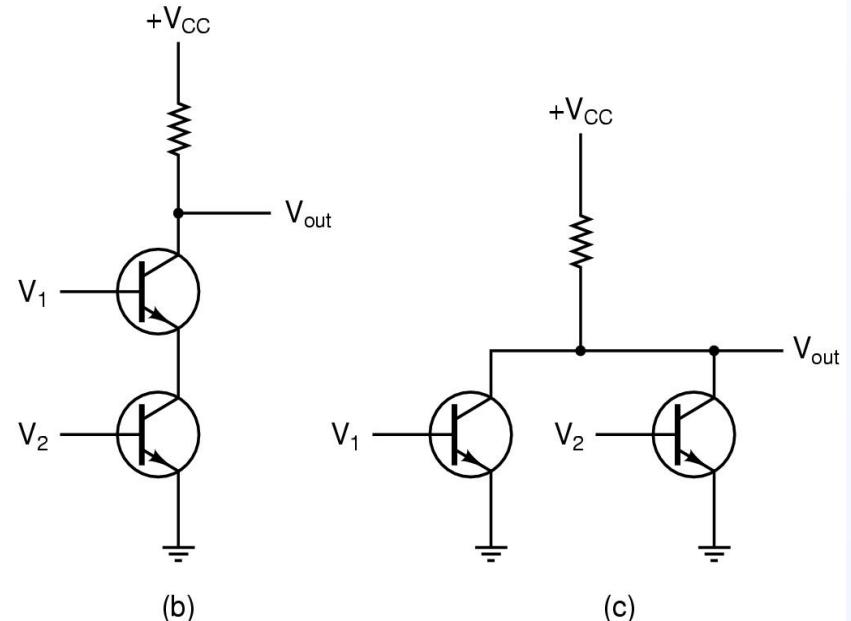
- quando V_{in} è basso, il transistor si disabilita (circuito aperto) e si comporta come *una resistenza infinita*, quindi V_{out} è alto; viceversa quando V_{in} è alto, il transistor si attiva e si comporta come *un filo* mettendo a terra V_{out}
- pochi **nanosecondi** per passare da uno stato all'altro: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



Transistor bipolare

Porte logiche

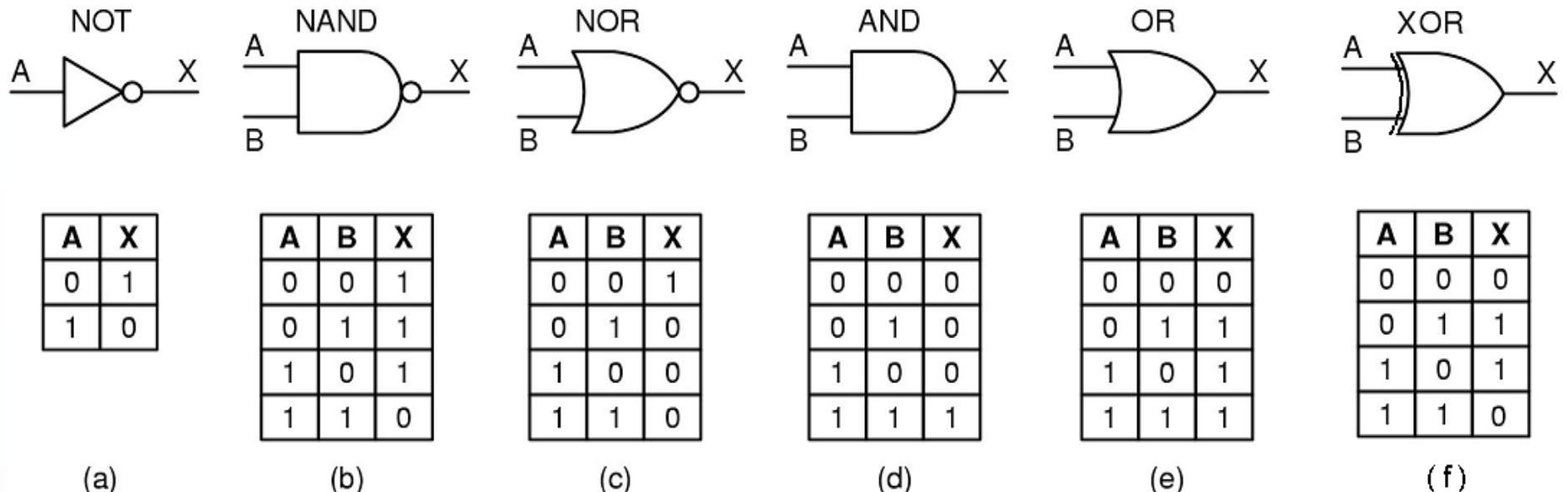
- **Logica positiva** ($0 = \text{bassa tensione}$; $1 = \text{alta tensione}$)
 - (b) una **porta NAND**: due transistor collegati in serie
 - (c) una **porta NOR**: due transistor collegati in parallelo



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



- 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 ($A \text{ XOR } B = (A \text{ OR } B) \text{ AND } (A \text{ NAND } B)$)



PorteDiBase

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: $A\bar{B} + B\bar{C}$ (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:

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

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

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

- 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

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

$$\overline{A}BA + A\overline{B}B = 0B + A0 = 0$$

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:

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

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

- 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

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

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:

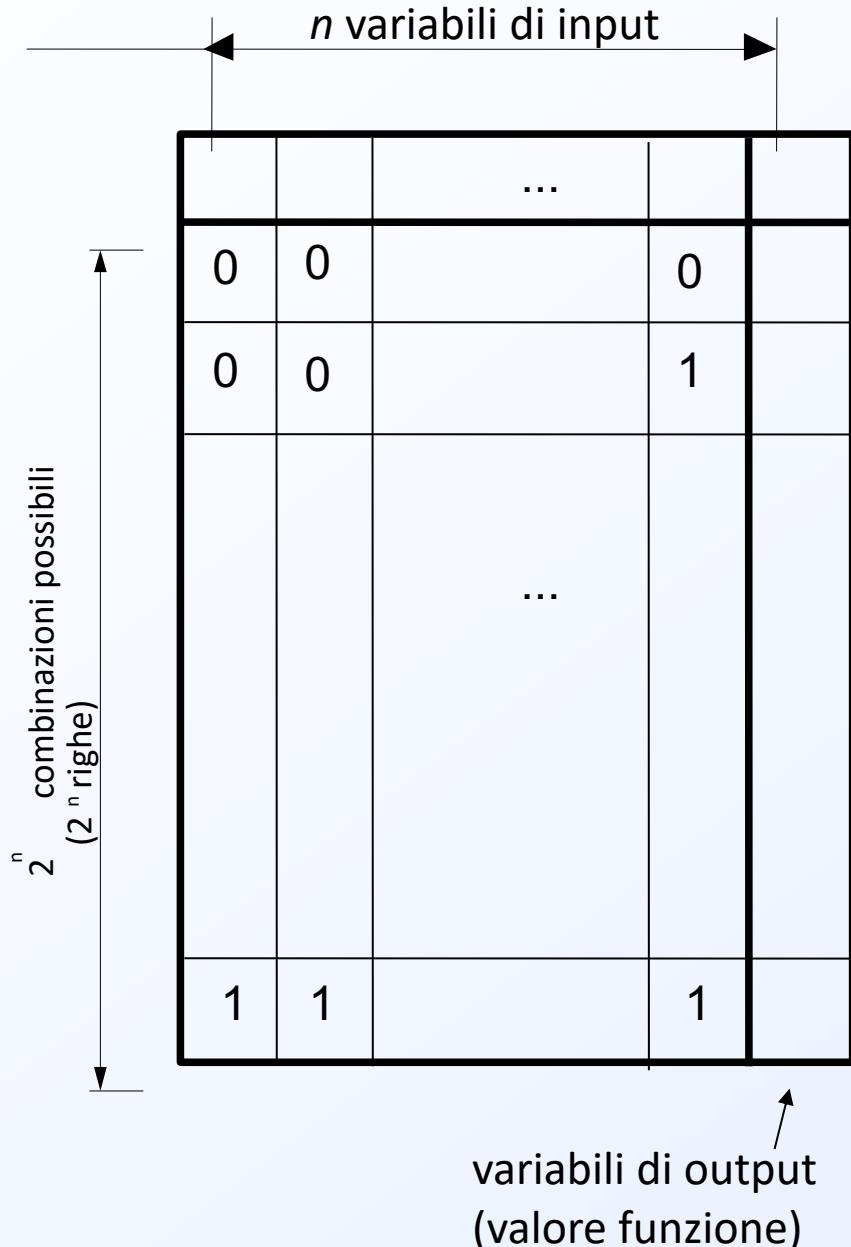
$$y = f(x_1, x_2, \dots, x_n)$$

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



Algebra Booleana: forme canoniche

- **Formula normale disgiuntiva (FND):**
 - Sommatoria di termini ciascuno dei quali è una produttoria 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 di termini ciascuno dei quali è una sommatoria 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: $\bar{A}BC$, $A\bar{B}C$, $A\bar{B}\bar{C}$, $\bar{A}BC$
- ✗ $M = \bar{A}BC + A\bar{B}C + A\bar{B}\bar{C} + ABC$
- ✗ 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 \leftrightarrow Formula algebrica

Da tabella di verità a formula algebrica

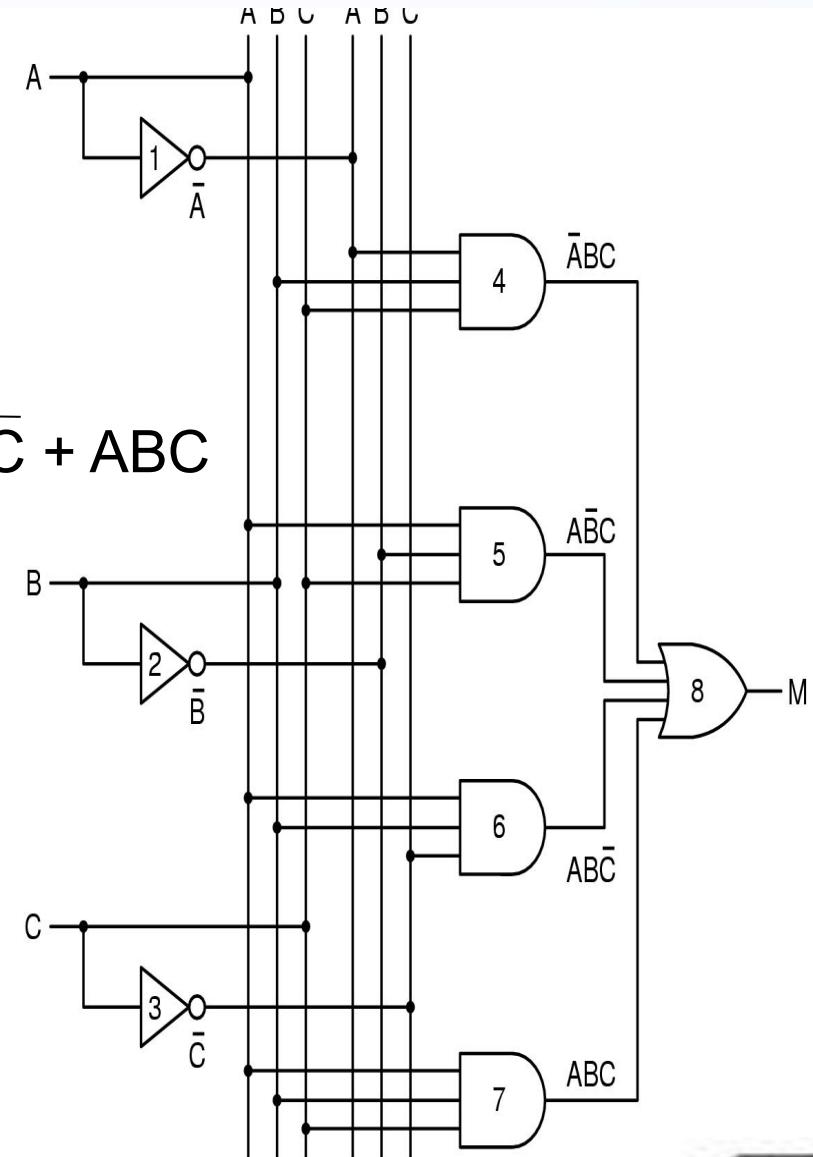
1. Scrivere la tabella di verità per la funzione
2. Disporre gli invertitori per generare il complemento di ogni input
3. Introdurre una porta AND per ogni termine con un 1 nella colonna dei risultati
4. Collegare le porte AND agli input appropriati
5. Inviare l'output di tutte le porte AND in una porta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

