Prestazioni dei Computer

Tempo di risposta e produttività

- **Tempo di risposta**: latenza, quanto ci vuole per eseguire una operazione.
- **Produttività**: Lavoro totale svolto per unità di tempo.
  In che modo sono influenzati?
- Sostituire il processore con una versione più veloce?
- Aggiungere più processori?

Tempo di risposta

Cosa determina le prestazioni di un programma?

- **Algoritmi**
  - Determina il numero di **operazioni** eseguite.
- **Linguaggio di programmazione**, **compilatore**, **architettura**
  - Determina il numero di **istruzioni macchina** eseguite per operazione.
- **Processore e memoria di sistema**
  - Determinano quanto *veloce* le istruzioni vengono eseguite.
- **Sistemi di I/O** (OS incluso)
  - Determina quanto velocemente le operazioni di I/O sono eseguite.

Misuriamo il tempo di risposta

Tempo trascorso:

- Tempo di risposta totale
  - Elaborazione, I/O, overhead del SO, tempo di inattività.
- **Tempo di CPU** (Clock):
  - Tempo speso elaborando un dato lavoro:
    - Sconti sul tempo di I/O, quote di altri lavori
  - Comprende il tempo CPU dell'utente + tempo CPU del sistema
  - I programmi sono influenzati dalle prestazioni della CPU e del sistema

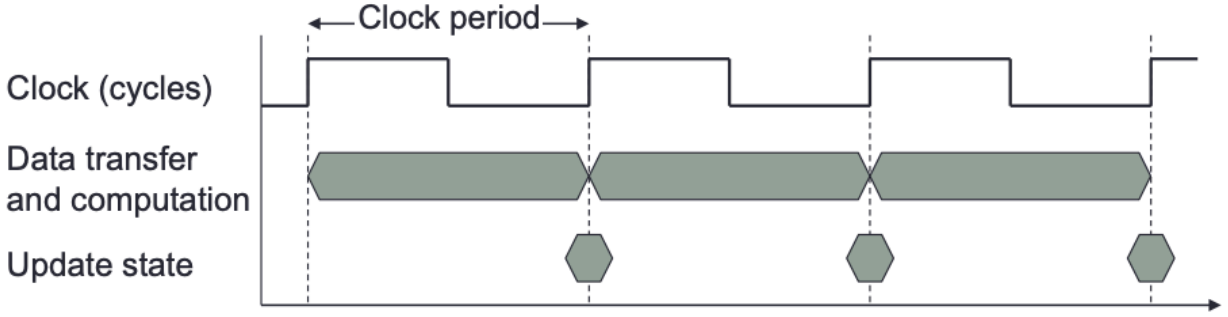
CPU Clock

È la frequenza operativa di un processore, cioè la velocità con cui la CPU può eseguire le istruzioni.

$$T = \frac{1}{f}$$

Dove:

- T è il **periodo di clock** (misurato in secondi)
- f è la **frequenza del clock** (misurata in hertz, Hz)



Esempio:
Se la frequenza del clock della CPU è **2 GHz** (2 miliardi di cicli al secondo), il periodo di clock sarà:

$$T = \frac{1}{2 \times 10^9} = 0,5 \times 10^{-9} = 0,5 \text{ nanosecondi}(ns)$$

Ogni ciclo di clock dura **0,5 nanosecondi**.

$$\text{Clock Rate} = \frac{\text{Clock Cycles}}{\text{CPU Time}}$$

GHz:

$$1GHz = 10^9 Hz$$

ISA: Instruction Set Architecture.
Il **conteggio delle istruzioni** per un programma sono determinati dal programma, ISA e compilatore.
Il numero medio di **cicli per istruzione** (CPI), sono determinati dall'hardware della CPU. Se istruzioni differenti hanno CPI differenti, il CPI medio è influenzato dal mix di istruzioni.

Esempio:

- Computer A: $\text{Cycle Time} = 250ps, CPI = 2.0$
- Computer B: $\text{Cycle Time} = 500ps, CPI = 1.2$
- ISA identico
  Quale è più veloce? Di quanto?

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= 1 \times 2.0 \times 250ps = 1 \times 500ps \\ \text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= 1 \times 1.2 \times 500ps = 1 \times 600ps\end{aligned}$$

A è più veloce, di quanto?

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{1 \times 600ps}{1 \times 500ps} = 1.2$$

A è 1.2 volte più veloce di B.

Se classi di istruzione diverse richiedono un numero diverso di cicli:

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

Media pesata dei CPI:

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles = 2×1 + 1×2 + 2×3 = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles = 4×1 + 1×2 + 1×3 = 9
  - Avg. CPI = 9/6 = 1.5

IPC: Istruzioni per ciclo:

$$\text{IPC} = \frac{\text{Instruction Count}}{\text{Clock Cycle}} = \frac{1}{\text{CPI}}$$

RIASSUMENDO:

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$

Power Trends

Nella tecnologia **CMOS IC**:

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

Rappresentazione dell'informazione

Rappresentazione binaria

Tutta l'informazione interna ad un computer è codificata con sequenze di due soli simboli: 0 e 1.
L'unità elementare di informazione si chiama *bit* (da *binary digit*).
Byte: 8 bit.
Word: sequenza di 32, 64, ... bits (4, 8, ... *Bytes*)

Sistema decimale posizionale (1)

La rappresentazione di un numero intero in base 10 è una sequenza di cifre scelte fra l'insieme $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
Il valore di una rappresentazione è dato da:
Parte intera:

$$a_N \cdot 10^N + a_{N-1} \cdot 10^{N-1} + \ldots + a_1 \cdot 10^1 + a_0 \cdot 10^0 +$$

