Tecnica Digitale

SUPSI Dipartimento Tecnologie Innovative

Gianni Grasso

7 novembre 2024

Classe: I1B

Anno scolastico: 2024/2025

Indice

1 Sistemi numerici

Nella vita di tutti i giorni siamo abituati ad usare un sistema numerico decimale, qui di seguito è riportato un esempio di come funziona invece il sistema numerico binario:

- Numeri decimali

$$\begin{aligned} 5374_{10} &= 5 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0 \\ &= 5000 + 300 + 70 + 4 \end{aligned}$$

- Numeri binari

$$\begin{aligned} 1101_2 &= 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 8 + 4 + 0 + 1 \\ &= 13 \end{aligned}$$

- Conversione da decimale a binario

$$47_{10} \text{ to binary:}$$

$$32 \cdot 1 + 16 \cdot 0 + 8 \cdot 1 + 4 \cdot 1 + 2 \cdot 1 + 1 \cdot 1 = 101111_2$$

1.1 Il sistema binario

Sapendo il numero di bit di un numero binario possiamo determinare:

- Il numero di valori che può generare $\rightarrow 2^N$
- Il range di numeri possibili da generare $\rightarrow [0; 2^N - 1]$

Ad esempio se si hanno a disposizione 3 bit:

- $2^3 = 8$ possibili valori
- $[0; 7] = [000_2; 111_2]$

1.2 Potenze di due

È quindi importante essere facilmente in grado di determinare le potenze di due, o almeno una parte di esse:

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

Figura 1: Memorizzare fino a 2^9

1.3 Sistema esadecimale

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Figura 2: Numeri esadecimali

In questo caso il sistema è simile a quello binario, di seguito sono riportati degli esempi di alcune conversioni:

- Da esadecimale a binario:

$$4AF_{16} \rightarrow \text{Ogni cifra ha 4 bit}$$

$$0100\ 1010\ 1111_2$$

- Da esadecimale a decimale:

$$4AF_{16} \rightarrow \text{Come per i numeri binari}$$

$$16^2 \cdot 4 + 16^1 \cdot 10 + 16^0 \cdot 15 = 1199_{10}$$

Nota: Ogni cifra di un numero esadecimale é composto da un **nibble**, ovvero 4 bit. Solitamente con i numeri esadecimali non si parla più di bit più significativo e meno significativo ma di **byte**. Un byte sono 8 bit di conseguenza il MSB (Most Significant Byte) saranno le due cifre più a sinistra.

2 Operazioni aritmetiche con numeri binari

2.1 Addizione

Le addizioni con numeri binari sono leggermente diverse da quella alla quale siamo abituati, solitamente svolgiamo questa operazione in colonna, ecco un esempio pratico:

$$\begin{array}{r} 11 \\ 1011 \\ +0011 \\ \hline 1110 \end{array}$$

In generale valgono le seguenti regole per l'addizione tra due numeri binari, A e B sono i due addendi mentre C è il carry (riporto) e R il risultato:

A	B	C	R
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

2.1.1 Overflow

Potrebbe succedere che il risultato di un operazione binaria sia più grande del numero massimo permesso dal numero di bit degli addendi, in questo caso si ha un problema di overflow:

$$\begin{array}{r} 111 \\ 1011 \\ +0110 \\ \hline 10001 \end{array}$$

Infatti con 4 bit si possono rappresentare i numeri nell'intervallo $[0; 2^4 - 1] = [0; 15]$. Nel caso di prima il numero $10001_2 = 17_{10}$ è fuori dal range di numeri che si possono rappresentare.

2.2 Segno/modulo

La rappresentazione in segno/modulo ci permette di rappresentare i numeri binari negativi. In questa rappresentazione il MSB rappresenta il segno del numero (**0 = positivo** e **1 = negativo**). Di fatto stiamo sacrificando un bit per rappresentare il segno, il range di numeri che è possibile rappresentare con $N - bit$ quindi cambia, diventando $[-(2^{N-1} - 1), 2^{N-1} - 1]$.

Nota: Utilizzando questa rappresentazione non è possibile sommare due numeri, il risultato non sarà esatto.

