## Circuiti e reti combinatorie

Appendice A + dispense

1

## Linguaggio del calcolatore

- Solo assenza o presenza di tensione: 0 o 1
- Tante componenti interconnesse che si basano su 0 e 1
- Anche per esprimere concetti complessi
- Bit: binary digit (0 o 1)

2

## Algebra di Boole

- Strumento matematico su cui si basano i sistemi digitali (George Boole, 1854)
- Variabili che possono avere solo uno di due valori: 1 (vero) o 0 (falso)
- Operazioni di base tra le variabili: AND, OR, NOT
  - A AND B = A x B
  - A OR B = A + B
  - NOT A = $\bar{A}$

3

## AND, OR, NOT

- AND: risultato 1 se e solo se entrambi gli operando sono 1
- OR: risultato 1 se almeno uno dei due operandi e' 1
- NOT: inverte il valore dell'operando
- Esempio: A + ($\overline{B}$ x C) = D
  - D =1 se A=1 o se B=0 e C=1

4

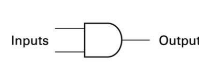
## Notazione

- Senza parentesi, AND ha precedenza sull'OR
  - Esempio: A + (B x C) = A + B x C
- Spesso scritto senza x:
  - Esempio: A + BC
- Tabella di verita'
  - Risultato per ogni possibile combinazione dei valori degli operandi

5

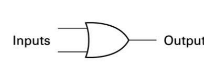
## And e or

**AND**




| Inputs | Output |
|---|---|
| 0 0 | 0 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 1 |

**OR**

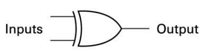



| Inputs | Output |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 1 |

6

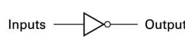
## Xor e not

XOR




| Inputs | Output |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

NOT



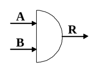

| Inputs | Output |
|---|---|
| 0 | 1 |
| 1 | 0 |



AND

| A | B | A AND B |
|---|---|---|
| falso | falso | falso |
| falso | vero | falso |
| vero | falso | falso |
| vero | vero | vero |

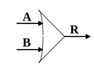
| A | B | R |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




OR

| A | B | A OR B |
|---|---|---|
| falso | falso | falso |
| falso | vero | vero |
| vero | falso | vero |
| vero | vero | vero |

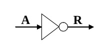
| A | B | R |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




NOT

| A | NOT A |
|---|---|
| falso | vero |
| vero | falso |

| A | R |
|---|---|
| 0 | 1 |
| 1 | 0 |




## Proprieta' dell'algebra di Boole

- Regole di base (postulati):
  - Commutativita'
    - A + B = B + A
    - A x B=B x A
  - Distributivita'
    - A + (B x C) = (A + B) x (A + C)
    - A x (B + C) = (A x B) + (A x C)
  - Elementi neutri
    - 1 x A = A
    - 0 + A = A
  - Elementi inversi
    - A x Ā = 0
    - A + Ā = 1



## Altre proprieta' dell'algebra di Boole

- Assorbimento
  - 0 x A = 0
  - 1 + A = 1
- Idempotenza
  - A x A = A
  - A + A = A
- Associativita'
  - A x (B x C) = (A x B) x C
  - A + (B + C) = (A + B) + C
- Leggi di De Morgan
  - $\overline{A \times B} = \bar{A} + \bar{B}$
  - $\overline{A + B} = \bar{A} \times \bar{B}$



## Porte logiche

- Circuito elettronico che, dati dei segnali in ingresso, produce un segnale (0 o 1) ottenuto effettuando una operazione Booleana sugli ingressi
- Ogni porta ha 1 o 2 input e 1 output
- Dati gli input, l'output corrispondente appare quasi istantaneamente (ritardo di commutazione)
- Di solito, solo pochi tipi di porte ➔ identificare insieme di porte funzionalmente completi