Parte frazionaria:

$$a_{-1} \cdot 10^{-1} + a_{-2} \cdot 10^{-2} + a_{-3} \cdot 10^{-3} + \ldots$$

Sistema decimale posizionale (2)

$$\begin{aligned} 253 &= 2 \times 100 + 5 \times 10 + 3 \times 1 \\ &= 2 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 \end{aligned}$$

Notazione in base 2 (1)

La rappresentazione di un numero intero in base 2 è una sequenza di cifre scelte fra $\{0, 1\}$:
Parte intera:

$$a_N \cdot 2^N + a_{N-1} \cdot 2^{N-1} + \ldots + a_1 \cdot 2^1 + a_0 \cdot 2^0 +$$

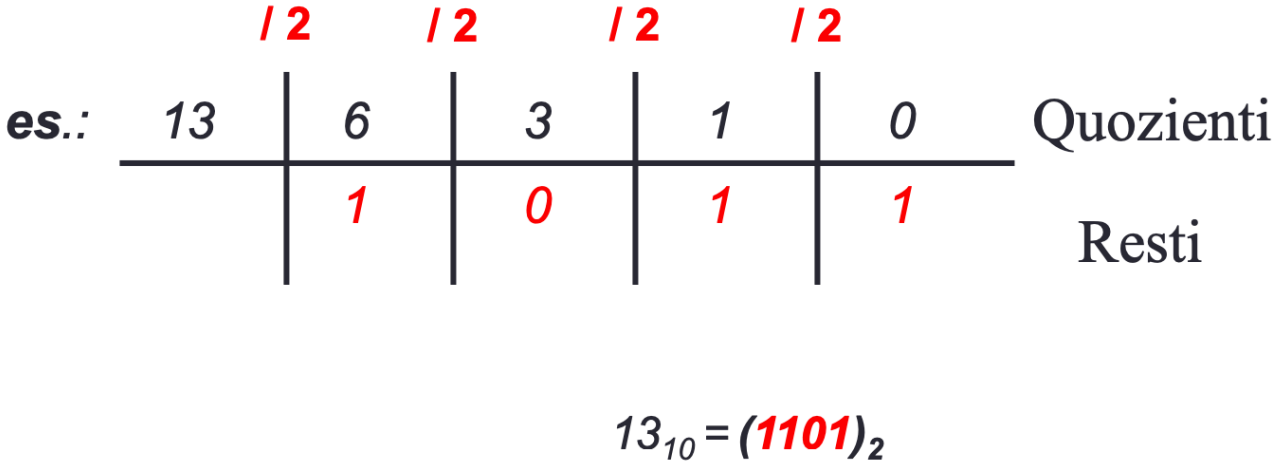
Parte frazionaria:

$$a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + a_{-3} \cdot 2^{-3} + \ldots$$

Notazione in base 2 (2)

$$110 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 4 + 2 + 0 = 6$$

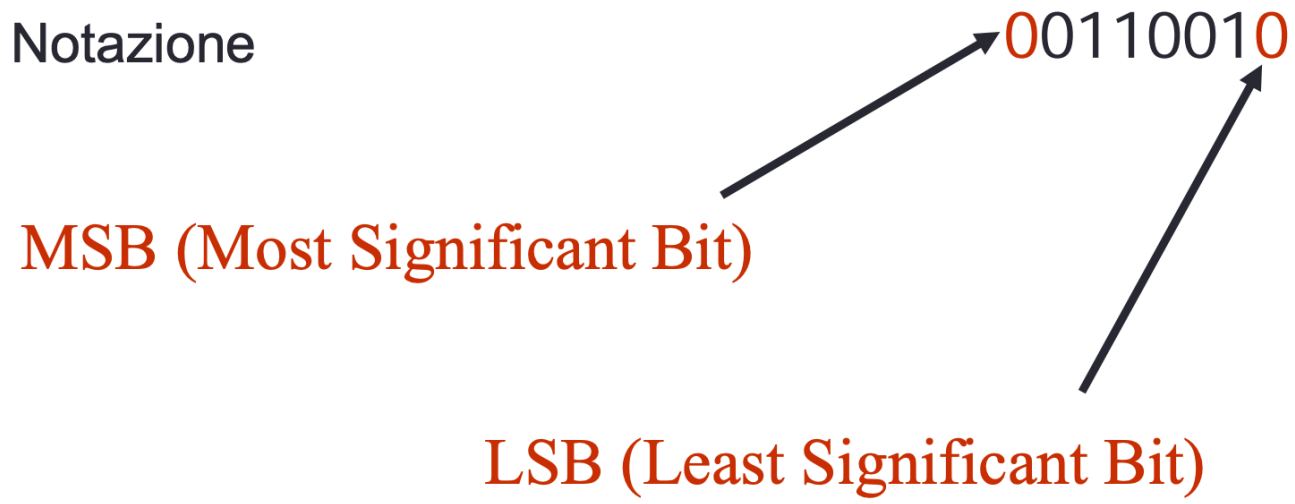
Conversione di interi: Base 10 -> Base 2



$13_{10} = (1101)_2$

La rappresentazione dei numeri all'interno di un computer

Gli interi positivi si rappresentano usando 4 o 8 byte.



- Se uso 4 byte (32 bit) posso rappresentare tutti i numeri da 0 a $2^{32}-1$.

