



Università degli Studi di Milano «La Statale»
Dipartimento di Informatica

Architettura degli elaboratori - Lez 5:

Porte logiche e circuiti combinatori (part C)

Marco Tarini
marco.tarini@unimi.it

116



Tabelle di verità con valori indeterminati

- A volte, vogliamo un circuito che abbia un determinato comportamento...
ma solo con alcune delle possibili combinazioni di input
 - ▶ Le altre combinazioni sono impreviste, errate.
Sappiamo che non verranno mai fornite in input al circuito
 - ▶ Non ci interessa il comportamento del circuito
al di fuori degli input previsti
 - ▶ In presenza di input imprevisti, l'output può essere qualsiasi
 - ▶ In una tabella della verità, denotiamo questo output con X
- Vediamo un esempio concreto

Architettura degli elaboratori - 117 - Porte logiche

117



Esempio: il “7 segment display” per cifre decimali (cioè quello consueto)

- ▶ costruiamo un piccolo circuito combinatorio che calcola lo stato (acceso, spento) di uno dei sette led del display
- ▶ dobbiamo far apparire sul display una cifra in base 10 (da 0 a 9)
- ▶ Codificare una cifra in base 10, ci servono 4 bit
- ▶ Le cifre sono codificate con valori da 0 a 9, cioè da 0000_2 a 1001_2



0000 0001 0010 0011 0100 0101 0110 0111 1000 1001

Architettura degli elaboratori

- 118 -

Funzioni e circuiti combinatori

118



Esempio: tabella di verità per il led in basso



0000 0001 0010 0011 0100 0101 0110 0111 1000 1001

A	B	C	D	OUT
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Architettura degli elaboratori

- 119 -

Porte

119



Circuiti per tabelle con valori indeterminati

- Un circuito che implementa una tabella delle verità con valori indeterminati è corretto sia se produce 1, sia se produce 0, quando i valori indeterminati sono immessi in input
- La presenza di valori indeterminati può essere sfruttata per ottimizzare il circuito che implementa una data tabella
- Non vedremo tecniche di ottimizzazione specializzate in questo senso
 - ▶ Tuttavia, nella traduzione da tabella ad espressioni in forma normale, per le «X» non è mai necessario aggiungere min-termini (né per usando la 1ma né usando la 2nda)
 - ▶ Cioè, le «X» possono essere comodamente considerate come zeri nella 1ma forma, e come uni nella 2nda forma

Architettura degli elaboratori

- 120 -

Porte logiche

120



Reti logiche a più output

- Fin'ora ci siamo concentrati su circuiti combinatori con un output, ma tutto si estende facilmente a circuiti combinatori a più output.
- Esempio: costruiamo un circuito che, presi in input tre bit A, B, C, restituisce due bit:
 - ▶ $D = 1$, sse esattamente due dei tre bit di input sono 1
 - ▶ $E = 1$, sse tutti i bit di input sono 1
- Quindi, una funzione che restituisce, ad esempio:
 - $110 \rightarrow D=1, E=0$
 - $100 \rightarrow D=0, E=0$
 - $111 \rightarrow D=0, E=1$

Architettura degli elaboratori

- 121 -

Porte logiche

121



Una rete logica a più output: tabella delle verità

- D = 1, sse esattamente due bit (dei tre in input) sono 1
- E = 1, sse tutti e tre i bit in input sono 1

INPUT			OUTPUT	
A	B	C	D	E
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Architettura degli elaboratori

- 122 -

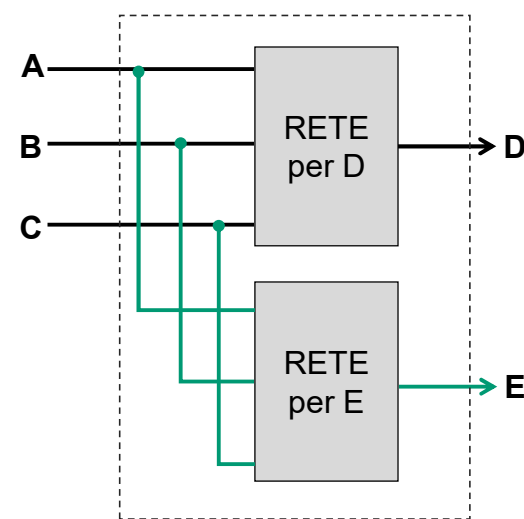
Porte logiche

122



Una rete logica a più output: schema a livello più astratto

INPUT			OUTPUT	
A	B	C	D	E
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1



Architettura degli elaboratori

- 123 -

Porte logiche

123



Reti logiche a più output: osservazioni

- Le 2 sotto-reti dell'esempio lavorano *in parallelo*!
 - Nel **software**, il calcolo è (tipicamente) *sequenziale*:
per es, se una funzione in Go deve calcolare N valori di ritorno,
li calcola *uno dopo l'altro*
 - L'**hardware** è invece il regno del parallelismo:
se un circuito deve calcolare N output, li può calcolare *in parallelo*
- Il design **non ottimizzato** di un circuito con N output è molto semplice:
basta realizzare N circuiti indipendenti, uno per ciascun output,
e diramare l'input a ciascuno di loro.
- La loro **ottimizzazione** tuttavia può essere complicata, perché valori
intermedi calcolati in uno dei due circuiti possono essere utili nell'altro.
 - In generale, identificare e mettere "a fattor comune"
sotto-espressioni condivise non è triviale
 - Non vedremo tecniche specializzate per far questo
 - Tuttavia, nella sintesi delle espressioni in forma normale è banale
almeno condividere gli eventuali min-termini usati dai diversi output

Architettura degli elaboratori

- 124 -

Porte logiche

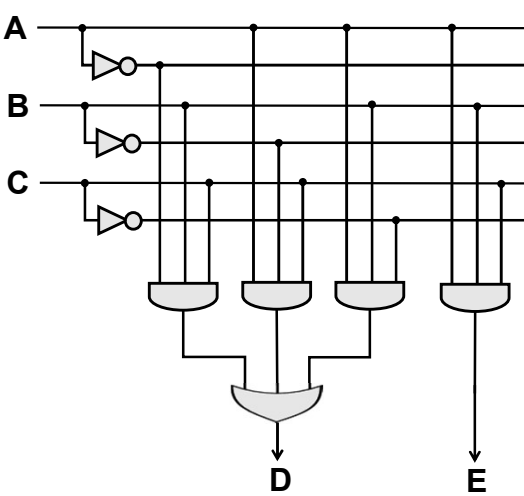
124



Una rete logica a più output: meno astrattamente (usando la 1ma forma normale)

INPUT			OUTPUT	
A	B	C	D	E
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

$$D = \neg A \neg B C + A \neg B C + A B \neg C$$
$$E = A B C$$



Architettura degli elaboratori

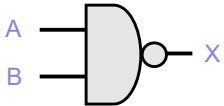
- 125 -

Porte logiche

125

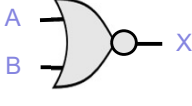


Altri operatori booleani (e porte corrispondenti), non “classici” ma utili



NAND
«l'opposto di AND»
«non entrambi»

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



NOR
«l'opposto di OR»
«nessuno dei due»

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

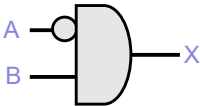
Architettura degli elaboratori - 126 - Porte logiche

126



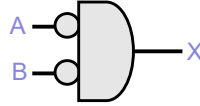
Altri operatori booleani (e porte corrispondenti). Non “classici” ma utili.

- In generale possiamo modificare una qualsiasi porta (a N ingressi) negando (graficamente, con dei pallini) i suoi ingressi o la sua uscita
- Esempi:



!A AND B

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



!A AND !B

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

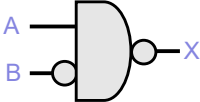
Architettura degli elaboratori - 129 - Porte logiche

129



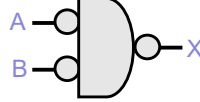
Altri operatori booleani (e porte corrispondenti). Non “classici” ma utili.

- In generale possiamo modificare una qualsiasi porta (a N ingressi) negando (graficamente, con dei pallini) i suoi ingressi o la sua uscita
- Esempi:



$\neg (A \text{ AND } B)$

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



$\neg (\neg A \text{ AND } \neg B)$

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

Architettura degli elaboratori

- 130 -

Porte logiche

130



Altri operatori booleani (e porte corrispondenti). non “classici” ma molto utili.



XOR
(«or esclusivo»)

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

uno o l'altro,
ma
NON entrambi

$X = A \oplus B$

Significati intuitivi di A XOR B:

- A oppure B, *ma non entrambi* (in latino: A *aut* B)
- vero se A e B diversi
falso se A e B uguali
- La negazione di A («!A»...
ma condizionata su B:
inverti A, se B = 1
ma lascia A invariato, se B = 0
(e viceversa!)
- In logica classica: $(\bar{A}B + A\bar{B})$

Architettura degli elaboratori

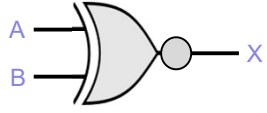
- 131 -

Porte logiche

131



Altri operatori booleani (e porte corrispondenti). Non “classici” ma utili.



NXOR
(«fa il contrario di XOR»)

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

Significati intuitivi di **A NXOR B**:

- uno **XOR** seguito da un NOT
 $A \text{ NXOR } B = \neg(A \text{ XOR } B)$
- cioè vale se...
«entrambi veri oppure entrambi falsi»
- cioè...
 $AB + \neg A \neg B$
- cioè...
operatore di uguaglianza fra A e B:
vero se A e B sono uguali.
falso se sono diversi

Architettura degli elaboratori

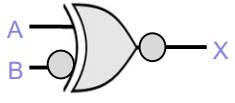
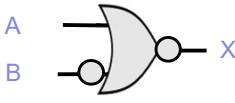
- 133 -

Porte logiche

133



Esercizi di riepilogo



- Scrivere la tabella di verità di queste due porte:
- Descrivine il funzionamento con un'espressione della logica classica
- Progetta un circuito che, dati due input, restituisce due bit:
 - ▶ il primo, vero se gli input sono diversi fra loro
 - ▶ il secondo, vero se gli input sono uguali fra loro

Ma ti è concesso di usare solo 2 porte logiche.
- Modifica la rete vista nell'esempio «da ABC a DE» in modo che D sia 1 anche quando tutti gli input sono 1 (cioè se *almeno* due input sono 1)
- Progetta un circuito che, preso in input un numero da 0 a 2 compresi (codificato in binario), produca in output il **successore** di quel numero (codificato in binario)
 - ▶ Hint: devi per prima cosa determinare quanti bit siano necessari in ingresso / uscita
 - ▶ Hint: nella tabella di verità compaiono delle «X»!
- [ragionamento] l'operatore $\oplus$ (XOR)... è commutativo? È associativo? (per scoprirlo, usa le tabelle di verità!)

Architettura degli elaboratori

- 135 -

Porte logiche

135