2.3 Complemento a due

Il complimento a due ci permette di ovviare a questo problema, esso consiste nell'invertire tutte le cifre di un numero binario e sommare ad esso 1, ad esempio per ottenere -3, considerando $3_{10} = 0011_2$ i passaggi saranno:

$$\begin{array}{r} 1100 \\ +0001 \\ \hline 1101 \end{array}$$

Anche in questo caso il range di numeri che si possono rappresentare cambia diventando $[-(2^{N-1}), 2^{N-1} - 1]$. **Nota:** in questo caso il bit più significativo non rappresenta solo il segno, fa anche parte della cifra, in questo modo è possibile sommare due cifre mantenendo il sistema coerente. Per convertire un numero negativo in positivo il procedimento è lo stesso, basta rifare il comlemento a due.

2.3.1 Estensione del segno

Ci sono due tipi di estensioni:

- Estensione di segno, il MSB viene copiato per tutti i bit aggiuntivi
  Ad esempio $3 = 0011$ diventa 00000011
  Oppure $-5 = 1011$ diventa 11111011
  Nota: possiamo eseguire il complemento a due sul numero esteso per far quadrare i conti
  $11111011 \rightarrow 00000100 + 1 = 00000101 = 5$
- Estensione di zero, vengono aggiunti zeri fino al numero di bit desiderato
  Ad esempio $1011_2 = 5_{10}$ diventa $00001011_2 = 11_{10}$

3 Porte logiche

3.1 Porte a input singolo

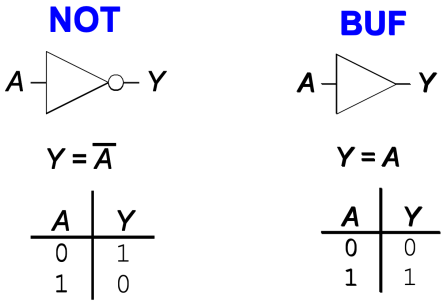


Figura 3: Porte logiche con un solo input

La porta **NOT** restituisce come output il valore negato rispetto all'input, la porta **BUF** invece restituisce la copia del valore di input.

3.2 Porte a doppio input

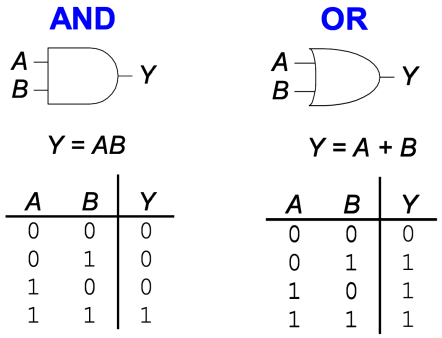


Figura 4: Porte logiche con due input

La porta logica **AND** restituisce 1 solo se il valore di entrambi gli input è 1, la porta **OR** invece restituisce 1 se almeno uno dei valori di input è 1.

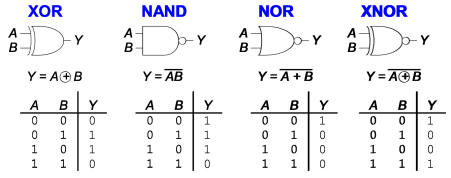


Figura 5: Altre porte logiche a due input

- **XOR**, (**OR** esclusivo), restituisce 1 solo quando un solo valore di input è 1
- **NAND**, restituisce l'output contrario di una porta **AND**
- **NOR**, restituisce l'output contrario di una porta **OR**
- **XNOR**, restituisce l'output contrario di una porta **XOR**

4 Livelli logici

Quando parliamo di livelli logici non si considerano i valori continui ma soltanto quelli discreti. Immaginiamo un grafico in cui per ogni valore di x la y può essere soltanto 1 o 0.

Se però abbiamo un valore intermedio (nè 1 nè 0), la lettura del valore diventerebbe ambigua

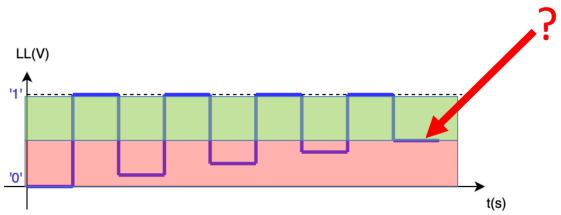


Figura 6: Livelli logici reali

4.1 Rumore

Questi valori intermedi possono esseri causati dal rumore. Il rumore è tutto ciò che disturba il segnare è può alterare il valore logico durante il passaggio di input o do output.

