

# Corso di Architettura degli Elaboratori

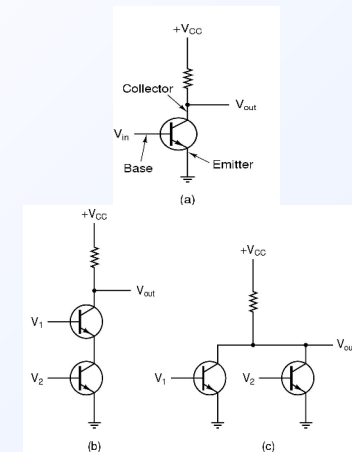
## Il livello logico digitale: Algebra Booleana e Circuiti logici digitali di base

Matteo Baldoni

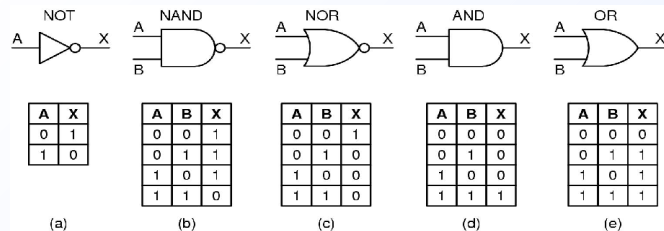
Dipartimento di Informatica  
Università degli Studi di Torino  
C.so Svizzera, 185 – I-10149 Torino  
baldoni@di.unito.it  
<http://www.di.unito.it/~baldoni>

# Porte logiche

- × (a) invertitore a transistor: quando  $V_{in}$  è basso,  $V_{out}$  è alto e viceversa
- × Pochi **nanosecondi** per passare da uno stato all'altro: "alto" (tensione  $V_{CC}$ ) 1 logico, "basso" (terra) 0 logico
- × **Logica positiva**
- × (b) una **porta NAND**: due transistor collegati in serie
- × (c) una **porta NOR**: due transistor collegati in parallelo



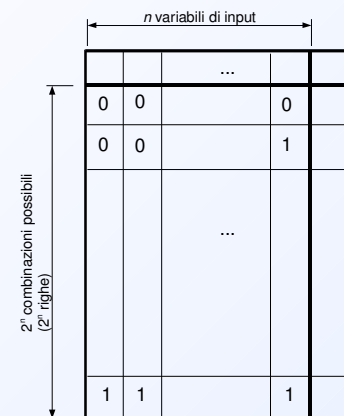
# Porte logiche



- × NOT: un transistor
- × NAND e NOR: due transistor
- × AND e OR: tre transistor (quelli del NAND e NOR, rispettivamente, più un invertitore)

# Algebra Booleana

- × Funzioni nell'algebra Booleana (George Boole, 1815-1864): funzioni a due soli valori, 0 e 1
- × **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:  $2^n$  funzioni (es., se  $n = 2$  allora 16 funzioni diverse)



## Algebra Booleana

- x AB: A "and" B
- x A + B: A "or" B
- x  $\bar{A}$ : "not" A

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

- x *Esempio:*  $\bar{A}B + \bar{A}BC$  (e' vero solo quando  $A = 1$  e  $B = 0$  oppure  $B = 1$  e  $C = 0$ )

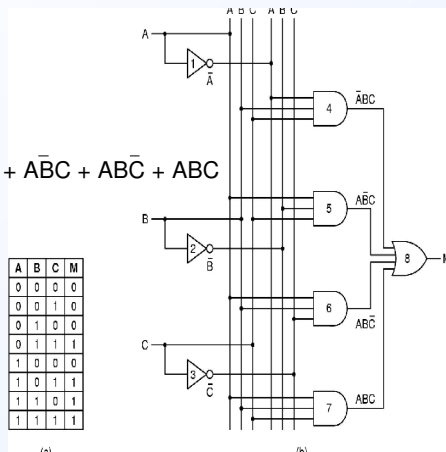
## Algebra Booleana

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

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

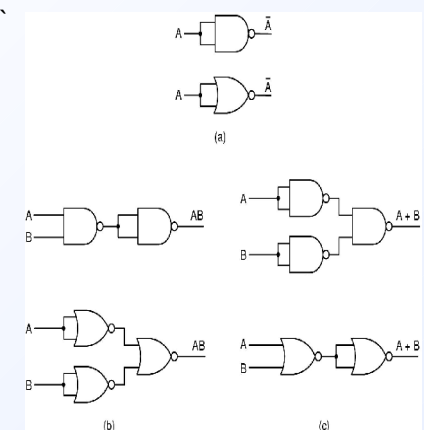
## Algebra Booleana: implementazione

1. Scrivere la tabella di verita' 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



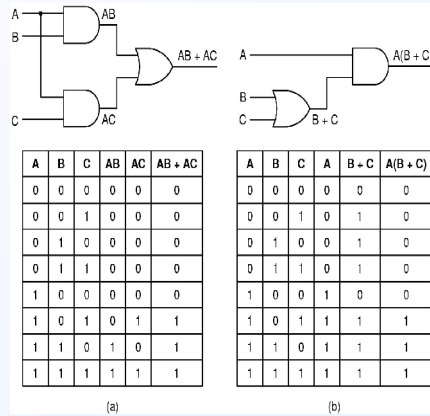
## Algebra Booleana: implementazione

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



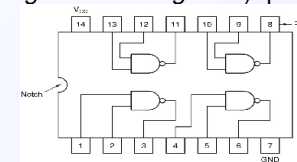
## Algebra Booleana: implementazione

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



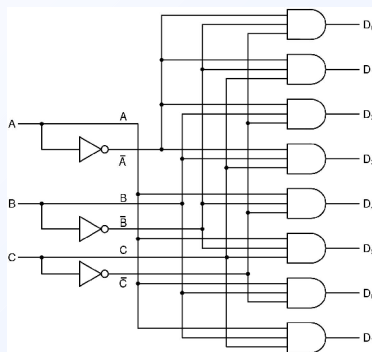
## Circuiti di base

- × Circuiti integrati (IC) o *chip*
- × Dual Inline Package (DIP)
- × 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*): più di 100.000 porte



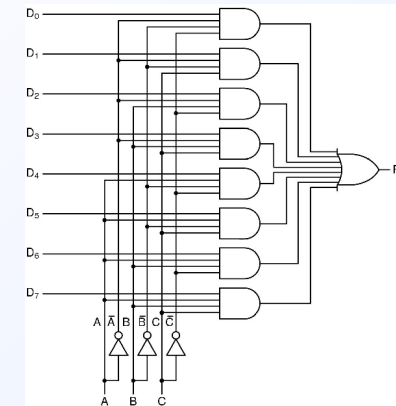
## Circuiti combinatori: decoder

- × **Circuito combinatorio:** l'output viene determinato solo dagli input del momento
- × **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.



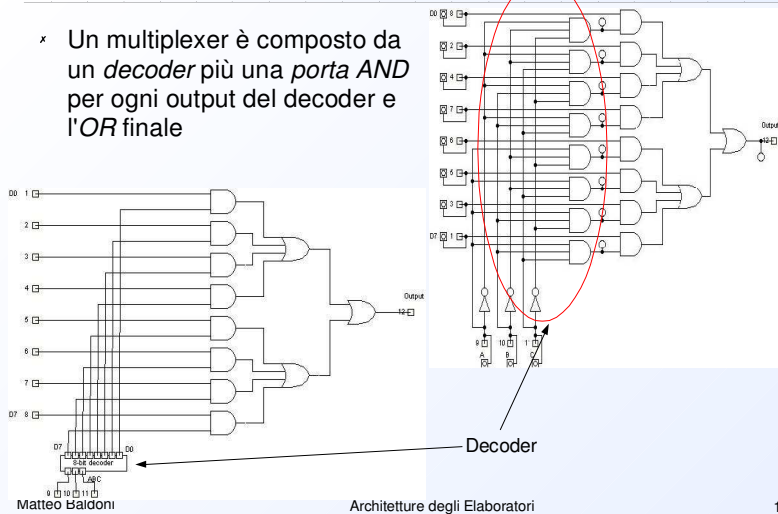
## Circuiti combinatori: multiplexer

- × **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
- × **Demultiplexer:** invia il segnale di input ad uno dei  $2^n$  output, a seconda dei valori delle  $n$  linee di controllo



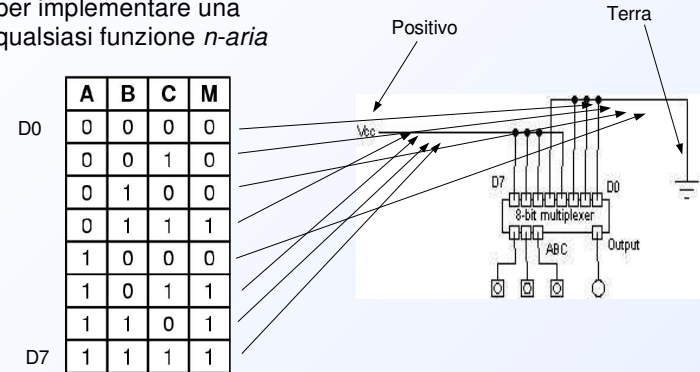
# Circuiti combinatori: multiplexer

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



# Circuiti combinatori: multiplexer

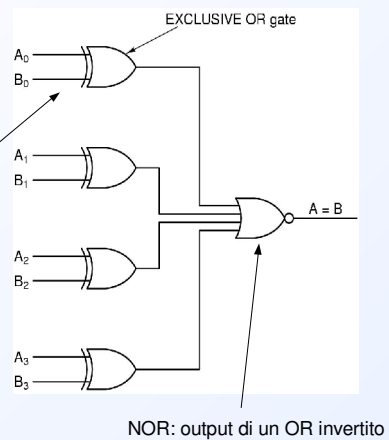
- Un multiplexer con  $n$  input di controllo può essere utilizzato per implementare una qualsiasi funzione  $n$ -aria



# Circuiti combinatori: comparatori

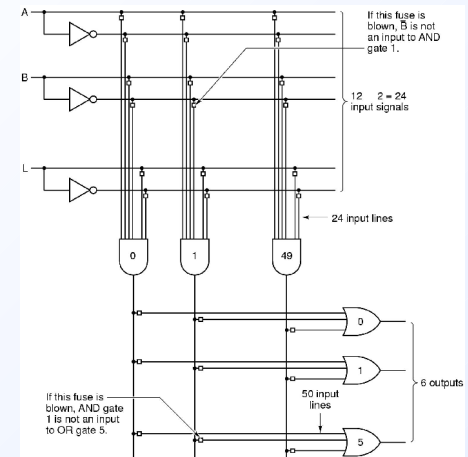
- Confronta due serie di bit in input: produce 1 se gli input sono uguali, 0 altrimenti
- Utilizza la XOR: ECLUSIVE OR

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

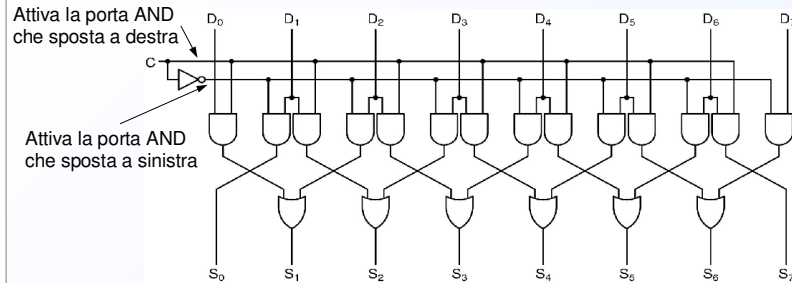


# Circuiti logici programmabili (PLA)

- 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



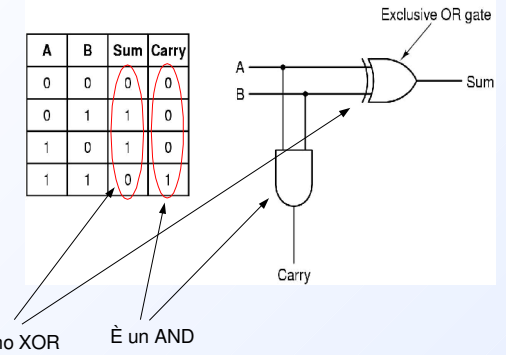
## Circuiti aritmetici: shifter



- x L'output è l'input spostato di un bit
- x Il controllo C determina la direzione dello "shift", a sinistra se C vale 0, a destra se C vale 1

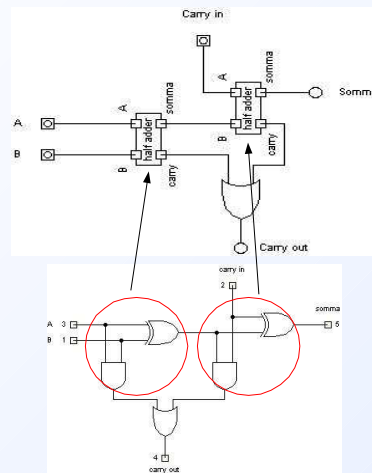
## Circuiti aritmetici: half adder

- x Somma due bit in input restituendo l'eventuale riporto



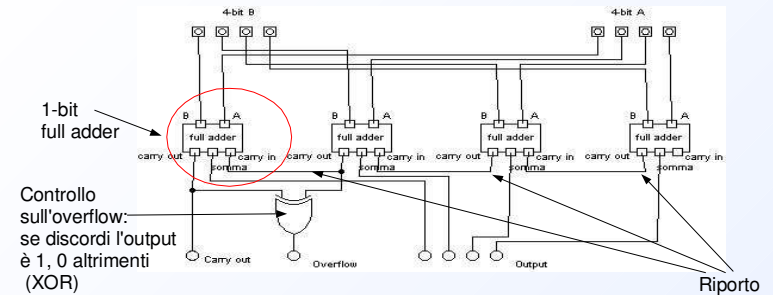
## Circuiti aritmetici: full adder

- x Il "mezzo sommatore" va bene solo per sommare due bit che si trovano all'inizio di una sequenza di bit (devo tenere conto del riporto generato a destra!)
- x Il **sommatore completo** è composto da due "mezzi sommatore"



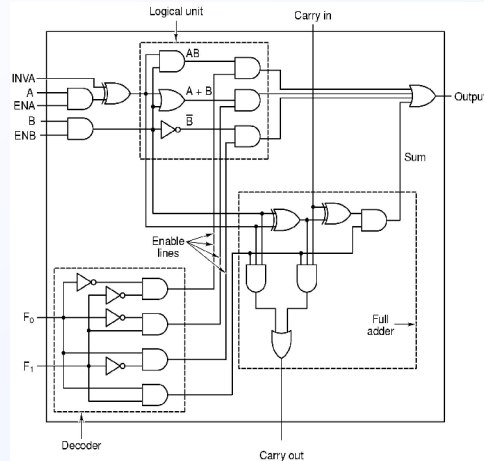
## Circuiti aritmetici: n-bit full adder

- x Un sommatore completo a  $n$  bit si può ottenere replicando in serie  $n$  volte un sommatore completo
  - x Il riporto (carry out) di un bit si usa come carry in dell'addizionatore completo alla sua sinistra
- Controllo sull'overflow: se discordi l'output è 1, 0 altrimenti (XOR)



