Calcolatori Elettronici

Esercitazione n 2

Codici a correzione di errore

Recupero degli errori hardware tramite codifiche ridondanti

Codifiche con n = m + r bit (**parola di codice**)

- **n** bit complessivi codifica
- **m** bit dati
- **r** check bit (ridondanti)

Si utilizza solo un sottoinsieme delle codifiche (**codifiche valide**)

Distanza di Hamming h = numero minimo di bit diversi tra due codifiche valide

Per **rilevare** errori su **d** bit occorre **h = d+1**

Per **correggere** errori su **d** bit occorre **h = 2d+1**

Rilevazione di errore singolo

Nel caso più semplice si vogliono solo rilevare errori singoli
Basta aggiungere un solo check bit **r = 1**, **n = m + 1**

Bit di parità: scelto in modo che il numero complessivo di 1 nella codifica sia sempre pari (o dispari)

Questo codice ha distanza h = 2

Errore rilevato da circuiti molto semplici
Le memorie segnalano ***parity error*** quando un errore si manifesta

Correzione di errore singolo

- m data bit, r check bit, n bit totali
- 2^m codifiche valide
- n codifiche errate a distanza 1 da ciascuna delle valide

Ogni codifica valida ne richiede in tutto n+1:

$$(n+1)2^m \leq 2^n \qquad \text{cioè} \qquad (m+r+1) \leq 2^r$$

Correzione di errore singolo

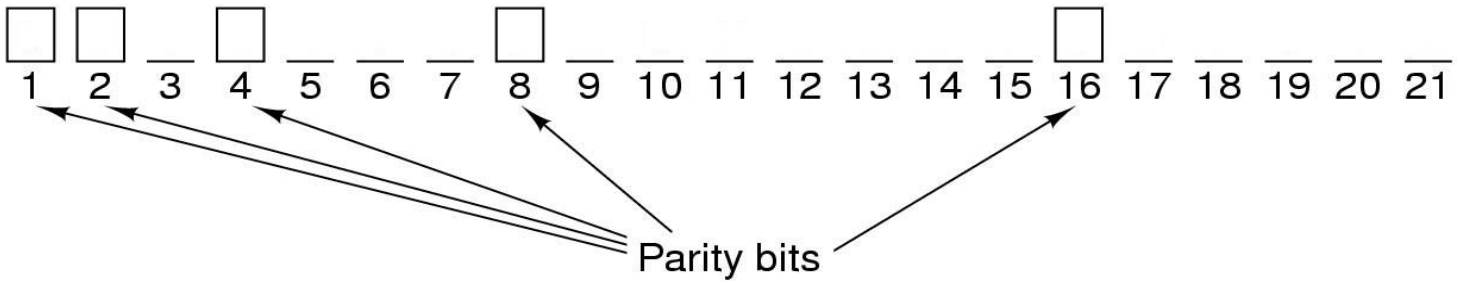
m r n=m+r r/m

Word size	Check bits	Total size	Percent overhead
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2

- Al crescere di m l'overhead scende

Correzione di errore singolo

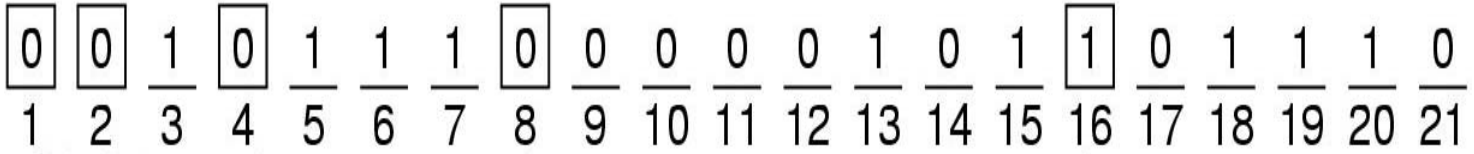
- Codice di Hamming per m = 16;
- r=5 => n=21;
- I 21 bit sono ordinati a partire da 1 con il bit 1 primo bit a sinistra
- I bit la cui posizione è una potenza di due sono bit di controllo (parità)



- Il bit b_i è controllato dai check bit la cui somma è pari a i

Correzione di errore singolo

- Costruzione del codice di Hamming per la parola 1111000010101110



il bit 1 controlla i bit 1,3,5,7,9,11,13,15,17,19,21
il bit 2 controlla i bit 2,3,6,7,10,11,14,15,18,19
il bit 4 controlla i bit 4,5,6,7,12,13,14,15,20,21
il bit 8 controlla i bit 8,9,10,11,12,13,14,15
il bit 16 controlla i bit 16,17,18,19,20,21

Si ottiene la parola di codice
001011100000101101110

- L'inversione del bit 5 genera la parola di codice
001001100000101101110
i bit di parità 1 e 4 sono errati => errore nel bit 1+4=5

L’algebra di Boole

George Boole:

- Matematico inglese del XIX secolo
- Algebra che descrive “le leggi del pensiero”
- Logica da cui è possibile verificare inequivocabilmente verità e falsità
- Sembrava pura teoria, ma costruì il supporto teorico per la trattazione dei circuiti elettronici.

L’algebra di Boole

- Consente di descrivere in forma algebrica le funzioni dei circuiti
- Fornisce dei metodi per l’analisi e la sintesi (a livello logico) dei circuiti
- Tramite l’algebra di Boole si stabilisce una corrispondenza biunivoca tra
  - operazioni dell’algebra e componenti elementari
  - espressioni algebriche e circuiti

Variabile booleana

- La variabile booleana può assumere solo 2 valori possibili:
  - 1 (true, vero)
  - 0 (false, falso).
  - $X \in B$ con $B = \{0, 1\}$

Funzione booleana

- Una funzione $z=f(x_1,...x_n)$ è una legge che fa corrispondere un valore binario di z ad ogni combinazione di valori delle variabili $x_1,...x_n$.

- Una funzione può essere rappresentata tramite una tabella di verità che elenca i valori di z per tutte le suddette combinazioni

- Con n variabili booleane indipendenti ci sono 2^n combinazioni e 2^{2^n} funzioni diverse.

Funzioni booleane per n=2 variabili

X	Y	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	X • Y		X		Y		X+Y			$\overline{Y}$		$\overline{X}$			1

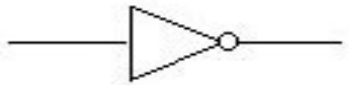
Le funzioni F1, F7, F10 e F12 rappresentano le ben note funzioni di base dell'algebra booleana:

- NOT (negazione)
- AND (prodotto)
- OR (somma)

Funzione NOT

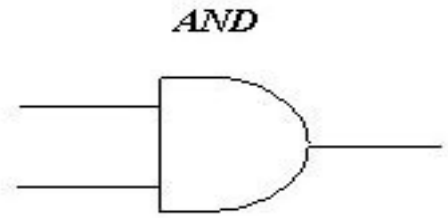
x	z
0	1
1	0

NOT



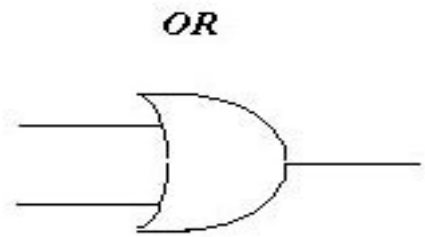
Funzione AND

x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1



Funzione OR

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1



Proprietà degli operatori NOT, AND, OR

$P_1)$ $x+0=x$

$P_2)$ $x+1=1$

$P_3)$ $x+x=x$

$P_4)$ $x+\bar{x}=1$

$P_5)$ $x_1+x_2=x_2+x_1$,

$P_6)$ $(x_1+x_2)+x_3=x_1+(x_2+x_3)$

$P_7)$ $(x_1\cdot x_2)+(x_1\cdot x_3)=x_1\cdot(x_2+x_3)$

$P_1')$ $x\cdot 1=x$

$P_2')$ $x\cdot 0=0$

$P_3')$ $x\cdot x=x$

$P_4')$ $x\cdot\bar{x}=0$

$P_5')$ $x_1\cdot x_2=x_2\cdot x_1$

$P_6')$ $(x_1\cdot x_2)\cdot x_3=x_1\cdot(x_2\cdot x_3)$

$P_7')$ $(x_1+x_2)\cdot(x_1+x_3)=x_1+(x_2\cdot x_3)$.

$P_8)$ $\bar{\bar{x}}=x$.

$P_9)$ $x_1\cdot x_2+x_1\cdot\bar{x}_2=x_1$

$P_{10})$ $x_1+x_1\cdot x_2=x_1$

$P_9')$ $(x_1+x_2)\cdot(x_1+\bar{x}_2)=x_1$

$P_{10}')$ $x_1\cdot(x_1+x_2)=x_1$.

$P_{11})$ $\overline{x_1+x_2+\ldots+x_n}=\bar{x}_1\cdot\bar{x}_2\cdot\ldots\cdot\bar{x}_n$

$P_{11}')$ $\overline{x_1\cdot x_2\cdot\ldots\cdot x_n}=\bar{x}_1+\bar{x}_2+\ldots+\bar{x}_n$.

Definizione (mintermine)

- Un *mintermine* p_i è una funzione che vale 1 in corrispondenza della sola configurazione *i* di valori delle variabili
- Una funzione *mintermine* p_i è rappresentata dal prodotto di tutte le variabili di ingresso, prese in forma diretta se nella configurazione di ingresso *i* compaiono col valore 1, in forma complementata se nella stessa configurazione compaiono col valore 0

Definizione (maxtermine)

- Un *maxtermine* s_i è una funzione che vale 0 in corrispondenza della sola configurazione *i* di valori delle variabili
- Una funzione *maxtermine* s_i è rappresentata dalla somma di tutte le variabili di ingresso, prese in forma diretta se nella configurazione di ingresso *i* compaiono col valore 0, in forma complementata se nella stessa configurazione compaiono col valore 1

Mintermini e maxtermini

	$a\ b\ c$	z	$p_0 p_2 p_4 p_5 p_6$	$s_1 s_3 s_7$
0	0 0 0	1	1 0 0 0 0	1 1 1
1	0 0 1	0	0 0 0 0 0	0 1 1
2	0 1 0	1	0 1 0 0 0	1 1 1
3	0 1 1	0	0 0 0 0 0	1 0 1
4	1 0 0	1	0 0 1 0 0	1 1 1
5	1 0 1	1	0 0 0 1 0	1 1 1
6	1 1 0	1	0 0 0 0 1	1 1 1
7	1 1 1	0	0 0 0 0 0	1 1 0

$$p_0 = \bar{a} \cdot \bar{b} \cdot \bar{c} \qquad s_1 = a + b + \bar{c}$$

$$p_2 = \bar{a} \cdot b \cdot \bar{c} \qquad s_3 = a + \bar{b} + \bar{c}$$

$$p_4 = a \cdot \bar{b} \cdot \bar{c} \qquad s_7 = \bar{a} + \bar{b} + \bar{c}$$

$$p_5 = a \cdot \bar{b} \cdot c$$

$$p_6 = a \cdot b \cdot \bar{c}$$

Forme canoniche SP e PS

- La *prima forma canonica* (detta *sp*) di una funzione *f* è costituita dall'OR di tutti i *mintermini* p_i, per le configurazioni *i* ove *f* = 1
- La *seconda forma canonica* (detta *ps*) di una funzione *f* è costituita dall'AND di tutti i *maxtermini* s_i, per le configurazioni *i* ove *f* = 0
- Per la funzione z dell'esempio precedente le forme canoniche sono:

$$z = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c};$$

$$z = (a + b + \bar{c})(a + \bar{b} + \bar{c})(\bar{a} + \bar{b} + \bar{c}).$$

Completezza funzionale di NOT, AND, OR

- L'esistenza delle forme canoniche è una dimostrazione che gli operatori NOT, AND, OR costituiscono un insieme *funzionalmente completo*
- Esistono vari insiemi di operatori funzionalmente completi
- Negando entrambi i membri della P_{11} si ottiene:

$$x_1 + x_2 = \overline{(\overline{x}_1 \cdot \overline{x}_2)}$$

che dimostra come l'OR sia ottenibile tramite AND e NOT. Essendo funzionalmente completa la scelta degli operatori NOT, AND, OR lo è anche la scelta NOT, AND

- In modo duale, negando entrambi i membri della P_{11}' si ottiene

$$x_1 \cdot x_2 = \overline{(\overline{x}_1 + \overline{x}_2)}$$

che dimostra la completezza funzionale della scelta di operatori NOT, OR

- L'insieme di operatori AND, OR non è funzionalmente completo, in quanto non è possibile ottenere con nessuna combinazione di AND e OR la funzione

$$f(x) = \overline{x}$$

Operatori NAND e NOR

- Si considerino i due seguenti operatori e le relative tabelle di verità

$x_1 x_2$	$x_1 \mid x_2$
0 0	1
0 1	1
1 0	1
1 1	0

NAND

$x_1 x_2$	$x_1 \downarrow x_2$
0 0	1
0 1	0
1 0	0
1 1	0

NOR

- E' immediato constatare che

$$P_{12}) \quad x_1 \mid x_2 = \overline{(x_1 \cdot x_2)}$$

$$P_{13}) \quad x_1 \downarrow x_2 = \overline{(x_1 + x_2)}$$

Operatori NAND e NOR

- Sia l'operatore NAND, sia l'operatore NOR costituiscono, singolarmente, scelte funzionalmente complete
- Si ha, infatti, che:

$$\overline{x} \overset{P_{12}}{=} x|x \quad \text{(indicato anche come } |x\text{, o semplicemente come } \overline{x}\text{)}$$

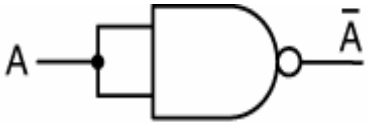
$$x_1 \cdot x_2 \overset{P_{12}}{=} |(x_1|x_2);$$

e

$$\overline{x} \overset{P_{13}}{=} x\downarrow x \quad \text{(indicato anche come } \downarrow x\text{, o semplicemente come } \overline{x}\text{)}$$

$$x_1 + x_2 \overset{P_{13}}{=} \downarrow(x_1 \downarrow x_2);$$

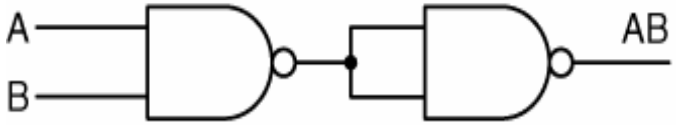
Universalità della porta NAND



$$\overline{AA} = \overline{A}$$

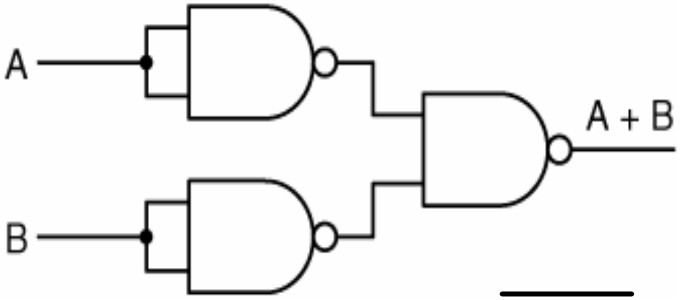
equivale a un NOT

Equivale a un AND



$$AB = \overline{\overline{AB}}$$

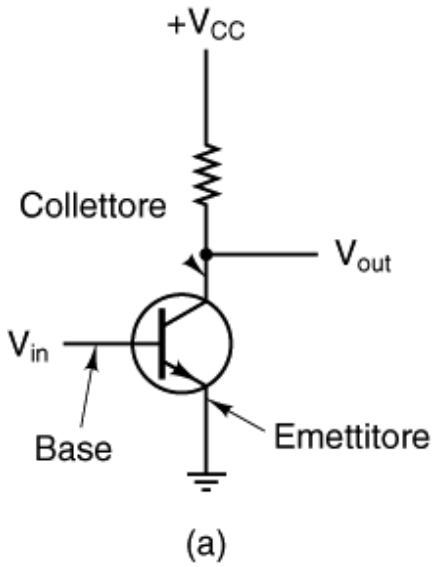
Equivale a un OR



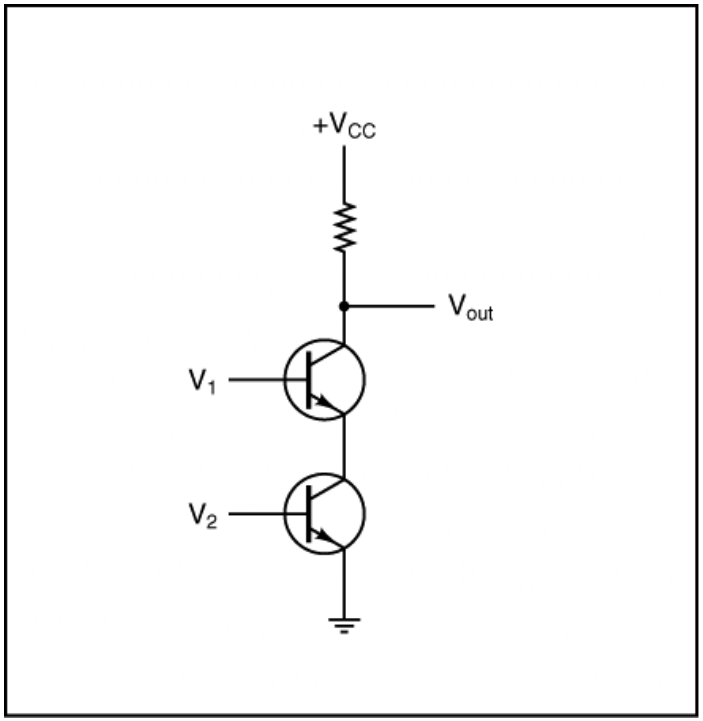
$$A + B = \overline{\overline{A + B}} = \overline{\overline{A}\ \overline{B}}$$

Porte NOT e NAND a livello fisico

Porta NOT



Porta NAND



Esempio: Funzione di Maggioranza

Sintetizzare (in 1a forma canonica) una funzione combinatoria dotata di 3 ingressi A, B e C,
e di un'uscita F, funzionante come segue:

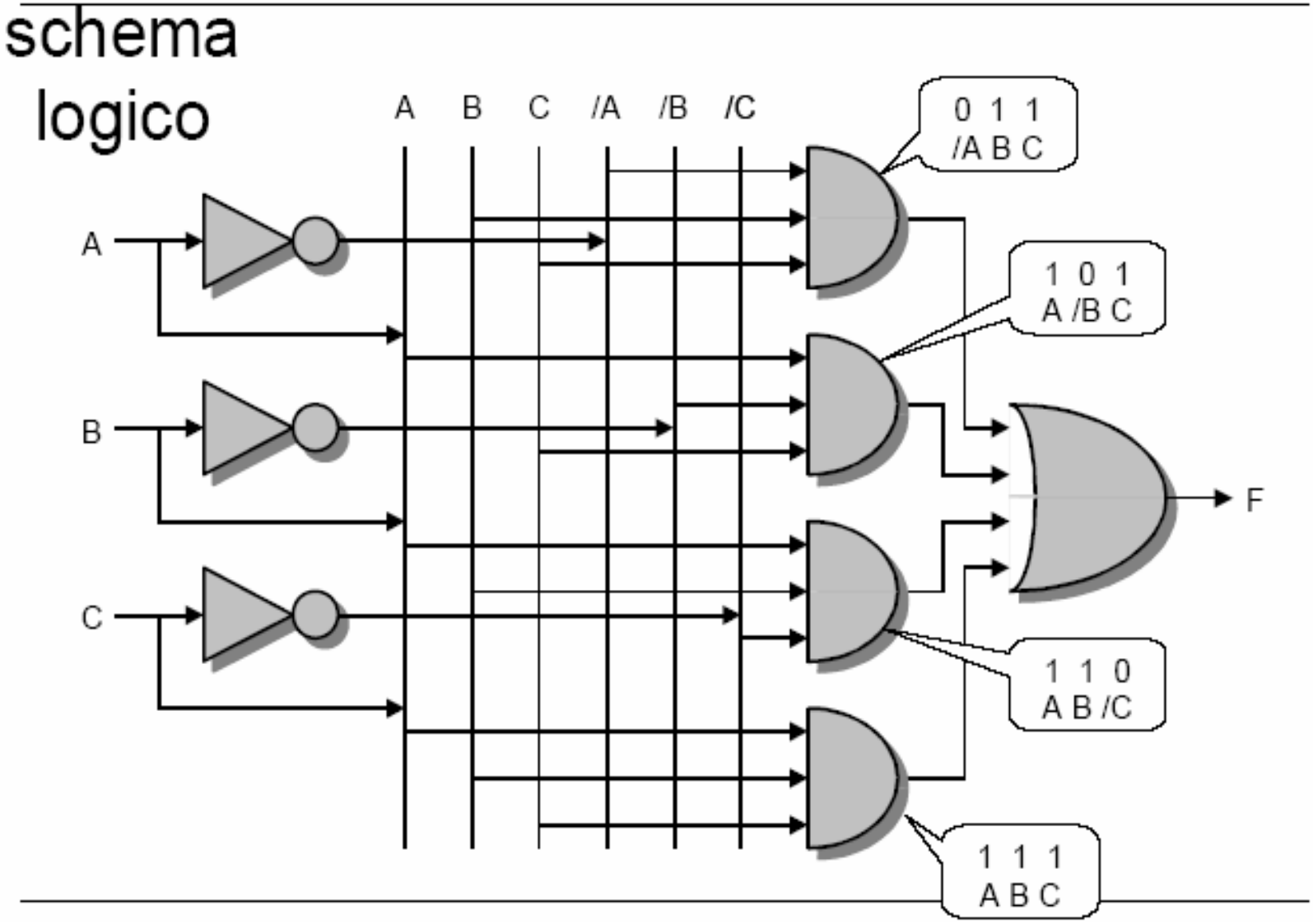
- Se la maggioranza degli ingressi vale 0, l'uscita vale 0
- Se la maggioranza degli ingressi vale 1, l'uscita vale 1

Funzione di Maggioranza: Tabella di verità

- La tabella delle verità della funzione maggioranza è mostrata a lato
- L'uscita vale 1 se e solo se 2 o tutti e 3 gli ingressi valgono 1 (cioè se e solo se il valore 1 è in maggioranza)

# riga	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Funzione di Maggioranza: Rete Combinatoria



Mappa di Karnaugh

- Una mappa di Karnaugh è una rappresentazione di funzioni logiche nella quale le possibili combinazioni di valori sono messe in corrispondenza con le caselle di una mappa rettangolare

- Una mappa per *n* variabili ha 2^n caselle che corrispondono alle 2^n configurazioni di valori delle variabili

- L'*adiacenza logica* fra due configurazioni si riflette in *adiacenza geometrica* tra due caselle

Esempi di mappe di Karnaugh

3 variabili

a \ bc	00	01	11	10
0	1	0	1	0
1	0	0	1	0

4 variabili

a b \ c d	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	1	1	0	0
10	1	0	0	1

Esempi di mappe di Karnaugh

5 variabili

a b \ c d	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	0	1	1	0
10	0	0	1	1

e = 0

a b \ c d	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	1	1	1	0
10	0	0	0	1

e = 1

Premessa teorica (funzioni prodotto)

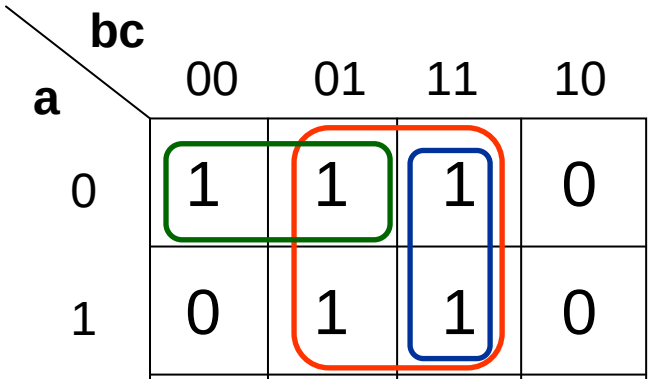
- Una funzione espressa come AND di *k* variabili ($1 \le k \le n$) dirette o negate si dice *funzione prodotto*
- Una funzione prodotto *p* di *k* variabili è rappresentata sulla mappa da 1 contenuti in un insieme di 2^{n-k} caselle adiacenti (*sottocubo relativo a p*)

$p_i = \bar{a}\bar{b}\bar{c}$ $p_j = bc$

a \ bc	00	01	11	10
0	1	0	1	0
1	0	0	1	0

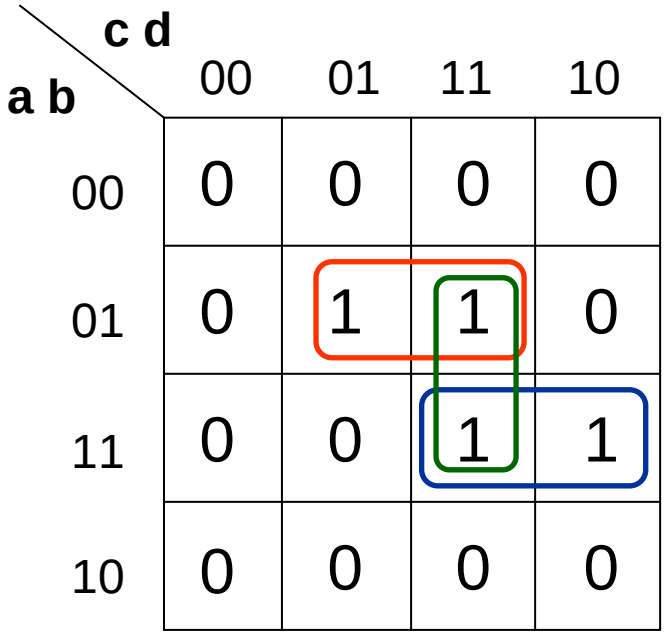
Premessa teorica (implicanti)

- Una funzione prodotto *p* si dice *implicante* di una funzione *f*, se *f*=1 almeno in tutti i *vertici* del sottocubo relativo a *p*
- Un implicante *p* si dice *implicante primo* di *f* se non esiste alcun altro implicante di *f* che copra *p*



Premessa teorica (implicanti primi essenziali)

- Un implicante primo di *f* si dice *implicante primo essenziale* di *f* se è l'unico a coprire un dato 1 della *f*



Sintesi ottima di reti a due livelli

- Unico parametro da minimizzare è il costo della rete, funzione crescente del numero N_b di blocchi AND e OR e del numero totale N_m dei morsetti di ingresso a tali blocchi

- Utilizzo delle mappe di Karnaugh per funzioni con un numero massimo di variabili pari a 5
  - individuazione degli implicanti primi essenziali
  - scelta degli implicanti primi che coprano gli 1 non coperti dagli implicanti primi essenziali

Forme canoniche

$$z = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$$

$$P_8 = \bar{a}\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$$

$$P_3 = \bar{a}\bar{c} + a\bar{b}\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$$

$$P_0 = \bar{a}\bar{c} + a\bar{b} + a\bar{b}\bar{c} + ab\bar{c}$$

$$P_7 = \bar{a}\bar{c} + a\bar{b} + a\bar{c}$$

$$P_9 = \bar{c} + a\bar{b}.$$

Forme canoniche

- Le forme canoniche possono essere semplificate mediante le proprietà dell'algebra
- La forma canonica sp vista in precedenza può essere semplificata nel seguente modo:

Esempio

- Calcolare il valore della seguente espressione booleana:

$$((A \wedge B) \quad \vee \quad ((A \vee C) \quad \wedge \quad (\neg B)))$$

Esempio: Costruzione della tabella di verità

A	B	C	(A∧B)	∨	(A∨C)	∧	(¬ B)
V	V	V	V		V		F
V	V	F	V		V		F
V	F	V	F		V		V
V	F	F	F		V		V
F	V	V	F		V		F
F	V	F	F		F		F
F	F	V	F		V		V
F	F	F	F		F		V

Esempio: Costruzione tabella di verità

A	B	C	$(A \wedge B) \vee ((A \vee C) \wedge (\neg B)$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	V
F	V	V	F
F	V	F	F
F	F	V	V
F	F	F	F

Esempio: Costruzione tabella di verità

A	B	C	$(A \wedge B)$	$\vee$	$(A \vee C) \wedge (\neg B)$
V	V	V	V		F
V	V	F	V		F
V	F	V	F		V
V	F	F	F		V
F	V	V	F		F
F	V	F	F		F
F	F	V	F		V
F	F	F	F		F

Dalla rete alla funzione logica

$$L = \overline{A}\,B + \overline{AC}\,D + BC + BC\,\overline{D}$$