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

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

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

1a)

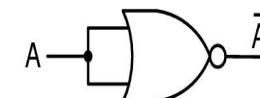
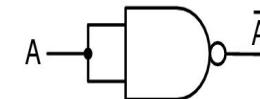


1b)

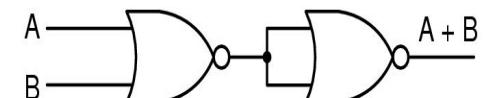
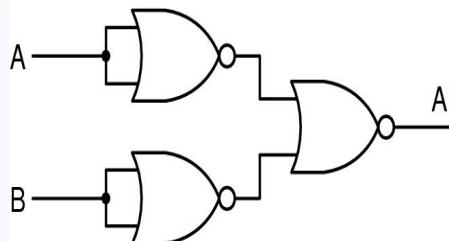
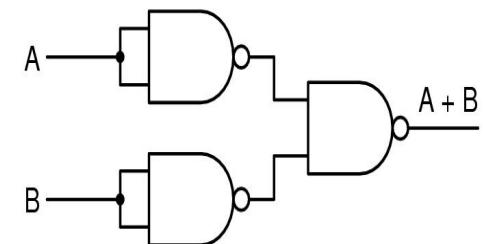
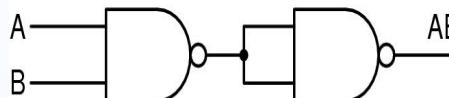


Implementazione di circuiti

- 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)

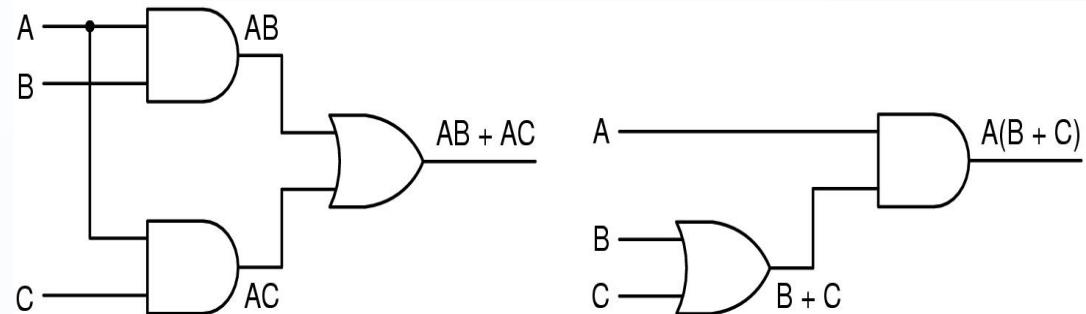


(b)

(c)

Implementazione di circuiti

- 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

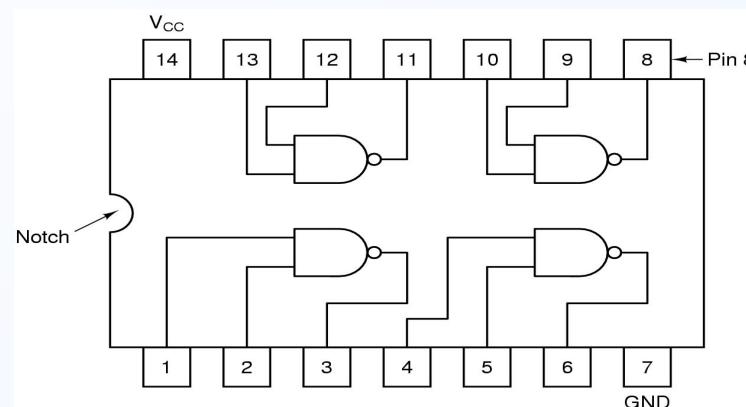
(a)

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)

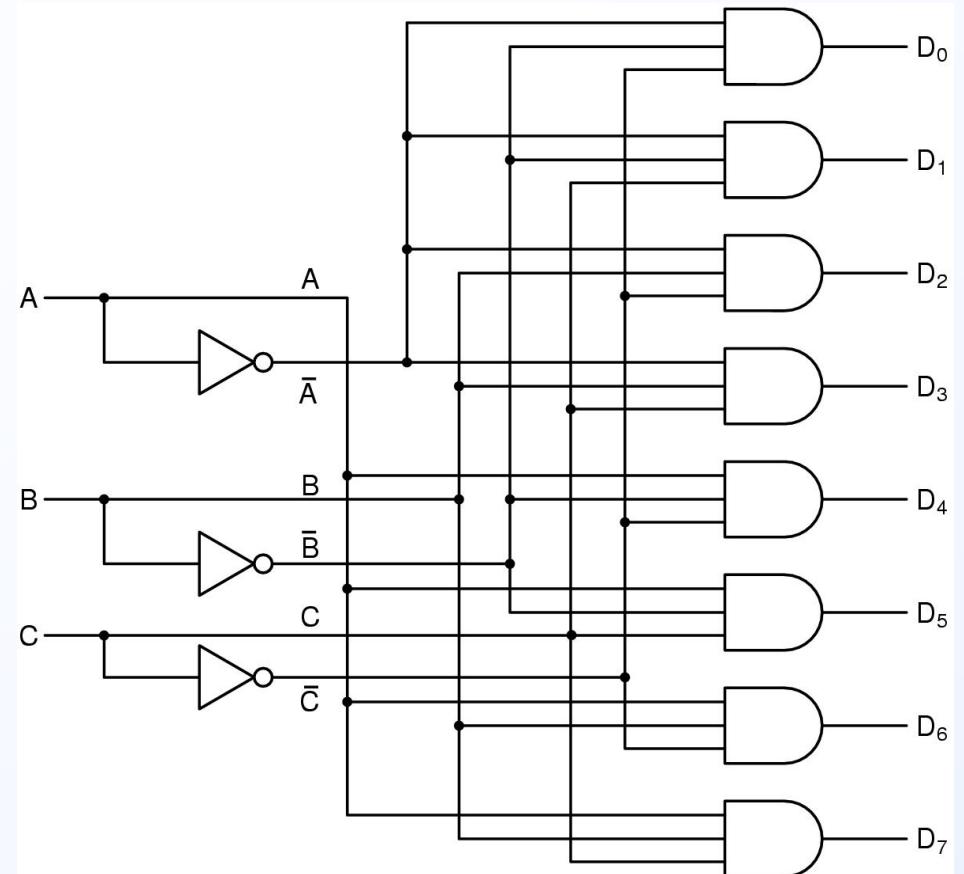
Circuiti di base

- Circuiti integrati (IC) o **chip**
- I chip si suddividono in:
 - SSI (*Small Scale Integrated*): da 1 a 10 porte
 - MSI (*Medium Scale Integrated*): da 10 a 100 porte
 - LSI (*Large Scale Integrated*): da 100 a 100.000 porte
 - VLSI (*Very Large Scale Integrated*): piu` di 100.000 porte



