Circuiti Logici Digitali

Operatori Logici e Porte Logiche

Prof. Fedeli Massimo

IIS Fermi Sacconi Ceci

28 novembre 2025

Indice

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

Introduzione alla Logica Digitale

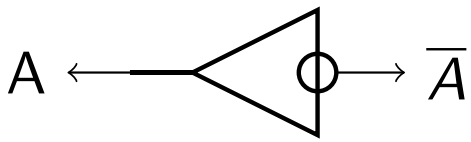
- La logica formale studia le proposizioni dichiarative
- Per proposizione si intende insieme di **soggetto** e **verbo**
- Una proposizione dichiarativa può essere **Vera** o **Falsa**
- Rappresentazione numerica:
  - Falso $= 0 = 0$V
  - Vero $= 1 =$ tensione di alimentazione
- Le variabili logiche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

Introduzione alla Logica Digitale

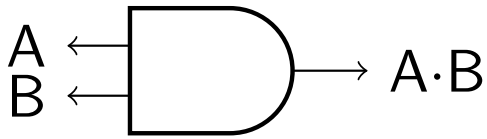
- Un proposizione che non si può dividere in altre proposizioni si dice **elementare**
- Il valore di una proposizione dichiarativa (Vero o Falso) può essere espresso in vari modi, a seconda del contesto.
- Generalmente, si at- tribuisce alla cifra numerica uno il significato di **Vero**, mentre a zero si attribuisce il valore **Falso**.

Operatori logici

- Si distinguono generalmente gli operatori logici in «**unari**» e in «**connettivi logici**»
- Gli operatori logici si possono vedere come delle scatoline, aventi uno o più ingressi, ma con una sola uscita: in tal caso, si chiamano porte logiche.



NOT (unario)



AND (connettivo)

Operatori Unari

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 = "Antonio mangia", NOT A = "Antonio non mangia"

Connettivo AND

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"

Connettivo OR

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)

Altri Operatori Logici

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.

Ordine di Precedenza

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)

Attraverso gli operatori logici si possono costruire delle espressioni complesse, nelle quali esiste la facoltà di applicare delle trasformazioni, di solito con lo scopo di cercare di semplificarle, mantenendo però lo stesso risultato.

Proprietà Fondamentali

$$a + 0 = a \qquad a \cdot 0 = 0$$

$$a + 1 = 1 \qquad a \cdot 1 = a$$

$$a + a = a \qquad a \cdot a = a$$

$$a + a' = 1 \qquad a \cdot a' = 0$$

$$a + b = b + a \qquad a \cdot b = b \cdot a$$

Teoremi di De Morgan

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

$$(a')' = a$$

Funzioni logiche e somma dei prodotti

Una funzione logica può essere descritta attraverso la **tabella di verità** che abbina i valori delle variabili al risultato atteso per ogni combinazione.

Metodo di sintesi - Somma dei prodotti:

1. Si parte dalla tabella di verità desiderata
2. Si elencano i **prodotti fondamentali logici** che descrivono ogni condizione degli ingressi
3. Si sommano (OR) tutti i prodotti ottenuti

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

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

Somma dei Prodotti (SOP) - Altro esempio

Esempio: Funzione di maggioranza a 3 ingressi

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

Procedimento:

1. Righe con output = 1:
   - Riga 4: $A' \cdot B \cdot C$
   - Riga 6: $A \cdot B' \cdot C$
   - Riga 7: $A \cdot B \cdot C'$
   - Riga 8: $A \cdot B \cdot C$
2. Somma dei prodotti:

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

La funzione restituisce 1 quando almeno due ingressi su tre sono uguali a 1

A cosa servono le funzioni logiche?

Definizione: Una funzione logica è una relazione che associa ad ogni combinazione di valori binari in ingresso (0 o 1) un valore binario in uscita.

