

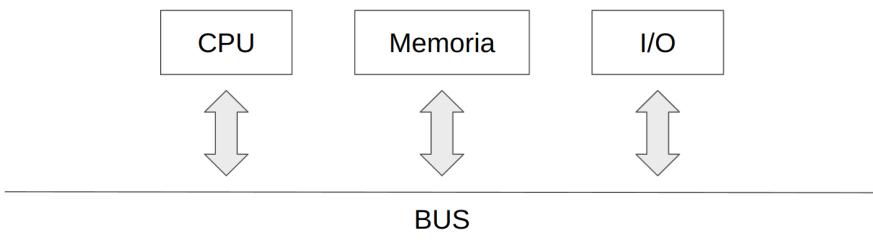
# PROGETTAZIONE DI SISTEMI DIGITALI

本笔记由九九精心整理汇编

## Programma del corso:

- Rappresentazione dell'informazione;
- **Circuiti** (reti) **combinatori** e algebra di **Boole**:
  - **Moduli standard**: MUX, DEMUX, ROM, PLA, codificatori e decodificatori;
- **Circuiti** (reti) **sequenziali** e **automi** a stati finiti:
  - **Moduli standard**: registri di memorizzazione e registri contatori.

## Curiosità: Architettura di Von Neumann:



## RAPPRESENTAZIONE DELL'INFORMAZIONE

Data una generica funzione:

$$f : A \rightarrow B$$

**A** = dominio

**B** = codominio

Funzione di **codifica**:

$$c : I \rightarrow C$$

**I** = informazione

**C** = codice

Funzione di **decodifica**:

$$c^{-1} : C \rightarrow I$$

L'alfabeto è un insieme di simboli utilizzati.

**Bit (binary digit)**: nelle rappresentazioni per il computer useremo solo **{0, 1}**.

**Cardinalità**: numero di elementi di un insieme, viene indicato con  $|I|$  oppure  $N_I$ :

- $N_I = N_C$ : codifica **non ambigua** e **non ridondante**;
- $N_I > N_C$ : codifica **ambigua**;
- $N_I < N_C$ : codifica **ridondante**.

**Computer**:  $\Sigma = \{0, 1\}$ , massimo numero di simboli per comporre parole di codice.

**Proprietà** di una codifica:

- **economicità**: minor numero di simboli;
- **efficienza** nella rappresentazione;
- **efficienza** nell'esecuzione delle operazioni.

Se **lunghezza** = 3 simboli:

- con  $\Sigma = \{0, 1\}$  si possono rappresentare  $2^3$  parole (000, 001, 010, 011, 100, 101, 110, 111);
- con  $\Sigma = \{\text{alfabeto italiano}\}$  si possono rappresentare  $21^3$  parole (aaa, aab, aac, aad, ..., zzz).

Se **lunghezza** = 4 bit, si possono rappresentare  $2^4 = 16$  numeri:

0 → 0000      1 → 0001      2 → 0010      3 → 0011      4 → 0100      5 → 0101      ...      15 → 1111

**Composizione** decimale e binaria:

$$3872_{(10)} \rightarrow 111100100000_{(2)}$$

$$3872_{(10)} = 3 \times 10^3 + 8 \times 10^2 + 7 \times 10^1 + 2 \times 10^0$$

$$111100100000_{(2)} = 1 \times 2^{11} + 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

**Prefissi** di grandezza dei bit:

$$2^{10} = 1024 \text{ bit} \rightarrow \text{kilobit} \quad 2^{20} = 1.048.576 \text{ bit} \rightarrow \text{megabit} \quad 2^{30} = 1.073.741.824 \text{ bit} \rightarrow \text{gigabit}$$

La denominazione del prefisso deriva da:

$$10^3 \rightarrow \text{kilo} \text{ (mille)}$$

$$10^6 \rightarrow \text{mega} \text{ (milione)}$$

$$10^9 \rightarrow \text{giga} \text{ (miliardo)}$$

### Decodificare:

$$\begin{array}{lll} 101101_{(2)} = 2^5 + 2^3 + 2^2 + 2^0 = 32 + 8 + 4 + 1 = 45_{(10)} & \rightarrow & \text{da base 2 a base 10} \\ 312_{(4)} = 3 \times 4^2 + 1 \times 4^1 + 2 \times 4^0 = 48 + 4 + 2 = 54_{(10)} & \rightarrow & \text{da base 4 a base 10} \\ 312_{(5)} = 3 \times 5^2 + 1 \times 5^1 + 2 \times 5^0 = 75 + 5 + 2 = 82_{(10)} & \rightarrow & \text{da base 5 a base 10} \end{array}$$

### Sistemi numerici posizionali:

Data una base  $b$ , il sistema numerico posizionale utilizza le cifre da 0 a  $b - 1$ :

$$\Sigma = \{0, \dots, b - 1\}$$

**Lunghezza n:** con  $n$  simboli si possono formare  $b^n$  parole distinte.

Intervallo delle parole =  $[0; b^n - 1]$

### Codificare:

- Si divide ripetutamente per la **base** il **valore** e i **quotienti** ottenuti fino a quando il quoziente = 0;
- La rappresentazione è data dalla sequenza di resti in ordine **inverso**.

$$45_{(10)} \rightarrow 101101_{(2)}$$

Diviso 2	
45	Resto
22	1
11	0
5	1
2	1
1	0
0	1

$$26_{(10)} \rightarrow 11010_{(2)}$$

Diviso 2	
26	Resto
13	0
6	1
3	0
1	1
0	1

$$82_{(10)} \rightarrow 312_{(5)}$$

Diviso 5	
82	Resto
16	2
3	1
0	3

Le basi molto utilizzati sono **base 8** e **base 16** (potenza di 2):

$$\text{base 8} = 2^3 \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7\}$$

$$\text{base 16} = 2^4 \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\} \rightarrow (A = 10, B = 11, C = 12, D = 13, E = 14, F = 15)$$

Usando i numeri per  $\{10, \dots, 15\}$  nella la base 16 provocano confusioni perché hanno 2 caratteri, quindi si usano lettere.

### Esempi:

base 2 → base 8

$$\underbrace{101}_{5} \underbrace{110}_{6} \rightarrow 56_{(8)}$$

base 2 → base 16

$$\underbrace{0010}_{2} \underbrace{1110}_{E} \rightarrow 2E_{(16)}$$

La **cifra meno significativa (LSB)** è quella più a **destra** (associata alla potenza 0);

la **cifra più significativa (MSB)** è quella più a **sinistra**.

base 8 → base 2

$$\begin{array}{c} 365_{(8)} \\ \downarrow \\ 011 \quad 110 \quad 101 \end{array} \rightarrow 011110101_{(2)}$$

base 16 → base 2

$$\begin{array}{c} A5B_{(16)} \\ \downarrow \\ 1010 \quad 0101 \quad 1011 \end{array} \rightarrow 101001011011_{(2)}$$

$$\begin{array}{c} 365_{(16)} \\ \downarrow \\ 0011 \quad 0110 \quad 0101 \end{array} \rightarrow 3 \times 16^2 + 6 \times 16^1 + 5 \times 16^0 = 768 + 96 + 5 = 869_{(10)}$$

$$2^9 + 2^8 + 2^6 + 2^5 + 2^2 + 2^0 = 512 + 256 + 64 + 32 + 4 + 1 = 869_{(10)}$$

### Moltiplicazioni e divisioni per potenze della base:

$$\text{base } = 10 \quad 23 \times 10^4 = 230000 \text{ (aggiungere 4 zeri finali)}$$

$$7800 / 10^2 = 78 \text{ (togliere 2 zeri finali)}$$

$$\text{base } = 2 \quad 101 \times 2^3 = 101000 \text{ (aggiungere 3 zeri finali)}$$

$$01100 / 2^2 = 011 \text{ (togliere 2 zeri finali)}$$

$$\text{base } b \quad \text{n cifre} \quad \sum_{i=0}^{n-1} a_i b^i = a_0 b^0 + a_1 b^1 + \dots + a_{n-1} b^{n-1}$$

未完待续 ( 这里缺一大部分, 过段时间补全 )

## CIRCUITI COMBINATORI E ALGEBRA DI BOOLE

### Porte logiche di base:



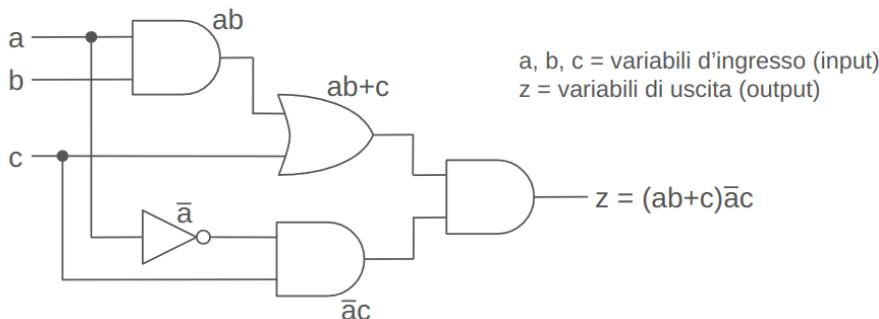
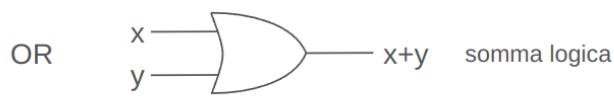
x	$\bar{x}$
0	1
1	0



x	y	$xy$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1



### Algebra di Boole (o Booleana):

- alfabeto di supporto = {0, 1}
- operazioni o operatori = **OR** (+, ∨), **AND** (·, ∧), **NOT**
- assiomi:
  - **associatività:**  $(x + y) + z = x + (y + z)$   $(xy)z = x(yz)$
  - **commutatività:**  $x + y = y + x$   $xy = yx$
  - **distributività:**  $x(y + z) = xy + xz$   $x + yz = (x + y)(x + z)$
  - **complemento:**  $x + \bar{x} = 1$   $x \cdot \bar{x} = 0$
  - **elemento neutro:**  $x + 0 = x$   $x \cdot 1 = x$
- proprietà:
  - **involuzione:**  $\bar{\bar{x}} = x$
  - **idempotenza:**  $x + x = x$   $x \cdot x = x$
  - **elemento nullo:**  $x + 1 = 1$   $x \cdot 0 = 0$
  - **assorbimento:**  $x + xy = x$   $x \cdot (x + y) = x$
  - **leggi di De Morgan:**  $\overline{x + y} = \bar{x} \cdot \bar{y}$   $\overline{x \cdot y} = \bar{x} + \bar{y}$

### Espresione Booleana (E.B.):

- la singola variabile;
- la negazione di una variabile;
- la somma e il prodotto di due variabili;
- posso iterare le due definizioni precedenti e usare parentesi per ottenere le espressioni booleane.

**Identità** si ottiene eguagliando due diverse E.B.

**Osservazione:** le identità negli assiomi e nelle proprietà scritte sono legate da una relazione di **dualità** che si ottiene scambiando gli operatori + e · e le costanti 0 e 1.

**Esempio:**  $(a + b)c(d + \bar{a})$       la sua duale =  $ab + c + d\bar{a}$

## Verifica di identità:

- metodo dell'**induzione perfetta**: verifica per ogni possibile valore delle variabili:

$x$	$y$	$x + y$	$\bar{x} \cdot \bar{y}$	$\bar{x} + \bar{y}$
0	0	1	1	1
0	1	0	0	1
1	0	0	0	1
1	1	0	0	0

- metodo basato su trasformazione tramite **assiomi e proprietà**:

$$\begin{aligned} x + xy &= x && (\text{distribuzione}) \\ x(1 + y) &= x && (\text{elemento nullo}) \\ x(1) &= x && (\text{elemento neutro}) \\ x = x & \rightarrow && \text{identità verificata} \end{aligned}$$

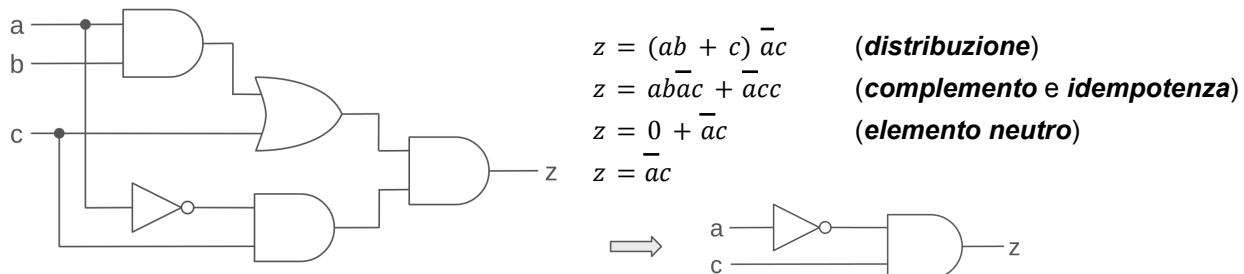
**Esercizi:** verificare che  $a + \bar{a}b = a + b$  e  $ab + \bar{a}\bar{b} + ab\bar{b} + \bar{a}\bar{b} = 1$

$$\begin{array}{lll} a + \bar{a}b = a + b & (\text{distribuzione}) & ab + \bar{a}\bar{b} + ab\bar{b} + \bar{a}\bar{b} = 1 \\ (a + \bar{a})(a + b) = a + b & (\text{complemento}) & b(a + \bar{a}) + \bar{b}(a + \bar{a}) = 1 \\ (1)(a + b) = a + b & (\text{elemento neutro}) & b(1) + \bar{b}(1) = 1 \\ a + b = a + b & & b + \bar{b} = 1 \\ & & 1 = 1 \end{array} \quad \begin{array}{lll} & & (\text{distribuzione}) \\ & & (\text{complemento}) \\ & & (\text{elemento neutro}) \end{array}$$

**Regola del consenso:**  $ab + bc + \bar{a}\bar{c} = ab + \bar{a}\bar{c}$

**Verifica:**

$$\begin{aligned} ab + bc + \bar{a}\bar{c} &= ab + \bar{a}\bar{c} && (\text{elemento neutro e complemento}) \\ ab + (a + \bar{a})bc + \bar{a}\bar{c} &= ab + \bar{a}\bar{c} && (\text{distribuzione}) \\ ab + abc + \bar{a}\bar{b}c + \bar{a}\bar{c} &= ab + \bar{a}\bar{c} && (\text{assorbimento}) \\ ab + \bar{a}\bar{c} &= ab + \bar{a}\bar{c} && \end{aligned}$$

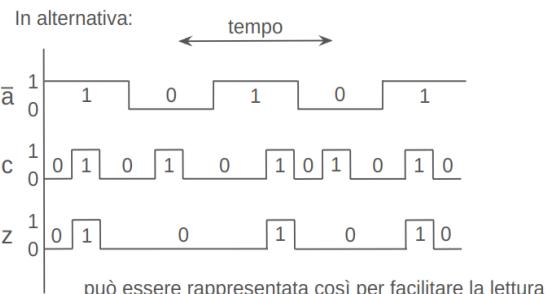
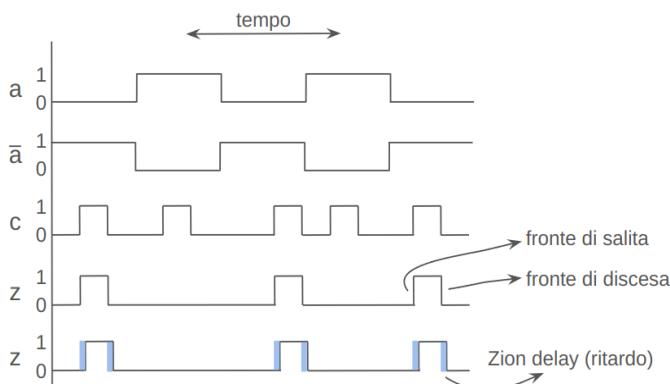