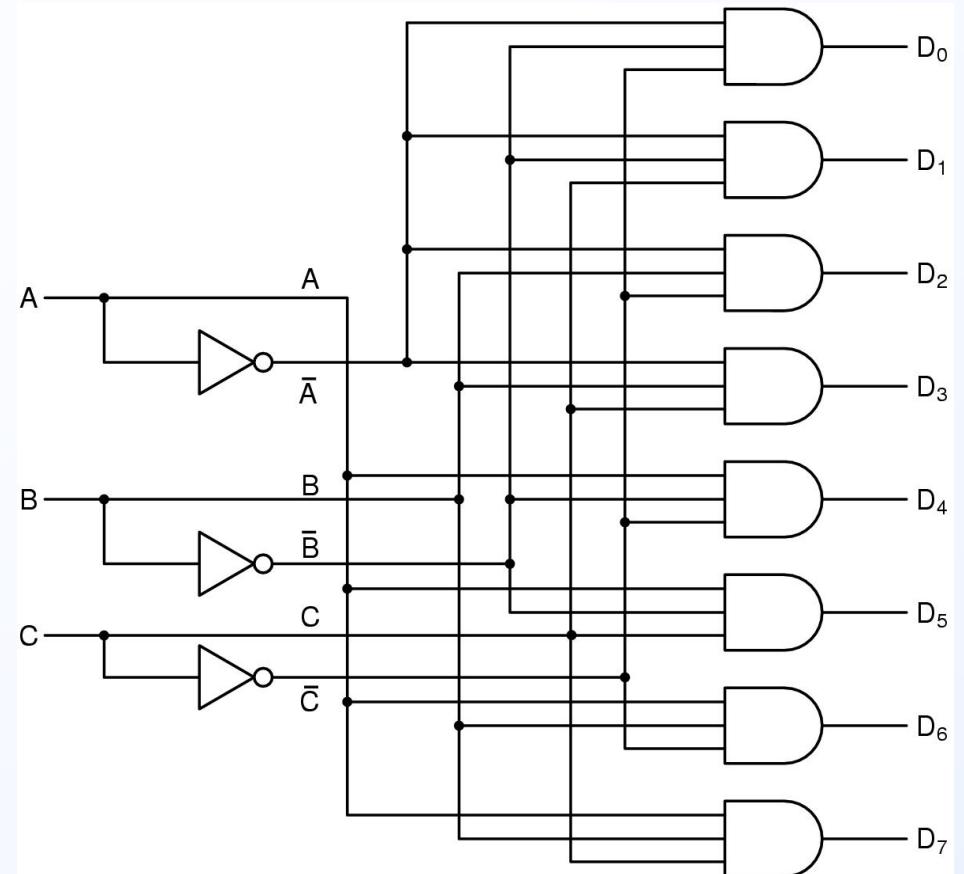
Circuiti combinatori

- **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



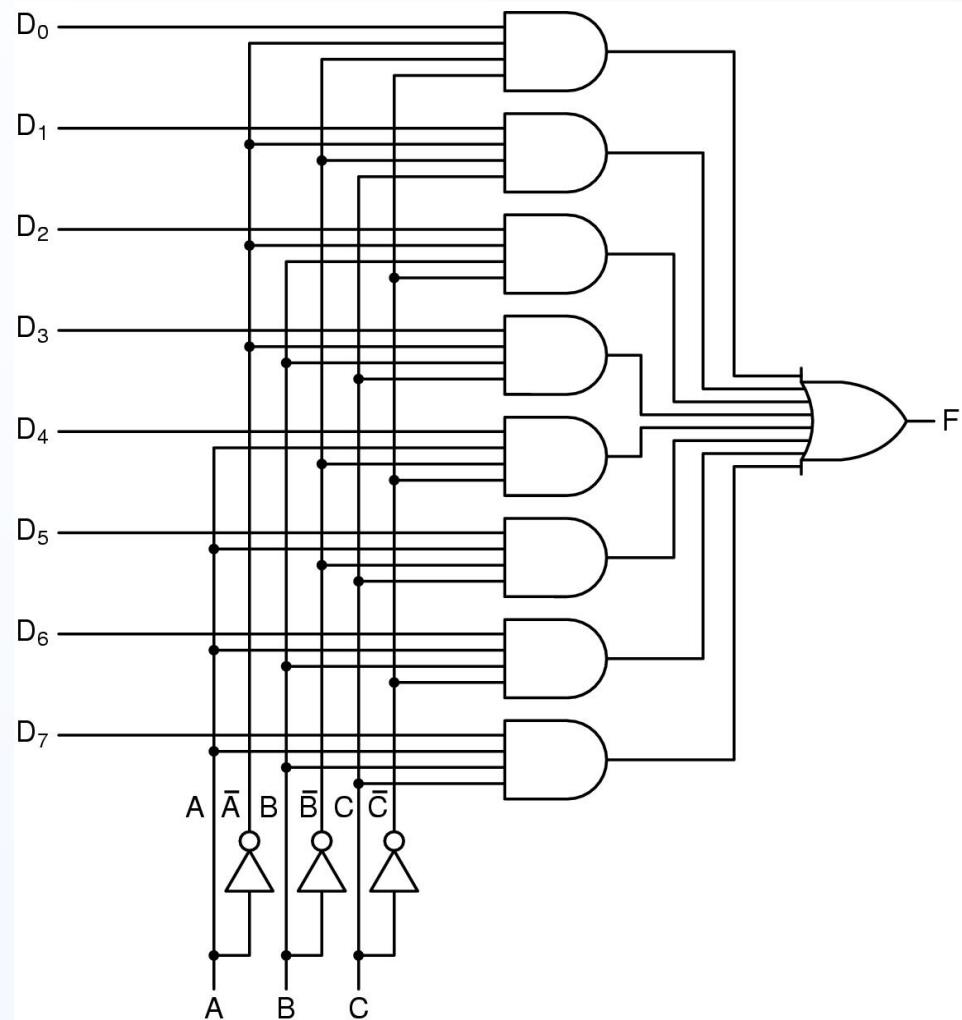
Circuiti combinatori: decoder

- × **Decoder:** prende un numero di n bit come input e lo usa per selezionare (mettere a 1) una delle 2^n linee di output
- × Può essere utilizzato per attivare una certa componente (vedi ALU più avanti), oppure un banco di memoria, ecc.
- × Interpretiamo ABC come le cifre di un numero in base 2 con A quella più significativa



Circuiti combinatori: multiplexer

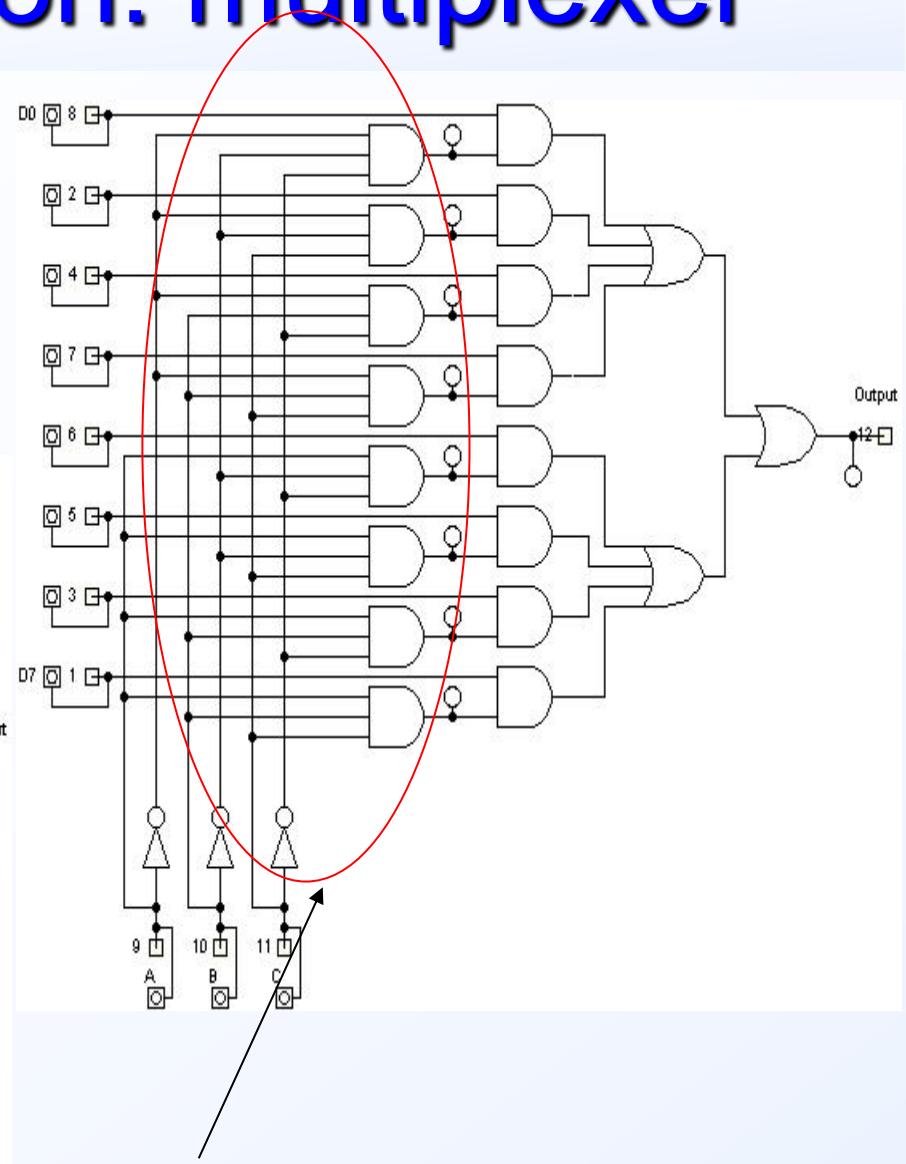
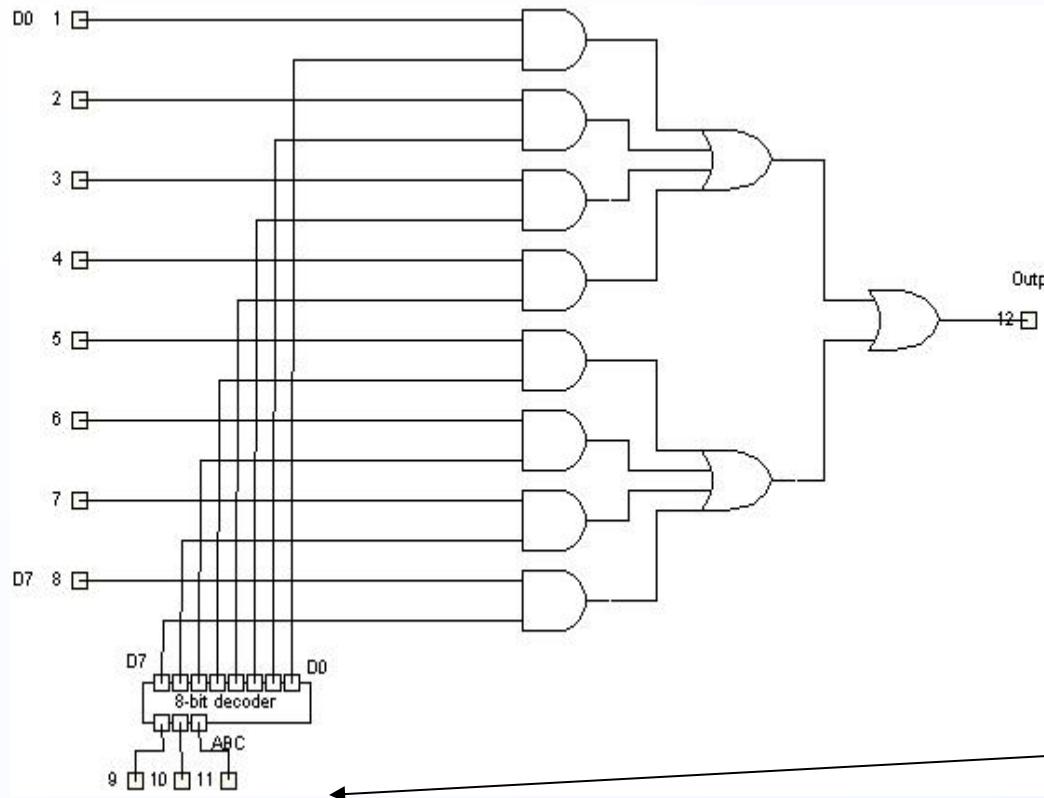
- **Multiplexer:** 2^n input, 1 output e n input di controllo
- Le linee di controllo determinano quale dei 2^n input deve essere selezionato per essere inviato all'output



