UNIVERSITÀ DEGLI STUDI DI TRIESTE

Circuiti in logica combinatoria

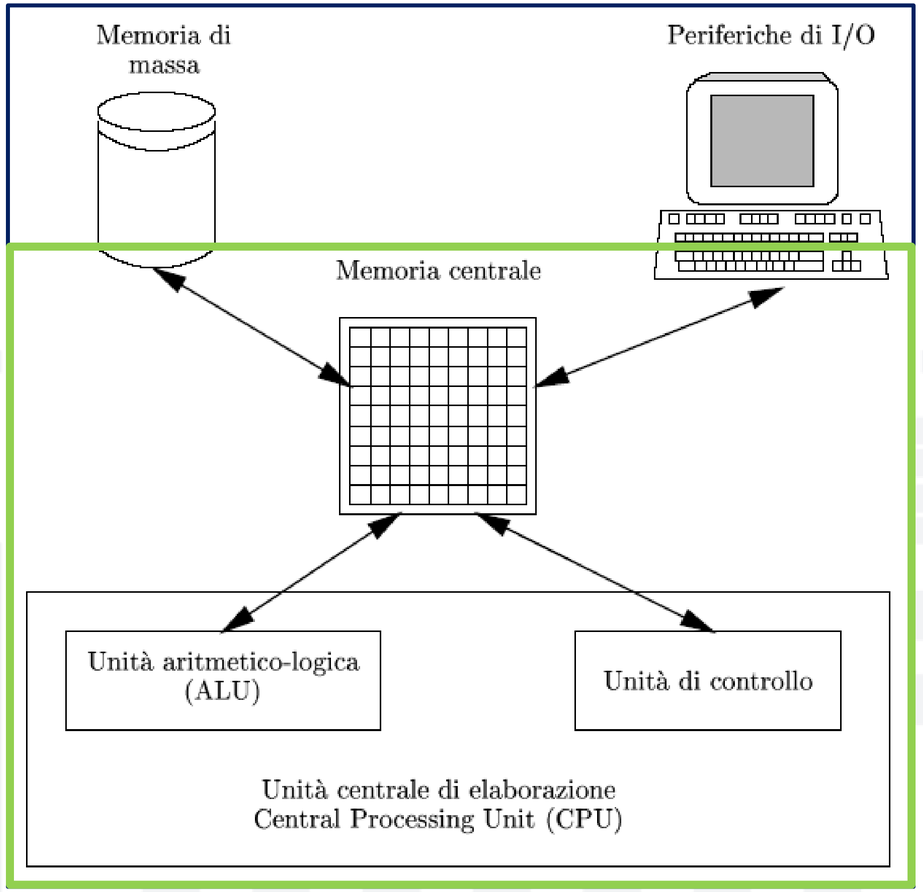
Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 17 marzo 2026

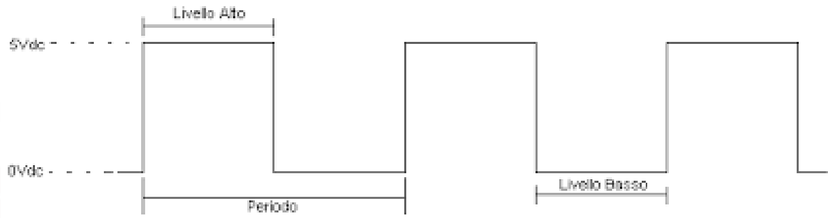
MODULO 1: Architettura degli elaboratori

John Von Neumann
Matematico ed informatico di origine ungherese che viveva e lavorava negli Stati Uniti negli anni ‘40



Architettura di un elaboratore di Von Neumann

Le varie parti dell’architettura devono sincronizzare le proprie attività: serve un **«CLOCK»**



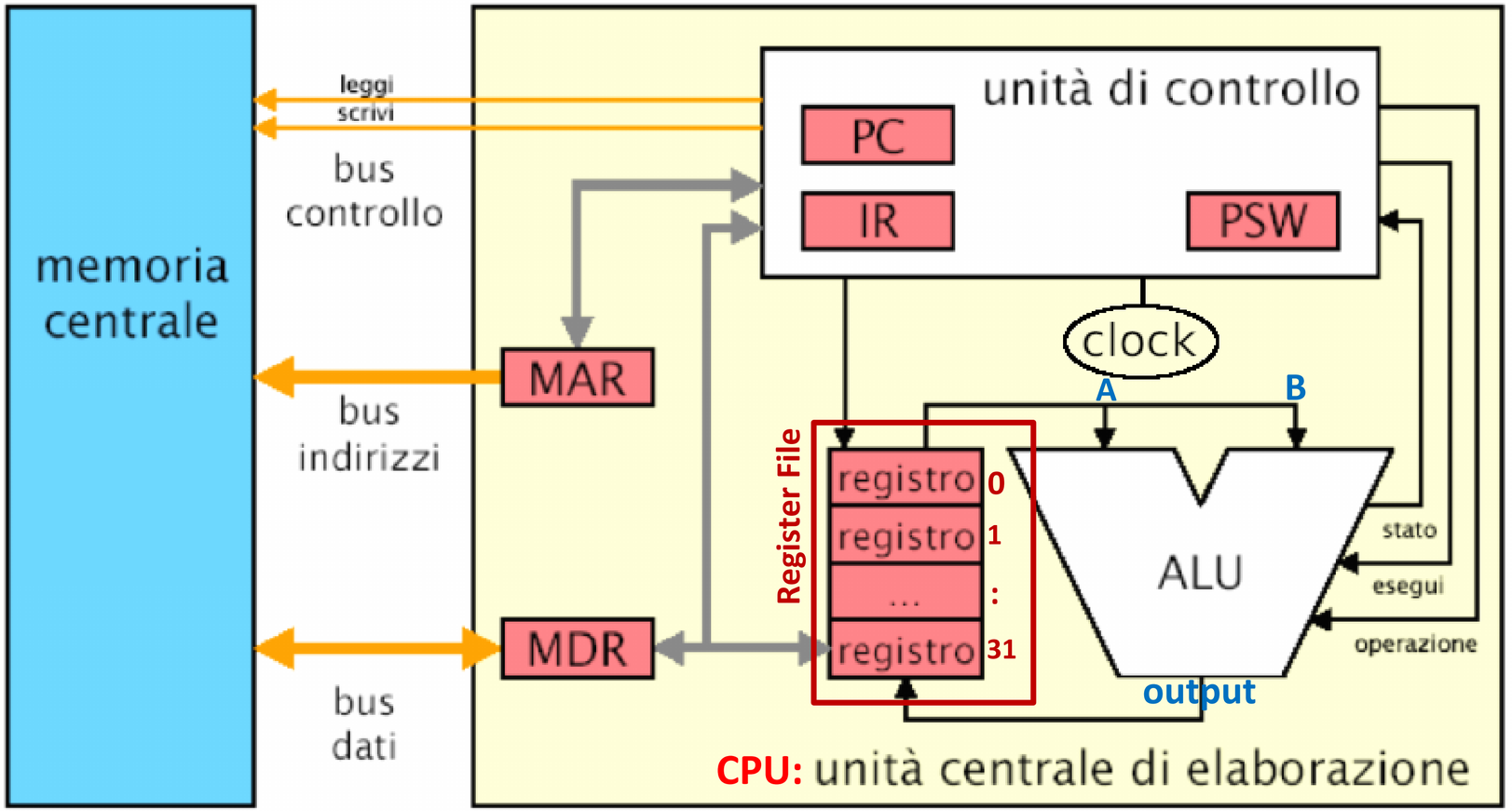
E’ un segnale periodico.

Più è veloce, più attività si possono fare nell’unità di tempo!

1GHz = 10^9 Hz

UNIVERSITÀ DEGLI STUDI DI TRIESTE

CONTROLLO+DATAPATH: ASTRAZIONE



IR: Instruction Register

→ È **l'istruzione** che la CPU sta *decodificando/eseguendo*.

PC: Program Counter

→ Contiene **l'indirizzo** della prossima istruzione.

MAR: Memory Address Register

→ Contiene l'indirizzo di memoria da cui leggere o in cui scrivere, va sul bus indirizzi.

MDR: Memory Data Register

→ Contiene il dato letto dalla memoria o da scrivere in memoria, va su bus dati.

PSW: Program Status Word

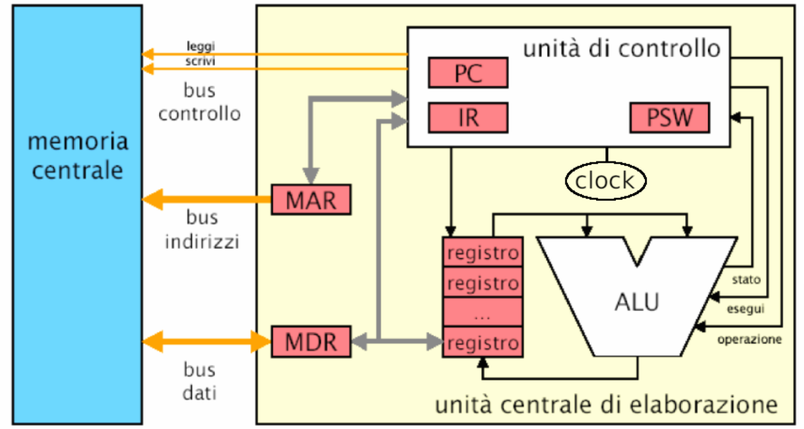
→ Serve per decisioni di controllo (branch, condizioni). Il registro di stato: contiene flag (zero, carry, overflow, segno, ecc.).

lw $10, 4($8)

carica in $10 la word che si trova all'indirizzo contenuto nel registro $8 + un offset di 4 byte

1. Fetch dell'istruzione

- Il **PC** (Program Counter) contiene l'indirizzo dell'istruzione da eseguire.
- Il valore del **PC** viene copiato nel **MAR** (Memory Address Register).
- Il MAR invia l'indirizzo alla memoria tramite il **bus indirizzi**.
- La **control unit** attiva sul **bus di controllo** il segnale **read**.
- La memoria legge l'istruzione e la manda sul **bus dati**.
- Il dato passa nel **MDR** (Memory Data Register).
- Dal MDR l'istruzione viene caricata nell'**IR** (Instruction Register).
- Il **PC viene incrementato** per puntare alla prossima istruzione.



lw $10, 4($8)

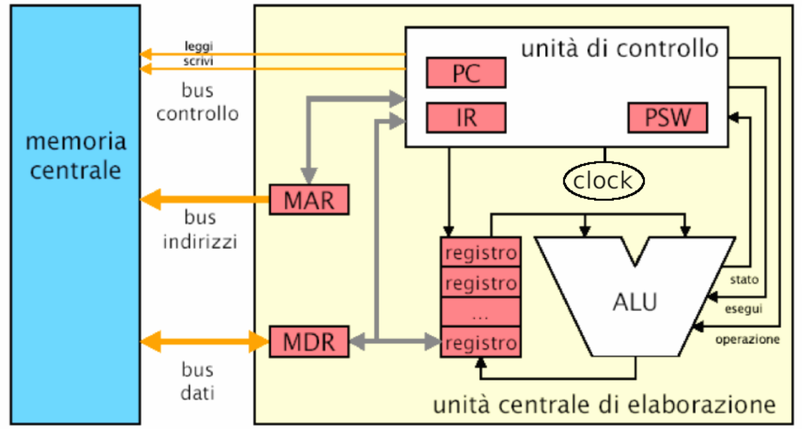
carica in $10 la word che si trova all'indirizzo contenuto nel registro $8 + un offset di 4 byte

2. Decode dell'istruzione

- L'istruzione contenuta nell'**IR** viene decodificata dalla **control unit**.

Da qui si identificano:

opcode = lw
rs = $8
rt = $10
immediate = 4



Operazioni:

- Il **Register File** usa **rs = $8** come indice
- Accede (legge) al contenuto del registro **$8**
- Il valore letto viene inviato alla **ALU** (input A)
- L'immediato 4 viene esteso di segno (32 bit) e inviato all'altra entrata della ALU (input B).

lw $10, 4($8)

carica in $10 la word che si trova all'indirizzo contenuto nel registro $8 + un offset di 4 byte

3. Execute

3.1 Calcolo dell'indirizzo effettivo

La **ALU** esegue la somma: `indirizzo = $8 + 4`

Il risultato viene inviato al **MAR,** quindi sul **bus indirizzi** verso la memoria.

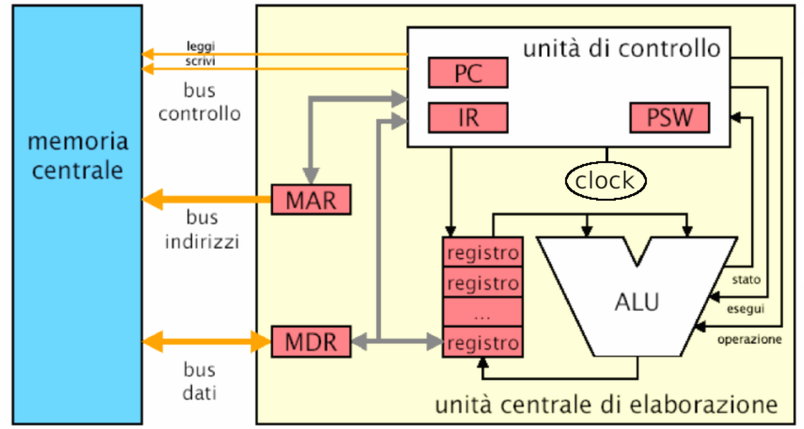
3.2 Accesso alla memoria

La **control unit** attiva il segnale **read** sul **bus di controllo**.

La memoria legge la word all'indirizzo calcolato.

Il dato viaggia sul **bus dati**.

Il dato entra nel **MDR**.



3.3 Scrittura nel registro destinazione

Il valore contenuto nel **MDR** viene inviato al Register File.

Il dato viene scritto nel registro **$10**.

-- Controllo + Datapath --

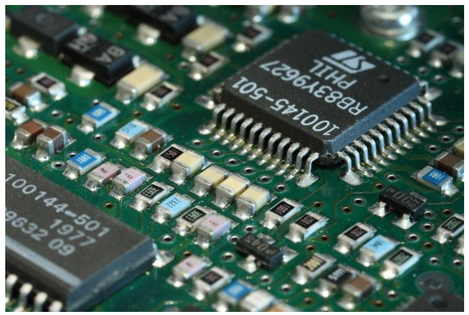
3 GRANDI DOMANDE

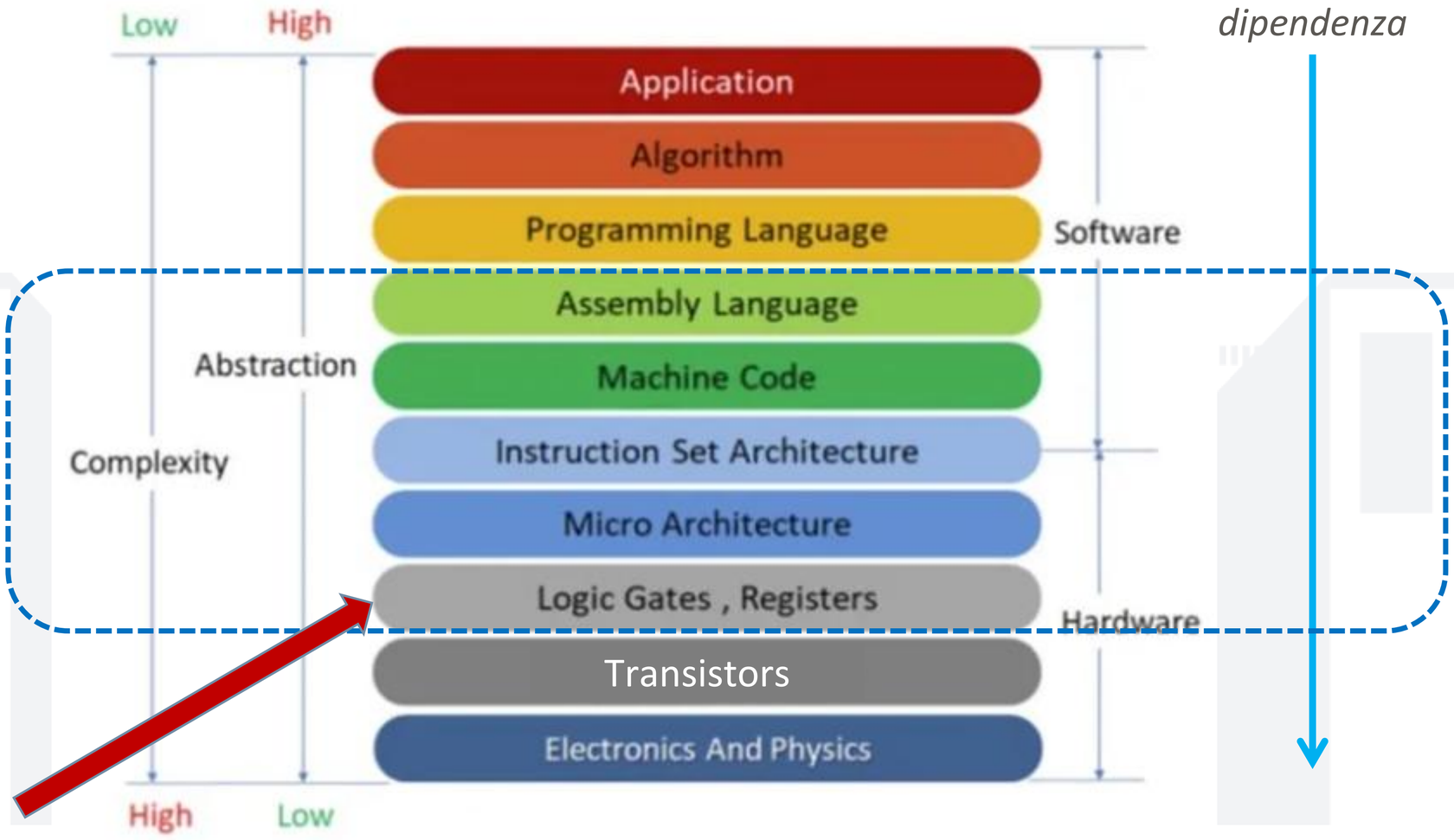
1.Come posso rappresentare l'informazione nel pc?

2.Come posso realizzare fisicamente tale rappresentazione?

3.Come posso costruire il pc a partire dagli elementi di base?

INSTRUCTION SET ARCHITECTURE



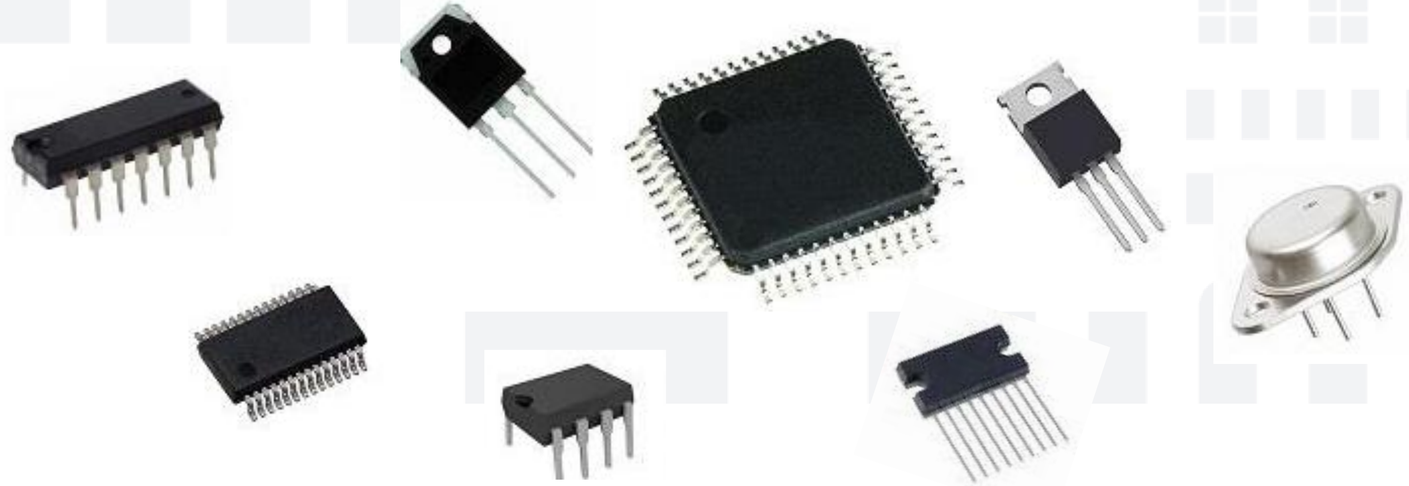


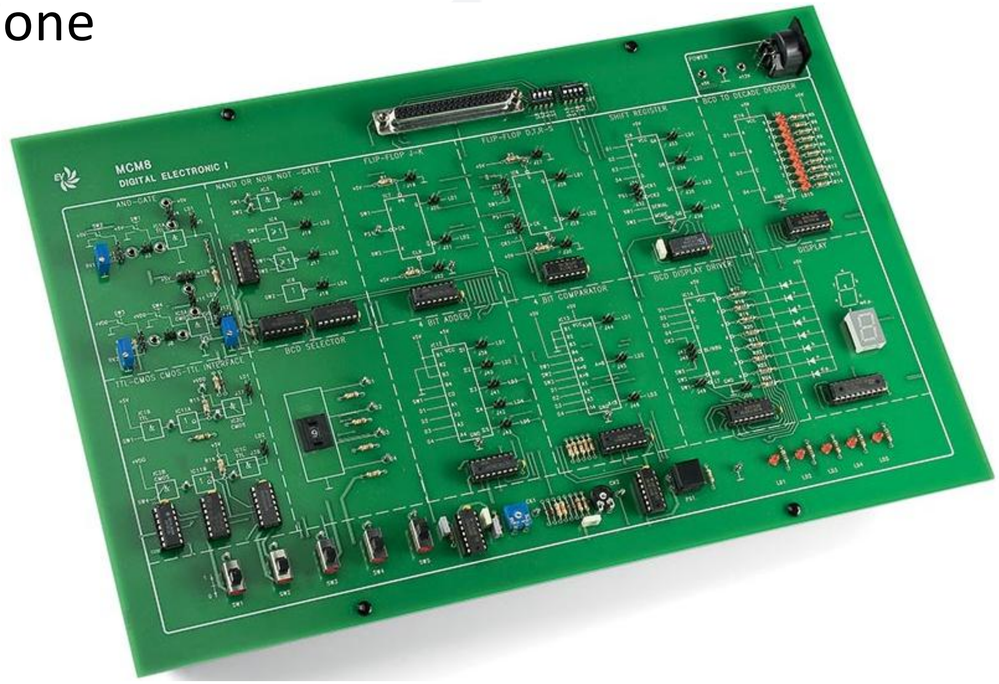
UNIVERSITÀ DEGLI STUDI DI TRIESTE

CIRCUITI LOGICI: INTRODUZIONE

I circuiti logici sono realizzati come circuiti integrati su chip di silicio (piastrina)

- **Porte (gate)** e fili depositati su chip di silicio, inseriti in un package e collegati all'esterno con un certo insieme di pin (piedini)
- I circuiti integrati si distinguono per grado di integrazione
- Da singole porte indipendenti a circuiti più complessi





UNIVERSITÀ DEGLI STUDI DI TRIESTE

CIRCUITI LOGICI: INTEGRAZIONE

- Integrazione
  - **SSI (Small Scale Integrated):** 1-10 porte
  - **MSI (Medium Scale Integrated):** 10-100 porte
  - **LSI (Large Scale Integrated):** 100-100.000 porte
  - **VLSI (Very Large Scale Integrated):** > 100.000 porte

- Con **SSI**, i circuiti integrati contenevano poche porte, direttamente collegate ai pin esterni

- Con tecnologia **MSI**, i circuiti integrati contenevano alcuni componenti base
  - circuiti comunemente usati nel progetto di un computer

- Con tecnologia **VLSI**, i circuiti integrati possono oggi contenere una CPU completa (o più)
  - microprocessore

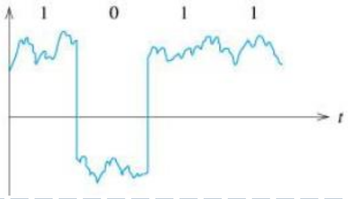
CIRCUITI LOGICI

Nell'elettronica digitale sia gli ingressi che le uscite possono assumere solo i valori di segnale alto (1 per convenzione) o basso (0 per convenzione).

In un circuito digitale i valori binari sono ottenuti tramite discretizzazione dei segnali:

Falso: segnali con voltaggio basso <= 1

Vero: segnali con voltaggio più alto >1



- Un circuito (o rete) combinatoria è quel circuito il cui lo stato delle uscite dipende solo dalla funzione logica applicata allo stato istantaneo (cioè in un determinato istante di tempo) delle sue entrate.

- Un circuito (o rete) sequenziale è quel circuito il cui lo stato delle uscite non dipende solo dalla funzione logica applicata ai suoi ingressi, ma anche sulla base di valori pregressi collocati in memoria.

PORTE LOGICHE

- Le porte logiche sono i componenti elettronici che permettono di svolgere le operazioni logiche primitive oltre che a quelle direttamente derivate.
  - Le porte logiche che realizzano le operazioni principali dell'**algebra booleana**
  - Una porta logica è un circuito elettronico che, dati dei segnali 0 e 1 in input, produce un segnale in output ottenuto effettuando una operazione booleana sugli ingressi
- **Le porte logiche hanno n input e generalmente 1 output**
  - A ogni combinazione di valori in ingresso corrisponde una e solo una combinazione in di valori uscita
  - *Dati gli input, l'output corrispondente appare quasi istantaneamente*

- Porte logiche fondamentali: AND, OR, NOT
- Porte logiche derivate: NAND, NOR, XOR

PORTA LOGICA **AND**

La porta logica AND svolge l'operazione logica di AND tra due bit, detta anche prodotto logico.

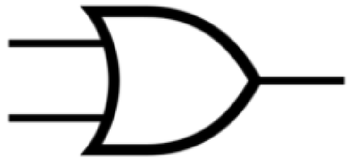
Considerando due entrate A e B, l'uscita A · B è data da:

A	B	A•B
0	0	0
0	1	0
1	0	0
1	**1**	**1**

PORTA LOGICA **OR**

La porta logica OR svolge l'operazione logica di OR tra due bit, detta anche somma logica.

Considerando due entrate A e B, l'uscita A + B è data da:

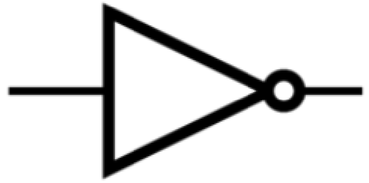


A	B	A+B
0	0	0
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

PORTA LOGICA **NOT**

La porta logica NOT svolge l'operazione logica di NOT su un bit, detta anche negazione logica.

Considerando un’entrata A, l'uscita Ā è data da:



A	Ā
0	**1**
1	0

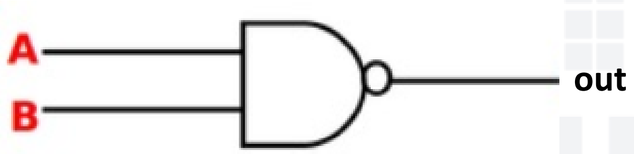
PORTE LOGICHE DERIVATE

Oltre alle porte logiche fondamentali (AND, OR, NOT) esistono altre porte che sono realizzate combinando le porte fondamentali, il cui principale scopo è la semplificazione dei circuiti (realizzando operazioni composte in un unico componente).

- Le porte logiche derivate sono **NAND, NOR e XOR.**

PORTA LOGICA **NAND**

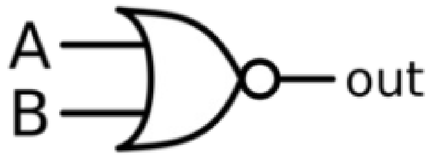
La porta logica NAND svolge l'operazione logica di NOT sul bit risultante dall'operazione AND sui bit in ingresso.



A	B	A•B	A NAND B
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

PORTA LOGICA **NOR**

La porta logica NOR svolge l'operazione logica di NOT sul bit risultante dall’operazione AND sui bit in ingresso.



A	B	A+B	A NOR B
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

PORTA LOGICA **XOR**

La porta logica XOR opera come disgiunzione esclusiva tra due input.

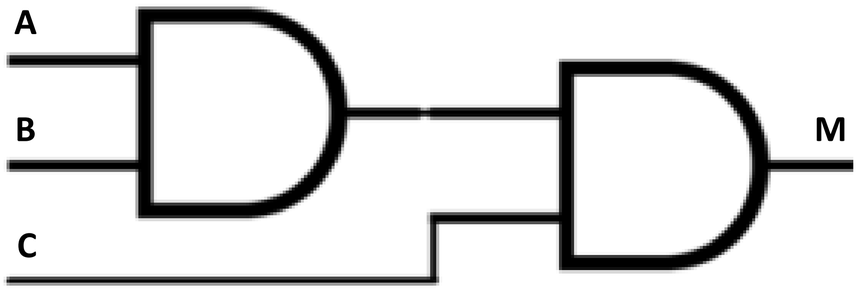
A	B	A⊕B
0	0	0
0	**1**	**1**
1	**0**	**1**
1	1	0

Funzione logica:

$\mathbf{A} \oplus \mathbf{B} = \text{not}(A) \cdot B + A \cdot \text{not}(B)$

PORTE CON PIÙ DI DUE INGRESSI

- Ad eccezione della porta NOT, le altre porte logiche possono esistere anche ad **N ingressi** (2, 3, 4,…,N).
- Queste porte svolgono l'**operazione logica associata su N bit invece che su 2**.
  - sono particolarmente comode nella rappresentazione grafica dei circuiti logici.
- Nella pratica, cioè nella realizzazione di circuiti, se si hanno a disposizione solo porte a 2 ingressi, è possibile realizzare porte a N ingressi collegando a cascata tra loro porte a 2 ingressi.
  - **Esempio**: una AND a 3 ingressi si può creare usando 2 AND a 2 ingressi come segue:



Funzione logica:

$M = A \cdot B \cdot C$

OSSERVAZIONI

Qualunque funzione logica può essere costruita usando le porte logiche AND, OR e NOT.

Le porte NOR e NAND svolgono la funzione di inverter e sono definite universali.

NAND → OR

$$A \text{ NAND } B = \text{NOT}(A \cdot B) = (\text{NOT } A) + (\text{NOT } B)$$

La porta NAND si può quindi realizzare anche con una NOT e un AND in cascata, oppure con due NOT e una OR.

NAND → NOT

$$A \text{ NAND } 1 = \text{NOT}(A \cdot 1) = (\text{NOT } A)$$

Quando uno dei due ingressi alla NAND è 1, allora la porta equivale a negare l'altro input. In tal caso basta una porta NOT.

PROPRIETA', ASSIOMI E TEOREMI DELL'ALGEBRA BOOLEANA

L'**algebra di Boole** (anche detta **algebra booleana**, **logica booleana** o **reticolo booleano**), in matematica e logica matematica, è il ramo dell'algebra in cui le variabili possono assumere solamente i valori *vero* e *falso* (valori di verità), generalmente denotati rispettivamente come 1 e 0.

https://it.wikipedia.org/wiki/Algebra_di_Boole

N°	Nome	Proprietà
1	**Proprietà associativa**	$a + (b + c) = (a + b) + c$; $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
2	**Proprietà commutativa**	$a + b = b + a$; $a \cdot b = b \cdot a$
3	**Proprietà distributiva**	$a + (b \cdot c) = (a + b) \cdot (a + c)$ $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
4	**Assioma dell'annullamento**	$a \cdot 0 = 0$; $a + 1 = 1$
5	**Assioma del Complemento**	$(a + \bar{a}) = 1$; $(a \cdot \bar{a}) = 0$
6	**Assioma dell'idempotenza**	$a \cdot a = a$; $a + a = a$
7	**Assioma doppia negazione**	$\bar{\bar{a}} = a$
8	**Teorema di DeMorgan**	$a + b = \overline{\bar{a} \cdot \bar{b}}$; $a \cdot b = \overline{\bar{a} + \bar{b}}$
9	**Teorema dell'assorbimento**	Se $Y = a + ab$ allora $Y = a$
10	**Teorema del consenso**	Se $Y = ab + \bar{a}c + bc$ allora $Y = ab + \bar{a}c$

Leggi di De Morgan

$$\overline{a + b} = \bar{a} \cdot \bar{b}$$

$$\overline{a \cdot b} = \bar{a} + \bar{b}$$

TEOREMA DI ESPRESSIONE CANONICA (O DEI MINTERMINI)

Qualunque funzione logica può essere costruita usando le porte logiche AND, OR e NOT.

Il **Teorema dei Mintermini** afferma che **ogni funzione booleana può essere espressa come una somma (operazione OR) di mintermini**.
Un **mintermine** è un prodotto (operazione AND) di tutte le variabili della funzione, ciascuna presente in forma diretta o negata, che rendono la funzione uguale a 1.

In altre parole, **per ogni combinazione di variabili in cui la funzione booleana restituisce 1, esiste un mintermine corrispondente**. La somma di tutti questi mintermini fornisce una rappresentazione completa della funzione, nota come **prima forma canonica o forma normale *dis*giuntiva (Sum of Products, SOP)**.

È un **modo standard per rappresentare una funzione logica** usando solo AND (prodotto logico), OR (somma logica), NOT (negazione).

TEOREMA DI ESPRESSIONE CANONICA **DUALE** (O DEI MAXTERMINI)

Qualunque funzione logica può essere costruita usando le porte logiche AND, OR e NOT.

Il **Teorema dei Maxtermini** afferma che **ogni funzione booleana può essere espressa come un prodotto (operazione AND) di maxtermini**.
Un **maxtermine** è una somma (operazione OR) di tutte le variabili della funzione, ciascuna presente in forma diretta o negata, che rendono la funzione uguale a 0.

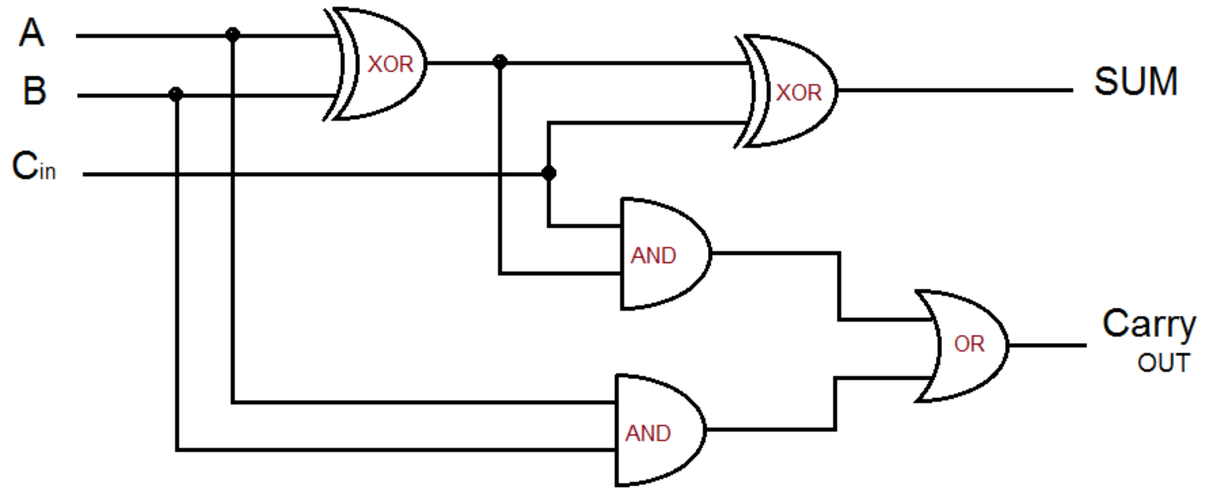
In altre parole, **per ogni combinazione di variabili in cui la funzione booleana restituisce 0, esiste un maxtermine (somma) corrispondente**. La somma di tutti questi maxtermini fornisce una rappresentazione completa della funzione, nota come **seconda forma canonica o forma normale *con*giuntiva (Product of Sums, POS)**.

È un altro **modo standard per rappresentare una funzione logica** usando solo AND (prodotto logico), OR (somma logica), NOT (negazione).

IL PRIMO CIRCUITO LOGICO

Full adder

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

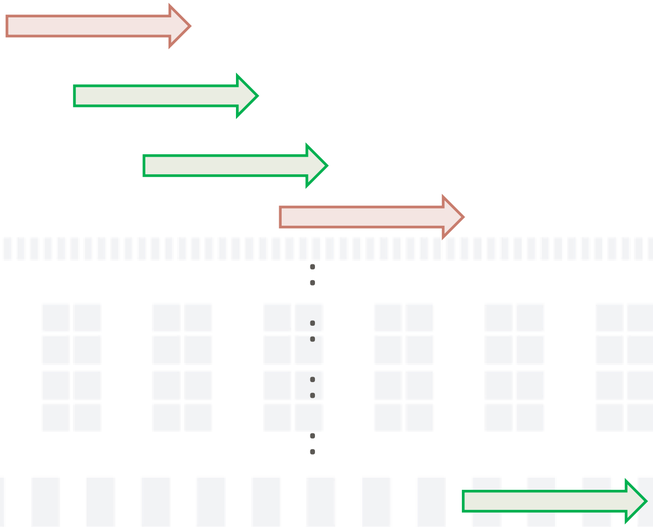


Questo circuito realizza la **SOMMA BINARIA** tra due bit (con riporto).

IL PRIMO CIRCUITO LOGICO

Full adder

Inputs			Outputs	
A	B	C_{in}	S	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	



Ogni riga della tabella di verità genera un MINTERMINE/MAXTERMINE.

Applichiamo il *Teorema dei Mintermini* **OPPURE** il *Teorema dei Maxtermini* usando, rispettivamente, i mintermini **OPPURE** i maxtermini e poi combinandoli tra loro:

(1) Nel caso della forma SOP, cerchiamo le righe dove S=1, formiamo i mintermini come prodotto delle variabili e poi li sommiamo.

(2) Nel caso della forma POS, cerchiamo le righe dove S=0, i maxtermini sono somma delle variabili e vengono moltiplicati tra loro.

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma SOP del Teorema.. (*continua*..)

Inputs			Outputs	
A	B	C_{in}	S	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

not(A) · not(B) · C

not(A) · B · not(C)

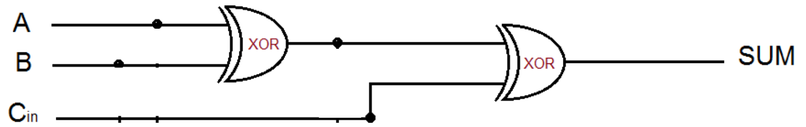
A · not(B) · not(C)

A · B · C

S = not(A) · not(B) · C + not(A) · B · not(C) + A · not(B) · not(C) + A · B · C

IL PRIMO CIRCUITO LOGICO

Full adder



Considerando la forma SOP del Teorema.. (*continua*..)

$$S = \text{not}(A) \cdot \text{not}(B) \cdot C + \text{not}(A) \cdot B \cdot \text{not}(C) + A \cdot \text{not}(B) \cdot \text{not}(C) + A \cdot B \cdot C$$

semplificazione (usando proprietà e assiomi dell'algebra booleana)

$$S = (\text{not}(A) \cdot \text{not}(B) + A \cdot B) \cdot C + (\text{not}(A) \cdot B + A \cdot \text{not}(B)) \cdot \text{not}(C)$$

pr. distrib. inversa

$$A \oplus B$$

$$\text{not}(A) \cdot \text{not}(B) + A \cdot B = \text{NOT}(A \oplus B)$$

$$S = \text{not}(A \oplus B) \cdot C + (A \oplus B) \cdot \text{not}(C)$$

not(X) **X**

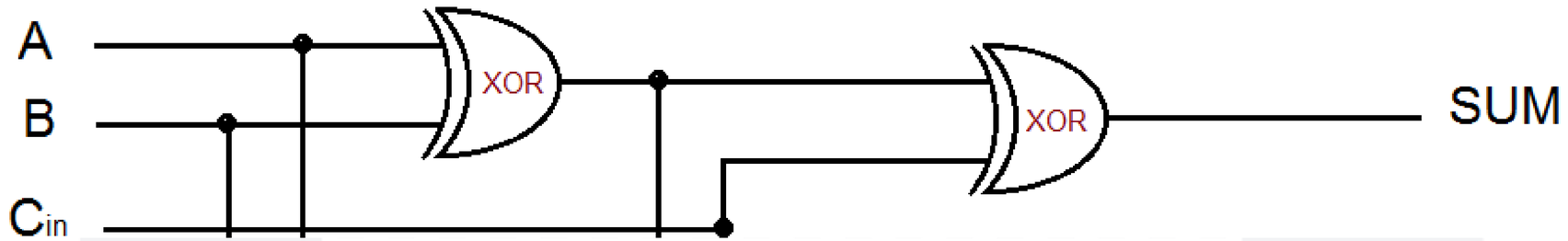
A	B	A XOR B	NOT(A XOR B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma SOP del Teorema..

S = (A ⊕ B) ⊕ C



IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma **POS** del Teorema.. (*continua..*)

Inputs			Outputs	
A	B	C_{in}	S	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

A + B + C

A + not(B) + not(C)

not(A) + B + not(C)

not(A) + not(B) + C

S = (A + B + C) · (A + not(B) + not(C)) · (not(A) + B + not(C)) · (not(A) + not(B) + C)

IL PRIMO CIRCUITO LOGICO

Full adder

Considerando la forma POS del Teorema.. (*continua..*)

$S = (A + B + C) \cdot (A + not(B) + not(C)) \cdot (not(A) + B + not(C)) \cdot (not(A) + not(B) + C) =$

$= [A + (B + C) \cdot (not(B) + not(C)] \cdot [not(A) + (B + not(C)) \cdot (not(B) + C)] =$

$= [A + (B \cdot not(B) + C \cdot not(B) + B \cdot not(C) + C \cdot not(C)] \cdot [not(A) + (B \cdot not(B) + not(C) \cdot not(B) + B \cdot C + C \cdot not(C)] =$

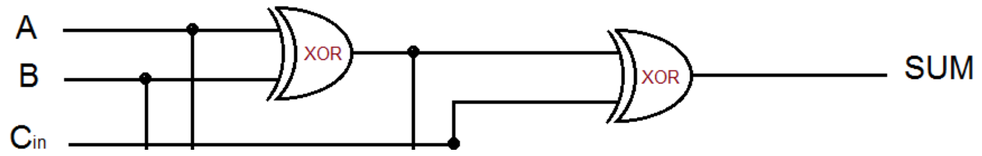
$= [A + C \cdot not(B) + B \cdot not(C)] \cdot [not(A) + not(C) \cdot not(B) + B \cdot C] =$

$B \oplus C$

$not(B \oplus C)$

$= A \cdot not(A) + (B \oplus C) \cdot not(A) + A \cdot not(B \oplus C) + (B \oplus C) \cdot not(B \oplus C) =$

$= not(A) \cdot (B \oplus C) + A \cdot not(B \oplus C) = A \oplus B \oplus C$



OSSERVAZIONI IMPORTANTI

- La tabella definisce una funzione logica unica
- **L’espressione canonica (SOP o POS) è unica**

- *La forma algebrica della funzione logica è unica:* ***NO!***
- *Ci possono essere circuiti equivalenti che implementano la stessa funzione logica.*

IL PRIMO CIRCUITO LOGICO

Full adder: circuito che realizza la SOMMA BINARIA tra due bit (con riporto).

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
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

Esercizio

Scrivete l'espressione della funzione logica C_{out} e provate ad integrare il circuito di prima con le porte mancanti

DECODER (1/2)

Un decoder è un componente elettronico caratterizzato dall'avere **n ingressi e 2^n uscite**.

Lo scopo del decoder è di ***impostare allo stato alto l'uscita corrispondente alla conversione in base 10*** *della codifica binaria a n bit ricevuta* ***in input*** *(e di impostare allo stato basso tutte le altre).*

In pratica:

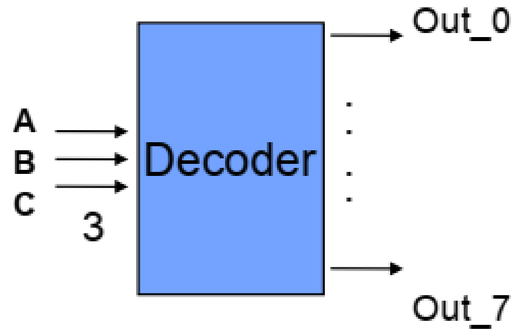
- gli n input sono interpretati come un numero unsigned
- se questo numero rappresenta il numero i, allora
  - solo il bit in output di indice i (i=0,1,...,2^{n-1}) verrà posto ad 1
  - tutti gli altri verranno posti a 0

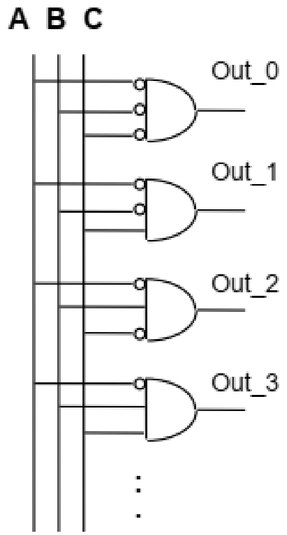
DECODER (2/2)

2^n uscite, solo 1 valore è attivo per ogni combinazione di input.

Quindi:

- **l'ingresso seleziona una delle uscite**;
- l'uscita selezionata ha valore 1 tutte le altre 0





A	B	C	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

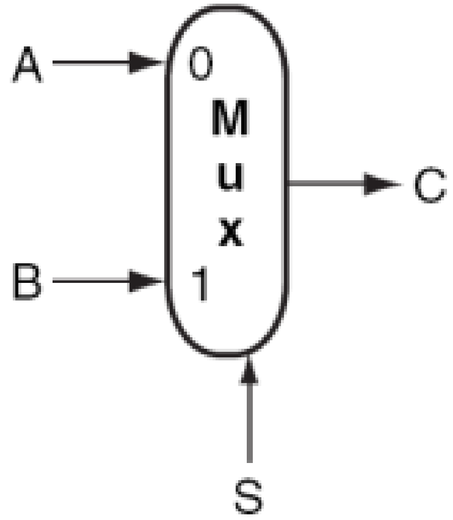
C'è un solo 1 per ogni riga

Con 3 bit possiamo selezionare UNA di $2^3 = 8$ uscite

MULTIPLEXER

Un multiplexer, detto anche selettore, è un componente elettronico caratterizzato da:

- 2^n entrate principali
- n entrate di controllo (selettore)
- 1 uscita



Il valore del selettore determina quale input diviene output.

MULTIPLEXER

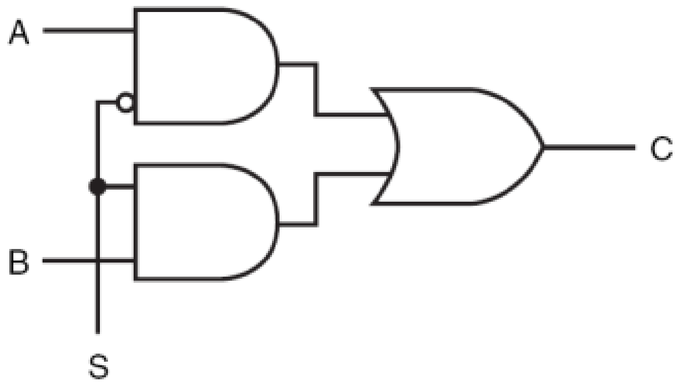
Esempio 1

Funzione logica

$$C = (A \cdot \bar{S}) + (B \cdot S)$$

Qual è la tabella di verità di questo circuito?

Circuito logico equivalente



S	A	B	C
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

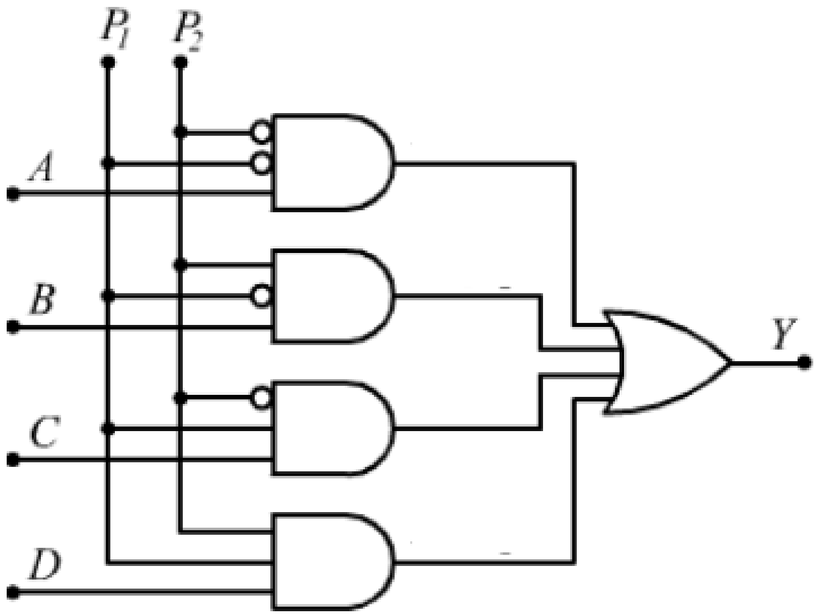
S ha la funzione di segnale di controllo

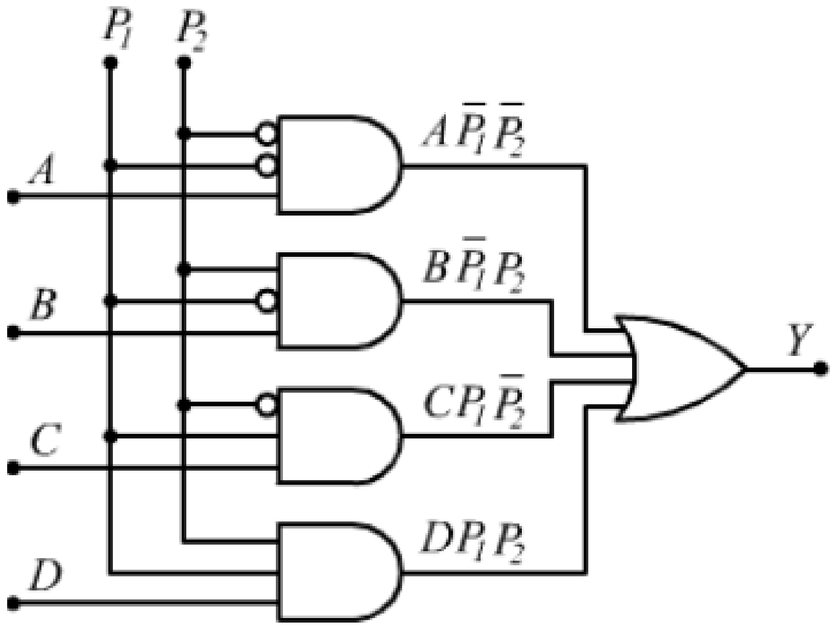
MULTIPLEXER

Esempio 2

Circuito logico

Quale funzione logica viene implementata??





$$Y = A\bar{P}_1\bar{P}_2 + B\bar{P}_1P_2 + CP_1\bar{P}_2 + DP_1P_2$$

MULTIPLEXER

Esempio 2

$$Y = A\bar{P}_1\bar{P}_2 + B\bar{P}_1P_2 + CP_1\bar{P}_2 + DP_1P_2$$

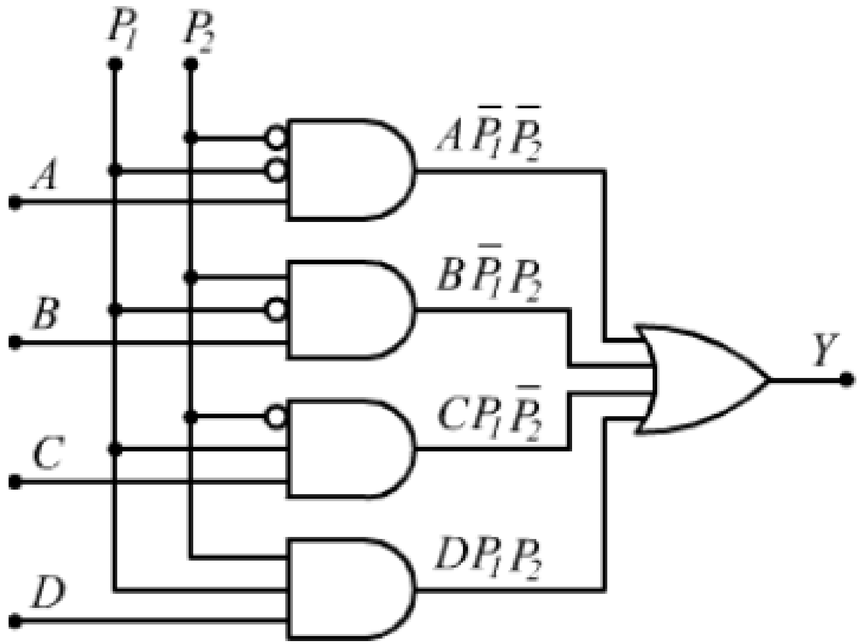
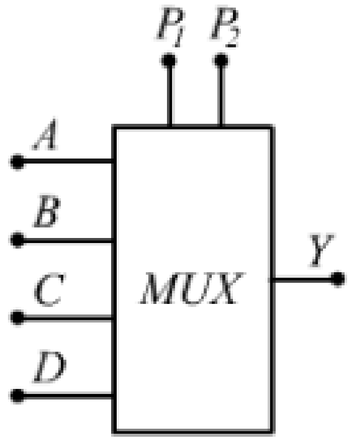


Tabella di verità

P_1	P_2	Y
0	0	A
0	1	B
1	0	C
1	1	D



Circuito con **2 segnali di controllo**

Materiale per la lezione

- Appendice B, Patterson & Hennessy

Prossima lezione: 18 ***marzo, h.16:00, aula 3B***