**Problema:** disegnare il circuito che produce accensione della luce (output = 1) se è notte e passa qualcuno.

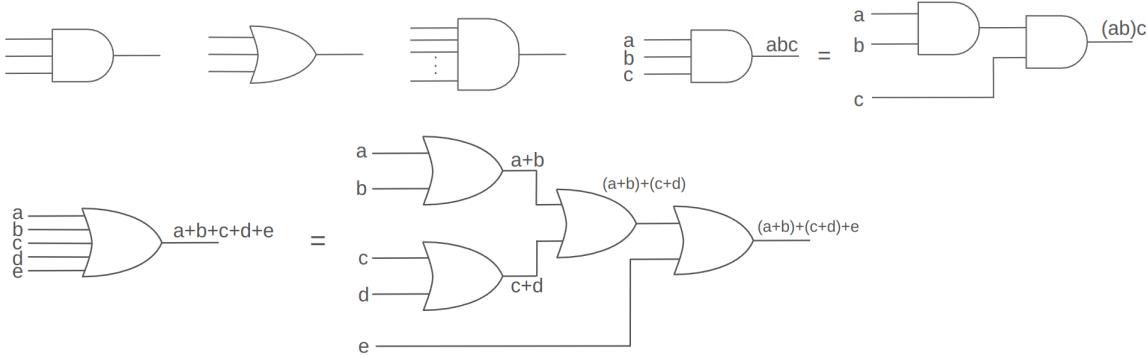
$$a = \begin{cases} 1 & \text{se è giorno} \\ 0 & \text{se è notte} \end{cases} \quad c = \begin{cases} 1 & \text{se passa qualcuno / rileva un movimento} \\ 0 & \text{altrimenti} \end{cases}$$

$$z = \bar{a}c$$

**Rappresentazione raggio quadra:**



## Porte a più di due ingressi:



**Espressione complementare:** data un'espressione booleana, si applica il complementare o negazione all'espressione.

**esempio:**  $f = ab + bc$

- **primo metodo** per ottenere l'espressione complementare:

$$\bar{f} = \overline{ab + bc} = \overline{ab} \cdot \overline{bc} = (\bar{a} + \bar{b})(\bar{b} + \bar{c}) = (\bar{a} + b)(\bar{b} + c) = \bar{a}\bar{b} + \bar{a}c + b\bar{b} + bc = \bar{a}\bar{b} + \bar{a}c + bc$$

(De Morgan) (De Morgan) (involuzione) (distribuzione) (complemento)

- **secondo metodo** per ottenere l'espressione complementare:

- si passa all'espressione duale;
- si complementa tutte le singole variabili;

$$f = ab + bc \quad \Rightarrow \quad \tilde{f} = (a + \bar{b})(b + \bar{c}) \quad \Rightarrow \quad \bar{f} = (\bar{a} + \bar{b})(\bar{b} + \bar{c}) = (\bar{a} + b)(\bar{b} + c)$$

(duale) (comp. su singole var.) (involuzione)

## Osservazione:

- espressione **POS** (Product of Sum) o in forma congiuntiva:  $(\bar{a} + b)(\bar{b} + c)$
- espressione **SOP** (Sum of Product) o in forma disgiuntiva:  $\bar{a}\bar{b} + \bar{a}c + bc$

## Da espressione a tavola di verità:

$a$	$b$	$c$	$f$	$\bar{f}$	$\tilde{f}$
0	0	0	0	1	1
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	0	1	1

Espressioni booleane:

$$f = \boxed{ab} + \boxed{bc}$$

SOP: mettiamo 1 nella tavola di verità

$$\bar{f} = \boxed{\bar{a}\bar{b}} + \boxed{\bar{a}c} + \boxed{bc}$$

SOP: mettiamo 1 nella tavola di verità

$$\tilde{f} = \boxed{(a + \bar{b})} \boxed{(b + \bar{c})}$$

POS: mettiamo 0 nella tavola di verità

## Da tavola di verità ad espressione:

SOP si ricava dai valori **1** della tavola:  $\tilde{f} = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + abc + abc$

**SOP in forma canonica:** in tutti i termini **somma** compaiono tutte le variabili, vere o complementate (**mintermini**).

POS si ricava dai valori **0** della tavola:  $\tilde{f} = (a + b + \bar{c})(a + \bar{b} + c)(a + \bar{b} + \bar{c})(\bar{a} + b + \bar{c})$

**POS in forma canonica:** in tutti i termini **prodotto** compaiono tutte le variabili, vere o complementate (**maxtermini**).

**Esercizio:** data  $f = (\bar{a} + c)(\bar{a} + \bar{b}) + c$

- portare in forma **normale SOP**:  $f = (\bar{a} + c)(\bar{a} + \bar{b}) + c = (\bar{a} + c)(\bar{a}\bar{b}) + c = \bar{a}\bar{a}\bar{b} + \bar{a}\bar{b}c + c = \bar{a}\bar{b} + c$  (**De Morgan**) (**distribuzione**) (**idempotenza+assorbimento**)
- ricavare la **duale** e portare in forma **SOP**:  $f = \bar{a}\bar{b} + c \Rightarrow \tilde{f} = (\bar{a} + \bar{b})c = \bar{a}c + \bar{b}c$  (**duale**) (**distribuzione**)
- tavola di verità** di  $f$  e  $\tilde{f}$ :

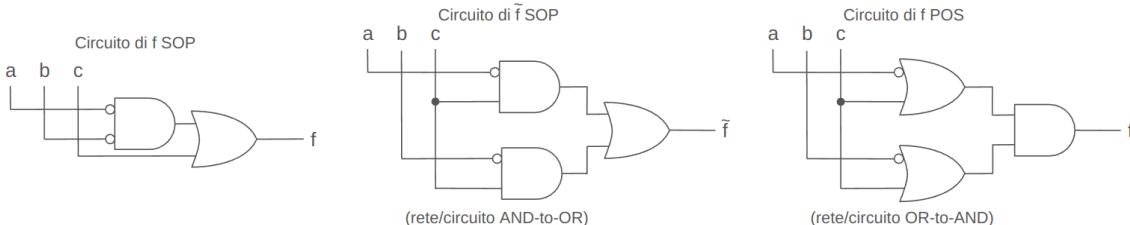
a	b	c	f	$\tilde{f}$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	0

- portare  $f$  in forma **normale POS**:  $f = \bar{a}\bar{b} + c = (\bar{a} + c)(\bar{b} + c)$  (**distribuzione**)

#### Procedimento per ottenere la forma normale SOP o POS:

- applicare **De Morgan** per ottenere complemento sulla singola variabile;
- applicare la proprietà **distributiva**;
- eliminare termini ridondanti o ripetuti: **assorbimento** o **idempotenza**.

#### Circuiti dell'esercizio precedente:



Le reti **AND-to-OR** hanno un livello di porte **AND** che confluiscano tutte in una porta **OR**.

Le reti **OR-to-AND** hanno un livello di porte **OR** che confluiscano tutte in una porta **AND**.

La **complessità** (numero di porte e tempo di attraversamento) del circuito dipende dal **numero di ingressi** (**FAN-IN**) delle porte.

#### Espressioni in forma canonica SOP e POS dell'esercizio precedente:

- canonica SOP:  $f = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + abc + ab\bar{c}$  (prodotto = 1 in tavola di verità)  
 $m_0 + m_1 + m_3 + m_5 + m_7$  oppure **OR**( $m_0, m_1, m_3, m_5, m_7$ )
- canonica POS:  $f = (a + \bar{b} + c)(\bar{a} + b + c)(\bar{a} + \bar{b} + c)$  (somma = 0 in tavola di verità)  
 $M_2 \cdot M_4 \cdot M_6$  oppure **AND**( $M_2, M_4, M_6$ )

#### Dalla forma normale alla forma canonica:

- $f = \bar{a}\bar{b} + c = \bar{a}\bar{b}(\bar{c} + c) + (\bar{a} + a)(\bar{b} + b)c = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + (\bar{a}\bar{b} + \bar{a}b + a\bar{b} + ab)c = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}\bar{b}c + \bar{a}bc + abc = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}\bar{b}c + \bar{a}bc + abc$
- $f = (\bar{a} + c)(\bar{b} + c) = (\bar{a} + \bar{b}b + c)(\bar{a}a + \bar{b} + c) = (\bar{a} + \bar{b} + c)(\bar{a} + b + c)(\bar{a} + \bar{b} + c)(a + \bar{b} + c) = (\bar{a} + \bar{b} + c)(\bar{a} + b + c)(a + \bar{b} + c)$

#### Procedimento per passare da normale SOP/POS a canonica:

- se in un termine prodotto manca una variabile, si moltiplica la somma della variabile e la sua complementata;
- se in un termine somma manca una variabile, si somma il prodotto della variabile e la sua complementata;
- si applica la proprietà distributiva e si ottengono i mintermini/maxtermini;
- si applica idempotenza per eliminare termini ripetuti.

**Esercizio:** stendere la tavola di verità e trovare le espressioni canoniche SOP e POS:

$$f(x, y) = f_1 + f_2$$

$$f_1(x, y) = 0 \text{ se } x = 1 \text{ e } y = 0$$

$$f_2 = \begin{cases} x \text{ se } y = 0 \\ \bar{x} \text{ se } y = 1 \end{cases}$$

x	y	f <sub>1</sub>	f <sub>2</sub>	f
0	0	1	0	1
0	1	1	1	1
1	0	0	1	1
1	1	1	0	1

**SOP canonica:**

$$f_1 = \bar{x}\bar{y} + \bar{x}y + xy$$

$$f_2 = \bar{x}y + x\bar{y}$$

**POS canonica:**

$$f_1 = \bar{x} + y$$

$$f_2 = (x + y)(\bar{x} + \bar{y})$$

**Esercizio:** verificare l'identità:  $x + y + (x + z)(\bar{x} + y) = x + y + z$

$$x + y + \bar{x}\bar{x} + xy + \bar{x}z + yz = x + y + z$$

$$x + y + \bar{x}z = x + y + z$$

$$(x + \bar{x})(x + z) + y = x + y + z$$

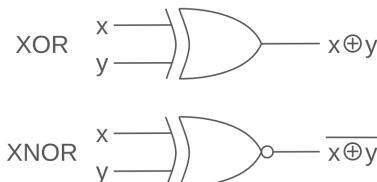
$$x + y + z = x + y + z$$

**Esercizio:** complementare di:  $x + y + (x + z)(\bar{x} + y)$  **(duale)**  
**(complementare di singole variabili)**

$$xy [ (xz) + (\bar{xy}) ]$$

$$\bar{x}\bar{y}(\bar{x}\bar{z} + xy)$$

**Altre porte logiche:**



x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

x	y	x ⊕ ȳ
0	0	1
0	1	0
1	0	0
1	1	1

$$x \oplus y = \bar{x}\bar{y} + x\bar{y}$$

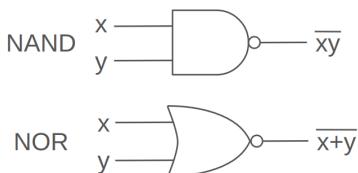
$$x \oplus 1 = \bar{x}$$

$$x \oplus 0 = x$$

$$\overline{x \oplus y} = xy + x\bar{y}$$

**XNOR:**  $\overline{x \oplus y} = \overline{\overline{xy} + xy} = \overline{(xy)(xy)} = (x + \bar{y})(\bar{x} + y) = \bar{xx} + xy + \bar{x}\bar{y} + y\bar{y} = xy + \bar{x}\bar{y}$  **(De Morgan)**

**XOR è associativo:**  $x \oplus (y \oplus z) = (x \oplus y) \oplus z$   
 $x \oplus (y \oplus z) = x \oplus (\bar{y}z + y\bar{z}) = \bar{x}(\bar{y}z + y\bar{z}) + x(\bar{y}z + y\bar{z}) = \bar{x}(\bar{y}z + y\bar{z}) + x(yz + \bar{y}\bar{z}) =$   
 $= \bar{x}\bar{y}z + \bar{x}y\bar{z} + xyz + x\bar{y}\bar{z} = (\bar{x}y + xy)z + (\bar{y}x + x\bar{y})\bar{z} = (\overline{x \oplus y})z + (x \oplus y)\bar{z} =$   
 $= (x \oplus y) \oplus z$



x	y	x̄ȳ
0	0	1
0	1	1
1	0	1
1	1	0

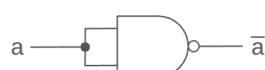
  

x	y	x̄+ȳ
0	0	1
0	1	0
1	0	0
1	1	0

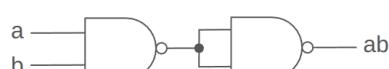
**NAND e NOR** sono **operatori universali**, quindi si possono realizzare **NOT**, **AND** e **OR** usando **solo NAND** oppure **solo NOR**, anche per i circuiti.

**NAND:**

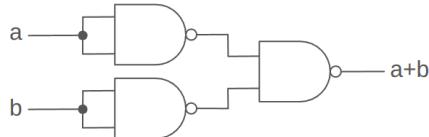
NOT  $\bar{a} = \overline{a \cdot a}$

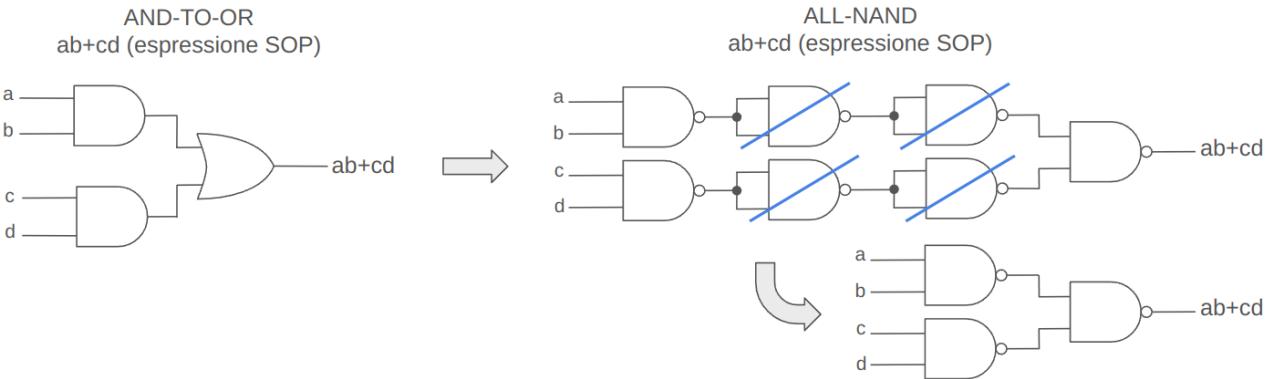


AND  $ab = \overline{\overline{ab}} = \overline{\overline{ab} \cdot \overline{ab}}$



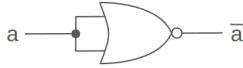
OR  $a + b = \overline{\overline{a} \cdot \overline{b}} = \overline{\overline{aa} \cdot \overline{bb}}$



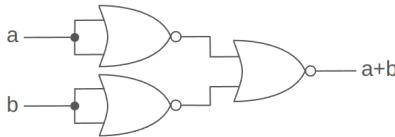


**NOR:**

NOT       $\bar{a} = \overline{\bar{a} + a}$



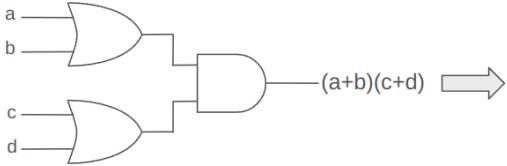
AND       $ab = \overline{\overline{\overline{a} + \overline{b}}} = \overline{\overline{a} + a + \overline{b} + b}$



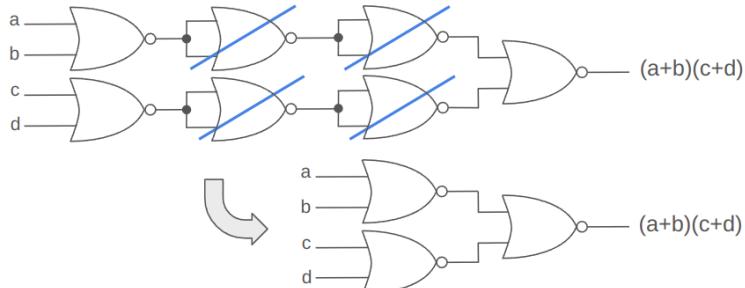
OR       $a + b = \overline{\overline{\overline{a} + \overline{b}}} = \overline{\overline{a} + b + \overline{a} + b}$



OR-TO-AND  
(a+b)(c+d) (espressione POS)



ALL-NOR  
(a+b)(c+d) (espressione POS)



**Esercizio:** progettare circuito di controllo apertura cancello.

Il cancello si apre solo se si utilizza il **telecomando** o la **chiave**, ma **non entrambi**, e che **non ci sono intralci**.