8bit-
multi-
plexer-WE

Circuiti combinatori: multiplexer

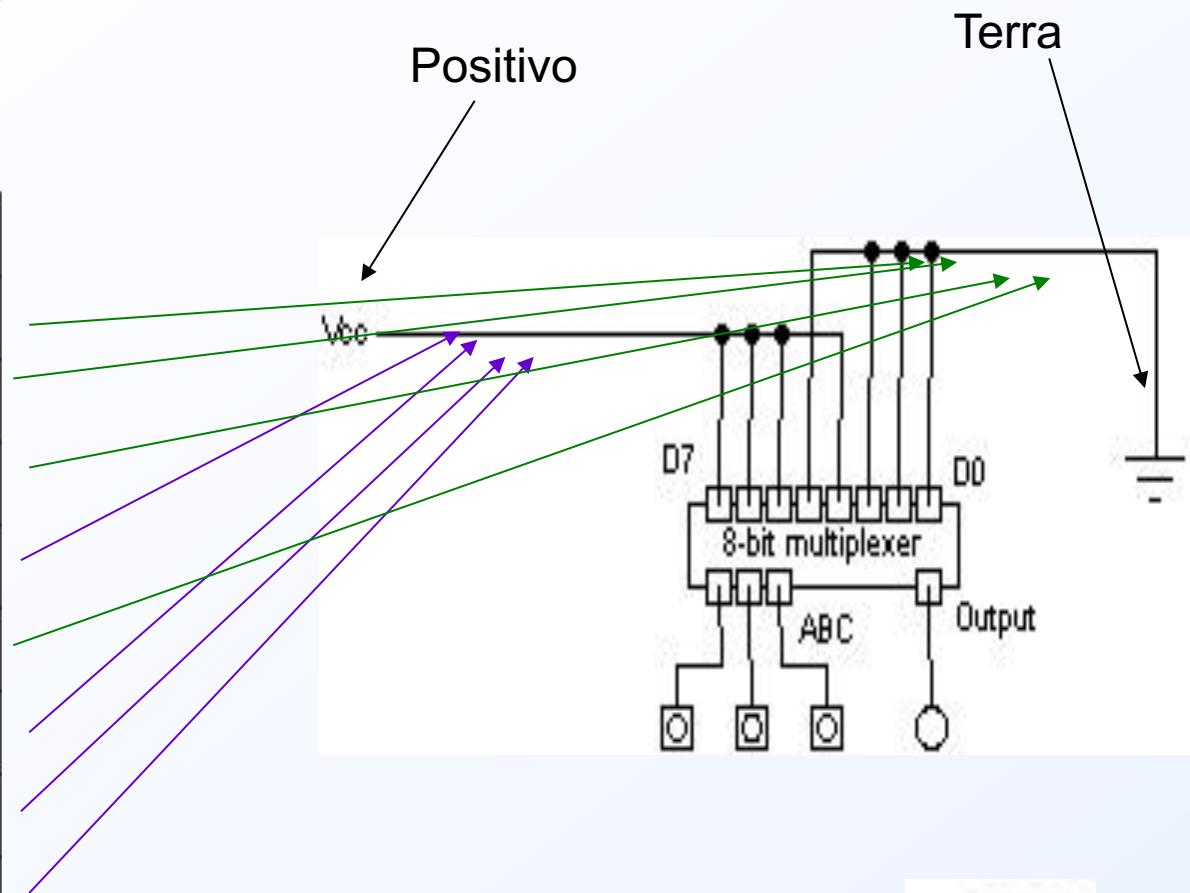
Un multiplexer è composto da un decoder più una porta AND per ogni output del decoder e l'OR finale



Circuiti combinatori: multiplexer

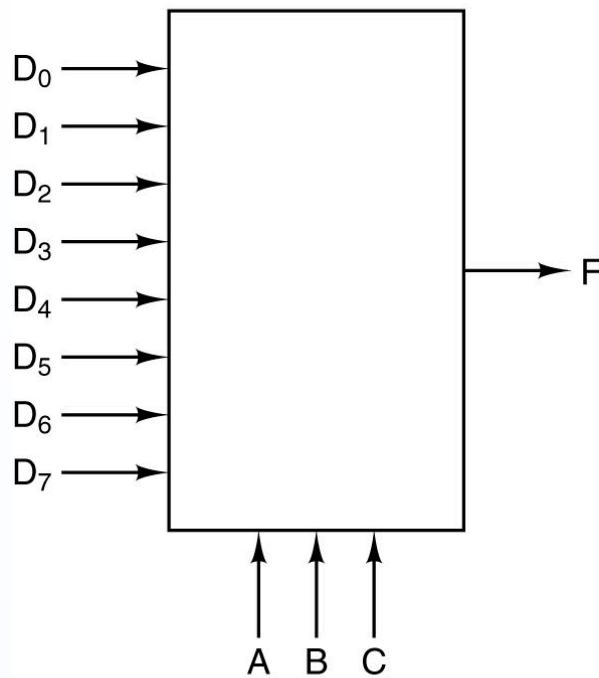
Un multiplexer con n input di controllo può essere utilizzato per implementare una qualsiasi funzione n -aria

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

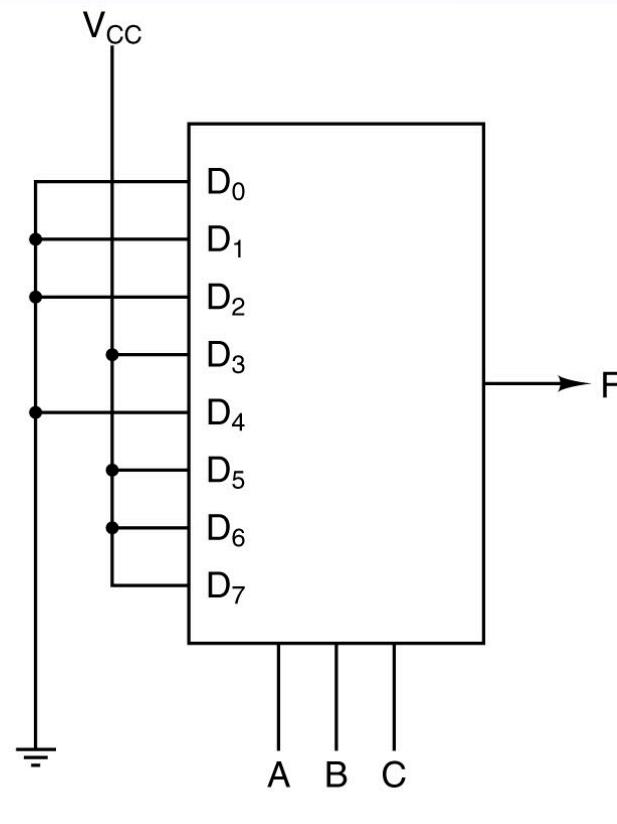


Maggioranza-
8bit-multiplexer

Circuiti combinatori: multiplexer



(a)



(b)

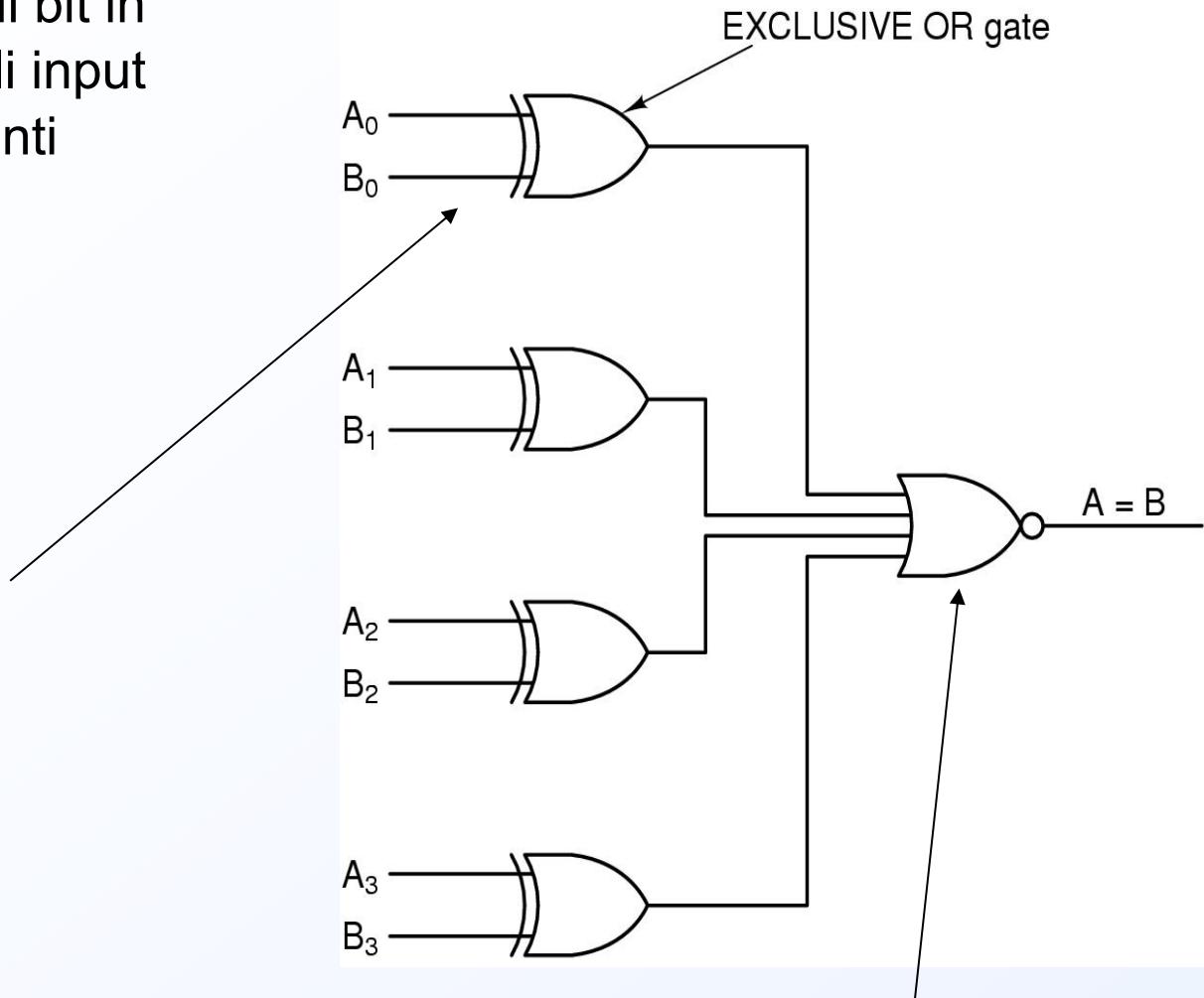
(a) MSI multiplexer.

(b) Lo stesso multiplexer usato per calcolare la funzione di maggioranza.

Circuiti combinatori: comparatori

- Confronta due serie di bit in input: produce 1 se gli input sono uguali, 0 altrimenti
- Utilizza la XOR:
EXCLUSIVE OR

