

Circuiti Logici Digitali

Operatori Logici e Porte Logiche

Corso di Scienze e Tecnologie Applicate

IIS Fermi Sacconi Ceci

27 novembre 2025

- 1 Introduzione
- 2 Operatori Logici
- 3 Tabelle di Verità
- 4 Algebra di Boole
- 5 Forme Canoniche
- 6 Mappe di Karnaugh
- 7 Applicazioni

- La logica formale studia le proposizioni dichiarative
- Una proposizione può essere **Vera** o **Falsa**
- Rappresentazione numerica:
 - Falso = 0 = 0V
 - Vero = 1 = tensione di alimentazione
- Le variabili logiche possono assumere solo valori booleani
- Gli operatori logici connettono i valori tra loro

Operatore NOT

- Interviene su un solo valore logico
- Inverte il valore in ingresso
- Simbolo: NOT, \neg , ' (apice)

| A | NOT A |
|---|-------|
| 0 | 1 |
| 1 | 0 |

Esempio: Se $A = \text{"Antonio mangia"}$, $\text{NOT } A = \text{"Antonio non mangia"}$

Operatore AND

- Opera su due o più ingressi
- Restituisce Vero solo se **tutti** gli ingressi sono Vero
- Simboli: AND, \wedge , \cdot

| A | B | A AND B |
|---|---|---------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Esempio: “Antonio mangia E Piero legge”

Operatore OR

- Opera su due o più ingressi
- Restituisce Vero se **almeno uno** degli ingressi è Vero
- Simboli: OR, \vee , +

| A | B | A OR B |
|---|---|--------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Esempio: “Antonio mangia E/O Piero legge”

Connettivo XOR (OR Esclusivo)

Operatore XOR

- Opera su due ingressi
- Restituisce Vero se **solo uno** degli ingressi è Vero
- Simboli: XOR, \oplus

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

Esempio: “Antonio mangia **OPPURE** Piero legge” (ma non entrambi)

NAND (NOT AND)

| A | B | NAND |
|---|---|------|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR (NOT OR)

| A | B | NOR |
|---|---|-----|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Nota: NAND e NOR sono operatori universali: qualsiasi funzione logica può essere realizzata usando solo porte NAND o solo porte NOR.

Regole di Valutazione

Le espressioni logiche seguono un ordine di precedenza:

- 1 **NOT** (negazione)
- 2 **AND** (prodotto logico)
- 3 **OR** (somma logica)

Esempi:

- $a + b \cdot c'$ si legge come $a + (b \cdot (c'))$
- Le parentesi modificano l'ordine di precedenza
- $A \cdot B + C$ significa $(A \cdot B) + C$

Valori Irrilevanti nelle Tabelle di Verità

- In alcune situazioni, il valore di certe variabili non influenza il risultato
- Si indica con “X” o “-” (don't care)
- Utile per semplificare i circuiti

Esempio: $A \cdot (B + C)$

| Tabella completa | | | |
|------------------|---|---|-----------|
| A | B | C | Risultato |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

| Tabella semplificata | | | |
|----------------------|---|---|-----------|
| A | B | C | Risultato |
| 0 | X | X | 0 |
| 1 | 0 | 0 | 0 |
| 1 | X | 1 | 1 |
| 1 | 1 | X | 1 |

Equivalenze dell'Algebra di Boole (1/2)

Proprietà Fondamentali

$$a + 0 = a$$

$$a + 1 = 1$$

$$a + a = a$$

$$a + a' = 1$$

$$a + b = b + a$$

$$a \cdot 0 = 0$$

$$a \cdot 1 = a$$

$$a \cdot a = a$$

$$a \cdot a' = 0$$

$$a \cdot b = b \cdot a$$

Teoremi Fondamentali

$$(a + b)' = a' \cdot b'$$

$$(a \cdot b)' = a' + b'$$

Applicazione: Trasformare funzioni tra forme diverse

- Convertire OR in AND (e viceversa)
- Utile per implementare circuiti con un solo tipo di porta
- Esempio: realizzare circuiti con sole porte NAND

Altre Proprietà dell'Algebra di Boole

Associatività

$$a + (b + c) = (a + b) + c$$

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

Distributività

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

Doppia Negazione

$$(a')' = a$$

Somma dei Prodotti (SOP)

- Ogni funzione logica può essere espressa come somma di prodotti
- Si parte dalla tabella di verità
- Si considerano solo le righe con output = 1
- Si crea un prodotto (AND) per ogni riga
- Si sommano (OR) tutti i prodotti

Esempio: Funzione NXOR

| A | B | NXOR |
|---|---|------|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$$F = A' \cdot B' + A \cdot B$$

Definizioni

- **Letterale:** una variabile o la sua negazione (es. A , A')
- **Mintermine:** prodotto di letterali per tutte le variabili
- Ogni combinazione di input ha un mintermine corrispondente

Mintermini per 2 variabili:

| A | B | Mintermine | Notazione |
|---|---|---------------|-----------|
| 0 | 0 | $A' \cdot B'$ | m_0 |
| 0 | 1 | $A' \cdot B$ | m_1 |
| 1 | 0 | $A \cdot B'$ | m_2 |
| 1 | 1 | $A \cdot B$ | m_3 |