Numeri relativi

- **Modulo e segno**
  - Prima cifra 0 -> $+$
  - Prima cifra 1 -> $-$

$$+2 \leftrightarrow 010 \ \text{ e } \ -2 \leftrightarrow 110$$

Complemento a due

Esempio con 4 bit:
Partiamo da $+5 = 0101 (4+1)$

1. si invertono gli 1 con gli 0: 1010
2. si aggiunge 1: $1011 = -5$

$$\begin{aligned} &1010 + 1 = 1011 = -5 \\ &-1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= -8 + 0 + 2 + 1 = -5 \end{aligned}$$

Conversione in base 8 da base 2

$$111000110101_2$$

In base 8:

$$\underbrace{111}\ \underbrace{000}\ \underbrace{110}\ \underbrace{101} = 7065_8$$

In base 16:

$$\underbrace{1110}\ \underbrace{0011}\ \underbrace{0101} = E35_{16}$$

BCD (Binary-Coded Decimal)

Si codificano in binario (4 bit) le singole cifre decimali.
254:

$$\underbrace{0010}_{2}\ \underbrace{0101}_{5}\ \underbrace{0100}_{4}$$

Rappresentazione in virgola fissa

Riservo X bit per la parte frazionaria.
Es:

$$\begin{aligned} 101.01 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 4 + 0 + 1 + 0 + 0.25 = 5,25 \end{aligned}$$

Problemi:

- *Overflow*
  - quando si sale al di sopra del massimo numero rappresenrtabile.
- *Underflow*
  - quando si scende al di sotto del minimo numero rappresentabile.

Rappresentazione in virgola mobile

Quando lavoro con *numeri molto piccoli* uso tutti i bit disponibili per rappresentare le cifre dopo la virgola.
Quando lavoro con *numeri molto grandi* le uso tutte per rappresentare le cifre in posizioni elevate.

- Ogni numero N è rappresentato da una coppia (*mantissa* M, *esponente* E) con il seguente significato

  $$N = M \times 2^E$$

  Esempio in base 10, con 3 cifre per la mantissa e 2 cifre per l'esponente:

  $$349\ 000\ 000\ 000 = 3.49 \times 10^{11}$$

  con la coppia $(3.49, 11)$ perché $M = 3,49$ e $E = 11$.

  $$0.000\ 000\ 002 = 2.0 \times 10^{-9}$$

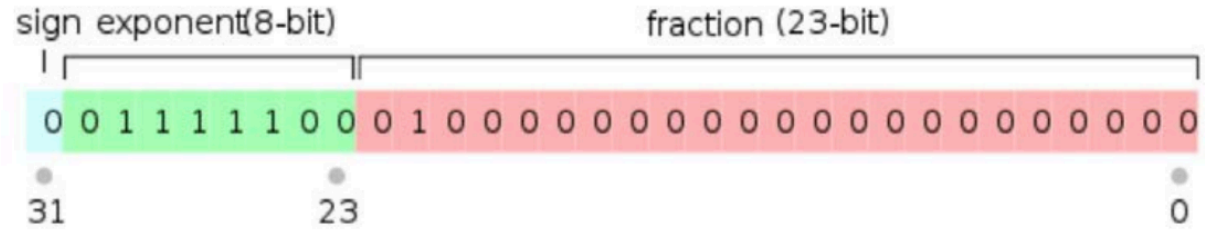
  con la coppia $(2.0, -9)$ perché $M = 2.0$ e $E = -9$.

Standard IEEE 754

Si specificano 3 parametri:

- P: precisione o numero di bit che compongono la mantissa.
- E_{max}: esponente massimo.
- E_{min}: esponente minimo.
  Per la **precisione singola** (32 bit):

- $P = 23, E_{max} = 127$ e $E_{min} = -126$
- 1 bit di *segno*; 8 bit *esponente*



- La mantissa viene normalizzata scegliendo l'esponente in modo che sia sempre nella forma $1, xxx\ldots$
- L'esponente è *polarizzato*, ovvero ci si somma E_{max}
  - costante di polarizzazione o **bias**
  Esempio:

$$0.15625_{(10)} = \frac{1}{8} + \frac{1}{32} = 2^{-3} + 2^{-5} = 0.00101_{(2)}$$

  Normalizzazione della manitssa:

$$0.00101_{(2)} = 1.01_{(2)} \times 2^{-3}$$

- *Parte frazionaria della mantissa*: $.01_{(2)}$
- *Esponente*: -3
- *Esponente polarizzato*: $-3 + 127 = 124$

Per la **precisione doppia** (64 bit)

- $P = 52, E_{max} = 1023, E_{min} = -1022$
- 1 bit *segno*; 11 bit *esponente*
- *Parte frazionaria della mantissa*: $.01_{(2)}$
- *Esponente*: -3
- *Esponente polarizzato*: $-3 + 1023 = 1020$
  ...
  ...
  ...

Introduzione alle Reti Logiche

Reti logiche

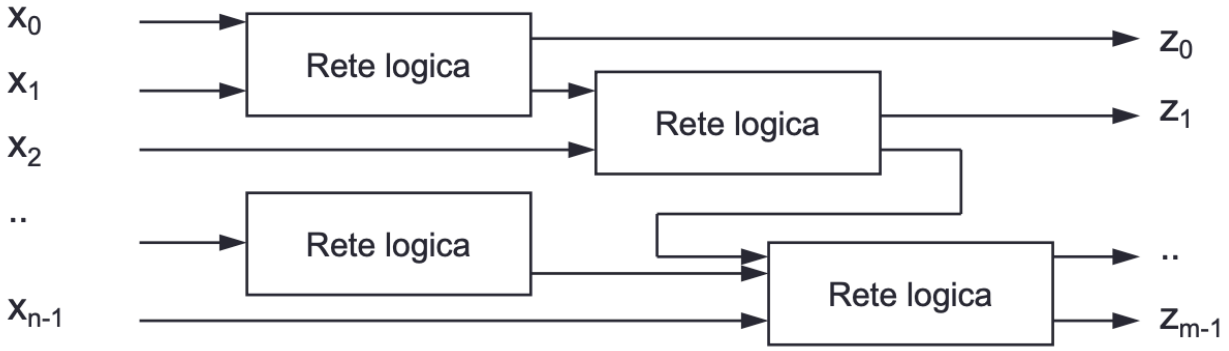
Sistema digitale avente n segnali binari di ingresso ed m segnali binari di uscita.
I segnali sono rigorosamente binari $(0/1)$.

I segnali sono grandezze funzioni del tempo

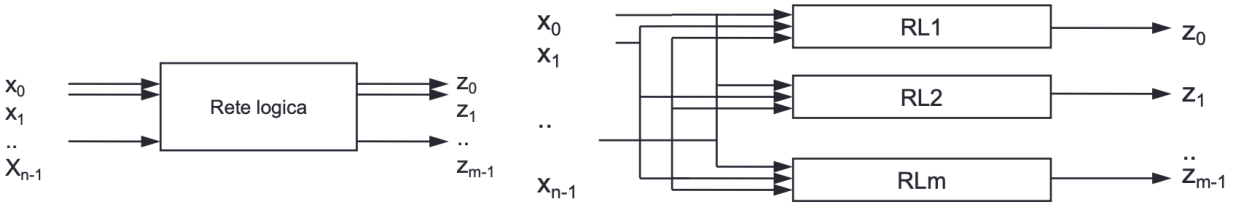
$$X = \{x_{n-1}(t), \ldots, x_0(t)\}$$

Proprietà delle reti logiche

- **Interconnessione**: l'interconnessione di più reti logiche, aventi per ingresso segnali esterni o uscite di altre reti logiche e per uscite segnali di uscita esterne o ingressi di altre reti logiche, è ancora una rete logica.



- **Decomposizione**: una rete logica complessa può essere decomposta in reti logiche più semplic.
- **Decomposizione in parallelo**: una rete logica a m uscite può essere decomposta in m reti logiche ad 1 uscita, aventi ingressi condivisi.

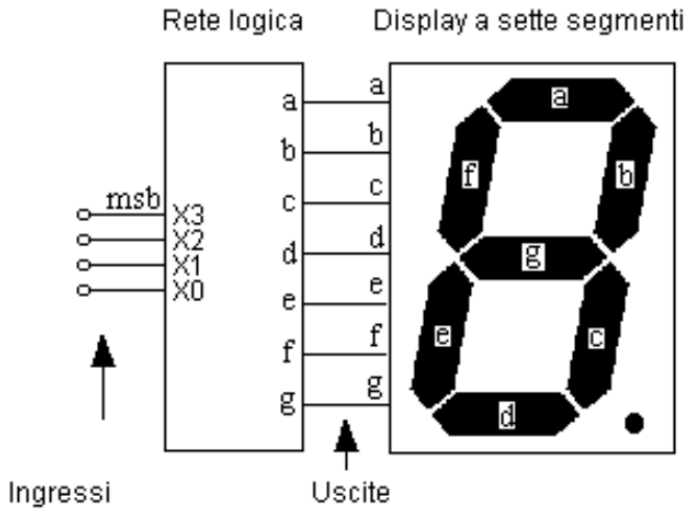


Reti combinatorie

- ogni segnale di uscita dipende solo dai valori degli ingressi in quell'istante.
- senza memoria, *non ha stato*, non ricorda gli ingressi precedenti, *transitori* a parte, basta conoscere gli ingressi in un istante per sapere esattamente quali saranno tutte le uscite nel medesimo istante.
  Esempio:
  Conversione di valori BCD su display a sette segmenti:
- Progettare una rete logica che permette la visualizzazione su un display a sette segmenti di un valore in codice BCD.
- **Codifica BCD**: impiego di 4 cifre binarie per la rappresentazione di un numero decimale da 0 a 9.

15	decimale
1111	binario
$0001\ 0101$	BCD

L'uscita $Z = \{a, b, \dots, g\}$ dipende in ogni istante dalla configurazione degli ingressi $\{x_3, x_2, x_1, x_0\}$.



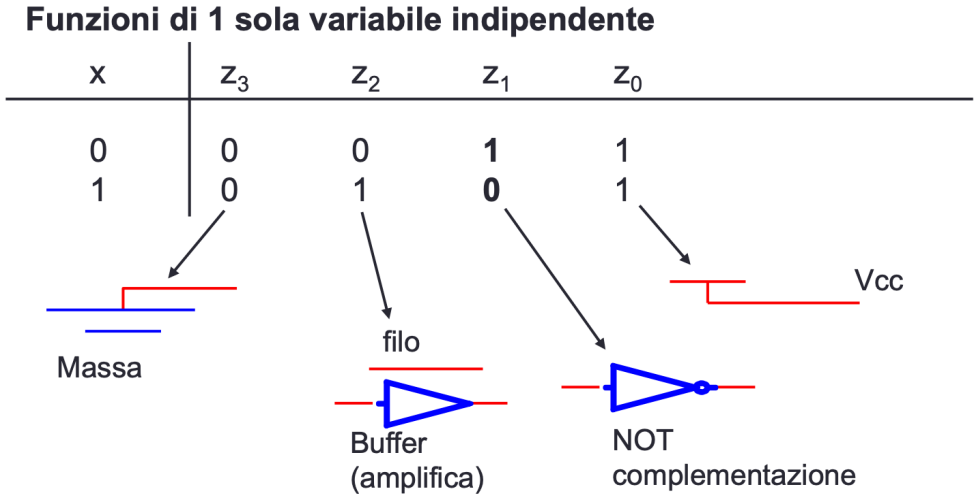
Descrizione reti combinatorie

- **Tabelle di verità**: associa le possibili combinazioni degli ingressi alle corrispondenti configurazioni delle uscite e indica il comportamento della rete logica.
  - Se la rete combinatoria ha n ingressi e m uscite, allora la tabella di verità ha $(n + m)$ colonne e 2^n righe.
  - **COMPLETAMENTE SPECIFICATE**: se ogni valore della tabella assume il valore logico di vero o falso.
  - **NON COMPLETAMENTE SPECIFICATE**: se contengono condizioni di indifferenza. Si verifica in due casi:
    - se alcune configurazioni di ingressi sono vietate.
    - se le uscite sono indifferenti per alcune configurazioni di ingresso.

Funzioni combinatorie e gate elementari

Le reti logiche combinatorie sintetizzano funzioni combinatorie.
Per ogni n, è finito il numero di funzioni combinatorie di n variabili di ingresso. Alcune funzioni

combinatorie elementari hanno una rappresentazione logica e grafica (*gate*).



Funzioni di 2 variabili indipendenti

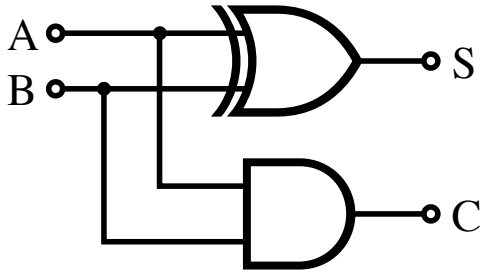
- **AND**
- **OR**
- **EXOR**
- **NOR**
- **EXNOR**
- **NAND**

  Quante sono le possibili funzioni binarie di n variabili?

$$\mathrm{N.conf} = 2^{2n}$$

  Esempio di rete logica con gate:

- **HALF ADDER**, un sommatore senza riporto in ingresso.

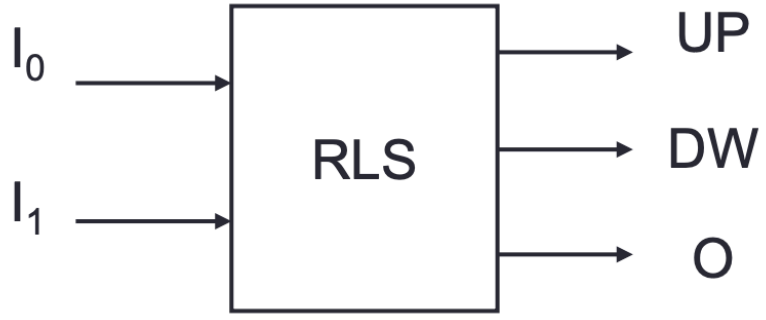


Reti sequenziali

- ogni segnale di uscita dipende dai valori degli ingressi in quell'istante E dai valori che gli ingressi hanno assunto negli istanti precedenti.
- rete con memoria, *ha stato*, è una rete in cui l'uscita cambia in funzione del cambiamento dell'ingresso e della specifica configurazione interna in quell'istante (*STATO*). Lo stato riassume la sequenza degli ingressi precedenti.
- Per sapere l'uscita in un certo istante ho due possibilità:
  - Mi ricordo *TUTTI* gli ingressi che si sono presentati alla rete dalla sua accensione.
  - Memorizzo uno *STATO* del sistema, che riassume in qualche modo tutti gli ingressi precedenti al fine di valutare il valore delle uscite.

Esempio:
Progettare la rete logica di gestione di un ascensore:

- La rete ha tre uscite *UP*, *DW* e *O*. *UP*, *DW* indicano le direzioni su e giù mentre *O* vale 1 se la porta deve essere aperta e 0 altrimenti. La rete ha come ingresso due segnali che indicano il piano $\{0, 1, 2, 3\}$ corrispondente al tasto premuto. Per calcolare l'uscita è necessario conoscere il piano corrente che indica lo stato interno.



Algebra di Boole

L'algebra di Boole è un sistema matematico che descrive funzioni di variabili binarie: è composto da:

- un insieme di simboli $B = \{0, 1\}$
- un insieme di operazioni $O = \{+, \cdot, ' \}$
  - $+$ somma logica (*OR*)
  - $\cdot$ prodotto logico (*AND*)
  - $'$ complementazione (*NOT*)
- un insieme P di postulati (assiomi).
  Proprietà di chiusura: per ogni $a, b \in B$:

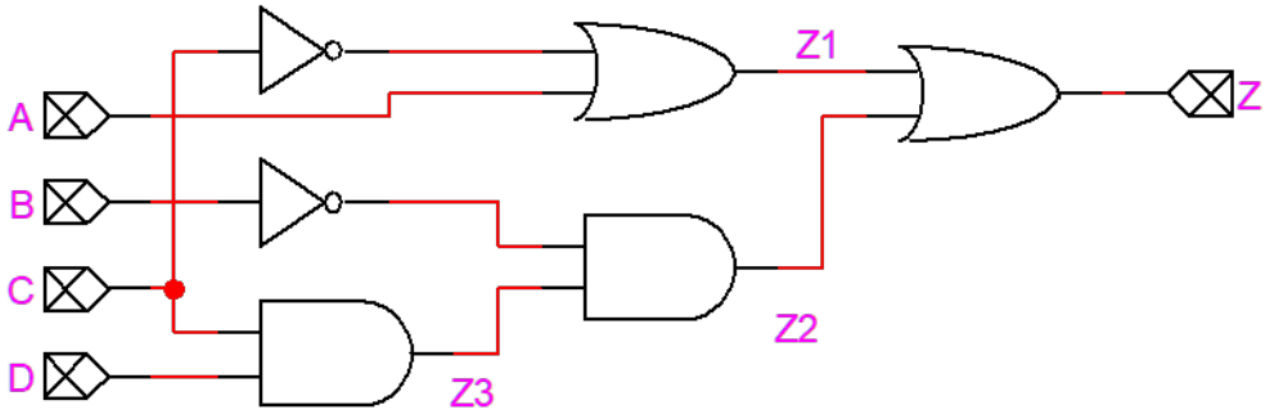
$$a + b \in B$$
$$a \cdot b \in B$$

Costanti dell'algebra: i simboli 0 e 1.
Variabile: un qualsiasi simbolo che può essere sostituito da una delle due costanti.
Un **espressione** secondo l'algebra di Boole è una stringa di elementi di B che soddisfa una delle seguenti regole:

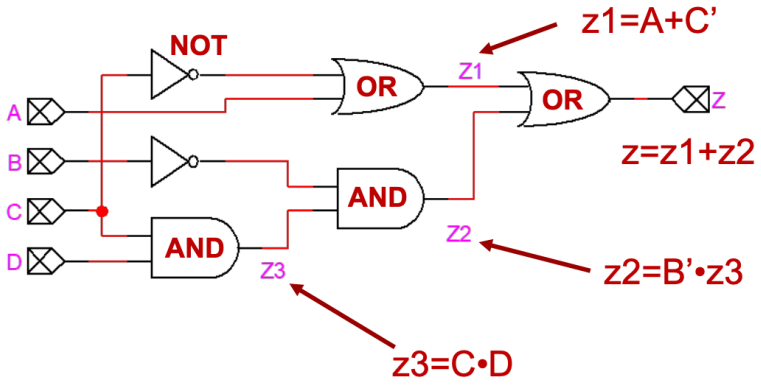
- una costante è un'espressione;
- una variabile è un'espressione;
- se X è un'espressione allora il complemento di X è un'espressione;
- se X, Y sono espressioni allora la somma logica di X e Y è un'espressione;
- se X, Y sono espressioni allora il prodotto logico di X e Y è un'espressione.
  Ogni *espressione* di n variabili descrive una funzione completamente specificata che può essere valutata attribuendo ad ogni variabile un valore assegnato.

Analisi di uno schema logico

Dallo schema logico tramite le espressioni è possibile ricavare il comportamento di una rete logica.
Esercizio: Eseguire l'analisi del seguente schema



Esercizio: Eseguire l'analisi del seguente schema



z=z1+z2
z1=A+C'
z2=B'•z3
z3= C•D

z=z1+z2
z1=A+C'
z2=B'•C•D

z=A+C'+B'•C•D

Teoremi dell'algebra di Boole

Principio di dualità:

- ogni espressione algebrica presenta una forma **duale** ottenuta scambiando l'operatore *OR* con *AND*, la costante 0 con la costante 1 e mantenendo i letterali invariati.
- ogni proprietà vera per un'espressione è vera anche per la sua **duale**.
- il principio di dualità è indispensabile per trattare segnali attivi alti e segnali attivi bassi.
  - Logica negativa
  - Logica positiva

  Teorema di Identità:

$$(\mathrm{T1})\ X + 0 = X \qquad (\mathrm{T1'})\ X \cdot 1 = X$$

Teorema di Elementi nulli:
Utili nella sintesi di reti logiche: gli elementi nulli permettono di "lasciar passare" un segnale di ingresso in determinate condizioni.

$$(\mathrm{T2})\ X + 1 = 1 \qquad (\mathrm{T2'})\ X \cdot 0 = 0$$

Idempotenza:

$$(\mathrm{T3})\ X + X = X \qquad (\mathrm{T3'})\ X \cdot X = X$$

Involuzione:

$$(\text{T4})\ (X')' = X$$

Complementarietà:

$$(\text{T5})\ X + X' = 1 \qquad (\text{T5'})\ X \cdot X' = 0$$

Proprietà commutativa:

$$(\text{T6})\ X + Y = Y + X \qquad (\text{T6'})\ X \cdot Y = Y \cdot X$$

Proprietò associativa:

$$(\text{T7})\ (X + Y) + Z = X + (Y + Z) = X + Y + Z$$

$$(\text{T7'})\ (X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z$$

Proprietà di assorbimento:

$$(\text{T8})\ X + X \cdot Y = X \qquad (\text{T8'})\ X \cdot (X + Y) = X$$

Proprietà distributiva:

$$(\text{T9})\ X \cdot Y + X \cdot Z = X \cdot (Y + Z \qquad (\text{T9'})\ (X + Y) \cdot (X + Z) = X + Y \cdot Z$$

Proprietà della combinazione:

$$(\text{T10})\ (X + Y) \cdot (X' + Y) = Y \qquad (\text{T10'})\ X \cdot Y + X' \cdot Y = Y$$

Proprietà del consenso:

$$(\text{T11})\ (X + Y) \cdot (X' + Z) \cdot (Y + Z) = (X + Y) \cdot (X' + Z)$$

$$(\text{T11'})\ X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$$

Teorema di De Morgan:

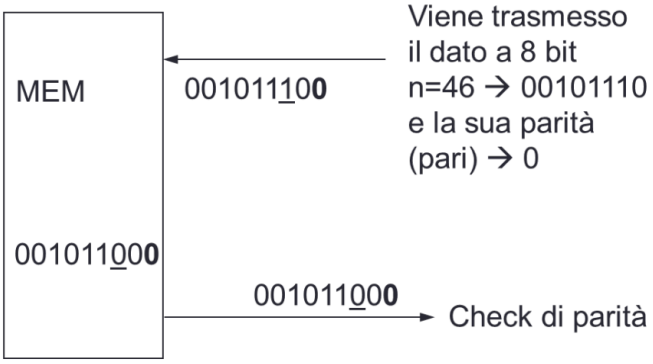
$$(\text{T12})\ (X + Y)' = (X' + Y') \qquad (\text{T12'})\ (X \cdot Y)' = (X' + Y')$$

generalizzabile per n variabili.