Utilizzi pratici:

- **Circuiti digitali**: progettare componenti elettronici (CPU, memorie, controller)
- **Controllo automatico**: sistemi di allarme, semafori, automazione industriale
- **Programmazione**: condizioni nei programmi (if, while, etc.)
- **Crittografia**: algoritmi di sicurezza e codifica

Esempio concreto - Sistema di allarme auto:

Porta aperta	Chiave inserita	Cintura allacciata	Allarme sonoro
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	X	1
1	1	0	1

Come si usa una funzione logica?

Il processo:

1. **Analisi**: identificare ingressi e uscita desiderata
2. **Tabella di verità**: elencare tutti i casi possibili
3. **Funzione logica**: ricavare l'espressione matematica
4. **Implementazione**: realizzare il circuito con porte logiche

Esempio - Lampada scale:

Due interruttori (A, B) controllano una lampada (L)

Int. A	Int. B	Lampada
0	0	0
0	1	1
1	0	1
1	1	0

$L = A \oplus B$ (XOR)

Cambiando uno qualsiasi dei due interruttori, la lampada cambia stato

Mintermini e Prodotti Fondamentali

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

Come realizzare una mappa di Karnaugh

Procedimento passo-passo:

1. **Determinare il numero di variabili**
   - 2 variabili $\rightarrow$ mappa 2×2 (4 celle)
   - 3 variabili $\rightarrow$ mappa 2×4 (8 celle)
   - 4 variabili $\rightarrow$ mappa 4×4 (16 celle)
2. **Creare la griglia con codifica Gray**
   - Etichettare righe e colonne con combinazioni delle variabili
   - Usare il codice Gray: tra celle adiacenti cambia un solo bit
3. **Riempire la mappa**
   - Inserire il valore della funzione (0 o 1) in ogni cella
   - Corrispondenza con la tabella di verità
4. **Raggruppare gli 1**
   - Creare gruppi di 1, 2, 4, 8... celle adiacenti contenenti 1
   - I gruppi devono essere rettangolari e di dimensione potenza di 2

Regole per i raggruppamenti nella mappa di Karnaugh

Regole fondamentali:

1. I gruppi devono contenere **solo celle con valore 1**
2. Le dimensioni dei gruppi devono essere **potenze di 2**: 1, 2, 4, 8, 16...
3. I gruppi devono essere **rettangolari** (non diagonali)
4. Si devono formare i **gruppi più grandi possibili**
5. Ogni 1 deve essere coperto da **almeno un gruppo**
6. Gli 1 possono appartenere a **più gruppi contemporaneamente**
7. I gruppi possono **"avvolgersi"** ai bordi della mappa (le celle ai bordi opposti sono adiacenti)

Risultato: Ogni gruppo corrisponde a un termine della funzione semplificata. Le variabili che cambiano nel gruppo vengono eliminate.

Mappa di Karnaugh

Rappresentazione e semplificazione

Una funzione logica può essere rappresentata in una **mappa di Karnaugh**, a partire dalla sua tabella di verità.

Obiettivo:

- Semplificare l'espressione che sintetizza la funzione
- Ridurre la complessità del circuito che la realizza

Come funziona:

La mappa di Karnaugh è una **rappresentazione bidimensionale** dei prodotti fondamentali di una tabella di verità: si dividono a metà i prodotti fondamentali e si rappresentano in un rettangolo.

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

Esempi di Semplificazione

Regole importanti:

- Gruppo di 2 celle adiacenti $\rightarrow$ elimina 1 variabile
- Gruppo di 4 celle adiacenti $\rightarrow$ elimina 2 variabili
- Gruppo di 8 celle adiacenti $\rightarrow$ 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

Applicazioni dei Circuiti Combinatori

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

Vantaggi della Minimizzazione

Perché semplificare i circuiti?

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

Obiettivo: Trovare il miglior compromesso tra:

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

Conclusioni

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

Bibliografia e Risorse

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

Grazie per l’attenzione!