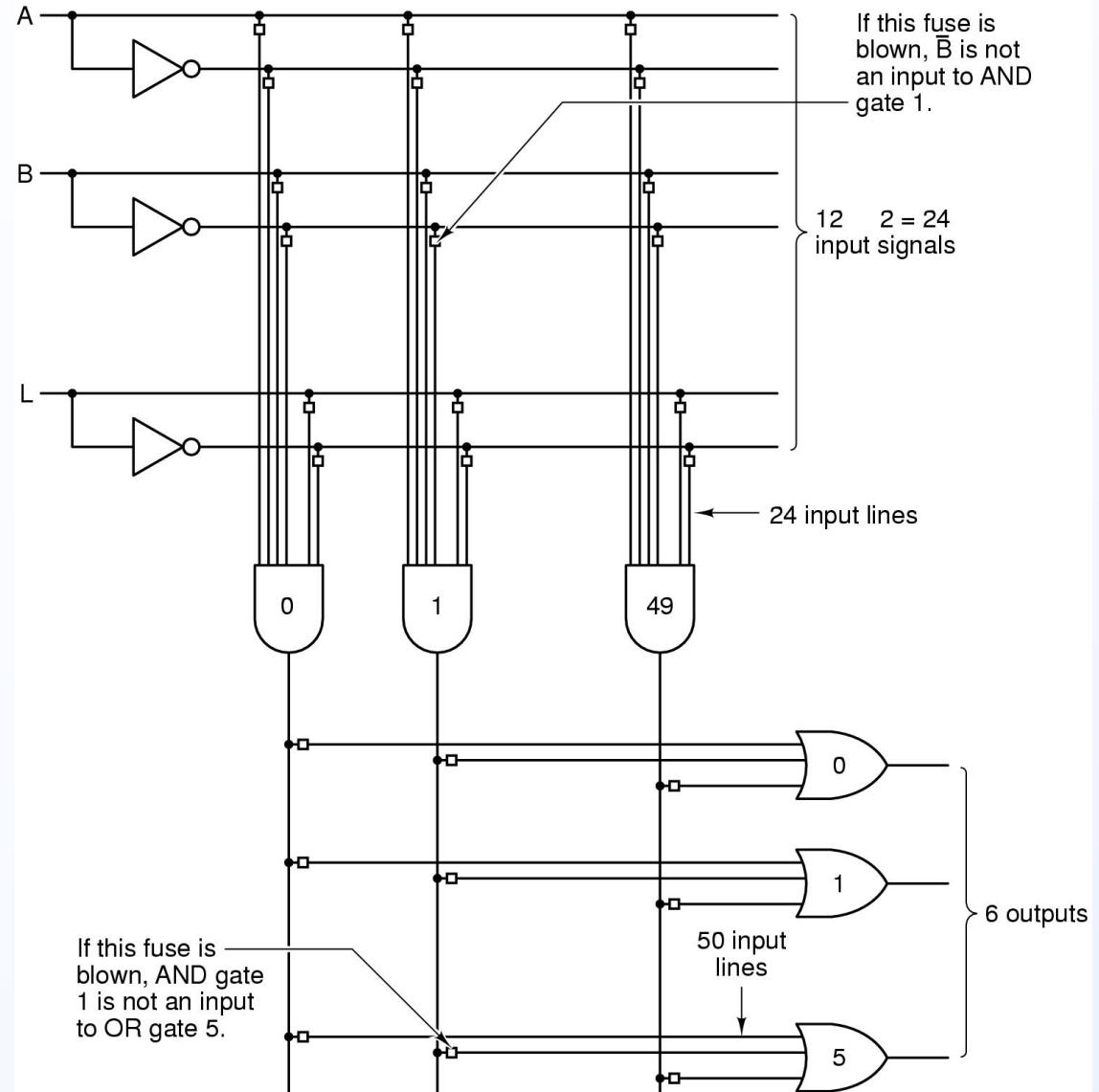
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



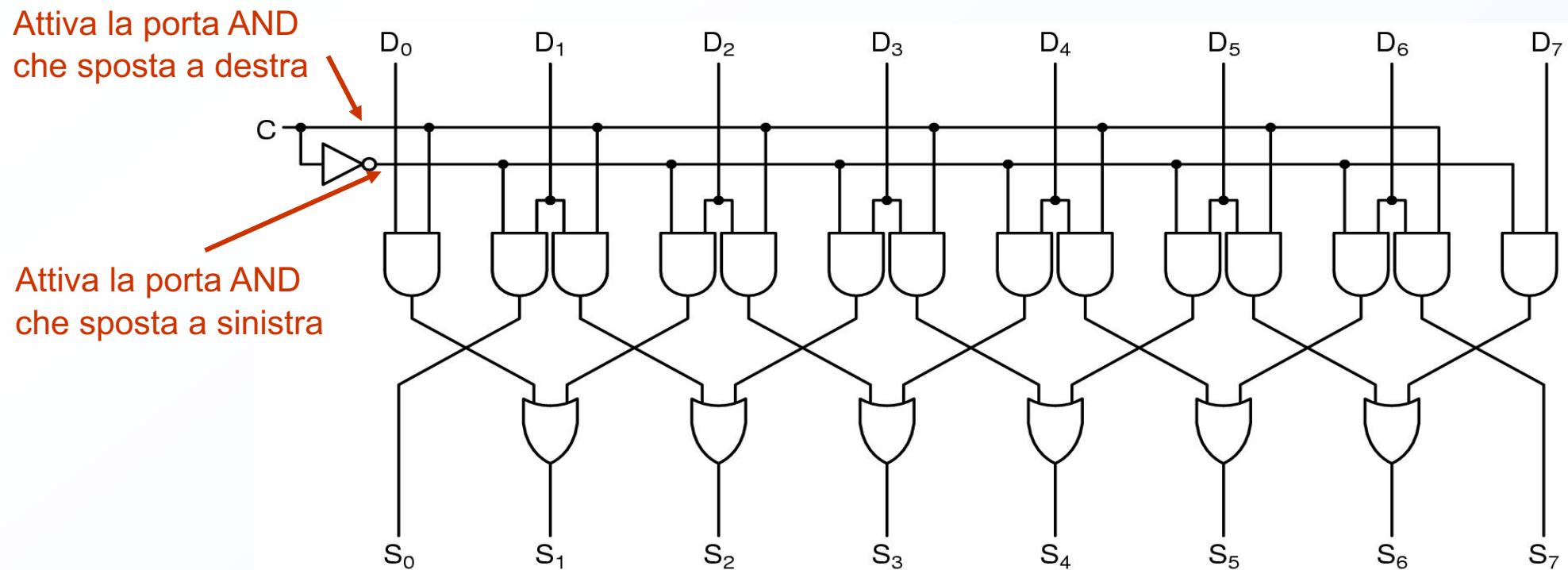
NOR: output di un OR invertito

Circuiti logici programmabili (PLA)

- Programmable Logic Array (PLA)
- Permette di implementare una tabella di verità qualsiasi (compatibilmente con gli input e output presenti nel chip)
- Fa uso dei fusibili
- Oggi non più convenienti per produzione in larga scala



Circuiti aritmetici: shifter



- L'output è l'input spostato di un bit
- Il controllo C determina la direzione dello “shift”; nel circuito presentato sopra lo shift è a sinistra se C vale 0 e il bit piu' significativo viene posto a 0, lo shift è a destra se C vale 1 e il bit meno significativo viene posto a 0

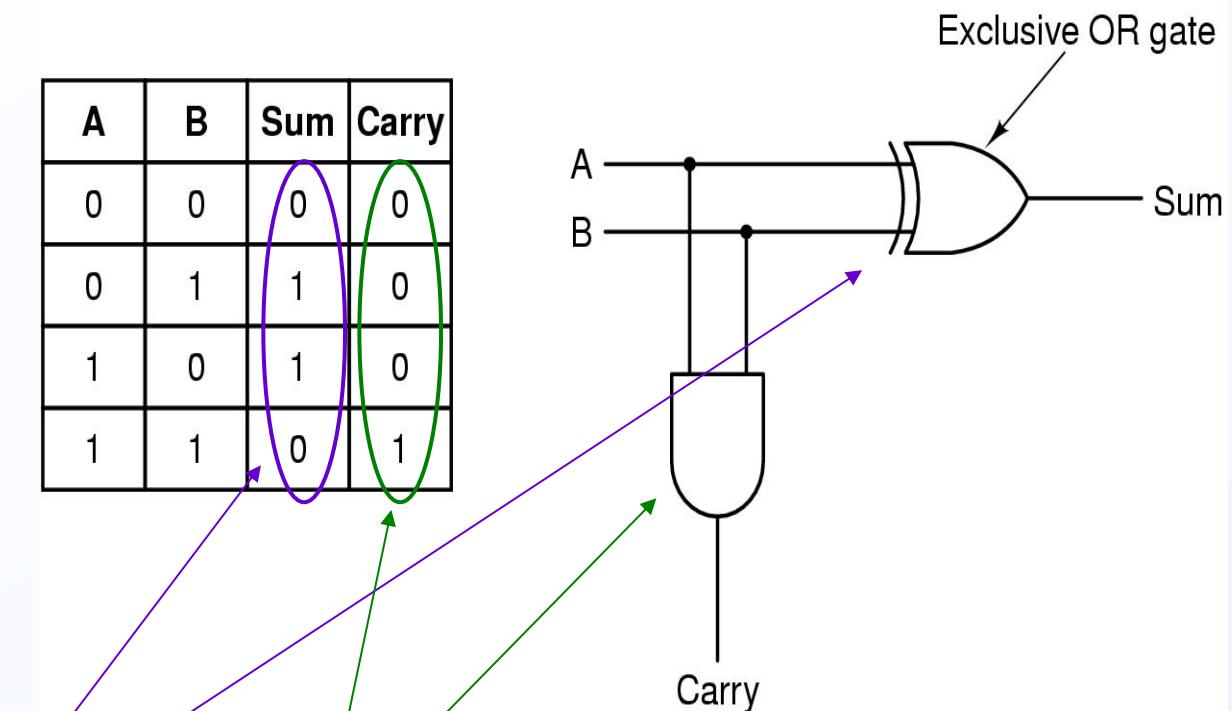
Circuiti numerici: addizionatori

Half adder

- Somma due bit in input restituendo l'eventuale riporto
- Il “semi addizionatore” va bene solo per sommare due bit che si trovano all'inizio di una sequenza di bit (altrimenti devo tenere conto del riporto generato a destra!)

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

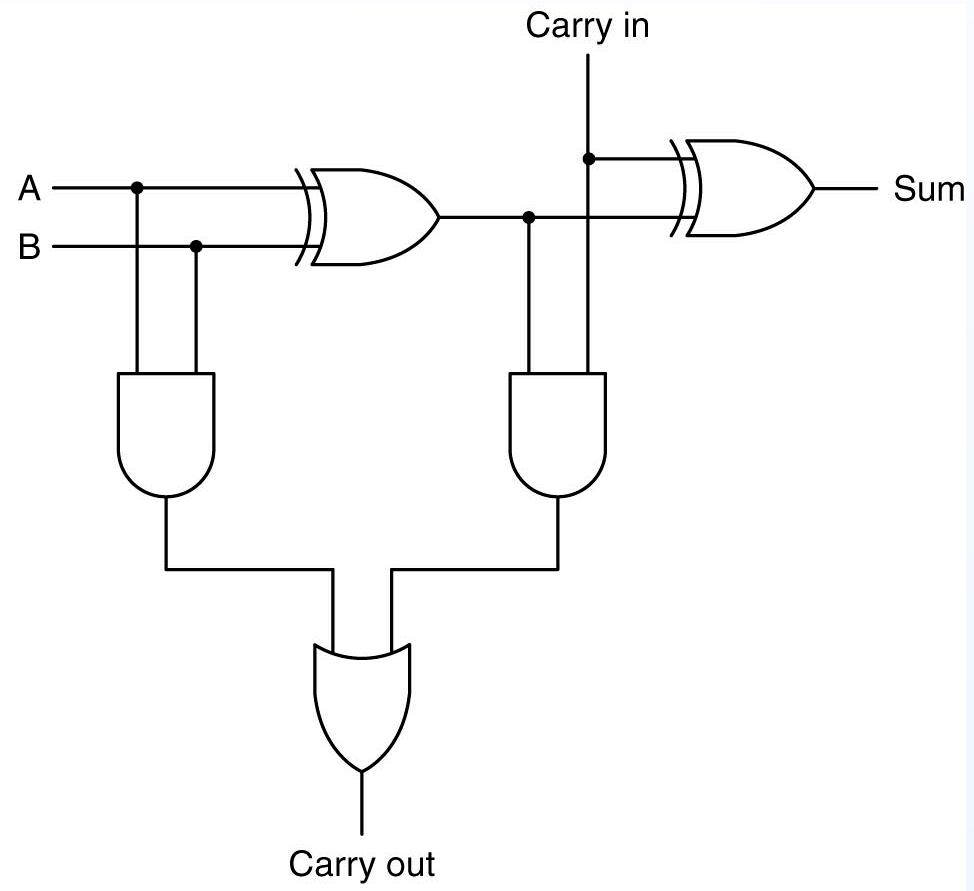
È uno XOR È un AND



Addizionatore completo

A	B	Carry in	Sum	Carry 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

(a)