Introduzione alle Mappe di Karnaugh

- Metodo grafico per semplificare funzioni logiche
- Alternativa all'algebra di Boole
- Rappresentazione bidimensionale dei prodotti fondamentali
- Efficace fino a 4 variabili
- Per più variabili serve rappresentazione multidimensionale

Vantaggi:

- Visualizzazione immediata
- Identificazione rapida di semplificazioni
- Riduzione del numero di porte logiche necessarie

Mappa di Karnaugh a 2 Variabili

Struttura:

| | B' | B |
|------|--------|-------|
| A' | $A'B'$ | $A'B$ |
| A | AB' | AB |

Esempio: XOR

| | B' | B |
|------|------|-----|
| A' | 0 | 1 |
| A | 1 | 0 |

$$F = A \oplus B$$

Nota: Celle adiacenti differiscono per una sola variabile

Mappa di Karnaugh a 3 Variabili

Attenzione all'ordine delle colonne: 00, 01, 11, 10 (Codice Gray)

| | $B'C'$ | $B'C$ | BC | BC' |
|------|----------|---------|--------|---------|
| A' | $A'B'C'$ | $A'B'C$ | $A'BC$ | $A'BC'$ |
| A | $AB'C'$ | $AB'C$ | ABC | ABC' |

Procedimento di semplificazione:

- 1 Inserire 1 nelle celle corrispondenti ai mintermini
- 2 Raggruppare celle adiacenti con 1 (gruppi di 1, 2, 4, 8)
- 3 Ogni gruppo elimina una variabile
- 4 Scrivere l'espressione minimizzata

Mappa di Karnaugh a 4 Variabili

| | $C'D'$ | $C'D$ | CD | CD' |
|--------|--------|-------|------|-------|
| $A'B'$ | 0 | 1 | 3 | 2 |
| $A'B$ | 4 | 5 | 7 | 6 |
| AB | 12 | 13 | 15 | 14 |
| AB' | 8 | 9 | 11 | 10 |

Regole di raggruppamento:

- I gruppi devono essere rettangolari
- Dimensioni: 1, 2, 4, 8, 16 celle
- La mappa è toroidale: i bordi sono adiacenti
- Massimizzare la dimensione dei gruppi

Regole importanti:

- Gruppo di 2 celle adiacenti → elimina 1 variabile
- Gruppo di 4 celle adiacenti → elimina 2 variabili
- Gruppo di 8 celle adiacenti → elimina 3 variabili

Processo:

- 1 Formare i gruppi più grandi possibili
- 2 Ogni 1 deve essere coperto da almeno un gruppo
- 3 I gruppi possono sovrapporsi
- 4 Minimizzare il numero totale di gruppi

Risultato: Espressione logica minimizzata = somma dei termini rappresentati dai gruppi

Condizioni Indifferenti (Don't Care)

- In alcune situazioni, certe combinazioni di input non si verificano mai
- Oppure il valore di output è irrilevante
- Si indicano con “X” nella mappa
- Possono essere considerati come 0 o 1 per ottenere la migliore semplificazione

Vantaggio:

- Permettono ulteriori semplificazioni
- Riducono il numero di porte necessarie
- Ottimizzazione del circuito

Mappe di Karnaugh con Funzione XOR

Pattern XOR nelle mappe

Le funzioni XOR hanno pattern caratteristici:

- Distribuzione a “scacchiera”
- Nessun raggruppamento efficace possibile
- Indicazione che la funzione contiene XOR

Esempio XOR a 2 variabili:

| | B' | B |
|------|------|-----|
| A' | 0 | 1 |
| A | 1 | 0 |

$$F = A \oplus B = A'B + AB'$$

La presenza di XOR rende difficile la semplificazione con Karnaugh

Principali Applicazioni

- **Decodificatori:** selezionano una linea di uscita in base all'input
- **Multiplexer:** selezionano uno tra più ingressi
- **Demultiplexer:** indirizzano un input verso una delle uscite
- **Addizionatori:** somma di numeri binari
- **Comparatori:** confronto tra valori
- **Unità logiche (ALU):** operazioni aritmetiche e logiche

Perché semplificare i circuiti?

- **Costo:** meno componenti → costo inferiore
- **Spazio:** circuiti più compatti
- **Velocità:** meno porte → minor ritardo di propagazione
- **Consumo:** minore dissipazione di potenza
- **Affidabilità:** meno componenti → minor probabilità di guasto

Obiettivo: Trovare il miglior compromesso tra:

- Numero di porte logiche
- Profondità del circuito (livelli di porte)
- Numero di connessioni

Riepilogo

- Gli operatori logici sono i mattoni dei circuiti digitali
- L'algebra di Boole fornisce le regole di manipolazione
- Le mappe di Karnaugh offrono un metodo grafico di semplificazione
- La minimizzazione è fondamentale per circuiti efficienti

Prossimi passi:

- Circuiti combinatori complessi
- Circuiti sequenziali (con memoria)
- Flip-flop e registri
- Progettazione di sistemi digitali

- Materiale didattico del corso
- “Appunti di Informatica Libera” - Daniele Giacomini
- Simulatori di circuiti logici: Tkgate, Logisim
- Risorse online per esercitazioni

Grazie per l'attenzione!