Producere: **tavola di verità, espressione SOP, espressione POS, AND-TO-OR, OR-TO-AND, ALL-NAND, ALL-NOR.**

- **Espressione booleana:**  $(t \oplus c) \bar{o}$

$$t = \begin{cases} 0 & \text{no telecomando} \\ 1 & \text{altrimenti} \end{cases} \quad c = \begin{cases} 0 & \text{no chiave} \\ 1 & \text{altrimenti} \end{cases} \quad o = \begin{cases} 0 & \text{no ostacolo} \\ 1 & \text{altrimenti} \end{cases}$$

- **Tavola di verità:**

t	c	o	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

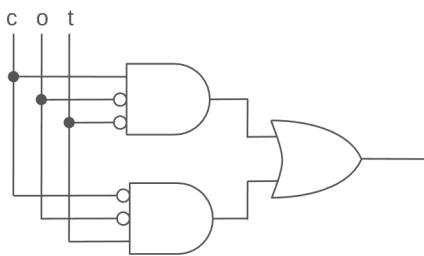
- **Espressione SOP:**  $(t \oplus c) \bar{o} = (\bar{t}c + t\bar{c}) \bar{o} = \bar{c}\bar{o}t + \bar{c}o\bar{t}$  (è anche **canonica** ( $m_4 + m_1$ ))

- **Espressione POS:**

$$(t \oplus c) \bar{o} = (\bar{t}c + t\bar{c}) \bar{o} = (\bar{t}c + t)(\bar{t}c + \bar{c}) \bar{o} = (\bar{t} + t)(c + t)(\bar{t} + \bar{c})(c + \bar{c}) \bar{o} = (c + t)(\bar{c} + \bar{t}) \bar{o}$$

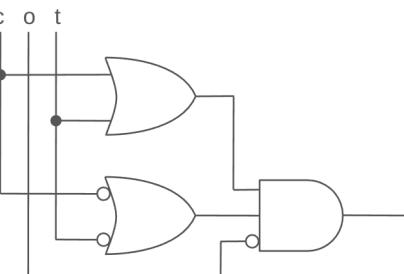
- **Espressione POS canonica:**  $(c + t)(\bar{c} + \bar{t})\bar{o} = (c + o\bar{o} + t)(\bar{c} + \bar{o}\bar{o} + \bar{t})(\bar{c}\bar{c} + \bar{o} + \bar{t}\bar{t}) =$   
 $= (c + o + t)(c + \bar{o} + t)(\bar{c} + o + \bar{t})(\bar{c} + \bar{o} + \bar{t})(c + \bar{o} + t)(c + \bar{o} + \bar{t})(\bar{c} + \bar{o} + t)(\bar{c} + \bar{o} + \bar{t}) =$   
 $= (c + o + t)(c + \bar{o} + t)(\bar{c} + o + \bar{t})(\bar{c} + \bar{o} + \bar{t})(c + \bar{o} + t)(c + \bar{o} + \bar{t}) = (\mathbf{M}_0 \cdot \mathbf{M}_2 \cdot \mathbf{M}_5 \cdot \mathbf{M}_7 \cdot \mathbf{M}_3 \cdot \mathbf{M}_6)$
- **AND-TO-OR, OR-TO-AND, ALL-NAND, ALL-NOR:**

AND-TO-OR di  $\bar{c}\bar{o}\bar{t} + \bar{c}\bar{o}\bar{t}$

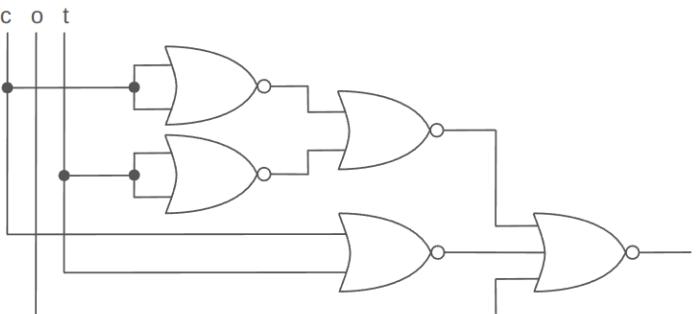
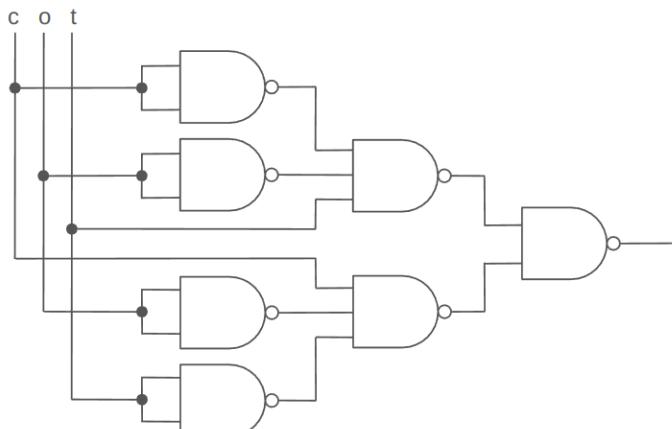


$$\text{ALL-NAND} = \overline{\overline{c} \cdot \overline{o} \cdot \overline{t} \cdot \overline{c} \cdot \overline{o} \cdot \overline{t}}$$

OR-TO-AND di  $(c + t)(\bar{c} + \bar{t})\bar{o}$



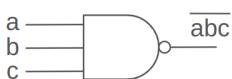
$$\text{ALL-NOR} = \overline{\overline{c} + \overline{c} + \overline{t} + \overline{t} + \overline{c} + \overline{t} + o}$$



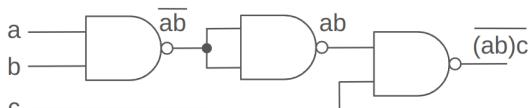
Per trasformare un'espressione **SOP / POS** in un'espressione **ALL-NAND / ALL-NOR** occorre sostituire ad ogni operatore (**NOT, AND, OR**) un operatore **NAND / NOR**; eccetto in casi particolari, quindi verificare con **De Morgan**.

$$\overline{\overline{c} + \overline{c} + \overline{t} + \overline{t} + \overline{c} + \overline{t} + o} = \overline{\overline{c} + \overline{c} + \overline{t} + \overline{t}} \cdot \overline{\overline{c} + \overline{t}} = (\overline{c} + \overline{c} + \overline{t} + \overline{t})(c + t) \bar{o} = (\bar{c} + \bar{t})(c + t) \bar{o}$$

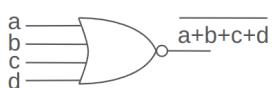
**NAND a 3 ingressi:**



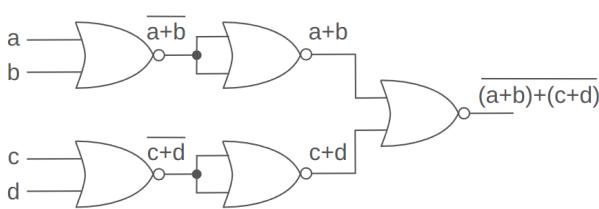
Realizzazione ALL-NAND



**NOR a 4 ingressi:**



Realizzazione ALL-NOR



**Esercizio:** stendere le tavole di verità della **funzione di maggioranza a 3 e 4 variabili**.

Funzione di maggioranza:  $f = \begin{cases} 1 & \text{se maggioranza è 1} \\ 0 & \text{se maggioranza è 0} \end{cases}$

- Tavola di verità della funzione di maggioranza a **3 variabili**:

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

- Tavola di verità della funzione di maggioranza a **4 variabili**:

$f_1 = 1$  se **almeno 2** variabili di ingresso valgono 1

$f_2 = 1$  se **più di 2** variabili di ingresso valgono 1

$f_1 \Rightarrow POS$  perché ha soli 5 **maxtermini**

$f_2 \Rightarrow SOP$  perché ha soli 5 **mintermini**

$$\begin{aligned} f_1 &= (a + b + c + d)(a + b + c + \bar{d})(a + b + \bar{c} + d)(a + \bar{b} + c + d)(\bar{a} + b + c + d) = \\ &= (a + b + c + dd)(a + b + cc + d)(a + bb + c + d)(aa + b + c + d) = \\ &= (a + b + c)(a + b + d)(a + c + d)(b + c + d) \end{aligned}$$

$$\begin{aligned} f_2 &= \bar{a}bcd + \bar{a}\bar{b}cd + ab\bar{c}d + abc\bar{d} + abcd = \\ &= abc(d + \bar{d}) + abd(c + \bar{c}) + acd(b + \bar{b}) + bcd(a + \bar{a}) = \\ &= abc + abd + acd + bcd \end{aligned}$$

a	b	c	d	$f_1$	$f_2$
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

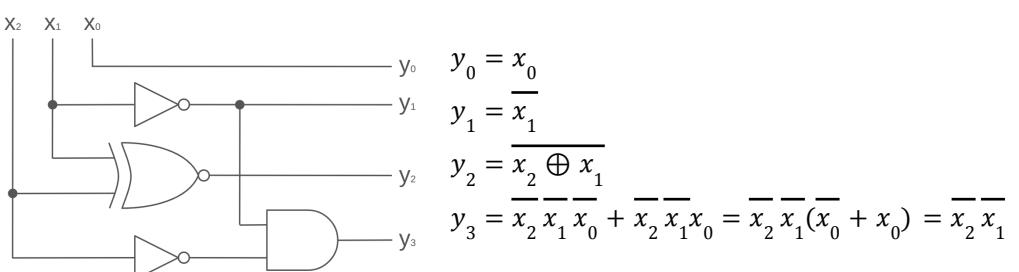
**Esercizio:** dato  $x \in [0, 7]$  rappresentato da  $x_2 x_1 x_0$ , stendere la **tavola di verità** per  $y = x - 2$  rappresentato in **complemento a 2** con il **minimo** numero di bit.

$x_2$	$x_1$	$x_0$	$x-2$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	-2	1	1	1	0
0	0	1	-1	1	1	1	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	1
1	0	0	2	0	0	1	0
1	0	1	3	0	0	1	1
1	1	0	4	0	1	0	0
1	1	1	5	0	1	0	1

Uso **4 bit** per rappresentare  $y$  perché:

3 bit in CA2 rappresenta  $[-4; 3]$

4 bit in CA2 rappresenta  $[-8; 7]$



**Funzioni non completamente specificate:**

- non tutte le combinazioni di **input** sono definite;
- non tutte le combinazioni di **output** sono definite, in corrispondenza dell'output si scrive  $\delta$  (**delta**) (**don't care**).

$x_2$	$x_1$	$x_0$	$x-2$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	-2	1	1 <sub>δ</sub>	1 <sub>δ</sub>	0 <sub>δ</sub>
0	0	1	-1	1	1	1	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	1
1	0	0	2	0	0	1	0
1	0	1	3	0	0	1	1
1	1	0	4	0	1 <sub>δ</sub>	0 <sub>δ</sub>	0 <sub>δ</sub>
1	1	1	5	0	1 <sub>δ</sub>	0 <sub>δ</sub>	1 <sub>δ</sub>

$x \in [1, 6]$   
y in CA2 con 3 bit

$$y_0 = x_0$$

$$y_1 = \overline{x_1}$$

$$y_2 = \overline{x_2} \overline{x_1}$$
 oppure  $\overline{x_2} \overline{x_1} x_0$

### Criterio di minimalità:

- una rete **AND-TO-OR / OR-TO-AND** è **minimale** se tra tutte le reti **AND-TO-OR / OR-TO-AND** ha:
  - il minimo numero di porte **AND/OR**;
  - il minimo numero di **ingressi** per ogni porta;
- un'espressione **SOP / POS** è **minimale** se tra tutte le espressioni booleane **SOP / POS** ha:
  - il minimo numero di **prodotti / somme**;
  - il minimo numero di **letterali** per ogni **prodotto / somma**.

**Minimizzazione di E.B.: mappe di Karnaugh o K-mappe** (fino a 4 variabili).

3 variabili a, b, c

a	b	c	f
0	0	0	$M_0 / m_0$
0	0	1	$M_1 / m_1$
0	1	0	$M_2 / m_2$
0	1	1	$M_3 / m_3$
1	0	0	$M_4 / m_4$
1	0	1	$M_5 / m_5$
1	1	0	$M_6 / m_6$
1	1	1	$M_7 / m_7$



		bc	00	01	11	10
		a	0	1	1	0
0		0	$M_0$ $m_0$	$M_1$ $m_1$	$M_3$ $m_3$	$M_2$ $m_2$
1		1	$M_4$ $m_4$	$M_5$ $m_5$	$M_7$ $m_7$	$M_6$ $m_6$

4 variabili a, b, c, d

		cd	00	01	11	10
		ab	00	1	3	2
00		0	$M_0$ $m_0$	$M_1$ $m_1$	$M_3$ $m_3$	$M_2$ $m_2$
01		1	$M_4$ $m_4$	$M_5$ $m_5$	$M_7$ $m_7$	$M_6$ $m_6$
11		12	$M_{12}$ $m_{12}$	$M_{13}$ $m_{13}$	$M_{15}$ $m_{15}$	$M_{14}$ $m_{14}$
10		13	$M_8$ $m_8$	$M_9$ $m_9$	$M_{11}$ $m_{11}$	$M_{10}$ $m_{10}$

Esempio

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$f_{SOP} = \bar{a}\bar{b} + \bar{a}bc + abc = \\ = \bar{a}b(\bar{c} + c) + bc(\bar{a} + a) = \\ = \bar{a}b + bc$$

$$f_{POS} = (a + b + c)(a + b + \bar{c})(\bar{a} + b + c)(\bar{a} + b + \bar{c})(\bar{a} + \bar{b} + c) = \\ = (a + b + \bar{c})(\bar{a} + (b + c)(b + \bar{c})(\bar{b} + c)) = \\ = (a + b)(\bar{a} + (b + c)(bc + \bar{b}c)) = \\ = (a + b)(\bar{a} + bc) = \\ = abc + \bar{a}b + bc = \\ = b(ac + \bar{a} + c) = \\ = b(\bar{a} + c)$$

SOP

		bc	00	01	11	10
		a	0	0	1	1
0		0	0	0	1	1
1		1	0	0	1	0

$\bar{a}b$

$bc$

POS

		bc	00	01	11	10
		a	0	1	1	0
0		0	0	0	1	1
1		1	0	0	1	0

$b$

$\bar{a} + c$

**Implicante**: insieme di **minstermini / maxtermini** che possono esprimere in forma **minima** (i cerchi in K-mappe).

**Dimensione implicanti**: 2, 4, 8.

Esempio

POS

	cd	00	01	11	10
ab					
00	0	0	1	0	
01	0	1	1	1	
11	1	1	1	1	
10	0	1	1	1	

SOP

	cd	00	01	11	10
ab					
00	0	0	1	0	0
01	0	1	1	1	1
11	1	1	1	1	1
10	0	1	1	1	1

$$f \text{ POS} = (a + b + c)(a + b + d)(a + c + d)(b + c + d)$$

$$f \text{ SOP} = ab + cd + bd + ad + bc + ac$$

Esempio

POS

	cd	00	01	11	10
ab					
00	1	1	0	1	
01	0	0	0	0	
11	0	1	0	0	
10	1	1	1	1	

SOP

	cd	00	01	11	10
ab					
00	1	1	0	1	
01	0	0	0	0	
11	0	1	0	0	
10	1	1	1	1	

$$f \text{ POS} = (a + \bar{b})(\bar{b} + d)(\bar{b} + \bar{c})(a + \bar{c} + \bar{d})$$

$$f \text{ SOP} = a\bar{b} + \bar{b}\bar{d} + \bar{b}\bar{c} + a\bar{c}\bar{d}$$

Esempio

a	b	c	d	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

POS

	cd	00	01	11	10
ab					
00	0	1	0	1	
01	1	0	1	0	
11	0	1	1	1	
10	1	1	0	1	

SOP

	cd	00	01	11	10
ab					
00	0	1	0	1	
01	1	0	1	0	
11	0	1	1	1	
10	1	1	0	1	

$$f \text{ POS} = (b + \bar{c} + \bar{d})(a + b + c + d)(\bar{a} + \bar{b} + c + d)(a + \bar{b} + c + \bar{d})(a + \bar{b} + \bar{c} + d)$$

$$f \text{ SOP} = a\bar{b}\bar{c} + \bar{b}\bar{c}\bar{d} + abd + bcd + \bar{b}cd + ac\bar{d} + \bar{a}bc\bar{d}$$

**Esercizio:** sia  $x$  un numero **BCD**, considerare la codifica  $y$  nel codice **2-SU-5**, scrivere le espressioni **minimali** per i singoli bit di  $y$ .

