Tecnica Digitale

SUPSI Dipartimento Tecnologie Innovative

Gianni Grasso

30 settembre 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} 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:

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

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:

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

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:

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

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

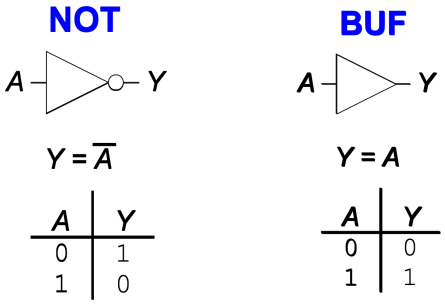

Figura 3: Porte logiche con un solo input

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

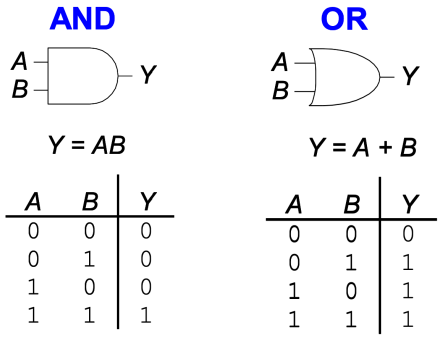

Figura 4: Porte logiche con due input

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.

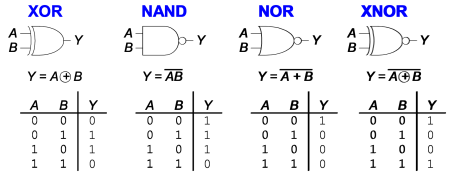

Figura 5: Altre porte logiche a due input

- **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 Porte logiche

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

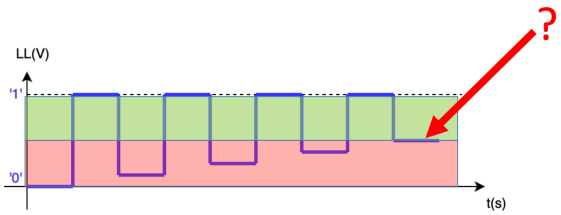

Figura 6: Livelli logici reali

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.

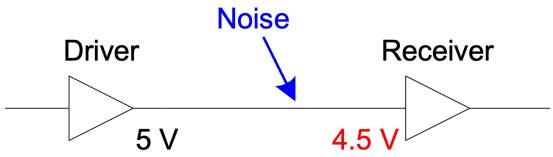

Figura 7: Esempio di rumore

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).

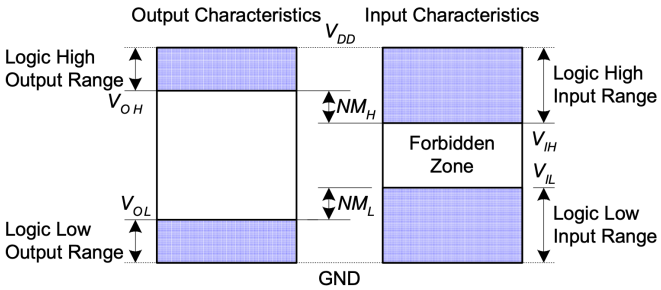

Figura 8: Schema noise margins

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

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