Architettura degli Elaboratori - Porte logiche e circuiti combinatori

Andrea Malvezzi

26 Settembre, 2024

Contents

1 Algebra di Boole

1.1 Espressioni booleane

Un'espressione booleana si costruisce usando:

- 0 e 1 (False e True);
- gli operatori booleani (o logici);
- delle variabili sempre con valore 0 oppure 1.

1.2 Proprietà dell'algebra di Boole

Nella tabella seguente sono presenti delle equivalenze per descrivere le proprietà dell'algebra di Boole.

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	$A\overline{A} = 0$	$A + \overline{A} = 1$
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	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Figure 1: Una visualizzazione delle proprietà dell'algebra di Boole.

1.2.1 La legge di De Morgan

La più importante tra le leggi presentate è sicuramente quella di De Morgan, in quanto permette di passare da una colonna della tabella all'altra in modo semplice e veloce.

1.2.2 Esempio di applicazione della legge di De Morgan

- Per cominciare, scriviamo la OR form della legge inversa: $A + \overline{A} = 1$;
- Seguentemente occorre pensare a com'è scritta l'espressione: siamo davanti ad una OR tra due variabili A, di cui una negata, il tutto pari ad 1;
- Ora osserviamo la legge di De Morgan nella forma OR. Questa afferma quanto segue: La negazione di una OR equivale ad una AND con entrambi gli input negati;
- Applichiamo quindi De Morgan: $\overline{A} + A = 1$ diventerà $\overline{A\overline{A}} = \overline{1}$, ovvero $\overline{A}A = 0$.

Ed ecco mostrato come passare da un lato all'altro della tabella tramite la formula di De Morgan.

1.3 Formula canonica

Una funzione booleana si può definire attraverso un'espressione basata solamente sulla AND, la OR e la NOT.
Inoltre, una funzione booleana è esprimibile in una forma detta "canonica". Per ricavarla occorre:

- identificare tutte le combinazioni per cui la funzione in esame è vera (queste son dette **mintermini**);
- fare la OR dei mintermini trovati;

1.3.1 Esempio di formula canonica

Analizziamo la seguente tabella:

A	B	C	F	
0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	0	$\overline{A}\,\overline{B}\,C$
0	1	0	1	$\overline{A}\,B\,\overline{C}$
0	1	1	0	$\overline{A}\,B\,C$
1	0	0	0	$A\,\overline{B}\,\overline{C}$
1	0	1	0	$A\,\overline{B}\,C$
1	1	0	1	$A\,B\,\overline{C}$
1	1	1	1	$A\,B\,C$

Figure 2: Nella tabella presentata si hanno 3 mintermini: $\overline{A}B\overline{C}$, $AB\overline{C}$, ABC.

Ora dobbiamo fare la OR tra i 3 mintermini trovati precedentemente:

$$\overline{A}B\overline{C} + AB\overline{C} + ABC$$

Questa espressione equivale alla forma canonica della funzione studiata.

2 I transistor

Un transistor è un dispositivo a 3 connettori: **collettore**, **emettitore** e **base**.
Quando non c'è tensione sulla base, il componente agisce come una resistenza infinita tra emettitore e collettore.
In caso contrario, si comporta da conduttore ideale.