(b)

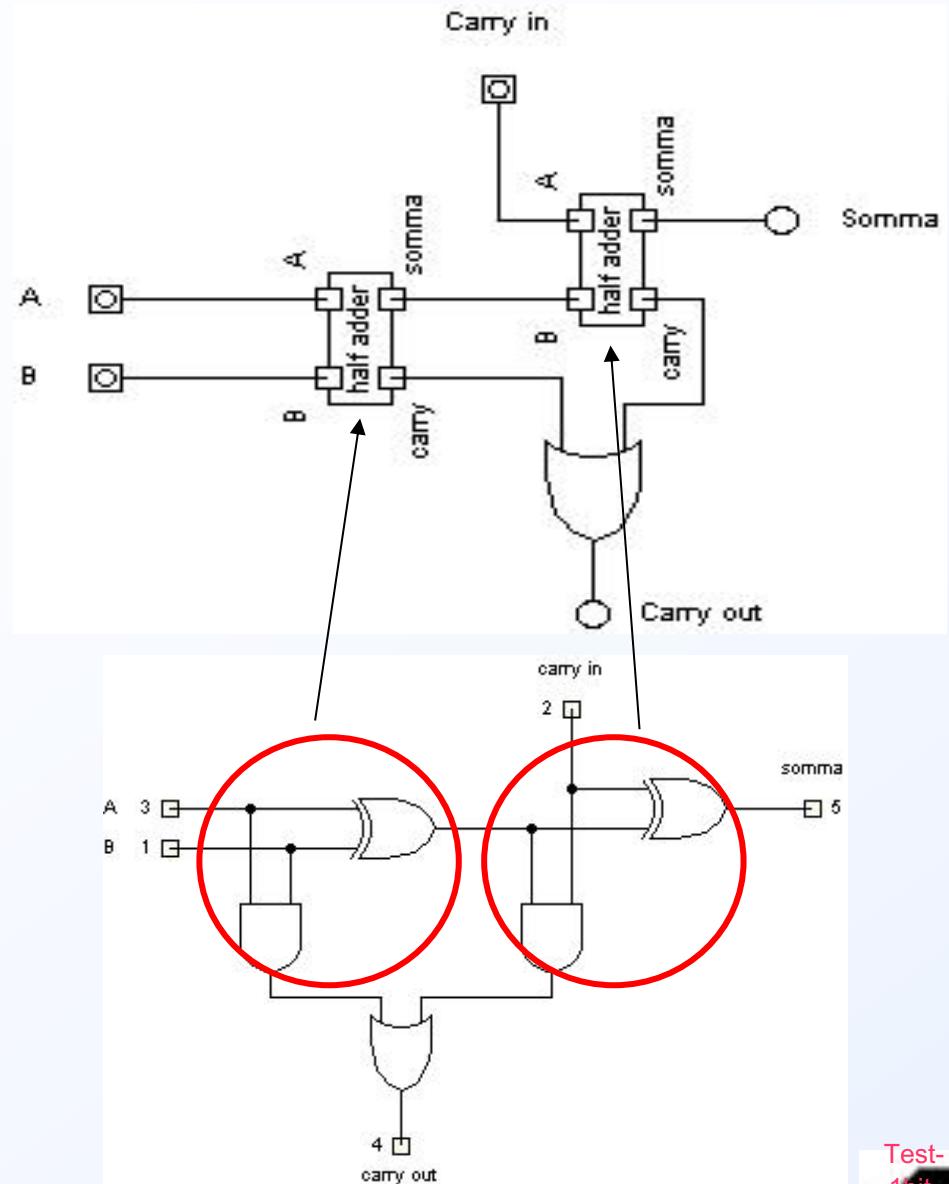
(a) Tabella di verità per l'addizionatore completo.

(b) Circuito per l'addizionatore completo.

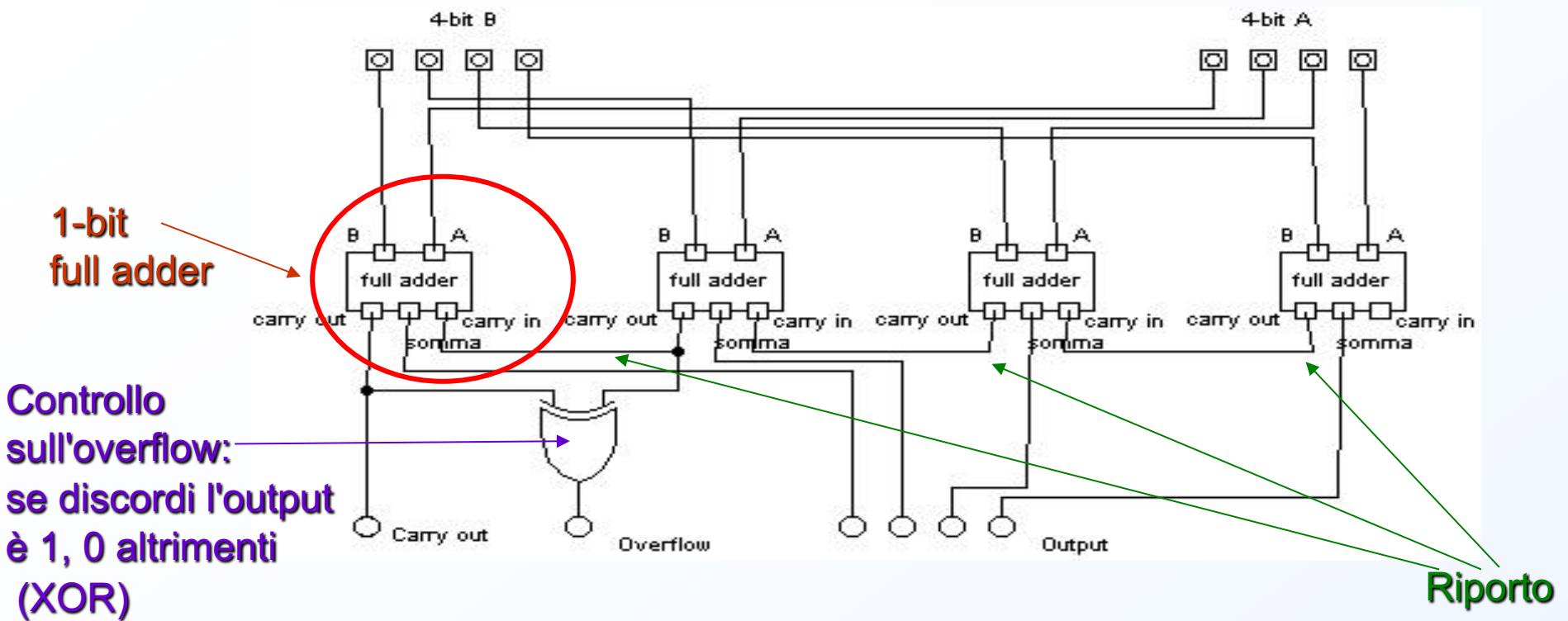
Circuiti numerici: addizionatori

Full adder

- L'addizionatore completo a 1 bit è composto da due "semi addizionatori"
-> **modularizzazione**



Circuiti numerici: addizionatori

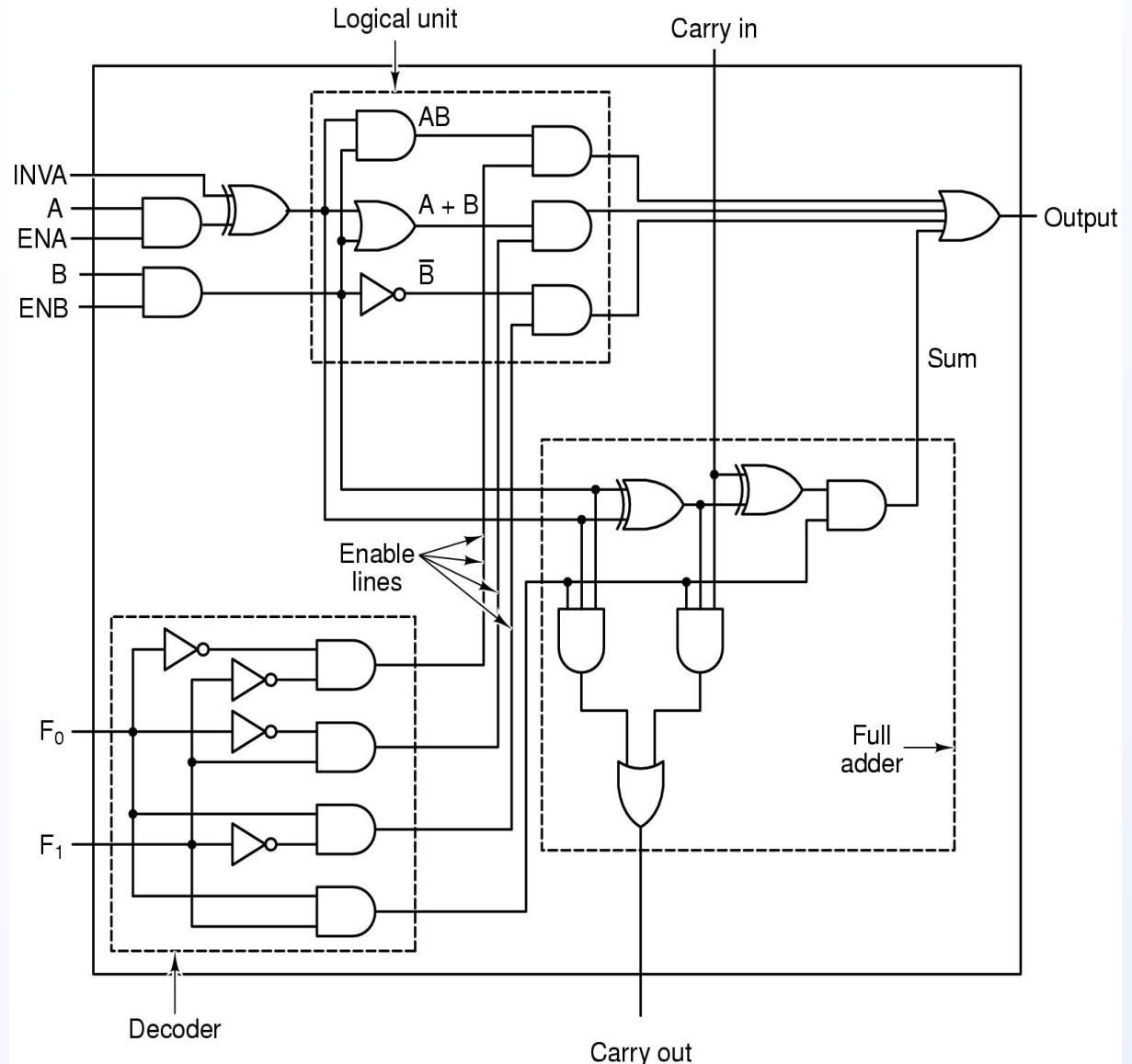


- **N bit full-adder:** un addizionatore completo a n bit si può ottenere replicando in serie n volte un addizionatore completo ad 1 bit
- Il riporto (carry out) di un bit si usa come carry in dell'addizionatore completo alla sua sinistra