Número BCD 4 bit				Peso codifica 2-su-5				
$x_3$	$x_2$	$x_1$	$x_0$	6	3	2	1	0
0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	1	0	1	0	0	1
0	1	0	0	0	1	0	1	0
0	1	0	1	0	1	1	0	0
0	1	1	0	1	0	0	0	1
0	1	1	1	1	0	0	1	0
1	0	0	0	1	0	1	0	0
1	0	0	1	1	1	0	0	0
1	0	1	0	δ	δ	δ	δ	δ
1	0	1	1	δ	δ	δ	δ	δ
1	1	0	0	δ	δ	δ	δ	δ
1	1	0	1	δ	δ	δ	δ	δ
1	1	1	0	δ	δ	δ	δ	δ
1	1	1	1	δ	δ	δ	δ	δ

$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
$x_1x_0$ $x_2x_1$	$x_1x_0$ $x_2x_3$	$x_1x_0$ $x_2x_2$	$x_1x_0$ $x_2x_2$	$x_1x_0$ $x_2x_3$
00 01 11 10	00 01 11 10	00 01 11 10	00 01 11 10	00 01 11 10
00 0 0 0 0	00 0 0 1 0	00 1 0 0 1	00 1 1 0 0	00 0 1 1 1
01 0 0 1 1	01 1 1 0 0	01 0 1 0 0	01 1 0 1 0	01 0 0 0 1
11 δ δ δ δ				
10 1 1 δ δ	10 0 1 δ δ	10 1 0 δ δ	10 0 0 δ δ	10 0 0 δ δ

$$SOP \quad y_4 = x_3 + x_2 x_1$$

$$SOP \quad y_3 = x_2 \overline{x}_1 + x_3 x_0 + \overline{x}_2 x_1 x_0$$

$$SOP \quad y_2 = \overline{x_2} \overline{x_0} + x_2 \overline{x_1} x_0$$

$$SOP \quad y_1 = \overline{\overline{x_3} \overline{x_2} \overline{x_1}} + \overline{\overline{x_3} \overline{x_1} \overline{x_0}} + x_2 \overline{x_1} \overline{x_0}$$

$$SOP \quad y_0 = x_1 \bar{x}_0 +$$

$$POS \quad y_4 = (x_3 + x_2)(x_3 + x_1)$$

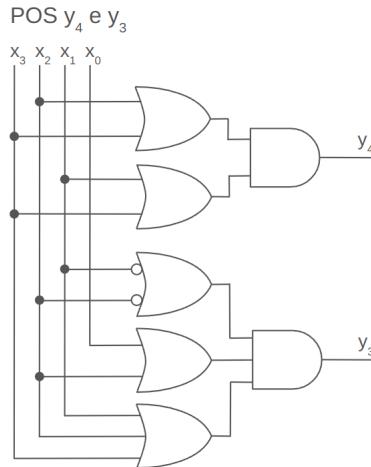
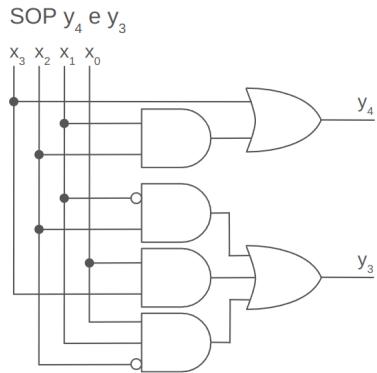
$$POS \quad y_3 = (\bar{x}_2 + \bar{x}_1)(x_2 + x_0)(x_3 + x_2 + x_1)$$

$$POS \ y_2 = (\bar{x}_2 + x_0)(x_2 + \bar{x}_0)(\bar{x}_2 + \bar{x}_1)$$

$$POS \quad y_1 = \overline{x_3}(\overline{x_1} + x_0)(x_2 + \overline{x_1})(\overline{x_2} + x_1 + \overline{x_0})$$

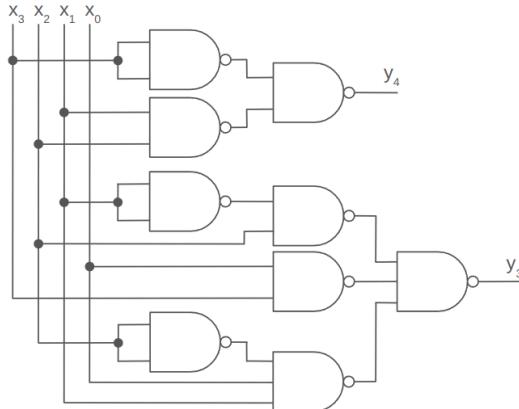
$$POS \quad y_0 = \overline{x}_2(x_1 + x_0)(\overline{x}_2 + \overline{x}_0)$$

## Alcuni disegni del circuito:



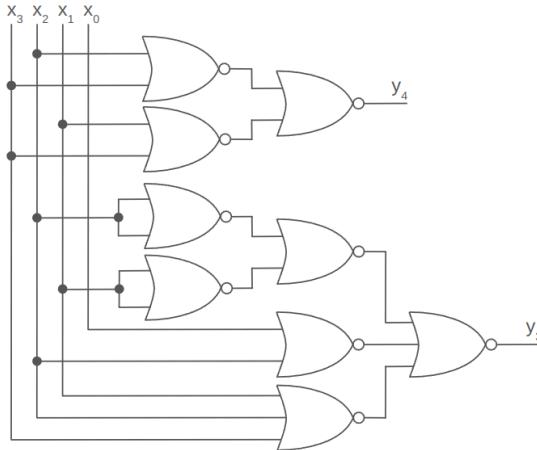
$$ALL\;NAND \quad y_4 = \overline{\overline{x_3x_3} \cdot \overline{x_2x_1}}$$

$$ALL\;NAND \quad y_3 = \overline{\overline{x_2}} \cdot \overline{x_1} \cdot \overline{x_3} \cdot \overline{x_0}$$



$$ALL\ NOR \quad y_4 = \overline{\overline{(x_3 + x_2)}} + \overline{\overline{(x_3 + x_1)}}$$

$$ALL\ NOR \quad y_3 = \overline{\overline{(x_2 + x_2)} + \overline{(x_1 + x_1)}} + \overline{(x_2 + x_0)} + \overline{(x_3 + x_2 + x_1)}$$



## Procedura di sintesi (o progettazione) di un circuito combinatorio:

- **descrizione verbale** di un problema;
- **espressione booleana**;
- **tavola di verità**;
- **espressioni booleane minimali** tramite **k-mappe**;
- **disegno del circuito**.

## Procedura di analisi di un circuito combinatorio:

- **rappresentazione di un circuito**;
- **espressioni booleane** delle **uscite**;
- **tavola di verità**;
- **descrizione verbale** caratterizzando le condizioni che danno **uscita 1 oppure 0** (dipende da quanti 1 e 0 sono presenti).

## Addizionatori:

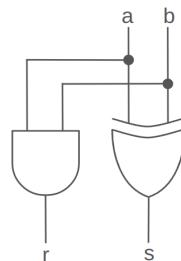
Half-Adder (2 bit in ingresso)

$$\begin{array}{r} \text{riporto} \\ \begin{array}{r} 1 \ 1 \\ 0 \ 1 \ 1 \ 0 \\ + \\ 0 \ 0 \ 1 \ 1 \\ \hline \text{somma} \end{array} \end{array}$$

		riporto	somma
a	b	r	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$s = a \oplus b$$

$$r = ab$$



Full-Adder (3 bit in ingresso)

a	b	c	r	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

		Somma				
		bc	00	01	11	10
a	bc	00	1	0	1	
		01	1	1	0	
a	bc	11	0	1	0	
		10	1	0	1	

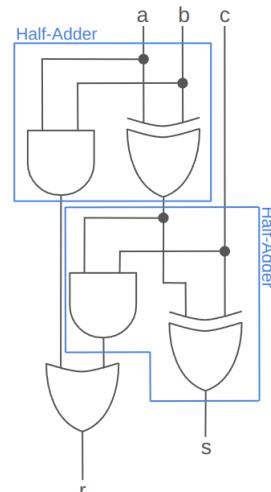
		Riporto				
		bc	00	01	11	10
a	bc	00	0	0	1	0
		01	1	1	0	1
a	bc	11	0	1	0	1
		10	1	1	1	1

Dalla mappa di Karnaugh si capisce direttamente che

$$r = ab + bc + ac = ab + c(a + b)$$

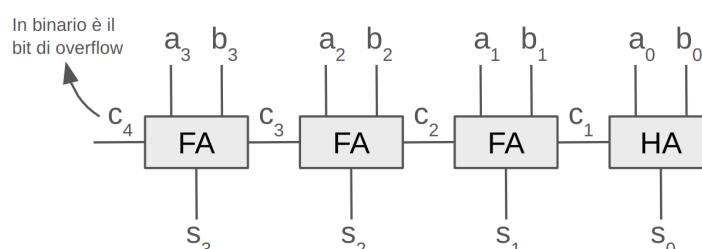
che equivale a  $\Rightarrow ab + c(a \oplus b)$

$$s = a \oplus b \oplus c$$



## Addizionatore a propagazione di riporto (Ripple Carry Adder):

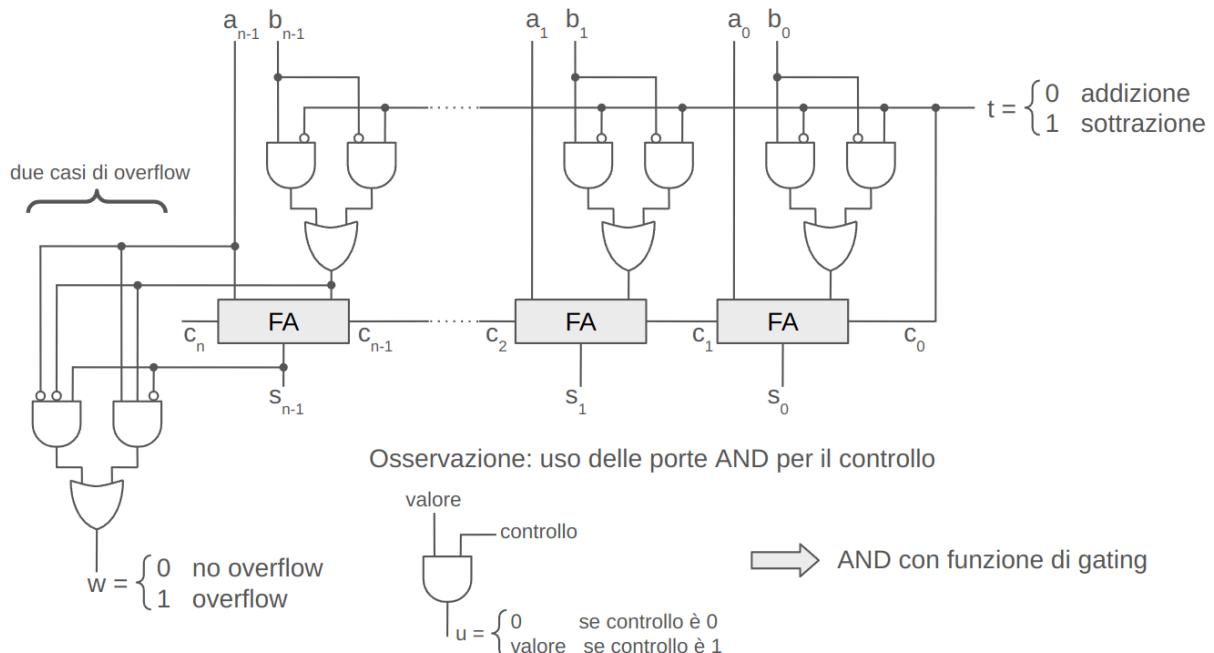
$$\begin{array}{r} \text{riporto C} \\ \begin{array}{r} c_3 \ c_2 \ c_1 \\ 1 \ 1 \ 0 \\ a_3 \ a_2 \ a_1 \ a_0 \\ 0 \ 1 \ 1 \ 0 \\ + \\ b_3 \ b_2 \ b_1 \ b_0 \\ 0 \ 0 \ 1 \ 1 \\ \hline \text{somma S} \end{array} \end{array}$$



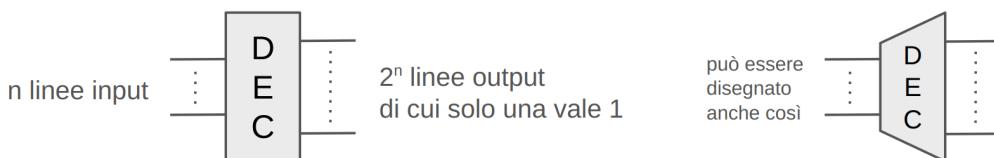
## Addizionatore a propagazione di riporto per valori in CA2:

Permette di effettuare:

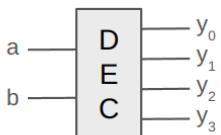
- **addizioni:** tra valori positivi / negativi;
- **sottrazioni:**  $A - B = A + (-B) = A + (\bar{B} + 1) \Rightarrow$  (in CA2).



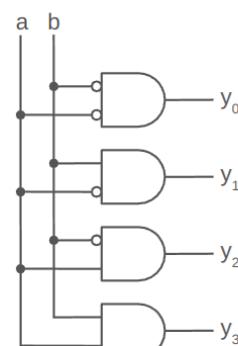
## Decodificatore:



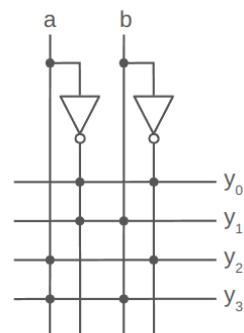
Decodificatore standard



a	b	$y_0$	$y_1$	$y_2$	$y_3$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Matrice di AND

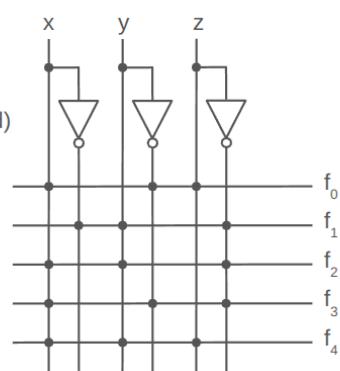


Esempio di decoder non standard

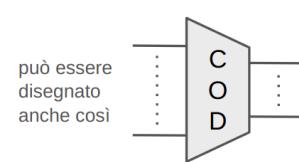
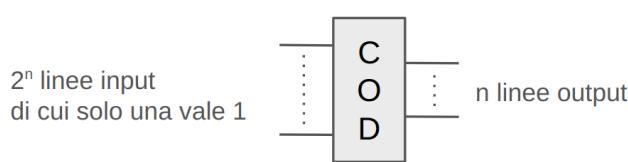
x	y	z	$f_0$	$f_1$	$f_2$	$f_3$	$f_4$
0	1	0	0	1	0	0	0
1	0	0	0	0	0	1	0
1	0	1	1	0	0	0	0
1	1	0	0	0	1	0	0
1	1	1	0	0	0	0	1

Mancano qualche combinazione di ingresso e non c'è una regola evidente.

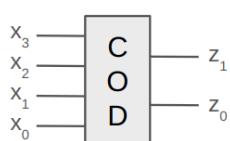
Matrice di AND  
(decodificatore non standard)



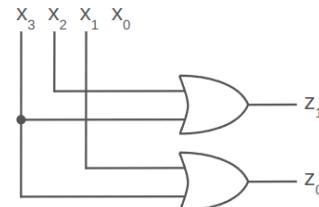
## Codificatore:



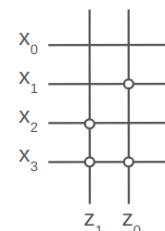
## Codificatore standard



$x_3$	$x_2$	$x_1$	$x_0$	$z_1$	$z_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



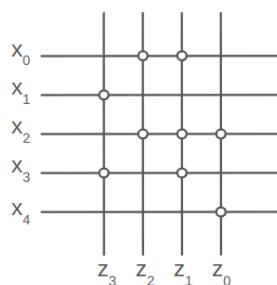
## Matrice di OR



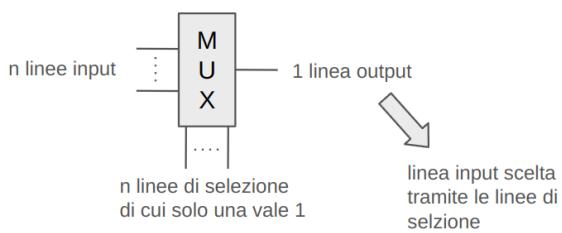
## Esempio di encoder non standard

$x_4$	$x_3$	$x_2$	$x_1$	$x_0$	$z_3$	$z_2$	$z_1$	$z_0$
0	0	0	0	1	0	1	1	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0
1	0	0	0	0	0	0	0	1

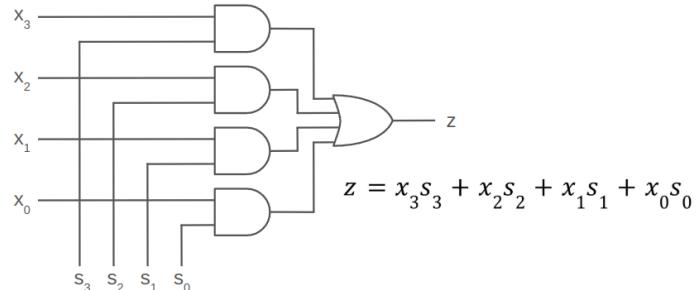
## Matrice di OR (codificatore non standard)



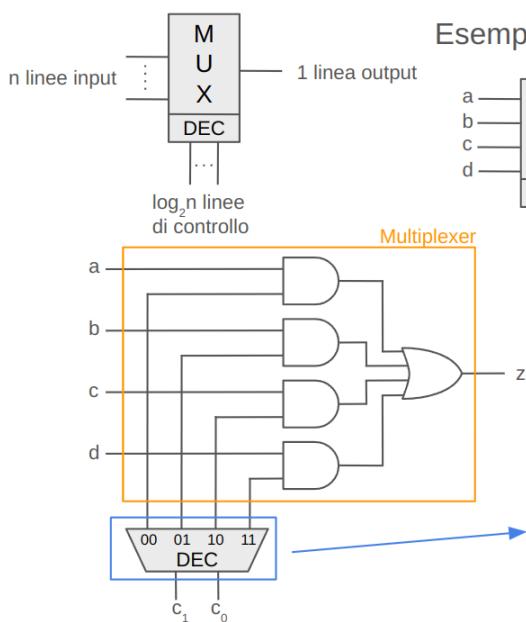
## Multiplexer:



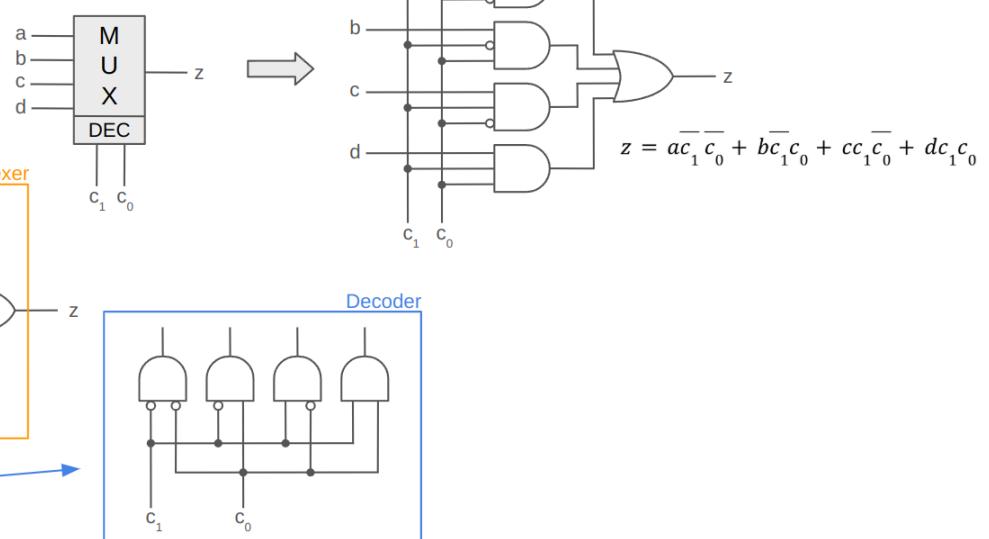
## Esempio con 4 ingressi



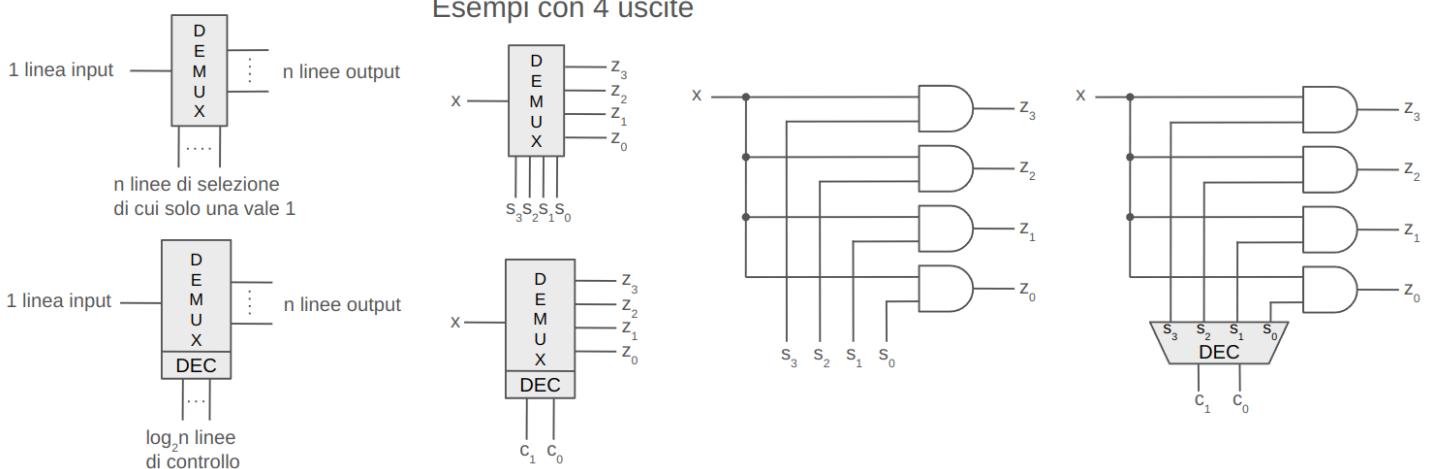
## Multiplexer con decodificatore:



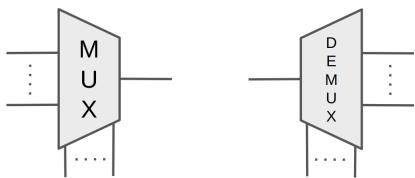
## Esempio con 4 ingressi



## Demultiplexer:



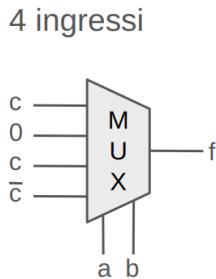
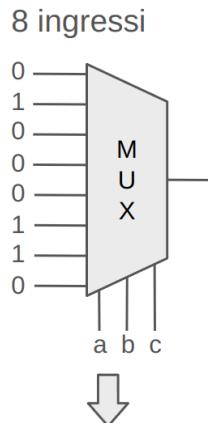
**Multiplexer e Demultiplexer possono essere disegnati anche così:**



**Uso di Multiplexer per realizzare funzioni booleane:**

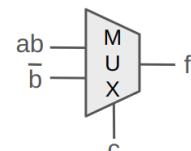
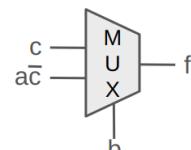
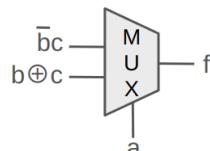
a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Ingressi del MUX = valori di f  
Linee di selezione = variabili di ingresso di f

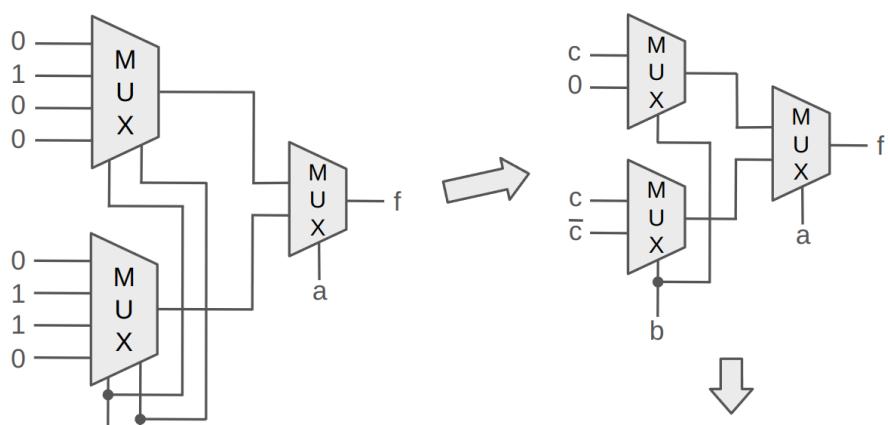


DEC: 8 AND da 3 ingressi (16 AND da 2 ingressi)  
MUX: 8 AND da 2 ingressi + 1 OR da 8 ingressi

2 ingressi

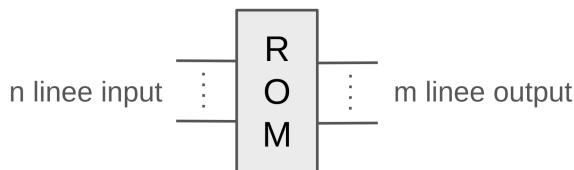


**2 livelli di Multiplexer:**



MUX:  $(2 \text{ AND da 2 ingressi} + 1 \text{ OR da 2 ingressi}) \times 3$   
 $= 6 \text{ AND da 2 ingressi} + 3 \text{ OR da 2 ingressi}$

## ROM (Read Only Memory):

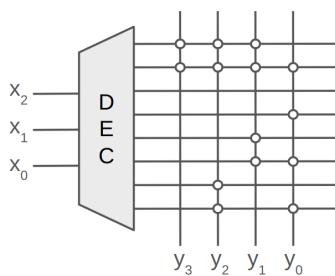


È composta da:

- **decodificatore;**
- insieme di porte OR ognuna per produrre un bit di uscita (**Matrice di OR**).

Esempio: uscita  $y = x - 2$  con 4 bit

$x_2$	$x_1$	$x_0$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	1	1	1	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	0	1
1	0	0	0	0	1	0
1	0	1	0	0	1	1
1	1	0	0	1	0	0
1	1	1	0	1	0	1



Si ottengono le formule in forma canonica

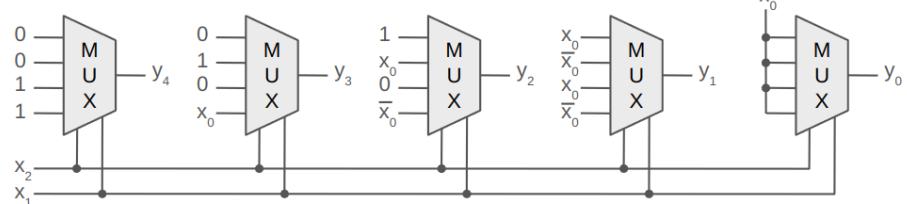
**Esercizio:** 3 linee di **ingresso**  $x_2, x_1, x_0 = x \Rightarrow 0 \leq x \leq 7$ , **uscita**  $y = 3x + 4$  con 5 bit.

Costruire:

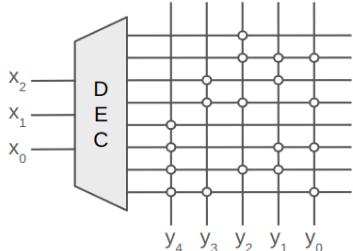
- **MUX da 4 ingressi;**
- **ROM;**
- **porte logiche minimali.**

$x_2$	$x_1$	$x_0$	$y$	$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	4	0	0	1	0	0
0	0	1	7	0	0	1	1	1
0	1	0	10	0	1	0	1	0
0	1	1	13	0	1	1	0	1
1	0	0	16	1	0	0	0	0
1	0	1	19	1	0	0	1	1
1	1	0	22	1	0	1	1	0
1	1	1	25	1	1	0	0	1

MUX da 4 ingressi (MUX 4-A-1)



ROM



Porte logiche

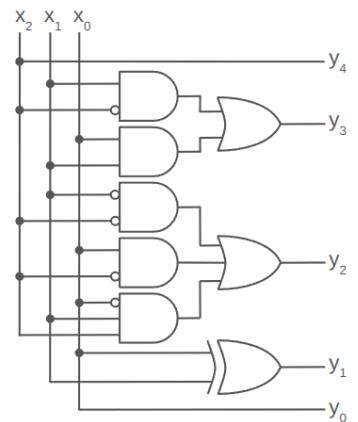
$$\begin{aligned} y_4 &= x_2 \\ y_3 &= \bar{x}_2 x_1 + x_1 x_0 \\ y_2 &= \bar{x}_2 \bar{x}_1 + \bar{x}_2 x_0 + x_2 x_1 \bar{x}_0 \\ y_1 &= x_1 \oplus x_0 = \bar{x}_1 x_0 + x_1 \bar{x}_0 \\ y_0 &= x_0 \end{aligned}$$

$x_2$	$x_1$	$x_0$	$y_3$	00	01	11	10
0	0	0	0	0	0	1	1
1	0	0	1	0	0	1	0

$x_2$	$x_1$	$x_0$	$y_2$	00	01	11	10
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1

$x_2$	$x_1$	$x_0$	$y_1$	00	01	11	10
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1

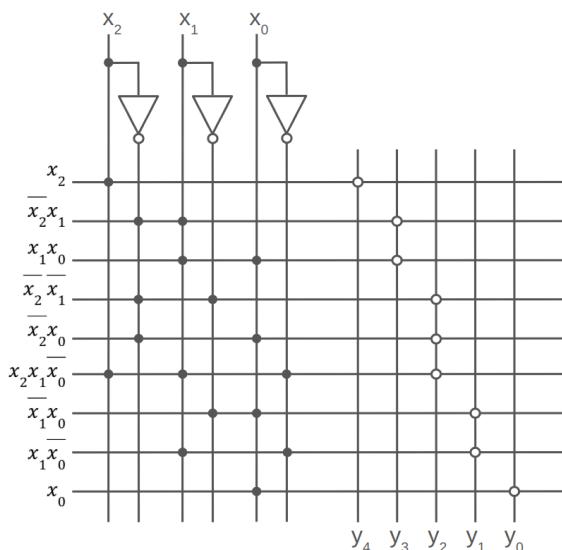
$x_2$	$x_1$	$x_0$	$y_0$	00	01	11	10
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1



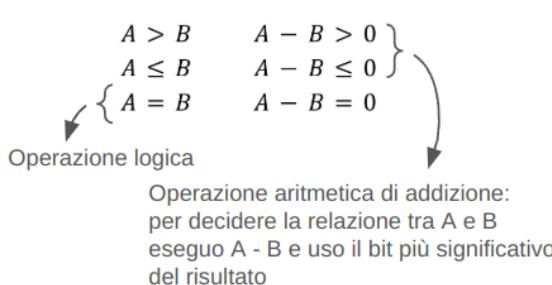
## PLA (Programmable Logic Array):

- **circuito integrato:** Matrice di AND + Matrice di OR;
- usato per realizzare insiemi di funzioni usando **E.B. SOP minimali**;
- **caratteristiche:**
  - **n** ingressi;
  - **m** uscite;
  - **p** termini prodotto (porte AND nella Matrice di AND).