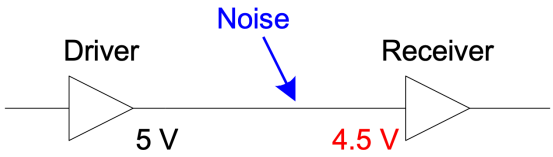


Figura 7: Esempio di rumore

4.2 Margini del rumore

Prendendo come riferimento l'immagine sopra, definiamo i concetti di **Voltage output high/-low** (V_{OH}, V_{OL}) e **Voltage input high/low** ($V_{IH,V_{IL}}$). Essi definiscono i range di valori per l'1 logico e lo 0 logico rispettivamente in entrata e in uscita.

Le differenza tra tensione di uscita del driver e tensione d'entrata del receiver viene chiamata **Noise Margin** (NM_H **e** NM_L).

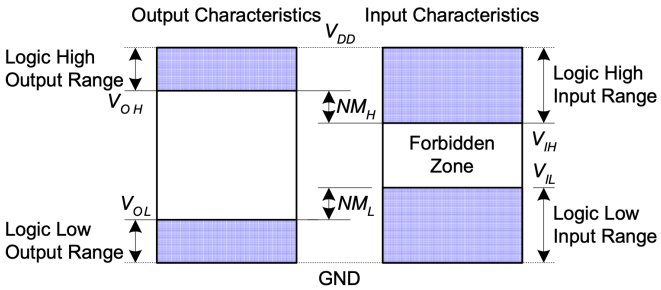


Figura 8: Schema noise margins

Se un valore logico non rientra nei margini esso entra nella **forbidden zone**, è importante definire bene questa zona. Se essa è troppo piccola si avrà uno scenario simile a quello iniziale, in cui non si sa che valore logico assegnare, se invece è troppo grande in caso di rumore si ha il rischio di non poter leggere il valore.

Per calcolare questi margini in generale valore

$$NM_H = V_{OH} - V_{IH}$$
$$NM_L = V_{IL} - V_{OL}$$

5 Transistors

Un transistor è un dispositivo elettronico semiconduttore che può funzionare come interruttore o amplificatore. Esso può essere di due tipi: **nMOS** e **pMOS**. nMOS è tipicamente collegato alla messa a terra (GND), mentre pMOS è generalmente collegato a Vdd.

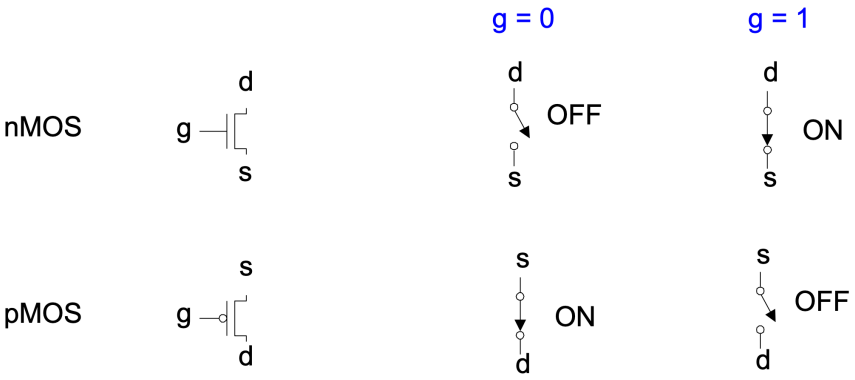
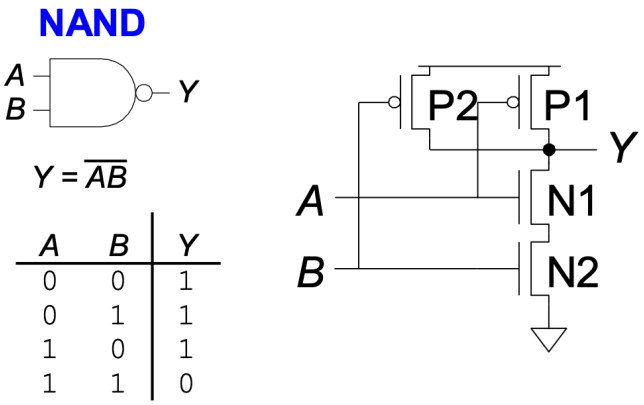


Figura 9: nMOS e pMOS

I transistor vengono utilizzati all'interno delle porte logiche per definirne il funzionamento, di seguito è riportato un esempio per la porta NAND:



A	*B*	P1	P2	N1	N2	*Y*
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

Figura 10: CMOS NAND Gate

Alcune porte logiche non sono strutturalmente definibili in questo modo, ad esempio per la porta AND è necessario realizzare una porta NAND seguita da una porta NOT.

6 Consumo energetico

La Potenza viene espressa in *Watt* e il suo consumo si divide in:

- Consumo dinamico
- Consumo statico

Il consumo dinamico rappresenta il consumo durante l'utilizzo di un dispositivo e vale:

$$P_{dynamic} = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f$$

Dove C rappresenta la capacità, V_{DD} la tensione e f la frequenza. Quest'ultima inoltre è il paramentro più facile da controllare quando vogliamo alzare o abbassare il livello di consumo dinamico.

Per quanto riguarda il consumo statico, esso rappresenta il consumo "a riposo" ed è determinato come:

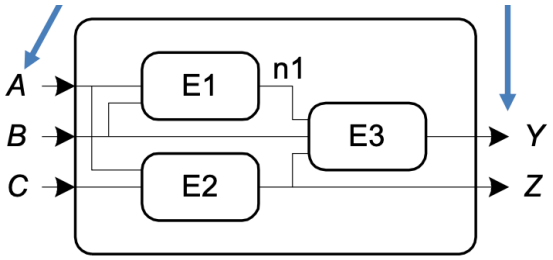
$$P_{static} = I_{DD} \cdot V_{DD}$$

E ovviamente il consumo totale è calcolato dalla somma degli altri due:

$$\begin{aligned} P_{total} =& P_{dynamic} + P_{static} \\ =& \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \quad + \quad I_{DD} \cdot V_{DD} \end{aligned}$$

7 Circuiti logici

Un circuito logico è composto da n input e m output (non c'è solo un ouput come per le porte logiche). Inoltre al suo interno possonoo esserci dei circuiti, ogni circuito è un elemento.



Esistono due tipi di circuiti logici:

- **Logica combinazionale (CL)**, non ha memoria, l'output è determinato solo dagli input
- **Logica sequenziale**, ha memoria, l'output è determinato anche dagli input precedenti

Per ora concentriamoci sui circuiti con **logica combinazionale**, ogni elemento di questo tipo di circuito deve essere anch'esso combinazionale. Inoltre ogni nodo di un circuito combinazionale è collegato sia a un ingresso che a un uscita.

7.1 Algebra booleana

Alcune definizioni:

- **Complemento**: una viariabile "negata", $\bar{A}, \bar{B}, \bar{C}$
- **Letterale**: una viariabile o il suo complemento, $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- **Implicante**: il prodotto di due o più letterali, $AB\bar{C}, A\bar{B}, BC$
- **Minterm**: un implicante che inlcude tutte le variabili di input esattamente una volta, $AB\bar{C}, A\bar{B}\bar{C}, ABC$
- **Maxterm**: la somma che include tutte le variabili di input esattamente una volta, $(A + \bar{B} + C), (\bar{A} + B + \bar{C}), (\bar{A} + \bar{B} + C)$

Nota: $(A + \bar{B} + \bar{A} + C)$ non è **Maxterm** perchè l'input A c'è due volte, anche se una di queste è negato.

Di seguito è riportata una tabella con tutti i teoremi e gli assiomi che possiamo utilizzare:

1	$x + y = y + x$	COMMUTATIVITÀ	$x.y = y.x$
2	$x + 0 = x$	IDENTITÀ	$x.1 = x$
3	$x + (y.z) = (x + y).(x + z)$	DISTRIBUTIVITÀ	$x.(y + z) = (x.y) + (x.z)$
4	$x + \overline{x} = 1$	COMPLEMENTARIETÀ	$x.\overline{x} = 0$
5	$x + x = x$	IDEMPOTENZA	$x.x = x$
6	$1 + x = 1$	DOMINANZA	$x.0 = 0$
7	$x + x.y = x$	ASSORBIMENTO	$x.(x + y) = x$
8	$x + (y + z) = (x + y) + z$	ASSOCIATIVITÀ	$x.(y.z) = (x.y).z$
9	$\overline{x + y} = \overline{x}.\overline{y}$	TEOREMA DI DE MORGAN	$\overline{x.y} = \overline{x} + \overline{y}$
10	$\overline{(\overline{x})} = x$	INVOLUZIONE	$\overline{(\overline{x})} = x$
11	$x + f(x, y, z...) = x + f(0, y, z...)$	TEOREMA DI SHANNON 1	$x.f(x, y, z...) = x.f(1, y, z...)$
12	$\overline{x} + f(x, y, z...) = \overline{x} + f(1, y, z...)$	TEOREMA DI SHANNON 2	$\overline{x}.f(x, y, z...) = \overline{x}.f(0, y, z..)$
13	$x.y + \overline{x}.z = x.y + \overline{x}.z + y.z$	CONSENSO	$(x + y).(\overline{x} + z) = (x + y).(\overline{x} + z).(y + z)$
14	$x.y + x.\overline{y} = x$	—	$(x + y).(x + \overline{y}) = x$

Figura 11: Proprietà e teoremi dell'algebra di Boole

7.1.1 Somma dei prodotti

La **SOP** (Sum-of-Products) è una forma nella quale possiamo scrivere una qualsiasi equazione booleana utilizzando i **minterm**.

$$Y = F(A,B) = \sum_{i=0}^{n}(mi) \qquad ; n = 2^N - 1$$

Guardiamo un esempio per capire come utilizzarala:

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

In questo caso

$$Y = F(A,B) = \bar{A}B + AB = \sum(1,3)$$

7.1.2 Prodotto delle somme

La **POS** (product-of-Sums) è una forma nella quale possiamo scrivere qualsiasi equazione booleana utilizzando i **maxterm**.

$$Y = F(A,B) = \prod_{i=0}^{n}(Mi) \qquad ; n = 2^N - 1$$

Guardiamo un esempio per capire come utilizzarala:

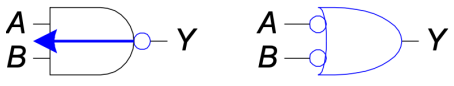
A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

In questo caso

$$Y = F(A,B) = (A+B)(\bar{A}+B) = \prod(0,2)$$

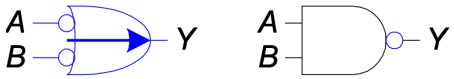
7.1.3 Teorema di DeMorgan

$$Y = \bar{A}B = \bar{A} + \bar{B}$$



ma anche

$$Y = A \bar{+} B = \bar{A} \cdot \bar{B}$$



Queste operazioni sono anche conosciute come **bubble pushing**. Esso si applica preferibilmente in modo tale che l'entrata sia uguale all'uscita. In altre parole se l'output è negato preferibilmente vogliamo avere anche l'input sucessivo negato in modo tale che sia facile semplificare l'espressione booleana.

7.2 Da espressione a porte logiche

Per passare da un espressione booleana ad uno schema logico seguiamo degli step:

1. Scriviamo tutti i letterali presenti nella funzione
2. Scriviamo le espressioni intermedie
3. Colleghiamo i letterali rendendo le espressioni coerenti
4. Scriviamo l'output

In generale durante la creazione di uno schema seguiamo le seguenti pratiche:

- I letterali (gli input) vanno scritti sulla sinistra o nella parte superiore
- Gli output vanno scritti sulla destra o sulla parte inferiore
- Il flusso delle porte va da sinistra verso destra
- Le linee devono essere preferibilmente dritte
- La giunzione di più cavi è rappresentata con un punto (i cavi che si intersecano senza punto non sono connessi)

7.3 Circuito prioritario

In questo tipo di circuito viene considerato solo il bit più significativo (**MSB**), in caso di più input con valore `1` viene considerato per l'output solo quello con MSB maggiore. Nella tabella della verità ci sarà sempre un solo bit con valore `1` per riga:

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

7.4 Don't Cares

Don't Cares è una tabella della verità che esclude i valori "inutili" di un'altra tabella della verità analoga. Nel caso del circuito prioritario la tabella sarebbe:

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

I don't cares vengono usati per semplificare i circuiti, decidiamo noi il criterio ma solitamente li usiamo per situazioni che non possono verificarsi.

7.5 Mappe di Karnaugh

7.5.1 Grey code

Il Grey Code è una rappresentazione alternativa per i numeri binari:

#	Binary	Gray	#
0	0 0 0 0	0 0 0 0	0
1	0 0 0 1	0 0 0 1	1
2	0 0 1 0	0 0 1 1	3
3	0 0 1 1	0 0 1 0	2
4	0 1 0 0	0 1 1 0	6
5	0 1 0 1	0 1 1 1	7
6	0 1 1 0	0 1 0 1	5
7	0 1 1 1	0 1 0 0	4
8	1 0 0 0	1 1 0 0	12
9	1 0 0 1	1 1 0 1	13
10	1 0 1 0	1 1 1 1	15
11	1 0 1 1	1 1 1 0	14
12	1 1 0 0	1 0 1 0	10
13	1 1 0 1	1 0 1 1	11
14	1 1 1 0	1 0 0 1	9
15	1 1 1 1	1 0 0 0	8

7.5.2 Come creare una mappa

Data una funzione

$$Y = f(A, B, C, D)$$

la sua mappa di Karnaugh si costruisce facendo una tabella contenente 2^n colonne, dove n è il numero di input per la parte superiore e 2^m righe, dove m è ikl numero di input per la parte inferiore.