Circuiti aritmetici: 1-bit ALU

Arithmetic Logic Unit (ALU)

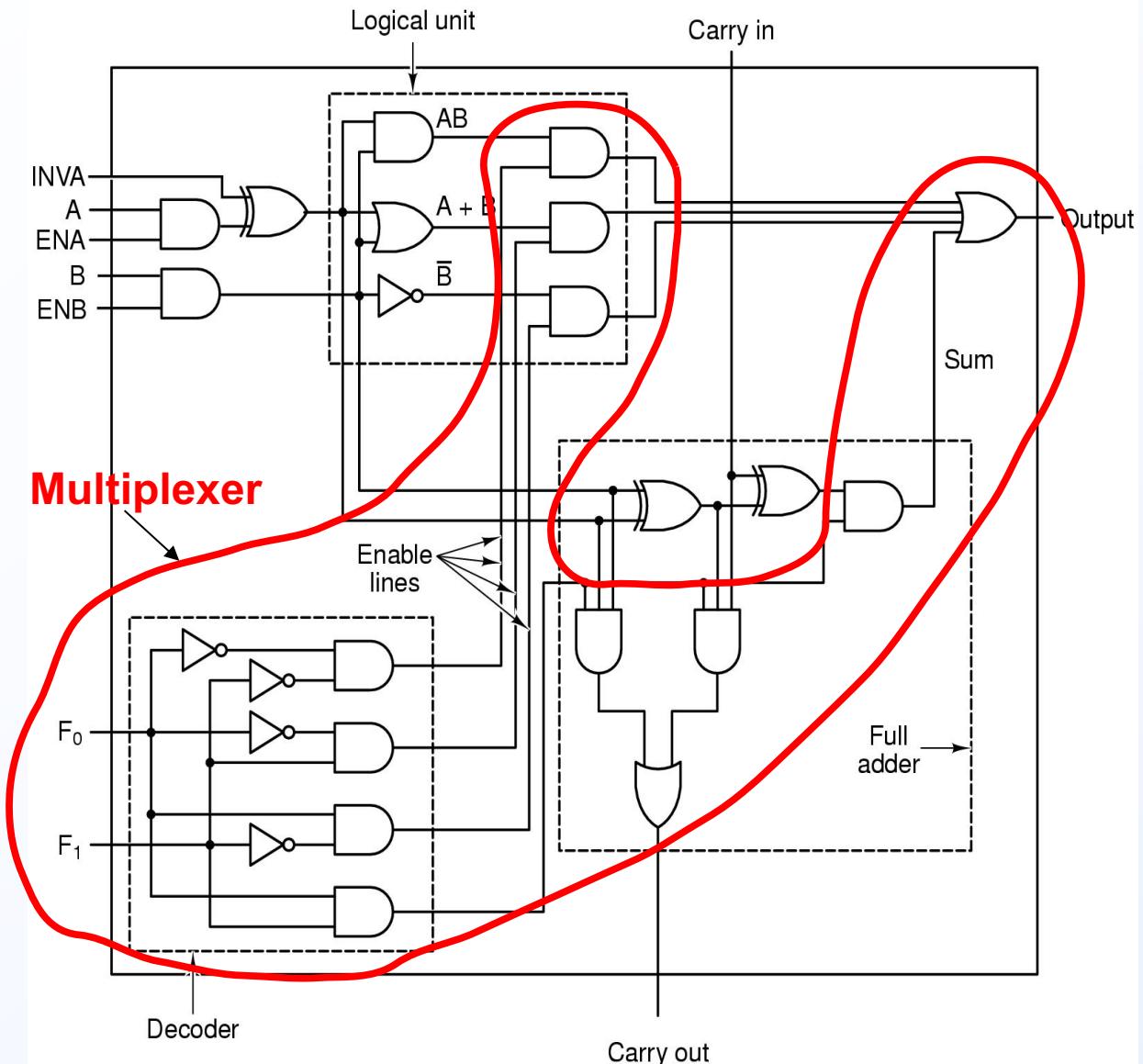
- Dati A e B è in grado di calcolare:
A “AND” B, A “OR”
B, “NOT” B,
A + B (somma)
- Input function select (un decoder!) per l'abilitazione dell'operazione desiderata (del corrispondente output)



Circuiti aritmetici: 1-bit ALU

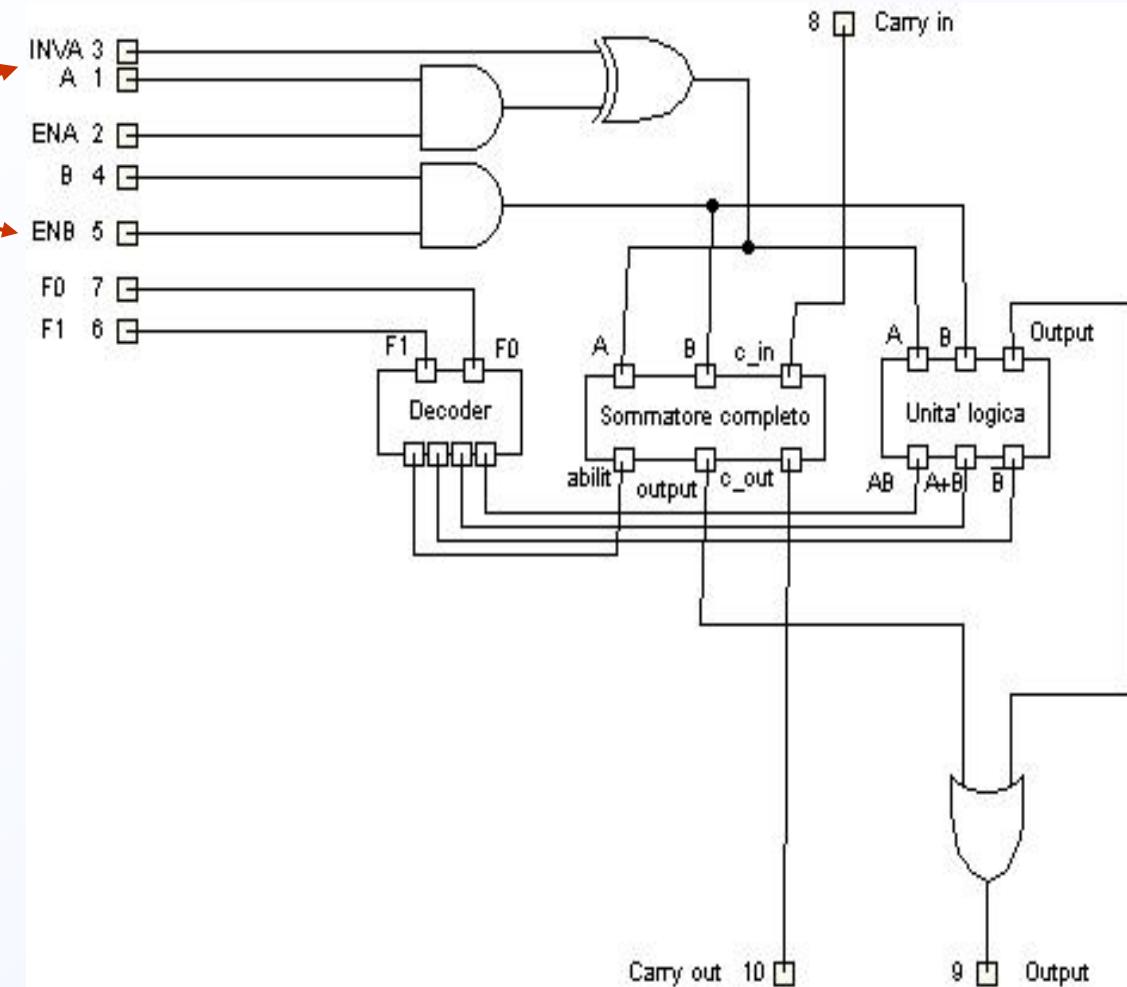
Arithmetic Logic Unit (ALU)

- Dati A e B è in grado di calcolare:
A “AND” B, A “OR” B, “NOT” B,
A + B (somma)
- Input function select (un decoder!) per l'abilitazione dell'operazione desiderata (del corrispondente output)

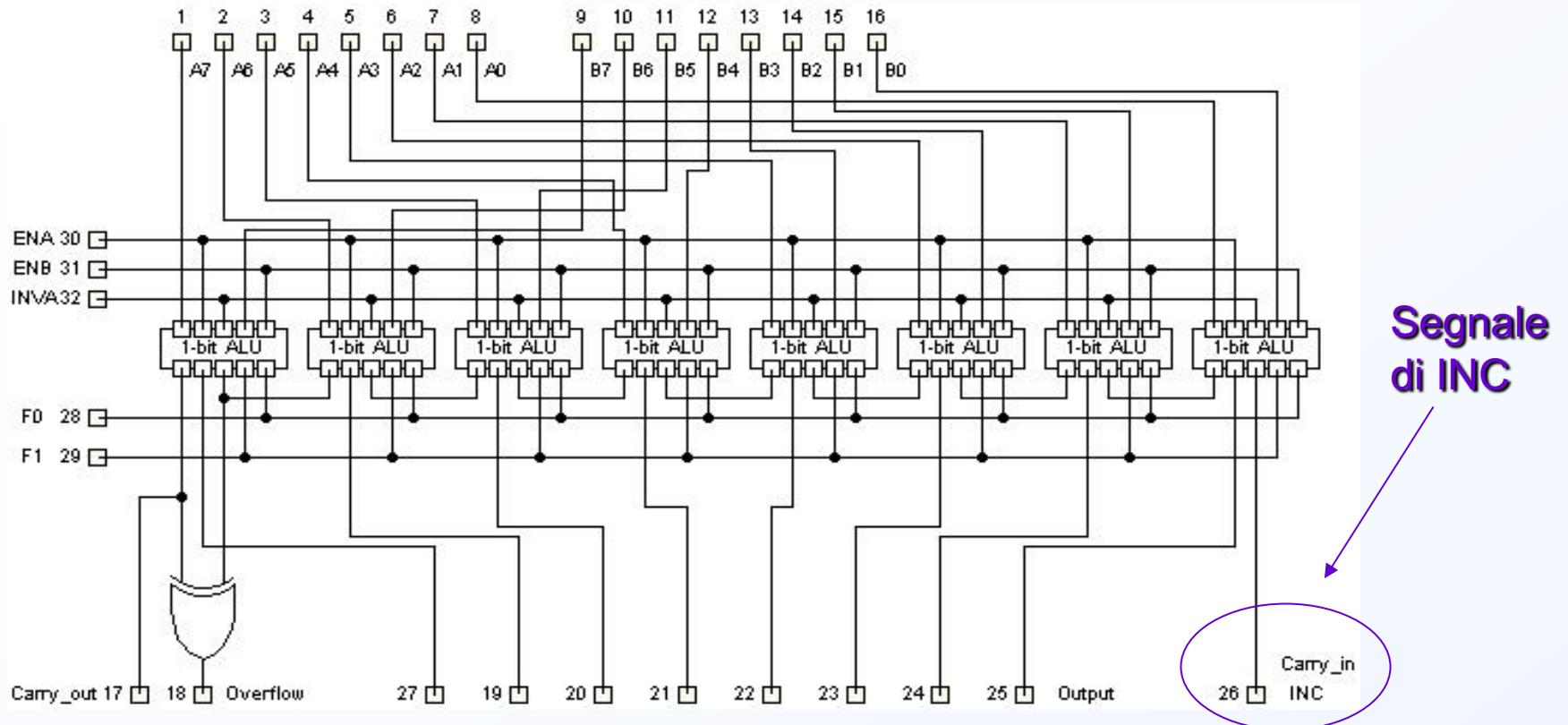


Circuiti aritmetici: 1-bit ALU

- Segnali di abilitazione anche per gli input A e B (ENA e ENB)
- Se INVA è a 1 viene passato in input il complemento di A anzichè A stesso
- Condizioni normali: ENA e ENB impostati a 1, INVA a 0
- **Bit slice**