Parità

- I *codici rilevatori d'errori* sono codici in cui è possibile rilevare se sono stati commessi errori nella trasmissione.
- *Codici ridondanti*: in cui l'insieme dei simboli dell'alfabeto è minore dell'insieme di configurazioni rappresentabili col codice.
- *Codici con bit di parità*: alla codifica binaria si aggiunge un bit di parità.
- **Parità pari**: rende pari il numero di 1 presenti nella parola.
- **Parità dispari**: rende dispari il numero di 1 presenti nella parola.
  Esempio:

Vogliamo trasmettere il dato a *8 bit*: $n = 46 \rightarrow 00101110$, la sua parità *pari* $\rightarrow 0$.



- Supponiamo un errore di trasmissione durante la scrittura in memoria così che il numero memorizzato sia 001011000.
- Quando il dato viene riletto ed utilizzato viene fatto il check di parità e si verifica che quel numero non è ammissibile per la codifica binaria con parità pari *perché la somma dei bit a 1 è dispari*.

Sintesi di reti logiche combinatorie

La più semplice rappresentazione delle funzioni Booleane è attraverso la *forma canonica*, che può essere ottenuta da qualsiasi rete logica combinatoria.

Forma canonica SP

Somma di prodotti:
Teorema: una funzione di n variabili può essere rappresentata in un solo modo come somme di prodotti di n variabili (*mintermini*).

- **Minitermine**: prodotto logico di n letterali.
- Da ogni tabella si deriva la forma SP, prendendo in *OR* tutti i mintermini corrispondenti alle righe in cui l'uscita vale 1, in cui ogni variabile è in forma diretta se nella colonna appare il valore 1 ed in forma complementata se 0.
  Indipendentemente dalla complessità della rete logica da realizzare, la rete logica ottenuta dalla forma canonica è una rete molto veloce, in quanto composta da soli due livelli e mezzo (livello dei not).

r	a	b	S	R	
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	**1**	r'ab
1	0	0	1	0	
1	0	1	0	**1**	ra'b
1	1	0	0	**1**	rab'
1	1	1	1	**1**	rab

R= r'ab +ra'b + rab' +rab

Forma canonica PS

Prodotto di somme:
Teorema: una funzione di n variabili può essere rappresentata in un solo modo come prodotto di

somme di n variabili (*maxtermini*).

- **Maxtermine**: somma logica di n letterali.
- Da ogni tabella si deriva la forma PS, prendendo in *AND* tutti i maxtermini corrispondenti alle righe in cui l'uscita vale 0, in cui ogni variabile e' in forma diretta se nella colonna appare il valore 0 ed in forma complementata se 1.

r	a	b	S	R	
0	0	0	0	0	(r+a+b)
0	0	1	1	0	(r+a+b')
0	1	0	1	0	(r+a'+b)
0	1	1	0	**1**	
1	0	0	1	0	(r'+a+b)
1	0	1	0	**1**	
1	1	0	0	**1**	
1	1	1	1	**1**	

  Dalla tabella precedente:
- $R = (r + a + b)(r + a + b')(r + a' + b)(r' + a + b)$

Funzioni non completamente specificate

(o **funzioni booleane incompletamente specificate**)
Se le uscite hanno condizioni di *indifferenza*.

x1	x2	Output
0	0	1
0	1	-
1	0	0
1	1	-

Sintesi e minimizzazione

Sintesi di reti logiche combinatori:

1. descrizione mediante tabella della verità
2. sintesi della espressione canonica SP o PS
3. corrispondenza 1 a 1 con schema logico

Normalmente una rete logica si dice in forma minima per indicare il minor numero di livelli e, a parità di livelli, il minor numero di gate e di ingressi dei gate.
Tecniche di minimizzazione:

- minimizzazione con manipolazione algebrica
- minimizzazione manuale (k-mappe)
- minimizzazione con algoritmi CAD o software appositi (Logisim)
  Perché minimizzare?
  Perché le forme canoniche richiedono troppi gate, troppo consumo di area.

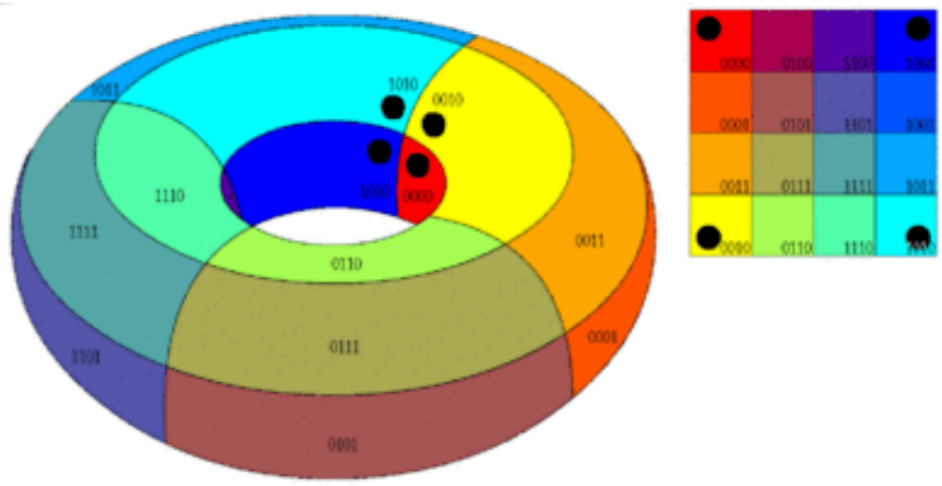
Mappe

Rappresentazione più compatta della tabella di verità, tramite matrici.
Le *righe* indicano tutte le possibili configurazioni di un sottoinsieme delle variabili di ingresso e le *colonne* tutte le configurazioni delle variabili.
Il valore nelle celle indica il valore dell'uscita nella configurazione corrispondente.

Mappe di Karnaugh

x_1x_0 \ x_3x_2	00	01	10	11
00	1	0	1	-
01	0	1	1	-
10	1	1	1	-
11	1	1	-	-

Le mappe vanno viste come «arrotolate» su se stesse. La prima riga risulta «adiacente» all'ultima riga. Stessa cosa per le colonne.



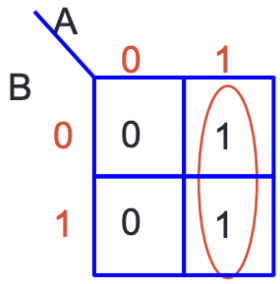
Minimizzazione con mappe di Karnaugh

Ogni casella della mappa è adiacente a caselle corrispondenti a *mintermini* (*maxtermini*) aventi distanza di Hamming unitaria dal *mintermine* (*maxtermine*) corrispondente alla casella considerata.

$$F = AB' + AB = A(B + B') = A \quad \text{proprietà distributiva}$$

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

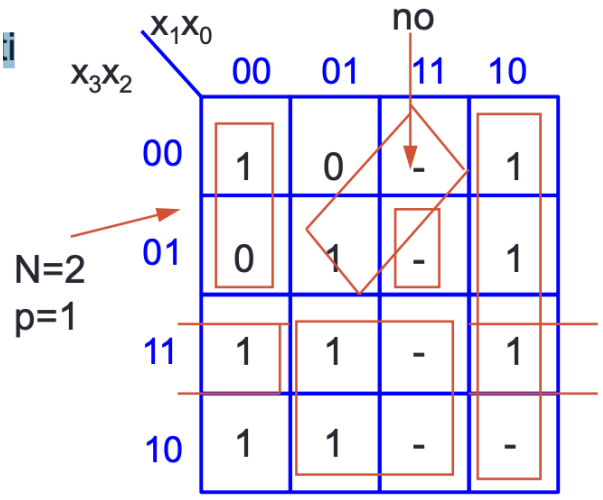
A	B	Output
1	1	1



Raggruppamenti rettangolari

Si dice **raggruppamento rettangolare** di ordine p una parte di una mappa a n variabili costituita da 2^p elementi (con $p \leq n$) tali da avere $n - p$ coordinate uguali fra loro, e di far assumere alle restanti p coordinate tutte le possibili configurazioni.
Ogni raggruppamento ha all'interno p celle adiacenti.

$$\text{RR ordine } 0 \rightarrow 1 \text{ cella}$$
$$\text{RR ordine } 1 \rightarrow 2 \text{ cella}$$
$$\text{RR ordine } 2 \rightarrow 4 \text{ cella}$$



Un Raggruppamento Rettangolare (*RR*) nel quale la funzione assume sempre valore 1 si dice **implicante** della funzione.
In modo duale, un *RR* nel quale la funzione assume sempre valore 0 si dice **implicato** della funzione.

- Si dice **copertura degli 1** un insieme di implicanti che contengono tutti gli 1 della funzione ed indifferenze.
- Un implicante non contenuto in nessun implicante di dimensioni maggiori prende il nome di **implicante primo**.
- **Implicanti essenziale**: un implicante primo contenente almeno un mintermine non contenuto in nessun altro implicante primo
- Ogni implicante essenziale deve essere contenuto nella *somma minima*. Vale il duale per gli implicati.

Una **copertura di 1** indica una forma SP.
Una **copertura di 0** indica una forma PS.

Complessità - Velocità

Per valutare la complessità di una rete logica in termini di complessità e velocità si utilizzano 3 indicatori:

- $N_{gate} =$ numero di gate,
- $N_{conn} =$ numero di connessioni,
- $N_{casc} =$ numero massimo di gate disposti in cascata
  Complessità: funzione *crescente* di N_{gate}, N_{conn}
  Velocità di elaborazione: funzione *decrescente* di N_{casc}

Forme normali e minime

Una espressione si dice:

- **normale SP** se è data dalla somma di prodotti non necessariamente di n variabili.
- **normale PS** se è data dal prodotto di somme non necessariamente di n variabili.
  Una espressione normale è equivalente alla forma canonica ma minimizzata.

Sintesi minima

(di costo minimo)

- minor numero di livelli
- minimo numero di gate
- minimo numero di connessioni
  l'espressione **minima normale** e non ridondante si ottiene con una *copertura* usando il **numero minimo di RR di ordine massimo**.
- Ordine massimo: minor numero di ingressi
- Minimo numero di RR: minimo numero di gate
- Forma normale irridondante: solo implicazioni essenziali
  Forma minima PS ed SP sono diverse.

Sintesi di reti combinatorie complesse

Esistono tecniche ed algoritmi per la sintesi automatica a più livelli:

- *Manipolazione algebrica*, ad esempio usando sistematicamente la proprietà distributiva.
- *Algoritmi di sintesi logica*.
- *CAD tools*.
- *Metodi empirici*.