Nota: la scelta di quanti e quali input usare per le righe e per le colonne è libera.

Ad ogni casella inoltre corrisponde una combinazione di Grey Code per le righe e una per le colonne. Il valore delle caselle è dato dalla combinazione di questi codici. La struttura consigliata per questo tipo di mappa è di **massimo 4 input**.

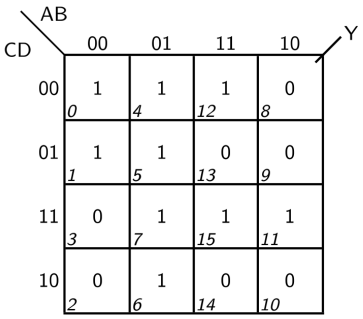
C D \ A B	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

7.5.3 Come usare una mappa

Possiamo utilizzare queste mappe per risolvere graficamente una funzione booleana. Sostituendo i valori del Grey Code con gli input e gli input negati della funzione possiamo facilmente stabilire dei pattern nella mappa per i **SOP** o i **POS** (casi più comuni).

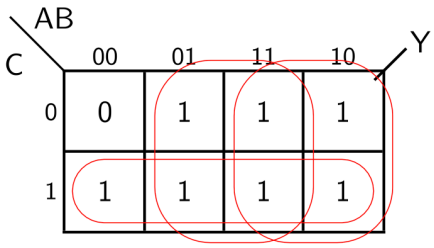
7.5.4 Da tabella della verità a mappa

	A	B	C	D	Y
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1



7.5.5 Formare i gruppi

Possiamo utilizzare una mappa di Karnaugh per trovare dei pattern. In generale due caselle adiacenti hanno sempre e solo un bit di differenza e sono quelle che quindi possono essere ragggruppate. Bisogna sempre cercare di fare meno gruppi possibili, prendendo quindi le combinazioni più grandi possibili.



Ogni gruppo formato rappresenta un'espressione booleana, se abbiamo formato correttamente i gruppi esse si semplificano ad un solo letterale. La funzione finale semplificata al massimo sarà l'unione dei letterali restituiti da ogni singolo gruppo.

Nota: la soluzione finale della funzione non è univoca, potrebbero esserci più soluzioni valide semplificate al massimo prendendo ad esempio gruppi diversi.l

7.6 Metodo Quine-McClusky

Il metodo Quine-McClusky è un algoritmo alternativo alla mappa di Karnaugh, lo si preferisce quando abbiamo più di 4 input. Per applicare McClusky bisogna partire da una specifica situazione iniziale:

1. Partire dalla forma canonica di una funzione (SOP o POS)
2.

7.6.1 Peso e distanza

$$HD(XY) = w(X \oplus Y)$$

- HD indica la distanza tra due parole binarie, ovvero la quantità di `bit` differenti tra loro
- w indica il peso di una parola binaria, ovvero quanti `bit` sono 1

7.6.2 Esempio

Facciamo ora un esempio dove applichiamo il metodo McClusky. Supponiamo di voler minimizzare la funzione

$$F(A,B,C,D) = \underbrace{\sum m(4,5,6,8,9,10,13)}_{\text{ON - Set}} + \underbrace{\sum d(0,7,15)}_{\text{DC - Set}}$$

1. Riempire la colonna 1 con i MIN termini di ON-set e DC-set raggruppati in base al loro peso
2. Confrontare gli elementi dei gruppi adiacenti alla ricerca di termini a distanza 1. Eliminare la variabile e mettere nella colonna successiva. *es.* 0000 e 0100 $\Rightarrow 0-00$.

   Segnare con ✓i termini che ne generano uno più semplice. Se non generano altri termini segnare con *IP* (**Implicante Primo**).

   Nota: ripetere questo step finchè non si possono avere più combinazioni.
3. Creare la tabella di copertura che permette di trovare la collezione minima di **IP** che copre l'ON-set.

7.7 Multiplexer