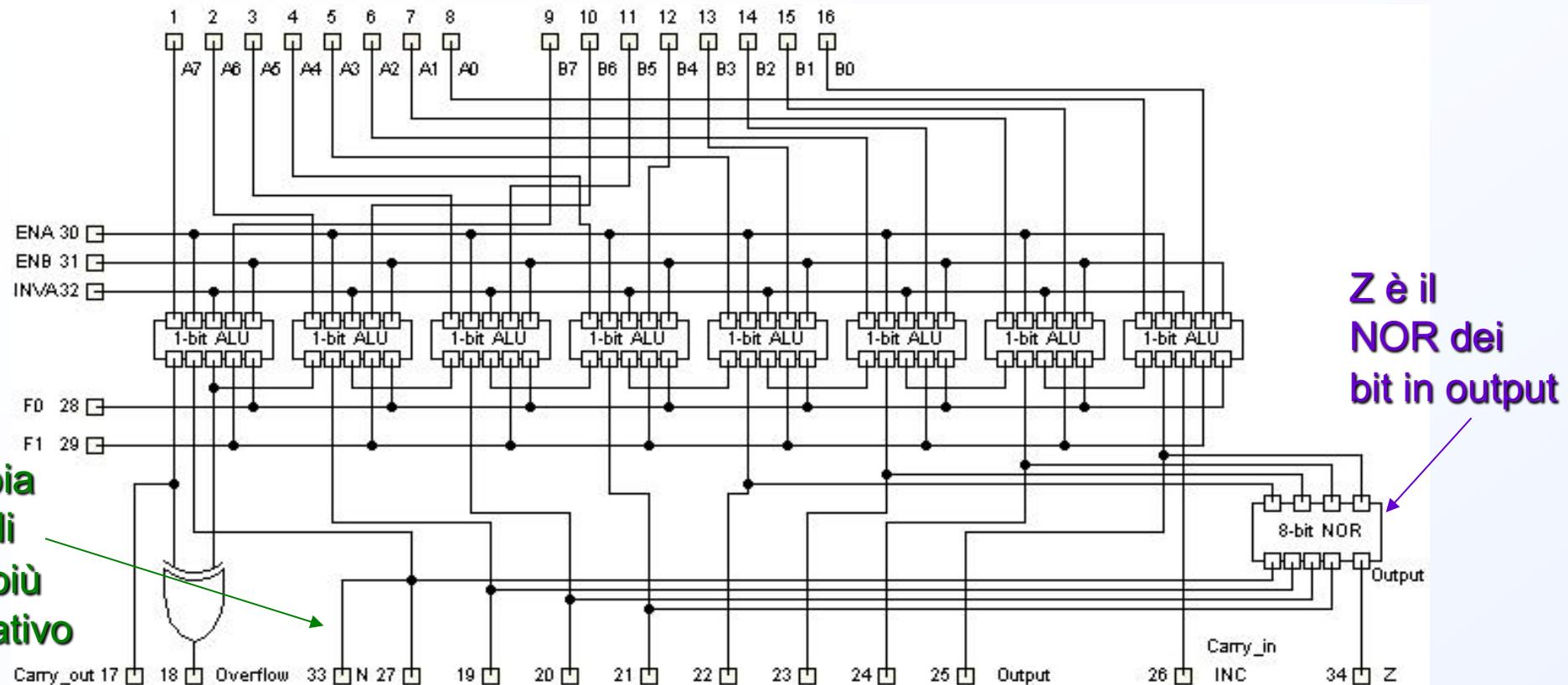


Circuiti aritmetici: 8-bit ALU



- Una n -bit ALU si ottiene collegando in serie n bit slice
- Il carry in del bit meno significativo può essere usato come segnale di INC: nell'addizione incrementa il risultato di 1 ($A + B + 1$, $A + 1$)

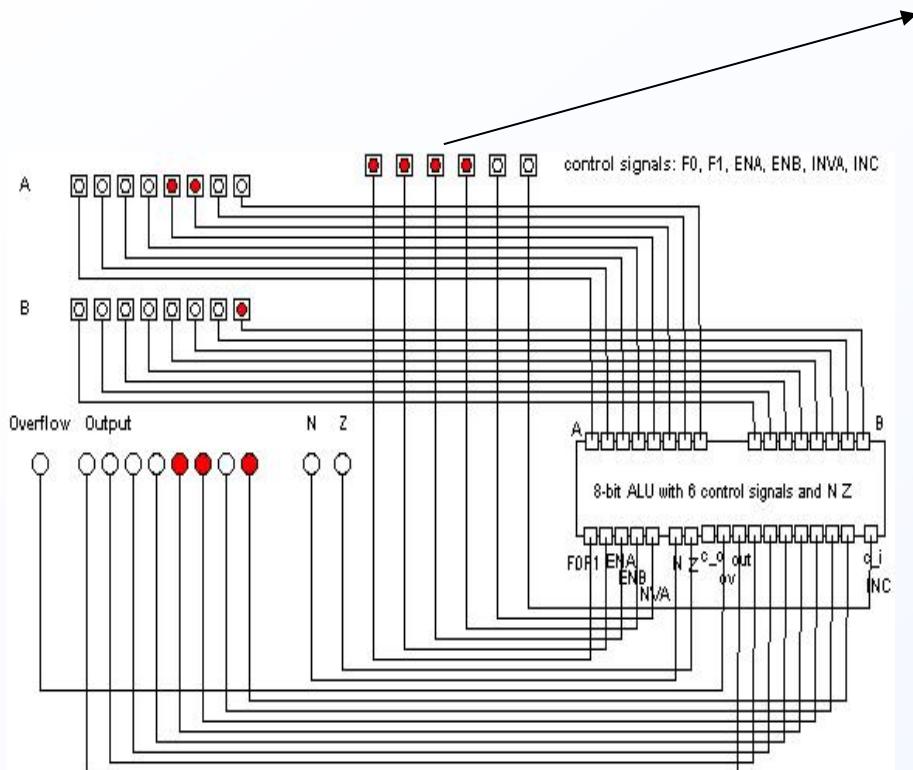
Circuiti aritmetici: 8-bit ALU with Z N



Segnali di output Z e N

- Z vale 1 se l'output è uguale a zero, 0 altrimenti
- N vale 1 se l'output è un numero negativo, 0 altrimenti

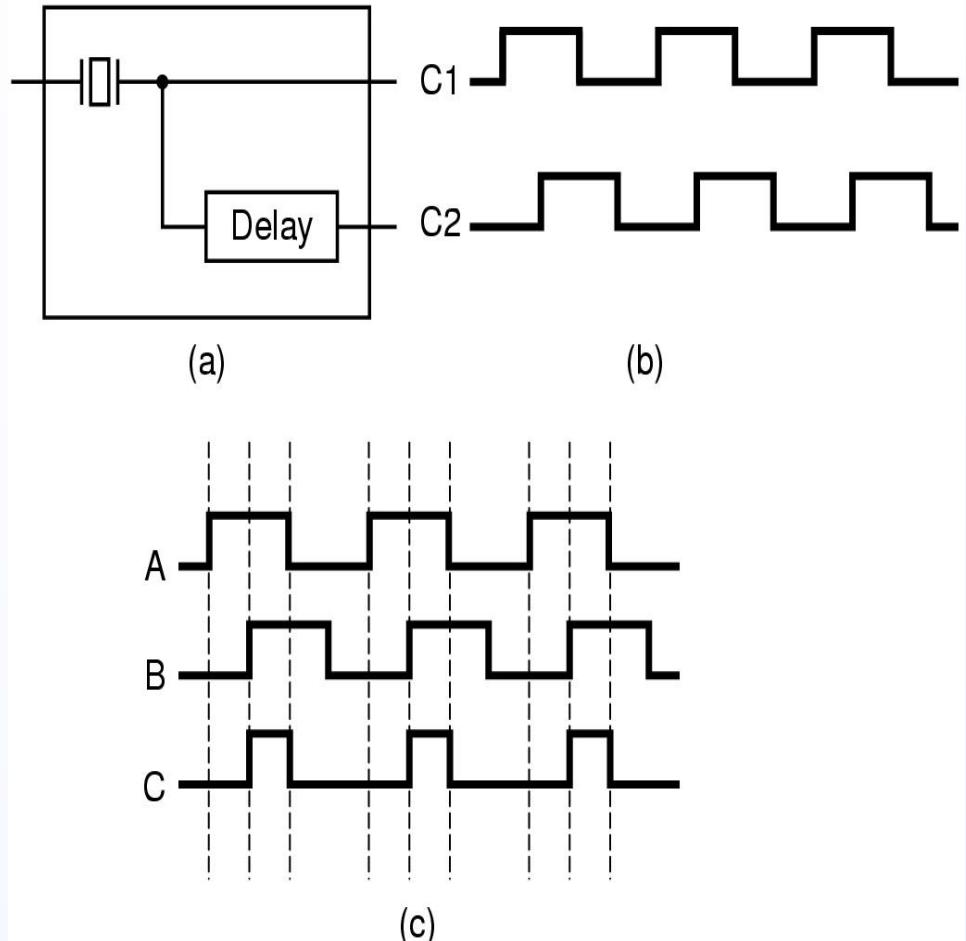
Circuiti aritmetici: 8-bit ALU with Z N



F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	\bar{A}
1	0	1	1	0	0	\bar{B}
1	1	1	1	0	0	$A + B$
1	1	1	1	0	1	$A + B + 1$
1	1	1	0	0	1	$A + 1$
1	1	0	1	0	1	$B + 1$
1	1	1	1	1	1	$B - A$
1	1	0	1	1	0	$B - 1$
1	1	1	0	1	1	$-A$
0	0	1	1	0	0	$A \text{ AND } B$
0	1	1	1	0	0	$A \text{ OR } B$
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	-1

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



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:

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

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

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