## 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)



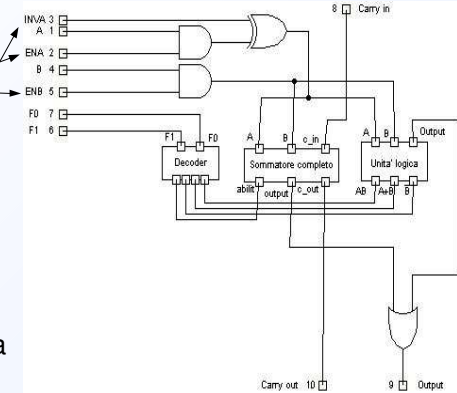
Matteo Baldoni

Architetture degli Elaboratori

21

## 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 anziché A stesso
- Condizioni normali: ENA e ENB impostati a 1, INVA a 0
- Bit slice**

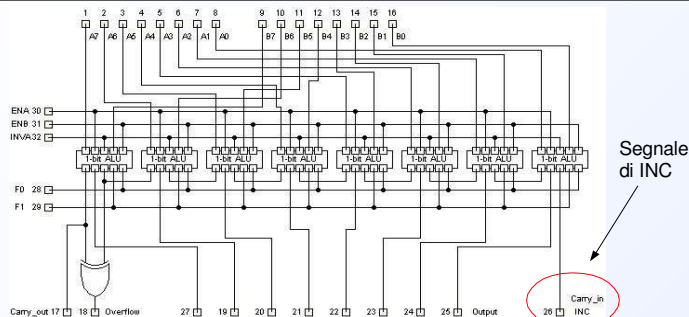


Matteo Baldoni

Architetture degli Elaboratori

22

## Circuiti aritmetici: 8-bit ALU



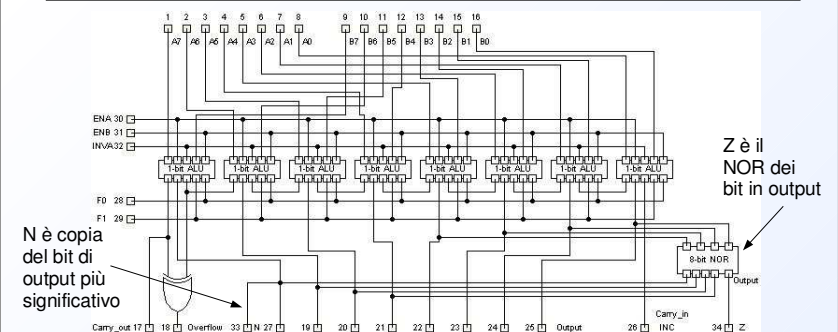
- Una  $n$ -bit ALU si ottiene collegando in serie  $n$  bit slice
- Il carry in del bit meno significativo può essere usato come segnale di INC: nell'addizione incrementa il risultato di 1 ( $A + B + 1$ ,  $A + 1$ )

Matteo Baldoni

Architetture degli Elaboratori

23

## Circuiti aritmetici: 8-bit ALU with Z N



- Segnali di output Z e N [Tanenbaum, Structured Computer Organization, Third Edition, pagina 166, sez. 4.1.4]:
  - Z vale 1 se l'output è uguale a zero, 0 altrimenti
  - N vale 1 se l'output è un numero negativo, 0 altrimenti

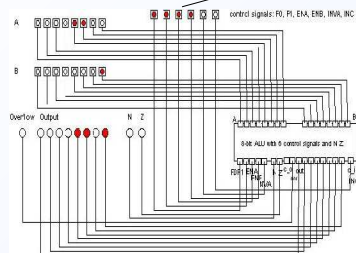
Matteo Baldoni

Architetture degli Elaboratori

24

# Circuiti aritmetici: 8-bit ALU with Z N

\* Il libro contiene due errori: si veda la pagina del corso per le spiegazioni



$F_0$	$F_1$	ENA	ENB	INVA	INC	Funzione
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	<del>A</del>
1	0	1	1	0	0	<del>B</del>
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 and B
0	1	1	1	0	0	A or B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
0	1	0	0	1	0	-1