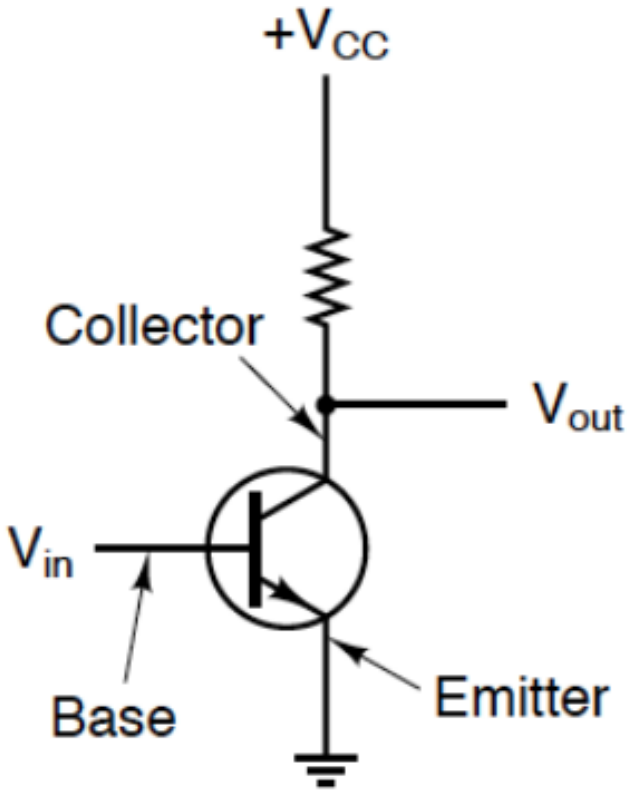


Figure 3: Esempio di porta NOT realizzata con transistor.

2.1 La porta NOT realizzata tramite transistor

La porta NOT della figura (3) funziona nella seguente maniera:

- quando l'ingresso V_{in} è *alto*, la corrente V_{cc} attraversa il transistor e lascia V_{out} basso (Input 1, output 0);
- quando l'ingresso V_{in} è *basso*, la corrente V_{cc} non riesce ad attraversare il transistor, il quale agisce da resistenza infinita, passando quindi per V_{out}, il quale diventa alto (Input 0, output 1).

2.2 La porta NAND realizzata tramite transistor

Per realizzare una porta NAND occorranno due transistor:

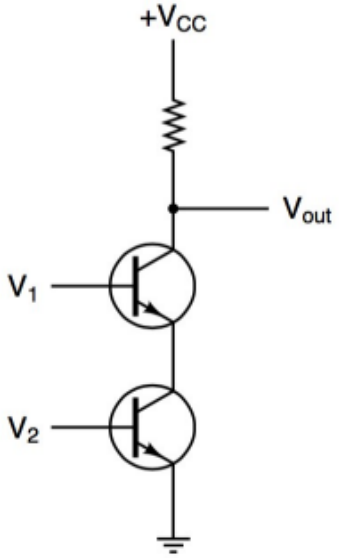


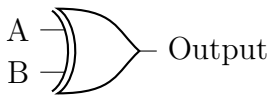
Figure 4: Esempio di porta NAND realizzata con 2 transistor.

Il funzionamento della porta logica in figura è simile a quello della NOT in (3):

- quando l'alimentazione di entrambi i transistor è alta, V_{out} non riceverà corrente (0);

- quando uno dei due transistor NON sarà alimentato, V_{out} sarà alimentato (1).

2.3 La porta XOR



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

La funzione logica XOR restituisce VERO quando soltanto uno dei due valori in ingresso è VERO.
Inoltre, la funzione logica XOR si può rappresentare con la seguente forma canonica:

$$XoR(a,b) = Or(And(a, NOT(b)), And(Not(a), b)) = (a \cdot \overline{b}) + (\overline{a} \cdot b) \quad (1)$$

3 Implementare una funzione logica

3.1 Dalla forma canonica all'implementazione

Per implementare una funzione logica (ovvero per realizzarla mediante porte logiche) occorre innanzitutto studiarne la tabella di verità, in modo da ricavarne la forma canonica.
Ad esempio, avendo la seguente tabella di verità:

A	B	SEL	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

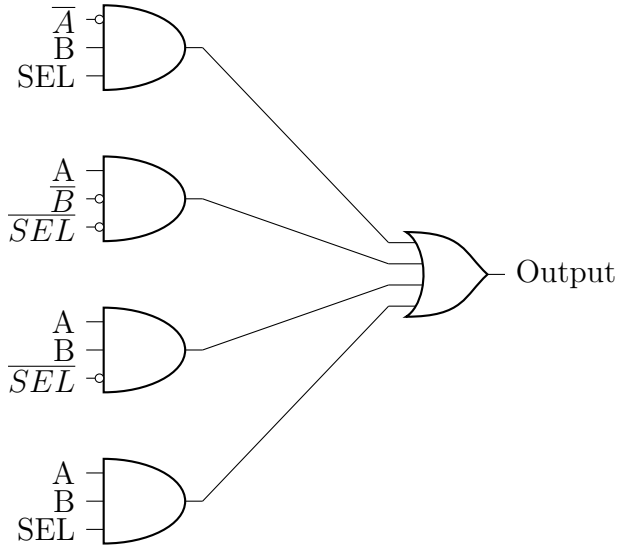
Occorrerà identificare tutti i mintermini della funzione ed effettuare una OR tra questi, ovvero:

- $\overline{A} \cdot B \cdot SEL$
- $A \cdot \overline{B} \cdot \overline{SEL}$
- $A \cdot B \cdot \overline{SEL}$
- $A \cdot B \cdot SEL$

Ed infine:

$$(\overline{A} \cdot B \cdot SEL) + (A \cdot \overline{B} \cdot \overline{SEL}) + (A \cdot B \cdot \overline{SEL}) + (A \cdot B \cdot SEL)$$

Ora che la forma canonica è nota, si può procedere a realizzare il circuito:



3.2 Semplificare un circuito logico

Per semplificare un circuito logico si può utilizzare un potente strumento chiamato **Mappa di Karnaugh**, o **K-map**.
Ecco come realizzare una mappa di Karnaugh:

3.2.1 Definire le variabili

Per realizzare una K-map occorrerà identificare le variabili usate nella funzione logica di cui si sta semplificando la tabella di verità.
In base al numero di variabili scelte (n), si avranno 2^n celle nella tabella.

3.2.2 Disegnare la mappa

La K-map è una griglia dove, in base al numero di variabili cambia il numero di celle.
Ad esempio, per 2 variabili, si avranno $2^2 = 4$ celle:

A/B	0	1
0	...	...
1	...	...

Con l'aumentare del numero delle variabili, invece la cosa cambierà leggermente.
Ad esempio, per 3 variabili:

A/BC	00	01	10	11
0	...	...	...	...
1	...	...	...	...

Mentre con 4:

AD/BC	00	01	10	11
00	...	...	...	...
01	...	...	...	...
10	...	...	...	...
11	...	...	...	...

Inoltre è importante notare come le coppie (00, 01, 10, 11, ...) siano arbitrariamente posizionate: l'obiettivo è creare gruppi di 1 posizionati adiacenti verticalmente, orizzontalmente o in modo toroidale.

3.2.3 Riempire la mappa

Ora bisogna riempire la mappa creata mettendo 1 dove la combinazione di Input restituisce 1 e 0 nelle altre celle. Ad esempio, prendendo la funzione presentata in (3.1):

A/B(SEL)	00	01	10	11
0	0	0	0	1
1	1	0	1	1

Osserviamo ora tuttavia come con questa disposizione delle coppie B,SEL si formino 3 gruppi, e come 2 di questi siano composti da 1 isolati.
Per evitare questa cosa, potremmo provare a cambiare disposizione delle accoppiate, ad esempio invertendo la coppia 10 con quella 11.

A/B(SEL)	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Ora abbiamo due gruppi di 1:

- A = 0 con B, SEL = 11 & A = 1 con B, SEL = 11 (verticalmente);
- A = 1 con B, SEL = 00 & A = 1 con B, SEL = 10 (toroidalmente);

3.2.4 Funzione semplificata

In base ai raggruppamenti effettuati nella sezione precedente, ora possiamo scrivere la forma canonica semplificata della funzione, ovvero: