Livello Logico-Digitale

Prof. Ing. Donato Impedovo

Applicazioni basate su regole booleane:

- Regole di funzionamento di centraline di controllo (ad esempio dei semafori..)
- Regole di movimentazione apparati meccanici

Algebra di Boole

Considerato un insieme **S di elementi**:

- Ognuno degli elementi può assumere solo uno dei due valori: 0, 1
- Nel quale esiste almeno una coppia di elementi $X,Y \in S \ni' X \neq Y$
- Per i quali sia definita una legge di composizione, detta **somma logica (+), tale che** $Z = X + Y$, con $X,Y, Z \in S$
- Per i quali sia definita una legge di composizione, detta **prodotto logico (•), tale che** $Z = X \cdot Y$, con $X,Y, Z \in S$
- Che contenga un elemento neutro rispetto alla somma (o zero), indicato dal simbolo 0, **tale che** $X + 0 = X$
- Che contenga un elemento neutro rispetto al prodotto (o unità), indicato dal simbolo 1 , **tale che** $X \cdot 1 = X$

Continua…

Algebra di Boole

Considerato un insieme S di elementi:
…continua

- Per il quale valgono le proprietà commutative delle operazioni ($+$ e $\cdot$), tali che $X + Y = Y + X$ e $X \cdot Y = Y \cdot X$

- Per il quale valgono le proprietà distributive della operazione $+$ rispetto alla operazione $\cdot$ e della operazione $\cdot$ rispetto alla operazione $+$, cioè $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$, $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

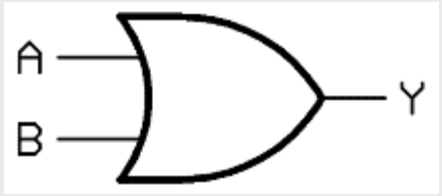
- Preso $X \in S \Rightarrow \exists (\underline{X} \in S) \ni' (X + \underline{X} = 1)$ e $(X \cdot \underline{X} = 0)$

  $\underline{X}$ è detto complemento di X

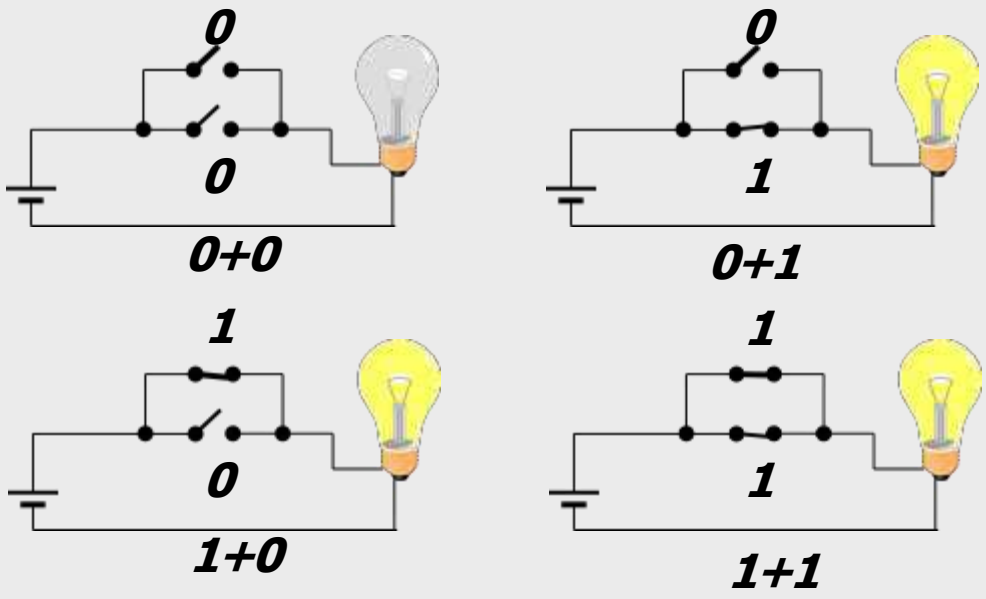
Le proprietà suddette definiscono su S una struttura algebrica comunemente detta Algebra di Boole.

Operatore OR

- Somma logica (OR): il valore della somma logica è 1 se il valore di almeno uno degli operandi è 1.



A	B	A+B A∪B A.OR.B
0	0	0
0	1	1
1	0	1
1	1	1

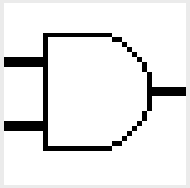


Date n variabili booleane indipendenti la loro somma logica (OR) è

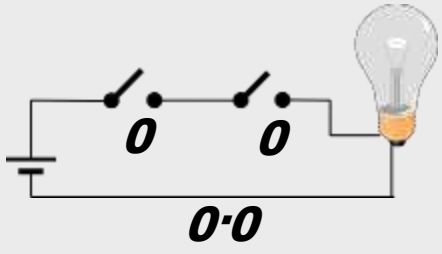
$$x_1+x_2+\ldots+x_n= \begin{cases} 1 \text{ se almeno una } x_i \text{ vale } 1 \\ 0 \text{ se } x_1=x_2=\ldots=x_n=0 \end{cases}$$

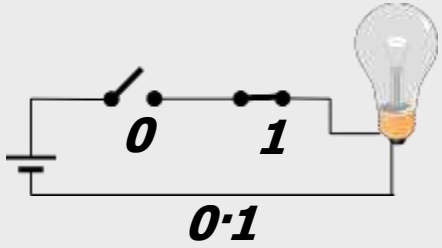
Operatore AND

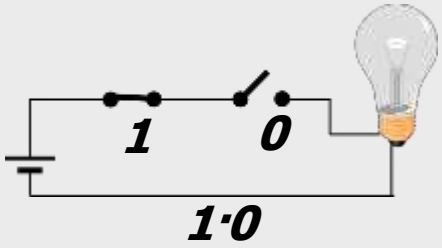
- Prodotto logico (AND): il valore del prodotto logico è 1 se il valore di tutti gli operandi è 1.

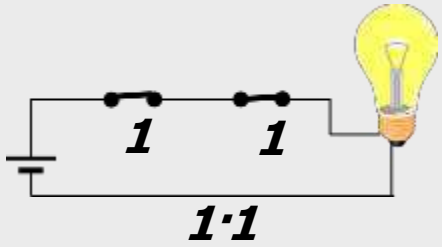


X	Y	X·Y X ∩ Y X.AND.Y
0	0	0
0	1	0
1	0	0
1	1	1









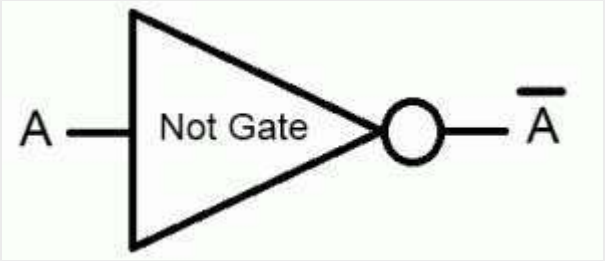
Date n variabili booleane indipendenti il loro prodotto logico (AND) è

$$x_1 \cdot x_2 \cdot \ldots \cdot x_n = \begin{cases} 0 \text{ se almeno una } x_i \text{ vale } 0 \\ 1 \text{ se } x_1 = x_2 = \ldots = x_n = 1 \end{cases}$$

Operatore NOT

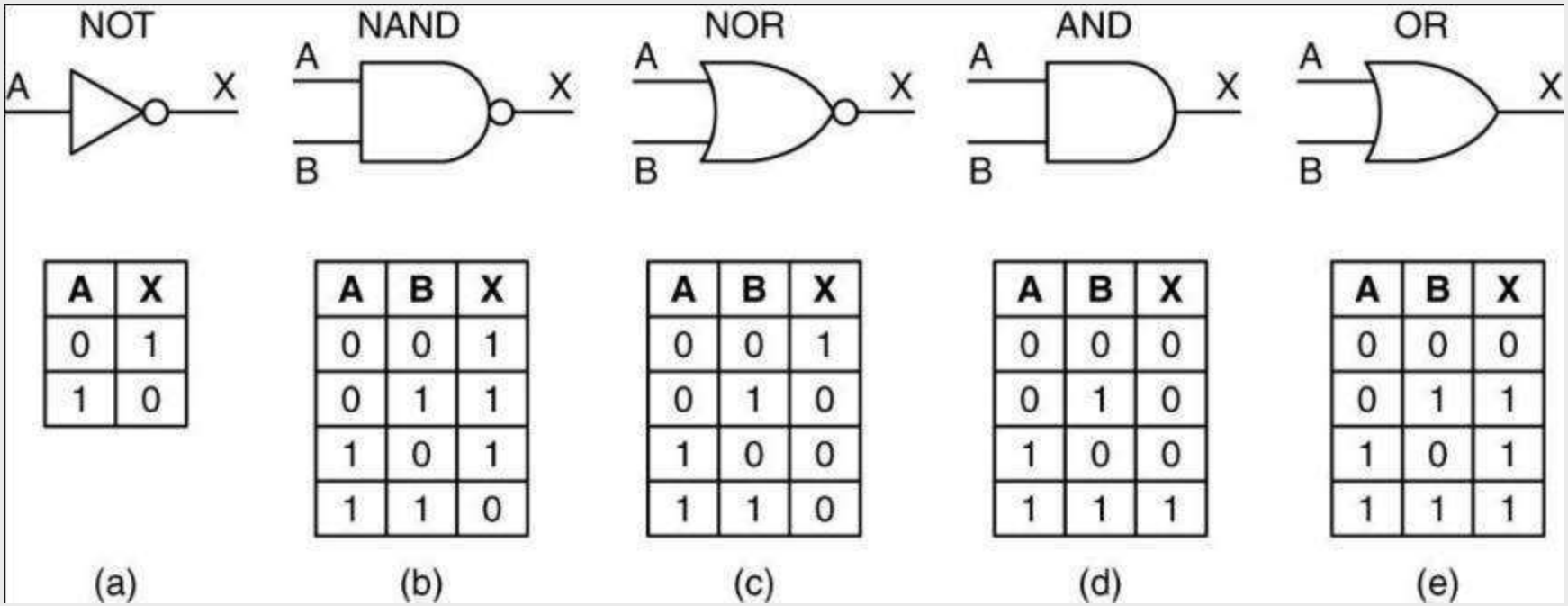
- Operatore di negazione (NOT): l'operatore inverte il valore della costante su cui opera. OPERAZIONE UNARIA.

A	$\overline{A}$!A NOT(A)
0	1
1	0



L'elemento $\overline{A}$ =NOT(A) viene detto complemento di A. Il complemento è unico.

PORTE



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

Funzioni logiche

- Una variabile y è una funzione delle n variabili indipendenti x_1, x_2,…, x_n, se esiste un criterio che fa corrispondere in modo univoco ad ognuna delle 2^n configurazioni delle xun valore di y

$$y = F(x_1, x_2, \ldots, x_n)$$

- Una rappresentazione esplicita di una funzione è la **tabella di verità**, in cui si elencano tutte le possibili combinazioni di x_1,…, x_n, con associato il valore di y

- Per costruire la tabella della verità di un'espressione booleana occorre:
  - semplificare, se possibile, l'espressione mediante i teoremi dell'algebra booleana
  - calcolare i termini parziali della funzione riducendoli alle operazioni fondamentali

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

a	b	c	a·b	a·b + c
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Proprietà

Idempotenza A.or.A=A A.and.A=A

Proprietà dell'idempotenza:

$A + A = A$ $A \cdot A = A$

Infatti:

A	A	A + A
0	0	0
1	1	1

A	A	A · A
0	0	0
1	1	1

Proprietà

Proprietà dell'elemento neutro per OR e AND:

$$A + 0 = A \qquad A \cdot 1 = A$$

Infatti:

A	0	A + 0
0	0	0
1	0	1

A	1	A · 1
0	1	0
1	1	1

Proprietà

Proprietà dell'elemento nullo per OR e AND:

$A + 1 = 1$ $\qquad$ $A \cdot 0 = 0$

Infatti:

A	1	A + 1
0	1	1
1	1	1

A	0	A · 0
0	0	0
1	0	0

Proprietà

Proprietà dell'elemento complementare per OR e AND:

$$A + \overline{A} = 1 \qquad A \cdot \overline{A} = 0$$

Infatti:

A	$\overline{A}$	$A + \overline{A}$
0	1	1
1	0	1

A	$\overline{A}$	$A \cdot \overline{A}$
0	1	0
1	0	0

Proprietà

OR e AND godono delle seguenti proprietà:

Commutativa $A+B=B+A$ $A \cdot B = B \cdot A$

Associativa
$$A+(B+C)=(A+B)+C=A+B+C$$
$$A \cdot (B \cdot C)=(A \cdot B) \cdot C=A \cdot B \cdot C$$

Assorbimento $A \cdot (A+B)=A$ (1) $A+(A \cdot B)=A$ (2)

Dimostrazione (1): $A \cdot (A+B) = A \cdot A + A \cdot B = A + A \cdot B = A \cdot (1+B) = A \cdot 1 = A$

Dimostrazione (2): $A+(A \cdot B)=(A+A) \cdot (A+B)=A \cdot (A+B)$ ..vedi sopra

Distributiva
$$A \cdot (B+C)=(A \cdot B)+(A \cdot C)$$
$$A+(B \cdot C)=(A+B) \cdot (A+C) \quad (*)$$

Dimostrazione di (*): $(A+B) \cdot (A+C) = A \cdot A + A \cdot C + B \cdot A + B \cdot C =$
$= A + A \cdot C + B \cdot A + B \cdot C = A(1+C) + B \cdot A + B \cdot C =$
$= A + B \cdot A + B \cdot C = A(1+B) + B \cdot C = A + B \cdot C$

Proprietà

DE MORGAN $\overline{A+B} = \overline{A} \cdot \overline{B}$ $\overline{A \cdot B} = \overline{A} + \overline{B}$

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

Proprietà

Nome	Forma AND	Forma OR
Elemento neutro	$1A = A$	$0 + A = A$
Assorbimento	$0A = 0$	$1 + A = 1$
Idempotenza	$AA = A$	$A + A = A$
Complementazione	$A\overline{A} = 0$	$A + \overline{A} = 1$
Proprietà commutativa	$AB = BA$	$A + B = B + A$
Proprietà associativa	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Proprietà distributiva	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Legge di assorbimento	$A(A + B) = A$	$A + AB = A$
Legge di De Morgan	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Figura 3.6 Identità dell'algebra booleana.

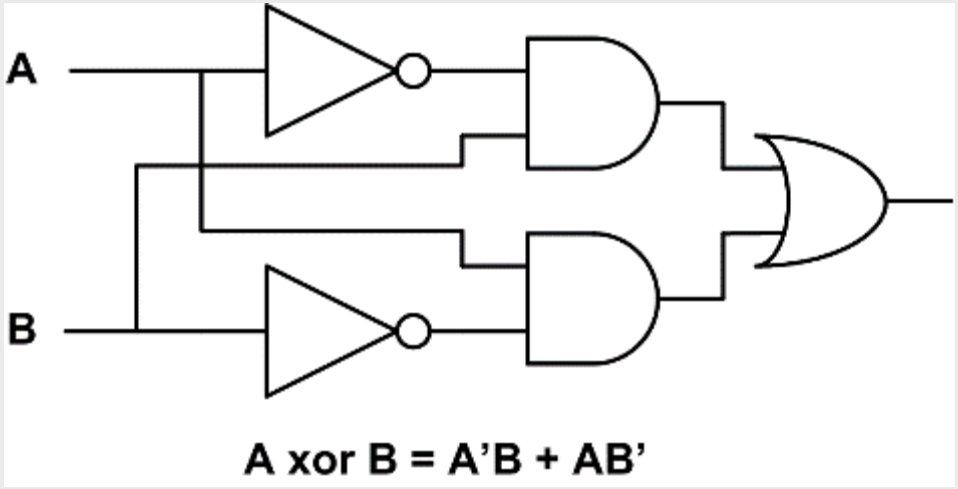
XOR

- La funzione XOR verifica la disuguaglianza di due variabili

Exclusive-OR gate

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

Esercizio:
Si verifichi che $A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$



ESERCIZI

Verificare l'equivalenza delle seguenti funzioni logiche:

- $\overline{A}\,\overline{B}\,\overline{C} + B\,\overline{C} + A\left(B + \overline{BC}\right) \equiv A + \overline{C}$
- $R = a\overline{b} + \overline{a}b$
  $S = \overline{\overline{a}\overline{b} + ab}$
  $T = (a+b)\left(\overline{a} + \overline{b}\right)$
- $F = \overline{a}b + a\overline{b} + \overline{a + bc}$
  $H = \overline{b} + \overline{a}$

ESERCIZI

Esprimere in forma simbolica la seguente preposizione logica: Il passaggio di un astronauta da una nave di servizio ad un satellite artificiale, è permesso se:

- La nave e il satellite sono uniti e alla stessa pressione interna, oppure se
- Sono separati e l'astronauta indossa la tuta pressurizzata.

In entrambi i casi occorre che le pile solari funzionino e giunga il consenso del controllo da terra.

variabili
P, passaggio dell'astronauta;
U, nave e satellite uniti;
I, stessa pressione interna;
T, l'astronauta indossa la tuta pressurizzata;
S, pile solari funzionanti;
C, consenso da terra

NB: veicoli separati = non uniti $(U = 0)$.

$$P = UISC + \overline{U}TSC =$$
$$= SC(UI + \overline{U}T)$$

ESERCIZI

Esprimere in forma simbolica la seguente preposizione logica:
l'avanzamento di un nastro trasportatore è permesso secondo due modi di funzionamento:

1. È inserito l'interruttore di alimentazione e vi sono pezzi da trasportare
2. È inserito l'interruttore, vi sono pezzi da trasportare e il numero di pezzi già trasportato è inferiore ad un limite N (prefissato)

Inoltre l'avanzamento si deve arrestare automaticamente in caso di incidente (es. caduta di un pacco, ecc.).

Variabili
M, modo di funzionamento (M= 1, primo modo; M= 0,secondo modo);
I, posizione interruttore;
P,ci sono pezzi da portare;
N, i pezzi trasportati sono meno di N;
C, c'è stato un incidente;
A, avanzamento del nastro

$$A = (IPM + IP\overline{M}N)\overline{C}$$

ESERCIZI

Progettare una rete combinatoria a tre ingressi che restituisca 1 solo se almeno due degli ingressi valgono 1

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

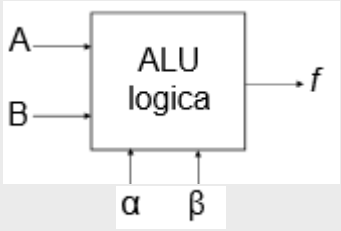
Espressione booleana

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

1. *Si identificano le variabili logiche*
2. *Si crea la parte sinistra della tavola di verità che ha un numero di righe pari a 2^N (N numero di variabili*
3. *Si scrivono tutte le combinazioni*
4. *Si interpreta ogni riga in base al problema e si riporta il valore che deve assumere l'output*
5. *Si considerano solo le righe in cui l'output vale 1 e si scrive la funzione come somma di prodotti logici*

ESERCIZI

- Si scriva, utilizzando gli operatori booleani AND, OR, NOT, la funzione booleana che ritorna in uscita il valore 1 se sono veri due dei quattro input
- Scrivere, utilizzando gli operatori booleani AND, OR e NOT, la funzione logica che riceve in ingresso un numero binario su quattro bit e restituisce VERO se e solo se il numero in ingresso è compreso tra 4 e 7.
- Progettare una rete combinatoria che realizzi un'ALU ad 1 bit capace di eseguire le operazioni logiche bit a bit di AND, OR, NOT, XOR



$f = AB$ se $\alpha=0\ \beta=0$
$f = A+B$ se $\alpha=0\ \beta=1$
$f = \bar{A}$ se $\alpha=1\ \beta=0$
$f = A \text{ xor } B$ se $\alpha=1\ \beta=1$

$$f = \bar{\alpha}\bar{\beta}AB + \bar{\alpha}\beta\bar{A}B + \bar{\alpha}\beta A\bar{B} + \bar{\alpha}\beta AB + \\ + \alpha\bar{\beta}\bar{A}\bar{B} + \alpha\bar{\beta}\bar{A}B + \alpha\beta\bar{A}B + \alpha\beta A\bar{B}$$

ESERCIZI

Trovare le espressioni per le funzioni booleane $f(x_1,x_2)$ e $g(x_1,x_2)$

Definite come segue

$f(x_1,x_2)=0$ se e solo se $x_1=1$ e $x_2=0$

$$g(x_1,x_2)=\begin{cases} x_1 & \text{se } x_2=0 \\ \bar{x}_1 & \text{se } x_2=1 \end{cases}$$

Verificare se la funzione *g* è equivalente alla seguente:

$$h(x_1,x_2)=\overline{f(x_1,x_2)\bullet f(x_2,x_1)}$$

Error Correcting Codes

Per vari motivi come il cambiamento di tensione di alimentazione del chip, accoppiamento tra le piste, fenomeni atmosferici o altre cause, i valori sulle linee di trasmissione possono essere modificati e i dati trasmessi errati.

I codici di rilevazione e/o correzione degli errori sono codici che consentono la rilevazione e/o correzione degli errori in una parola.

Esempio

Parola originaria (trasmessa) 100100

Parola finale (ricevuta) 100110

Error Correcting Codes

Date due parole di codice :

$$A: \; a_{n-1}\, a_{n-2} \ldots \; a_i \ldots \; a_1\, a_0$$

e

$$B: \; b_{n-1} b_{n-2} \ldots \; b_i \ldots \; b_1 b_0$$

La <u>distanza di Hamming</u> tra A e B è definita come:

$$H(A,B) = \sum_{i=0}^{n-1} d_h(a_i, b_i)$$

dove

$$d_h(a_i, b_i) = 0 \quad se \quad a_i = b_i$$

$$d_h(a_i, b_i) = 1 \quad se \quad a_i \neq b_i$$

Error Correcting Codes

Si definisce distanza di Hamming di un codice la minima distanza tra due parole di un codice.

Esempio. Il codice di quattro parole valide:

0000000000

1111100000

0000011111

1111111111

ha distanza di Hamming pari a 5.

Error Correcting Codes

Significato della **distanza di Hamming:** se tra due parole di codice vi è una distanza di Hamming pari a *d*, allora saranno necessari *d* errori singoli per trasformare una parola nell'altra.

A	B	H(A,B)
101	111	1
1100	0011	4
100011	100101	2

Error Correcting Codes

La distanza di Hamming gioca un ruolo chiave nella rilevazione e correzione di errori in un codice:

- Per rilevare d errori singoli è necessario un codice con distanza di Hamming d+1 (infatti in questo modo non esiste alcun modo in cui d errori singoli possono cambiare una parola valida in un'altra parola valida);

- Per correggere d errori singoli è necessario un codice con distanza di Hamming 2d+1 (infatti in questo modo anche con d cambiamenti la parola di codice originaria continua ad essere "più vicina" rispetto a tutte le altre non esiste alcun modo in cui d errori singoli possono cambiare una parola valida in un'altra parola valida);

Error Correcting Codes

Esempio. Dato il codice con distanza di Hamming pari a 5:

0000000000
1111100000
0000011111
1111111111

In questo caso è possibile:

- rilevare fino a 4 errori (d+1=5): 0000000001; 1111000011; 1010101000 (Non è possibile rilevare i 5 errori che modificano 0000000000 in 0000011111 !!!)
- correggere fino a 2 errori (2d+1=5): 0011111111 ➐ 1111111111; 0000010101 ➐ 0000011111 (Non è possibile correggere i 3 errori che modificano 0000000000 in 0000000111 !!!)

Error Correcting Codes

- Nei codici a rilevazione e/o correzione di errori si utilizzano alcuni bit extra (ridondanti) che vengono aggiunti alla parola stessa. Questi bit ridondanti si chiamano anche "bit di controllo"

- L'idea è quella di creare codici con distanza di Hamming maggiore al fine di poter rilevare e/o correggere errori.

Data una parola di m bit di dati, si aggiungono r bit extra di controllo.
Si ottiene così una unità di n=m+rbit(codeword)

Con una parola di m bit tutte le 2^m combinazioni sono legali ma, per via di come sono calcolati i bit di controllo, solo 2^m delle 2^n parole di codice sono valide.

Error Correcting Codes
Bit di parità

Esempio: Codice con controllo di parità

Dato + 1 bit "di parità"

Il bit di parità viene scelto in modo tale che il numero di bit 1 nella parola di codice sia pari (oppure dispari)

Dato	bit di parità
1001011	0
1011	1
1111011	0
1011001111	1

Un codice con bit di parità ha distanza di Hamming pari a 2: ogni singolo errore genera una parola di codice la cui parità è errata.

Esempio: *A: 10011* *1 (parity bit)*
B: 10001 *0 (parity bit)*

Servono due errori singoli per modificare una parola di codice valida in un'altra parola valida.

NB: so che si è verificato un errore singolo....ma non so dove!!

Error Correcting Codes
Codice di Hamming

Codice di Hamming

Problema: Si vuole realizzare un codice con m bit dati ed r bit di controllo, che sia capace di correggere tutti gli errori singoli.

Error Correcting Codes
Codice di Hamming

Esempio: Codice di Hamming

Ciascuna delle 2^m parole legali A ha:

- n (n=m+r) parole illegali a distanza 1 da essa.
- richiede quindi (n+1) stringhe di bit ad essa dedicate (1 per la parola corretta ed n per i possibili errori).

Quindi per poter rappresentare 2^m parole abbiamo bisogno di $(n+1)\cdot 2^m$ stringhe differenti.

Allora dovrà essere:

$$(n+1)\cdot 2^m \leq 2^n$$

Ovvero

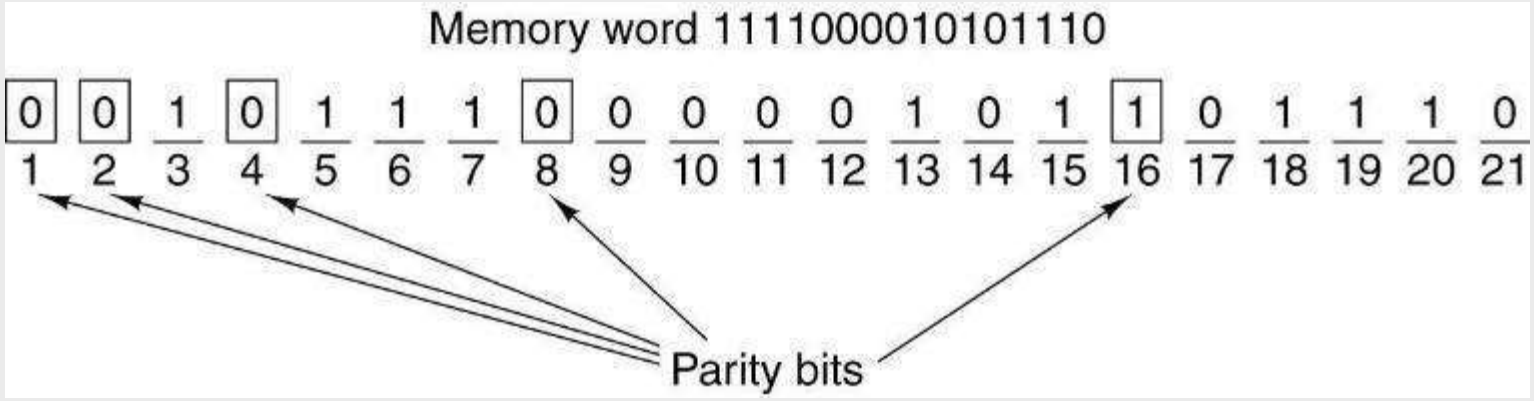
$$(m+r+1) \leq (2^n / 2^m) = 2^r$$

Error Correcting Codes
Codice di Hamming

Number of check bits for a code that can correct a single error ($(m+r+1) \leq 2^r$)

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

Error Correcting Codes
Codice di Hamming



- Nel codice di Hamming gli r bit di parità sono aggiunti a una parola di m bit, formando una nuova parola di n bit (n=m+r).
- I bit sono numerati da 1 (MSD – Most Significant Digit).
- Tutti I bit la cui posizione è potenza di 2 sono bit di parità (1,2, 4,8, 16, ecc.)
- Quelli restanti sono usati per i dati.

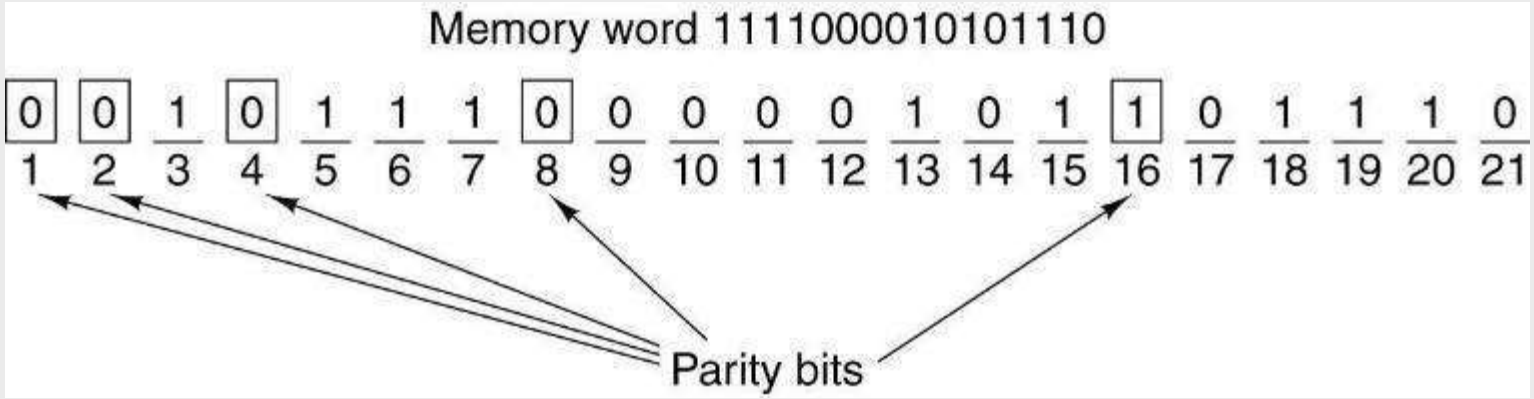
Error Correcting Codes
Codice di Hamming

Ciascun bit di parità controlla posizioni specifiche dei bit ed è impostato in modo che sia pari il numero totale dei bit che hanno valore 1 nelle posizioni controllate.

- Il bit 1 controlla i bit dispar: 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.

In particolare il bit di posto b è controllato dai bit di controllo $b_1,b_2,\ldots,b_j$ tale che $b= b_1+b_2+\ldots+b_j$.

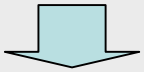
Error Correcting Codes
Codice di Hamming



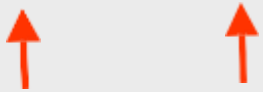
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	*		*		*		*		*		*		*		*		*		*		*
2		*	*			*	*			*	*			*	*			*	*		
4				*	*	*	*					*	*	*	*					*	*
8								*	*	*	*	*	*	*	*						
16																*	*	*	*	*	*

Error Correcting Codes
Codice di Hamming

0	0	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19



0	0	1	0	*0*	1	1	0	0	0	0	0	1	0	1	1	0	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	*		*		*		*		*		*		*		*		*		*		*
2		*	*			*	*			*	*			*	*			*	*		
4				*	*	*	*					*	*	*	*					*	*
8								*	*	*	*	*	*	*	*						
16																*	*	*	*	*	*