## Completezza di and, or, e not

- 16 operazioni logiche binarie (tante quante possibili scelte di 4 valori)
- 4 operazioni logiche unarie
- Tutte possono essere ottenute componendo and, or, e not

## Completezza

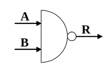
- A OR B = NOT((NOT A) AND (NOT B))
- Quindi anche {AND, NOT} e’ un insieme completo
- Lo stesso per {OR, NOT}

13

NAND

| A | B | A NAND B |
|---|---|---|
| falso | falso | vero |
| falso | vero | vero |
| vero | falso | vero |
| vero | vero | falso |

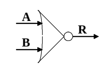
| A B | R |
|---|---|
| 0 0 | 1 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |



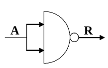

NOR

| A | B | A NOR B |
|---|---|---|
| falso | falso | vero |
| falso | vero | falso |
| vero | falso | falso |
| vero | vero | falso |

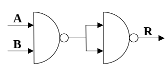
| A B | R |
|---|---|
| 0 0 | 1 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 0 |



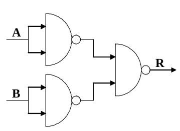

NOT




AND

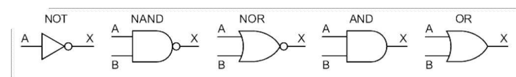



OR




Quindi NAND o NOR sono complete ➔ circuiti con solo porte NAND o solo porte NOR.

## Riassunto: porte logiche di base




NOT

| A | X |
|---|---|
| 0 | 1 |
| 1 | 0 |

NAND

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

AND

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

16

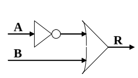
$\Rightarrow$

| A | B | A $\Rightarrow$ B |
|---|---|---|
| falso | falso | vero |
| falso | vero | vero |
| vero | falso | falso |
| vero | vero | vero |

| A B | A $\Rightarrow$ B |
|---|---|
| 0 0 | 1 |
| 0 1 | 1 |
| 1 0 | 0 |
| 1 1 | 1 |

A $\Rightarrow$ B equivale a (NOT A) OR B

| A B | NOT A | (NOT A) OR B |
|---|---|---|
| 0 0 | 1 | 1 |
| 0 1 | 1 | 1 |
| 1 0 | 0 | 0 |
| 1 1 | 0 | 1 |

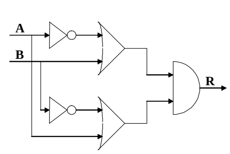



$\equiv$

| A B | A ■ B |
|---|---|
| 0 0 | 1 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 1 |

A $\equiv$ B equivale a
(A $\Rightarrow$ B) AND (B $\Rightarrow$ A)

| A B | A $\Rightarrow$ B | B $\Rightarrow$ A | (A $\Rightarrow$ B)AND(B $\Rightarrow$ A) |
|---|---|---|---|
| 0 0 | 1 | 1 | 1 |
| 0 1 | 1 | 0 | 0 |
| 1 0 | 0 | 1 | 0 |
| 1 1 | 1 | 1 | 1 |

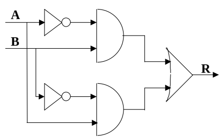


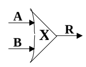
## ≠ o XOR

| A B | A ≠ B |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

A XOR B equivale a NOT (A ≡ B)

| A B | A ≡ B | NOT(A ≡ B) |
|---|---|---|
| 0 0 | 1 | 0 |
| 0 1 | 0 | 1 |
| 1 0 | 0 | 1 |
| 1 1 | 1 | 0 |

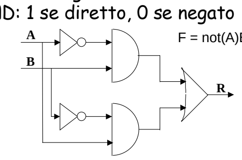



## Dalla tabella di verita' ad un circuito

- Implementazione di funzioni Booleane
- Tanti circuiti diversi per una stessa funzione
- Un metodo che funziona sempre (somma di prodotti):
  - Tanti input quante sono le dimensioni della tabella
  - Un solo output
  - Un OR la cui uscita e' l'output
  - Tanti AND quanti sono gli 1 della tabella
  - Input degli AND: 1 se diretto, 0 se negato

| A B | A ≠ B |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |




F = not(A)B + B not(A)



## Anche prodotto di somme

- Somma di prodotti: uscita 1 se si verifica qualche combinazione di ingressi che produce un 1
- Prodotto di somme: uscita 1 se non si verifica nessuna combinazione di ingresso che produce 0
- Un AND la cui uscita e' l'output
- Tanti OR quanti sono gli 0 della tabella
- Input degli OR: 0 se diretto, 1 se negato

| A B | A ≠ B |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

**Circuito corrispondente?**

F = (A + B) x (not(A) + not(B))



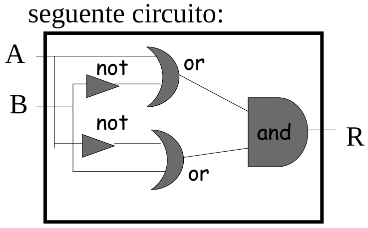
## Una funzione Booleana, tanti circuiti

- Esempio:
  - F = (not(A) x B) + (B x not(C))
  - Ma anche F = B x (not(A) + not(C))



## Esercizio

- Determinare la tavola di verità del seguente circuito:




è una tavola nota?



## Esercizio

- Partendo dalla tavola di verità dell'esercizio precedente, costruite un circuito che la realizza seguendo il metodo dela somma di prodotti e quello del prodotto di somme.

## Esercizio

•Si disegni un circuito logico che realizza la seguente tavola di verita':

- •A=0, B=0 ➔ R = 1
- •A=0, B=1 ➔ R = 1
- •A=1,B=0 ➔ R = 1
- •A=1, B=1 ➔ R = 0

25

## Esercizio

•Dare la tavola di verita' delle formule:

- •(A ➔ NOT(B)) OR (A AND B)
- •A OR (A AND NOT(B))
- •(NOT(A) ➔ NOT(B)) OR (NOT(A) AND B)

26

## Reti combinatorie

- I circuti che abbiamo visto non hanno cicli
- Sono rappresentabili da reti combinatorie
- Rete combinatoria: insieme di porte logiche connesse il cui output in un certo istante e' funzione solo dell'input in quell'istante
- N input binari e m output binari
- Ad ogni combinazione di valori di ingresso corrisponde **una ed una sola** combinazione di valori di uscita

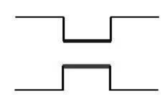
27

## Reti combinatorie (segue)

- Vediamo alcuni esempi di circuiti:
  - ✓ I segnali sono discretizzati e di solito assumono solo due stati:

0 / FALSO / [0..1] Volt

1 / VERO / [2..5] Volt

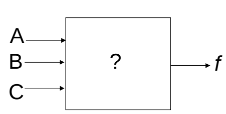


  - ✓ I circuiti piu' complessi sono realizzati attraverso la combinazione di circuiti semplici (porte logiche)

28

## Esercizio:
## dal problema alla rete combinatoria

Progettare una rete combinatoria a tre ingressi che restituisca in output 1 solo se ALMENO due ingressi sono a 1




29

SOLUZIONE : creazione della tabella

| A | B | C | *f* |
|---|---|---|---|
| 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 |

Espressione booleana

$f = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

**Somma di prodotti**

30

SOLUZIONE : riduzione della espressione

$f = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

$= \bar{A}BC + A\bar{B}C + AB(\bar{C} + C)$ distributiva

$= \bar{A}BC + A\bar{B}C + AB(1)$ complemento

$= \bar{A}BC + A\bar{B}C + AB$ identità

$= \bar{A}BC + A\bar{B}C + AB + ABC$ idempotenza

$= \bar{A}BC + AC(\bar{B} + B) + AB$ distributiva

$= \bar{A}BC + AC(1) + AB$ complemento

$= \bar{A}BC + AC + AB$ identità

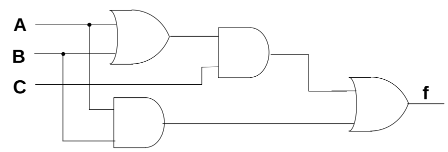
$= \bar{A}BC + AC + AB + ABC = \ldots$ idempotenza

$= BC + AC + AB = (B + A)C + AB$

31

SOLUZIONE : schema della rete combinatoria

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




32

# Implementazioni NAND e NOR

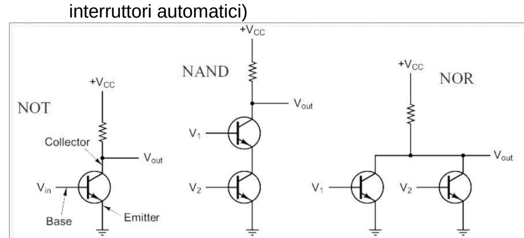
- Spesso si vuole usare solo porte NAND o solo porte NOR
- A volte non minimale, ma regolare
- Esempio:
- $F = B(\bar{A}+\bar{C})$
- $\bar{\bar{A}} = A$ e teorema di De Morgan

➔ $F = \text{nand}(\text{nand}(\bar{A},B),\text{nand}(B,\bar{C}))$

33

# Reti combinatorie (segue)

- Porte Logiche:
  - ✓ Sono realizzate tramite transistor (sono in pratica interruttori automatici)




34

# Reti combinatorie: specifica e progetto

- La specifica di una funzione logica da implementare mediante rete combinatoria può essere vista come un **programma**
- La progettazione diventa combinazione e complemento di reti già note
- Componenti standard
  - ✓ Confrontatore, commutatore, selezionatore
- ✓ Alcuni ingressi possono essere usati per controllare il funzionamento della rete combinatoria (bit di selezione o controllo)

35

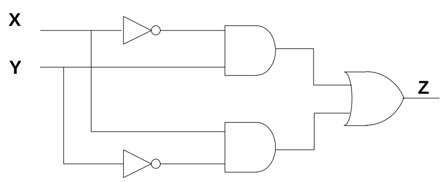
# Reti combinatorie piu' usate

- **Confrontatore**, a due ingressi (x,y) ed una uscita (z)
  - ✓ z := **not** (x = y)
- **Commutatore**, a due ingressi primari (x,y), un ingresso di controllo ($\alpha$) ed una uscita (z)
  - ✓ z := **if not** $\alpha$ **then** x **else** y
- **Selettore**, ad un ingresso primario (x), un ingresso di controllo ($\alpha$) e due uscite ($z_1$,$z_2$)
  - ✓ **if not** $\alpha$ **then** ($z_1$ := x ; $z_2$ := 0) **else** ($z_1$ := 0 ; $z_2$ := x)

36

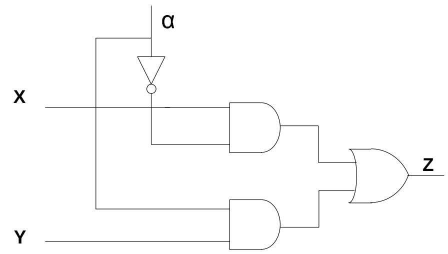
## Confrontatore

z := **not** (x = y)




37

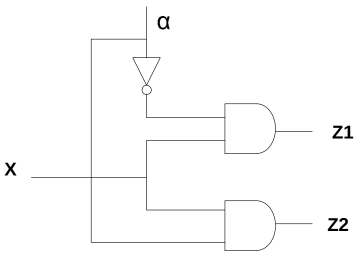
## Commutatore z := **if not** α **then** x **else** y




38

## Selettore

**if not** α **then (**z1 := x ; z2 := 0)
**else** (z1 := 0 ; z2 := x)



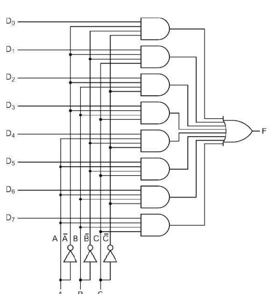

39

## Reti combinatorie multi-funzione

- Operatori aritmetico logici a specifica **diretta**
  - ✓ Addizione, sottrazione, traslazione, rotazione, incremento, decremento, etc.
- Reti aritmetico logiche **multi-funzione**
  - ✓ Eseguono **una** delle operazioni suddette a seconda del valore assunto da un certo numero di ingressi di controllo
  - ✓ Si usano per implementare le **ALU** (*arithmetic logic unit*)

40

## Multiplexer (o selettore) $2^n$ a 1




- Solo uno degli ingressi viene trasferito all'output
- n ingressi di controllo: indicano l'ingresso da trasferire
  - ✓ $2^n$ linee di input ( $D_0$ - $D_7$)
  - ✓ n linee di controllo (A,B,C)
  - ✓ 1 linea di output (F)
- ✓ Per ogni combinazione degli ingressi di controllo, 2n -1 delle porte AND hanno uscita 0, l'altra fa uscire l'ingresso

41

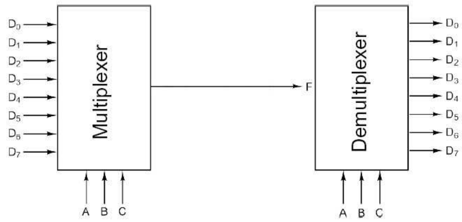
## Uso del multiplexer

- Caricamento del program counter, con valore proveniente da
  - □ Un contatore binario (incremento per successiva istruzione)
  - □ Registro istruzione corrente (istruzione di salto)
  - □ Output della ALU
  - □ Input primari tanti quante linee di ingresso, PC in output ad un multiplexer (1 multiplexer per ogni bit del PC)

42

## Demultiplexer

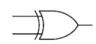
- ✓ E' il circuito inverso del Multiplexer ed è spesso usato in combinazione con quest'ultimo (seleziona comunicazione fra linee)




43

## Comparatore

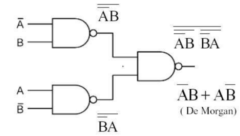
- ✓ Compara due ingressi e produce un output che indica la uguaglianza (0) o meno (1) degli ingressi
- ✓ Esempio di comparatore ad 1 bit: si realizza con una porta XOR



**XOR** (OR Esclusivo)

Simbolo ⊕

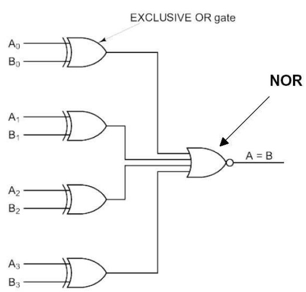
| A | B | XOR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




**Realizzazione con porte NAND**

44

## Comparatore a piu' bit

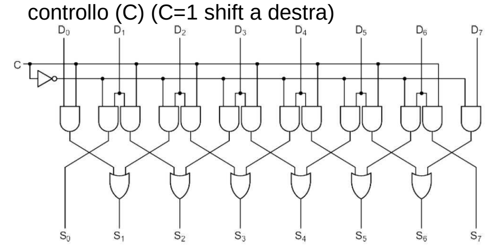



- ✓ Comparatori ad 1 bit vengono collegati tramite una porta NOR
- ✓ L'output vale 1 solo se tutti gli output dei singoli comparatori ad 1 bit valgono 0
- ✓ (Ai=Bi) per ogni i, cioe' A=B

45

## Traslatore (shifter)

- ✓ Trasla i bit in ingresso (D) di una posizione, a sinistra o a destra a seconda del valore del bit di controllo (C) (C=1 shift a destra)




46

## Sommatore

- Somma due numeri binari
- Prima bisogna capire cosa sono i numeri binari e come si sommano

47