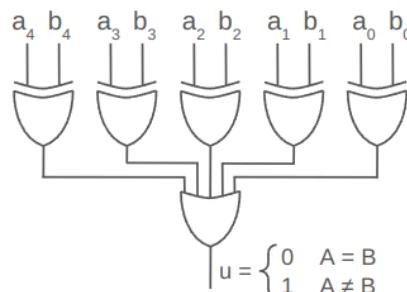
### PLA dell'esercizio precedente:



### Comparatore:

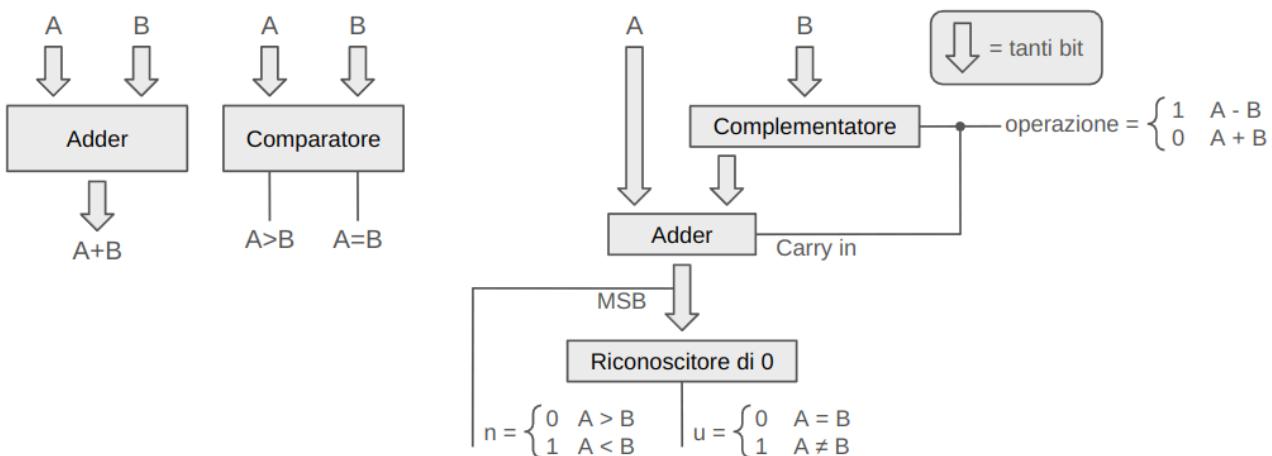


### Comparatore di uguaglianza (comparatore logico)



è veloce  
(esegue tutte le XOR insieme)

### Comparatore aritmetico



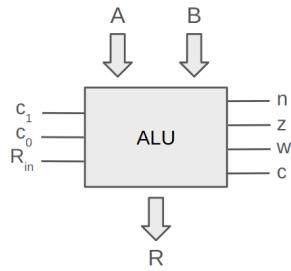
### ALU (Arithmetic Logic Unit):

#### Bit prodotti dall'ALU (sempre):

- **n**: risultato negativo;
- **z**: risultato è zero;
- **w**: overflow;
- **c**: carry in uscita.

#### Esempio di ALU con 3 bit in input:

- **c<sub>1</sub>**, pone a 0 operando A;
- **c<sub>0</sub>**, complemento logico di B;
- **R<sub>in</sub>**, riporto in entrata al sommatore.



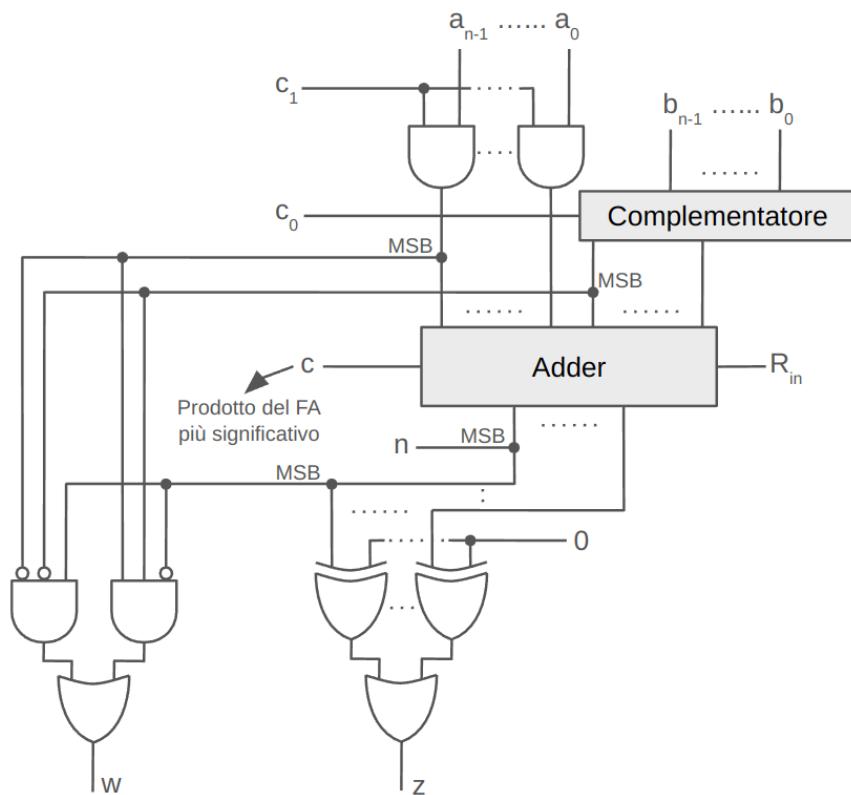
$$c_1 = \begin{cases} 1 & \text{passa A} \\ 0 & \text{pone a 0 A} \end{cases}$$

$$c_0 = \begin{cases} 1 & \text{complemento B} \\ 0 & \text{passa B} \end{cases}$$

$$R_{in} = \begin{cases} 0 & \text{riporto ingresso è 0} \\ 1 & \text{riporto ingresso è 1} \end{cases}$$

All'interno dell'ALU è presente un Adder

<b>c<sub>1</sub></b>	<b>c<sub>0</sub></b>	<b>R<sub>in</sub></b>	<b>risultato</b>
0	0	0	0+B=B
0	0	1	B+1
0	1	0	$\bar{B}$
0	1	1	$\bar{B}+1=-B$
1	0	0	A+B
1	0	1	A+B+1
1	1	0	A+ $\bar{B}$
1	1	1	A-B

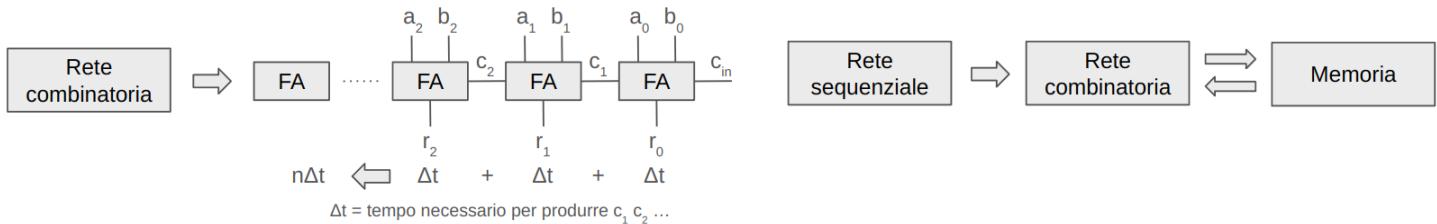


# CIRCUITI SEQUENZIALI E AUTOMI A STATI FINITI

## Reti sequenziali:

introduciamo il concetto di **tempo**:

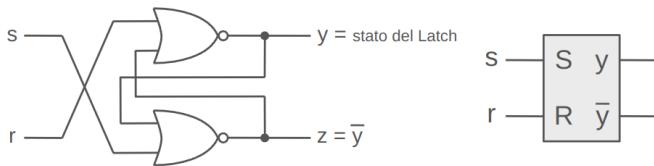
- **riduce** il numero di porte di circuiti combinatori;
- circuiti per la **memorizzazione**.



## Latch - cella di memoria:

- linea di **feed-back**;
- **memorizzare** un singolo bit.

## Latch SR (Set-Reset):



Stato attuale (tempo t)  $y = y(t)$

Stato futuro (tempo t+1)  $y' = Y = y(t + 1)$

(il tempo dipende dal tempo di attraversamento delle due porte NOR)

$$Y = \overline{\overline{r} + z} = \overline{\overline{r} + \overline{s} + y} = \overline{r}(s + y)$$

s	r	$Y = y(t+1)$
0	0	memorizzazione
0	1	reset
1	0	set
1	1	—

No stabilità nelle uscite  
y rappresenta il bit memorizzato

## Analisi dei Latch:

Memorizzazione

$$\begin{array}{lll} r = s = 0 & y = 1 & Y = \overline{r}(s + y) = 1(0 + 1) = 1 \\ r = s = 0 & y = 0 & Y = \overline{r}(s + y) = 1(0 + 0) = 0 \end{array}$$

Reset

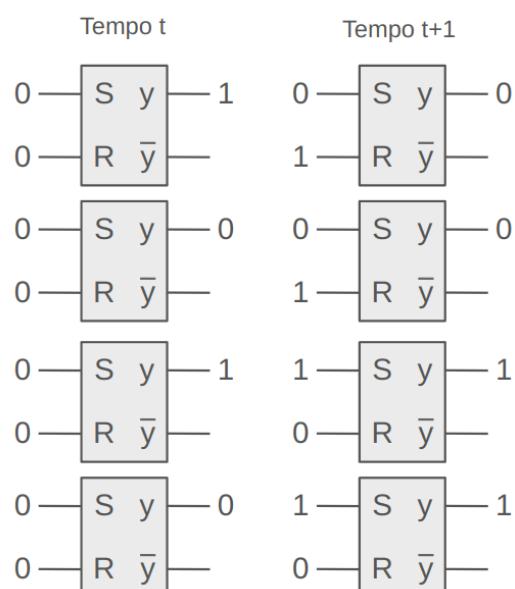
$$\begin{array}{lll} \text{Tempo t} & r = s = 0 & y = 1 \\ \text{Tempo t+1} & r = 1 \quad s = 0 & Y = \overline{r}(s + y) = 0(0 + 1) = 0 \end{array}$$

$$\begin{array}{lll} \text{Tempo t} & r = s = 0 & y = 0 \\ \text{Tempo t+1} & r = 1 \quad s = 0 & Y = \overline{r}(s + y) = 0(0 + 0) = 0 \end{array}$$

Set

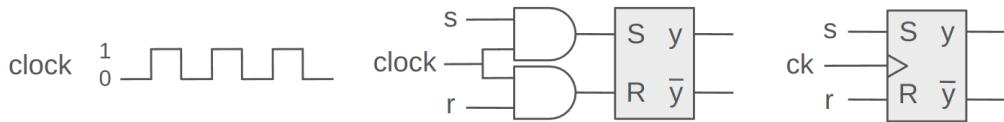
$$\begin{array}{lll} \text{Tempo t} & r = s = 0 & y = 1 \\ \text{Tempo t+1} & r = 0 \quad s = 1 & Y = \overline{r}(s + y) = 1(1 + 1) = 1 \end{array}$$

$$\begin{array}{lll} \text{Tempo t} & r = s = 0 & y = 0 \\ \text{Tempo t+1} & r = 0 \quad s = 1 & Y = \overline{r}(s + y) = 1(1 + 0) = 1 \end{array}$$

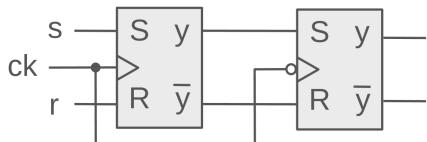


### Latch sincrono (Gated Latch):

- segnale orologio (**clock**):
  - segnale **periodico**;
  - usato per **sincronizzare**;
  - porte **AND** lasciano passare segnale quando vale **1**.



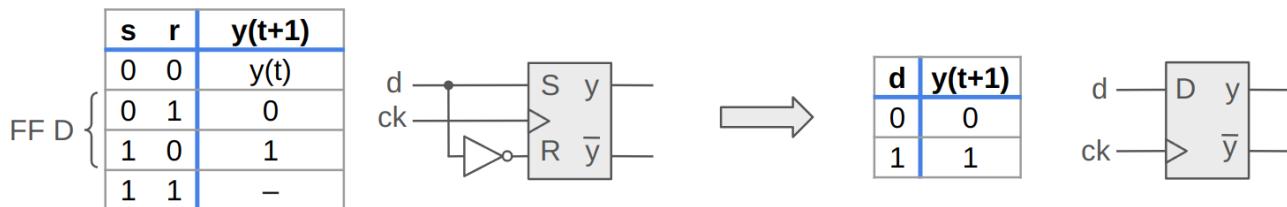
### Master Slave Latch:



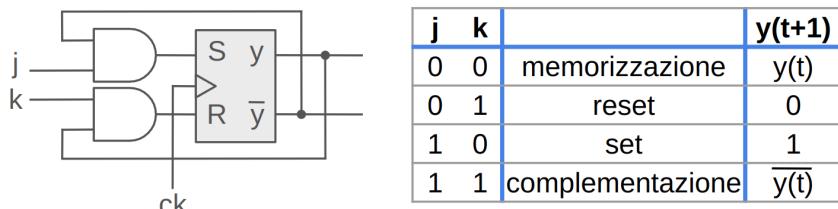
**Flip-Flop**: quando esiste il **clock**.

I latch SR precedenti che hanno il segnale clock possono essere chiamati anche **FF SR** (Flip-Flop Set-Reset).

### Flip-Flop D (Delay):

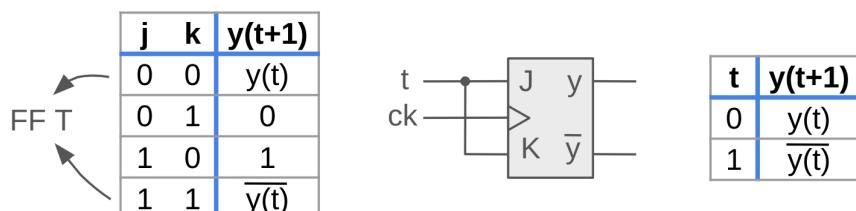


### Flip-Flop JK:

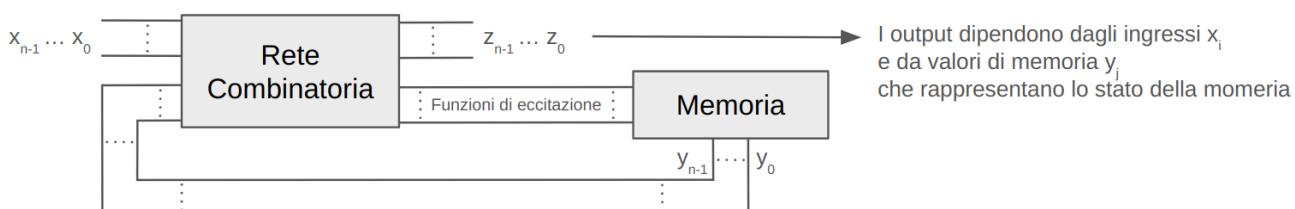


$$\begin{aligned}
 y(t + 1) &= \bar{r}(s + y) = (\bar{k}\bar{y})(j\bar{y} + y) = \\
 &= (\bar{k} + \bar{y})(j\bar{y} + y) = j\bar{k}\bar{y} + \bar{k}y + jy\bar{y} + \bar{y}y = \\
 &= jy + \bar{k}y
 \end{aligned}$$

### Flip-Flop T (Toggle):



### Reti sequenziali:

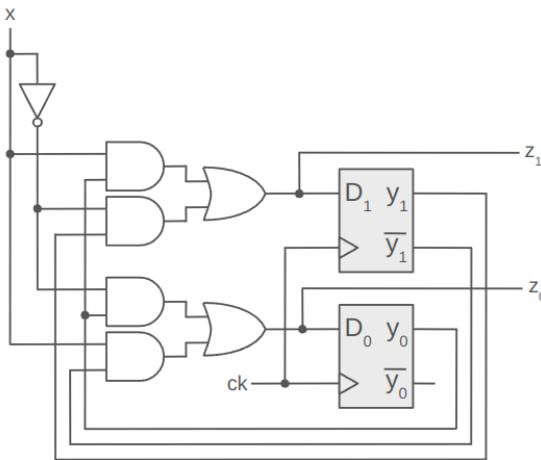


## Analisi di reti sequenziali:

data la rete sequenziale:

- **espressioni booleane** delle funzioni di **eccitazione** (ingressi dei FF) e delle **uscite**;
- **tavola degli stati futuri**, che contiene:
  - **input** ( $x_i$ ): tutte le combinazioni;
  - **stati dei FF** ( $y_j(t)$ ): espressioni booleane;
  - **uscite** ( $z$ ): espressioni booleane;
  - **stati futuri degli FF** ( $y_1(t+1)$ ): ricavo usando le tavole dei FF;
- **diagramma della rete** tramite automa a stati finiti (valori binari);
- **diagramma della "macchina" con astrazione** dai valori binari (automa a stati finiti);
- **minimizzazione** dell'automa;
- **descrizione verbale**.

## Esempio analisi di rete sequenziale:



Espressioni booleane

**Funzioni di eccitazione:**

- $d_1 = xy_0 + \bar{x}y_1$
- $d_0 = \bar{xy}_0 + x\bar{y}_1$

**Uscite:**

- $z_1 = d_1$
- $z_0 = d_0$

Tavola stati futuri

x	$y_1$	$y_0$	<b>Funzione di eccitazione</b>		<b>Uscite</b>		stato futuro	
			$d_1$	$d_0$	$z_1$	$z_0$	$Y_1$	$Y_0$
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0
0	1	1	1	1	1	1	1	1
1	0	0	0	1	0	1	0	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0
1	1	1	1	0	1	0	1	0

stato attuale

<b>d</b>	<b>Y</b>
0	0
1	1

Diagramma della rete (automa a stati finiti)

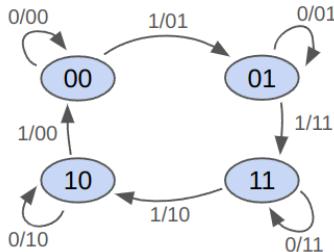
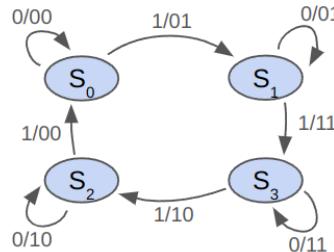


Diagramma della macchina

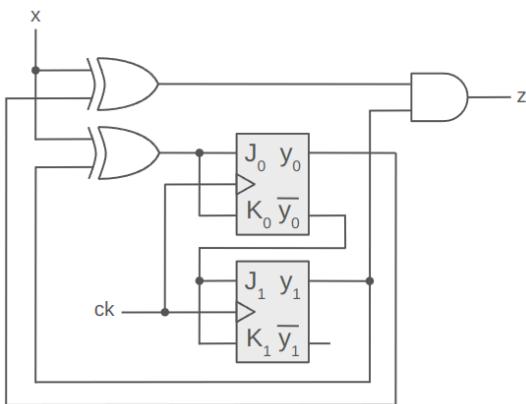
$00 \rightarrow S_0$   
 $01 \rightarrow S_1$   
 $10 \rightarrow S_2$   
 $11 \rightarrow S_3$



Descrizione verbale

Contatore di input uguali a 1  
Modulo 4 (da 0 a 3)

### Esempio analisi di rete sequenziale:



Espressioni booleane

$$\begin{aligned}j_0 &= k_0 = x \oplus y_1 \\j_1 &= k_1 = \bar{y}_0 \\z &= (x \oplus y_0) y_1\end{aligned}$$

Tavola stati futuri

		stato futuro							
x	y <sub>1</sub>	y <sub>0</sub>	j <sub>1</sub>	k <sub>1</sub>	j <sub>0</sub>	k <sub>0</sub>	z	Y <sub>1</sub>	Y <sub>0</sub>
0	0	0	1	1	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1
0	1	0	1	1	1	1	0	0	1
0	1	1	0	0	1	1	1	1	0
1	0	0	1	1	1	1	0	1	1
1	0	1	0	0	1	1	0	0	0
1	1	0	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0	1	1

j	k	Y
0	0	y
0	1	0
1	0	1
1	1	y

Diagramma della rete (automa a stati finiti)

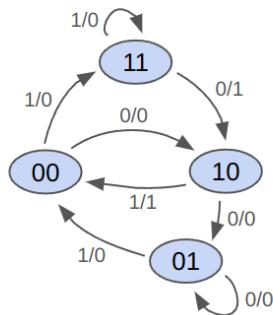


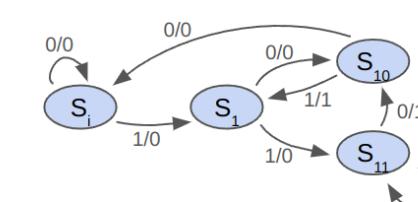
Diagramma della macchina

altri modi per chiamare	
01 → S <sub>i</sub>	stato iniziale A S <sub>0</sub>
00 → S <sub>1</sub>	B S <sub>1</sub>
10 → S <sub>10</sub>	C S <sub>2</sub>
11 → S <sub>11</sub>	D S <sub>3</sub>

Descrizione verbale

Riconoscitore di 101 e 110 con sovrapposizione

Input  
Output



Modello di Mealy

### Automi a stati finiti con output:

- automi con numero finito di stati;
- abbiamo bisogno di output.

$\Sigma$  alfabeto finito ingresso - **input**

$Q$  insieme finito di stati - **memoria** (rappresentazione binaria della memoria contenuta)

$\delta$  funzione di transizione - **archi**:

$$\delta : \Sigma \times Q \rightarrow Q$$

$$(0, 01) \rightarrow 01$$

$U$  alfabeto finito di uscita

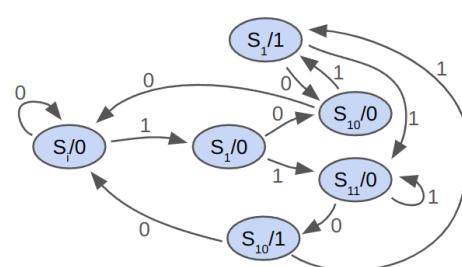
$\lambda$  funzione di uscita - modello di **Mealy**       $\lambda : \Sigma \times Q \rightarrow U$

$\lambda'$  funzione di uscita - modello di **Moore**       $\lambda' : Q \rightarrow U$

### Modello di Moore (dell'esercizio precedente):

	0	1	input
S <sub>i</sub>	S <sub>i</sub> / 0	S <sub>1</sub> / 0	
S <sub>1</sub>	S <sub>10</sub> / 0	S <sub>11</sub> / 0	stati futuro / uscita
S <sub>10</sub>	S <sub>i</sub> / 0	S <sub>1</sub> / 1	
S <sub>11</sub>	S <sub>10</sub> / 1	S <sub>11</sub> / 0	

Diagramma secondo Moore

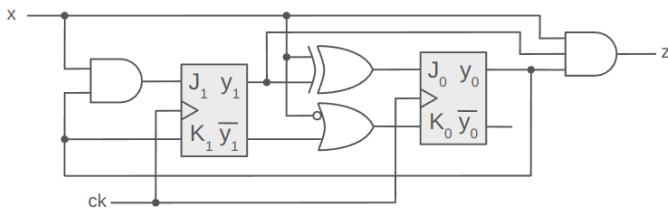


Modello di Moore ha più stati rispetto a quello di Mealy

Passo da uno all'altro attraverso tabella dell'automa



### Disegno con FF JK



### Variante con FF D

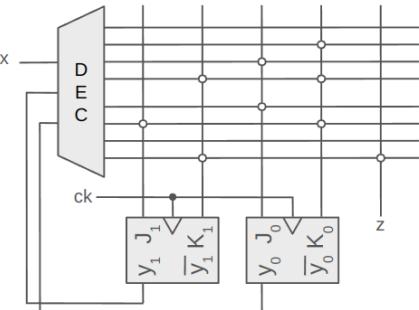
$d_1$	$y_1 y_0$	00	01	11	10
0	x	0	0	0	1
1	0	1	0	1	1

$$d_1 = y_1 \bar{y}_0 + x y_1 y_0$$

$d_0$	$y_1 y_0$	00	01	11	10
0	x	0	0	0	1
1	1	1	0	1	0

$$d_0 = \bar{x} y_1 \bar{y}_0 + x y_1 y_0 + \bar{x} y_1 \bar{y}_0$$

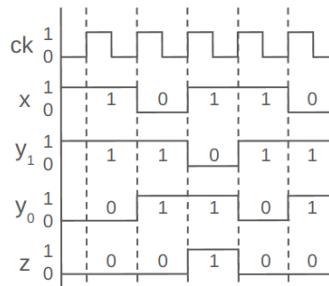
### Variante con parte combinatoria realizzata con ROM (JK)



### Diagramma temporale

- Compaiono:
- clock
  - gli input (bit)
  - i bit di stati
  - i bit di uscita

A fronte della sequenza di ingresso 10110110 a partire dallo stato 10



**Stati equivalenti di un automa:** due stati si dicono **equivalenti** se per ogni possibile valore di ingresso transitano nello **stesso stato successivo** e producono la **stessa uscita**.

### Esempio:

	a	b
$S_1$	$S_2 / 0$	$S_6 / 0$
$S_2$	$S_7 / 0$	$S_3 / 1$
$S_3$	$S_4 / 0$	$S_3 / 1$
$S_4$	$S_3 / 0$	$S_7 / 0$
$S_5$	$S_8 / 0$	$S_6 / 0$
$S_6$	$S_3 / 0$	$S_7 / 0$
$S_7$	$S_7 / 0$	$S_5 / 0$
$S_8$	$S_7 / 0$	$S_3 / 1$

- $S'_1 = \{S_1, S_5\}$
- $S'_2 = \{S_2, S_8\}$
- $S'_4 = \{S_4, S_6\}$
- $S'_3 = \{S_3, S_7\}$
- $S'_7 = \{S_7\}$
- $S'_5 = \{S_5\}$
- $S'_6 = \{S_6\}$

### Automa minimo

	a	b
$S'_1$	$S_2' / 0$	$S_4' / 0$
$S'_2$	$S_7 / 0$	$S_3 / 1$
$S'_3$	$S_4' / 0$	$S_3 / 1$
$S'_4$	$S_3 / 0$	$S_7 / 0$
$S'_7$	$S_7 / 0$	$S_1' / 0$

### Tabella triangolare

$S_2$	X	<del>07</del>
$S_3$	<del>23</del>	X
$S_4$	<del>23</del>	X
$S_5$	28	X X
$S_6$	<del>23</del>	X X
$S_7$	<del>27</del>	X X
$S_8$	X	<del>07</del> X X X X X
	$S_1$	$S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$

(se almeno una coppia presenta una non equivalente, non è necessario guardare le altre)

**Tabella triangolare:** per confrontare gli stati possiamo utilizzare una tabella triangolare.

Nelle **celle** riporto:

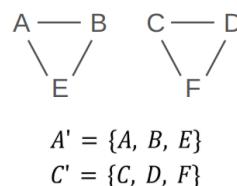
- **X:** se gli stati non sono equivalenti;
- **nulla:** se gli stati sono equivalenti;
- **nome delle coppie di stati per ogni possibile input:** se gli output sono uguali (per ogni possibile input).

### Esempio:

	0	1
<b>A</b>	G / 00	C / 01
<b>B</b>	G / 00	D / 01
<b>C</b>	D / 10	A / 01
<b>D</b>	C / 10	B / 01
<b>E</b>	G / 00	F / 01
<b>F</b>	F / 10	E / 01
<b>G</b>	A / 01	F / 11

### Tabella triangolare

B	CD
<b>C</b>	X X
<b>D</b>	X X AB
<b>E</b>	CF DF X X
<b>F</b>	X X DF AE CF BE X
<b>G</b>	X X X X X X
	<b>A</b> <b>B</b> <b>C</b> <b>D</b> <b>E</b> <b>F</b>



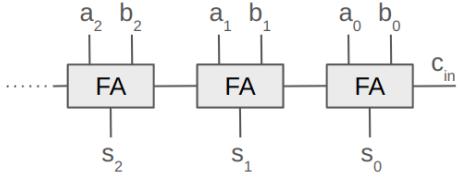
### Automa minimo

	0	1
<b>A'</b>	G / 00	C' / 01
<b>C'</b>	C' / 10	A' / 01
<b>G'</b>	A' / 01	C' / 11

$$A' = \{A, B, E\}$$

$$C' = \{C, D, F\}$$

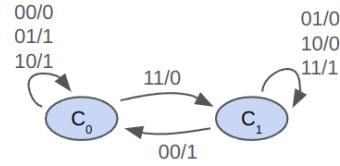
## Sintesi dell'addizionatore sequenziale:



Input  
Output  
 $a_i$   
 $b_i$   
 $s_i$   
 $C_0$  riporto 0  
 $C_1$  riporto 1

Automa

$y \backslash ab$	00	01	10	11
$C_0$	$C_0/0$	$C_0/1$	$C_0/1$	$C_1/0$
$C_1$	$C_0/1$	$C_1/0$	$C_1/0$	$C_1/1$



Codifica stati

$$\begin{aligned} C_0 &= 0 \\ C_1 &= 1 \end{aligned}$$

Tavola degli stati futuri

$y$	<b>a</b>	<b>b</b>	<b>y</b>	<b>Y</b>	<b>s</b>	<b>j</b>	<b>k</b>
	0	0	0	0	0	0	$\delta$
	0	0	1	0	1	$\delta$	1
	0	1	0	0	1	0	$\delta$
	0	1	1	1	0	$\delta$	0
	1	0	0	0	1	0	$\delta$
	1	0	1	1	0	$\delta$	0
	1	1	0	1	0	1	$\delta$
	1	1	1	1	1	$\delta$	0

Espressioni booleane

$s$	$y \backslash ab$	0	1
00	0	1	
01	1	0	
11	0	1	
10	1	0	

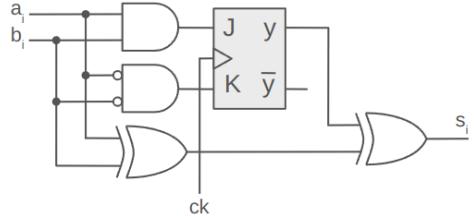
$j$	$y \backslash ab$	0	1
00	0	1	
01	0	0	$\delta$
11	1	0	
10	0	0	$\delta$

$k$	$y \backslash ab$	0	1
00	0	1	
01	$\delta$	0	
11	$\delta$	0	
10	$\delta$	0	

$$s = a \oplus b \oplus y$$

$$j = ab \quad k = \bar{a} \bar{b}$$

Disegno con FF JK



**Esercizio:** automa che riceve in **ingresso O, S, T** e produce in **uscita** (con **sovraposizioni**):

- **a**: se riconosce **STO**;
- **b**: se riconosce **OTO**;
- **c**: se riconosce **OST**;
- **0**: altrimenti.

	<b>O</b>	<b>S</b>	<b>T</b>
$Q_{in}$	$Q_O/0$	$Q_S/0$	$Q_{in}/0$
$Q_O$	$Q_O/0$	$Q_{OS}/0$	$Q_{OT}/0$
$Q_S$	$Q_O/0$	$Q_S/0$	$Q_{ST}/0$
$Q_{OS}$	$Q_O/0$	$Q_S/0$	$Q_{ST}/c$
$Q_{OT}$	$Q_O/b$	$Q_S/0$	$Q_{in}/0$
$Q_{ST}$	$Q_O/a$	$Q_S/0$	$Q_{in}/0$

$Q_O$	$Q_S$	$Q_{OS}$	$Q_{OT}$	$Q_{ST}$
$Q_O$	X	X	X	
$Q_S$	X	X	X	
$Q_{OS}$	X	X	X	
$Q_{OT}$	X	X	X	X
$Q_{ST}$	X	X	X	X
$Q_{in}$	$Q_O$	$Q_S$	$Q_{OS}$	$Q_{OT}$

Tabella triangolare

L'automa è già minimo

Codifica

3 bit per stati (ci sono 6 stati)  
2 bit per ingressi  
2 bit per uscite

**Versione semplificata dell'esercizio precedente:** riconoscere **STO** e **OTO** con **uscita 1** con **sovraposizioni**.

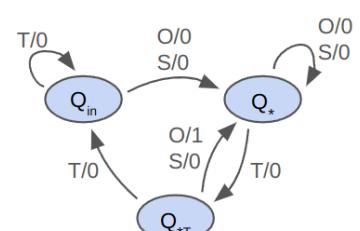
	<b>O</b>	<b>S</b>	<b>T</b>
$Q_{in}$	$Q_O/0$	$Q_S/0$	$Q_{in}/0$
$Q_O$	$Q_O/0$	$Q_S/0$	$Q_{OT}/0$
$Q_S$	$Q_O/0$	$Q_S/0$	$Q_{ST}/0$
$Q_{OT}$	$Q_O/1$	$Q_S/0$	$Q_{in}/0$
$Q_{ST}$	$Q_O/1$	$Q_S/0$	$Q_{in}/0$

Tabella triangolare

$Q_O$	$Q_S$	$Q_{OT}$	$Q_{ST}$
$Q_O$	X	X	
$Q_S$	X	X	
$Q_{OT}$	X	X	X
$Q_{ST}$	X	X	X
$Q_{in}$	$Q_O$	$Q_S$	$Q_{OT}$

Automa minimo

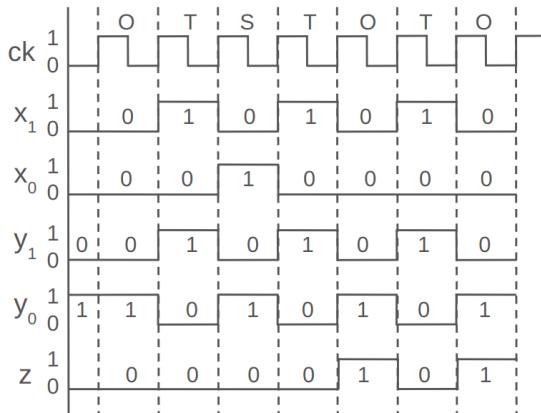
	<b>O</b>	<b>S</b>	<b>T</b>
$Q_{in}$	$Q_*/0$	$Q_*/0$	$Q_{in}/0$
$Q_*$	$Q_*/0$	$Q_*/0$	$Q_{*T}/0$
$Q_{*T}$	$Q_*/1$	$Q_*/0$	$Q_{in}/0$



## Codifica input Tavola stati futuri

	$x_1$	$x_0$
O	0	0
S	0	1
T	1	0

Diagramma temporale a partire dallo stato codificato con 01 e sequenza OTSTOTO



## Espressioni booleane

$j_1$	$y_1 y_0$	00	01	11	10
$x_1 x_0$	00	0	0	$\delta$	$\delta$
01	01	0	0	$\delta$	$\delta$
11	11	$\delta$	$\delta$	$\delta$	$\delta$
10	10	0	1	$\delta$	$\delta$

$y_1 y_0$	00	01	11	10
$x_1 x_0$	00	0	0	$\delta$
	01	0	0	$\delta$
	11	$\delta$	$\delta$	$\delta$
	10	0	0	$\delta$

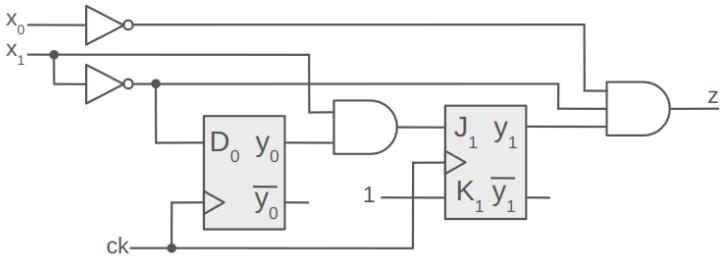
$$z = \overline{x_1} \overline{x_0} y_1$$

$$j_1 = x_1 y_0$$

$$k_1 =$$

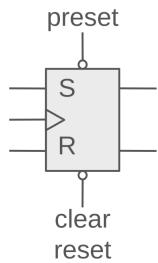
$$d_0 = \overline{x}_1$$

## Disegno con FF D e FF JK



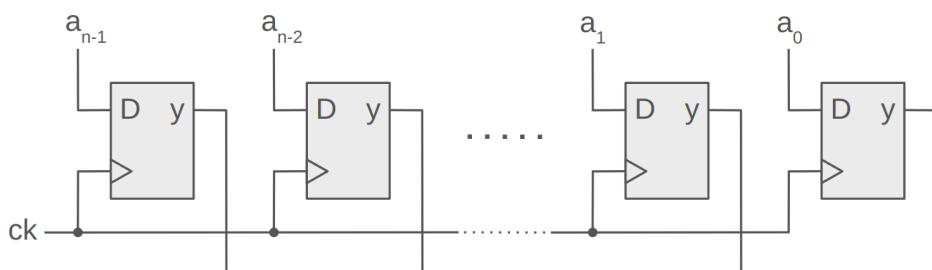
## Registri:

- insiemi di  $FF$ ;
  - $FF$  con *ingressi asincroni*.

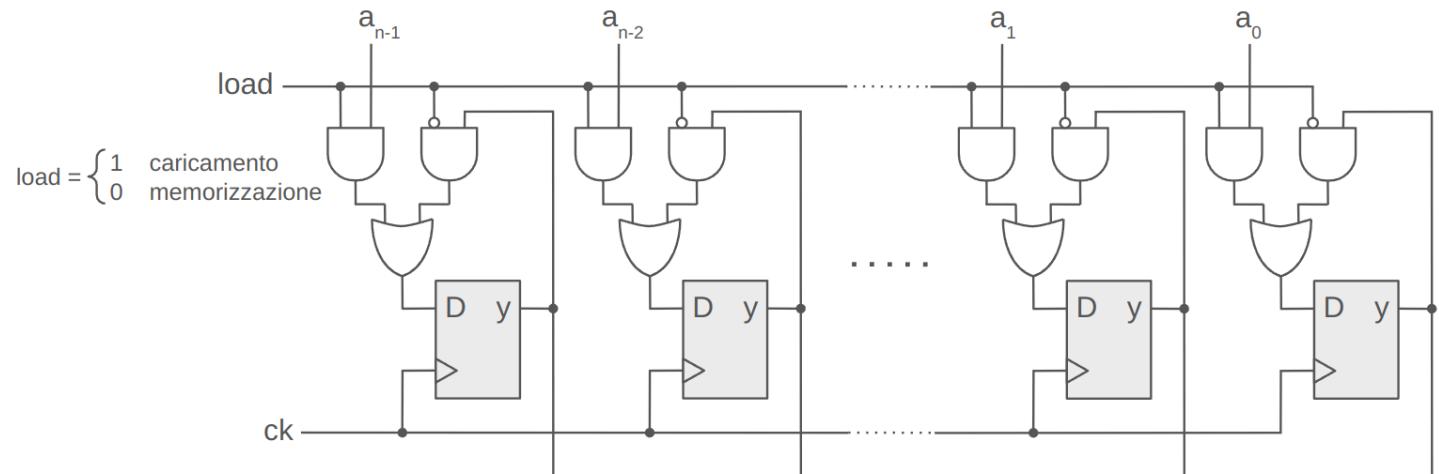


## Registri paralleli:

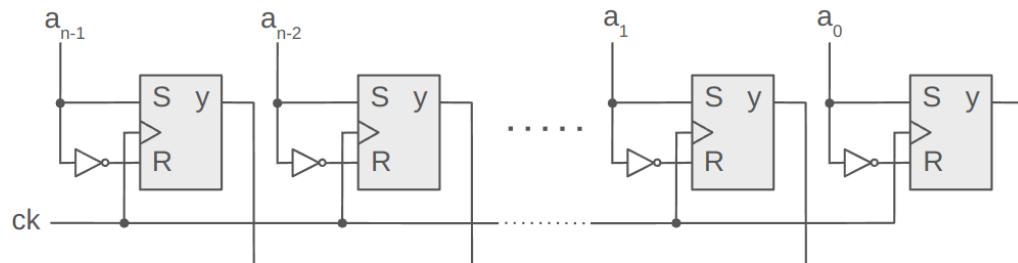
Caricamento parallelo (PIPO - Parallel Input Parallel Output)



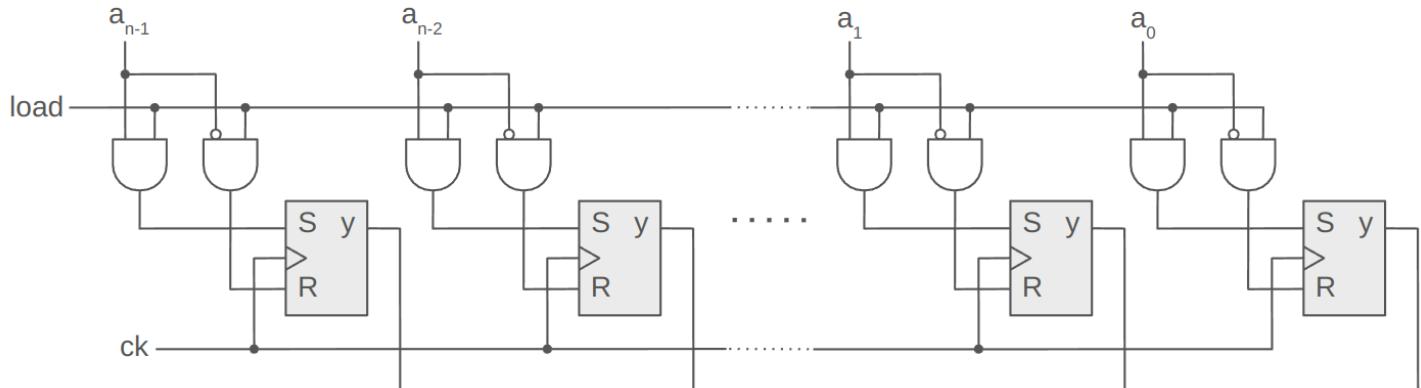
## Caricamento e memorizzazione



## Caricamento con FF SR

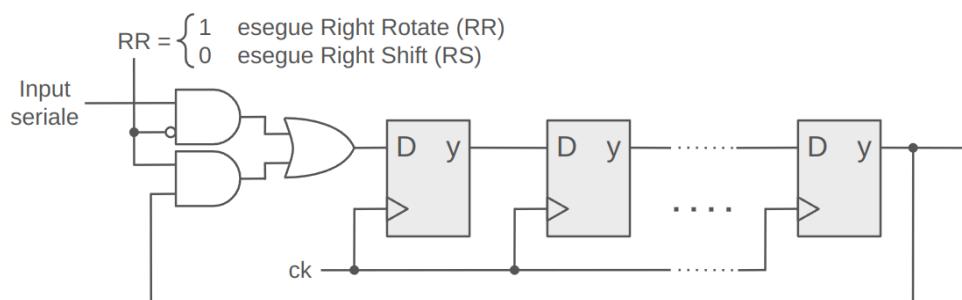


## Caricamento e memorizzazione con FF SR

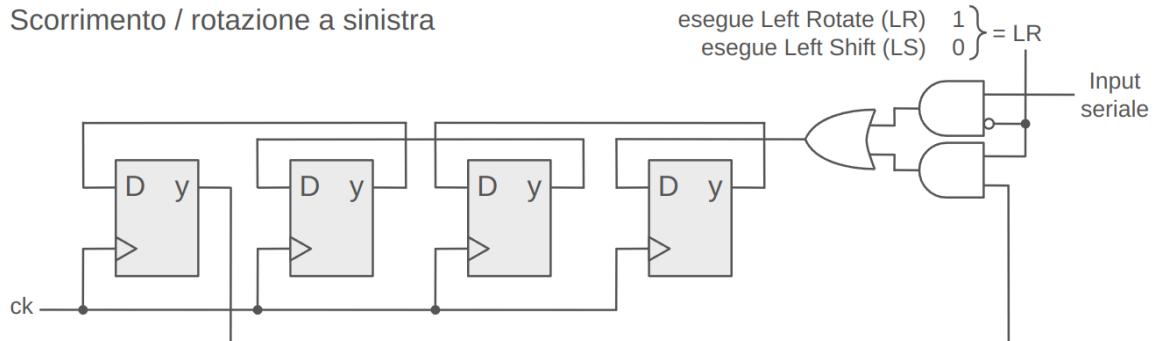


## Registri seriali / Registri a scorrimento (Shift Register):

Scorrimento / rotazione a destra



Scorrimento / rotazione a sinistra



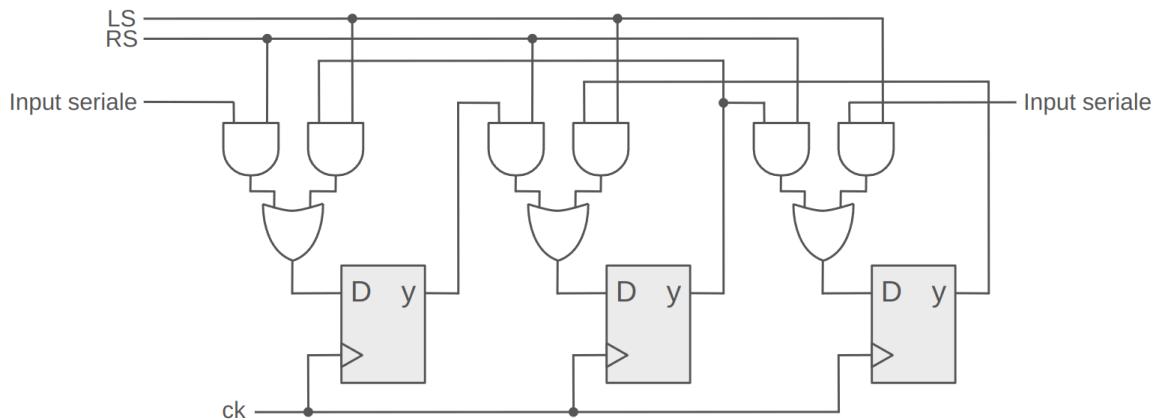
esegue Left Rotate (LR)  
esegue Left Shift (LS)

$\begin{matrix} 1 \\ 0 \end{matrix}$

= LR

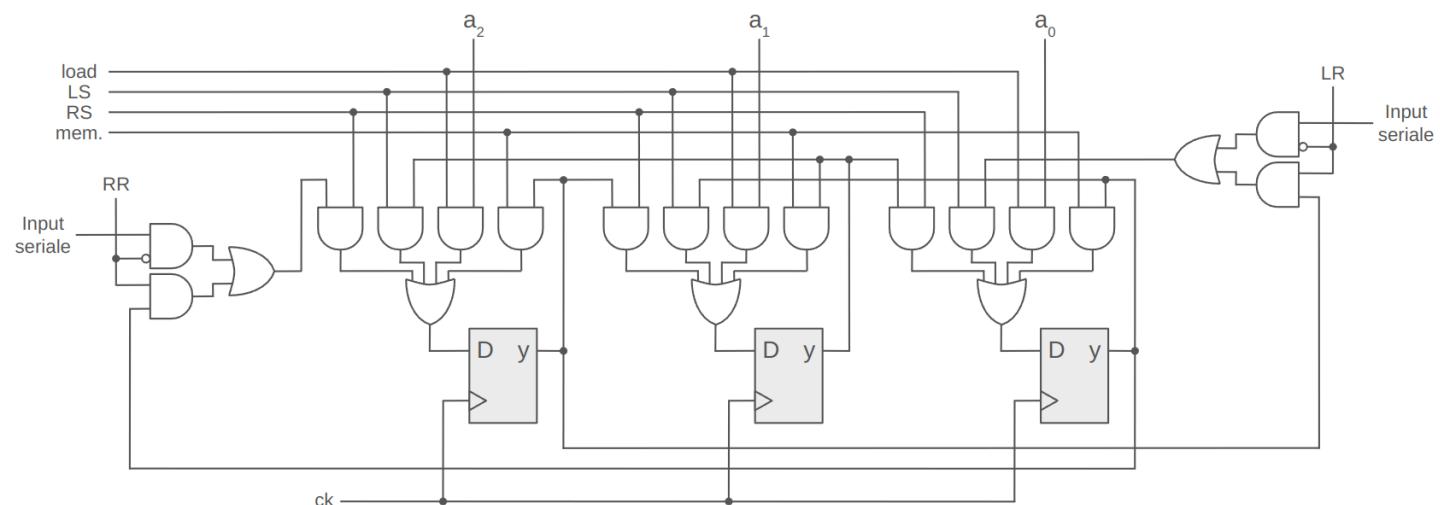
Input seriale

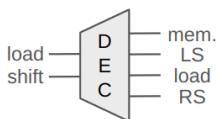
Scorrimento a destra e a sinistra



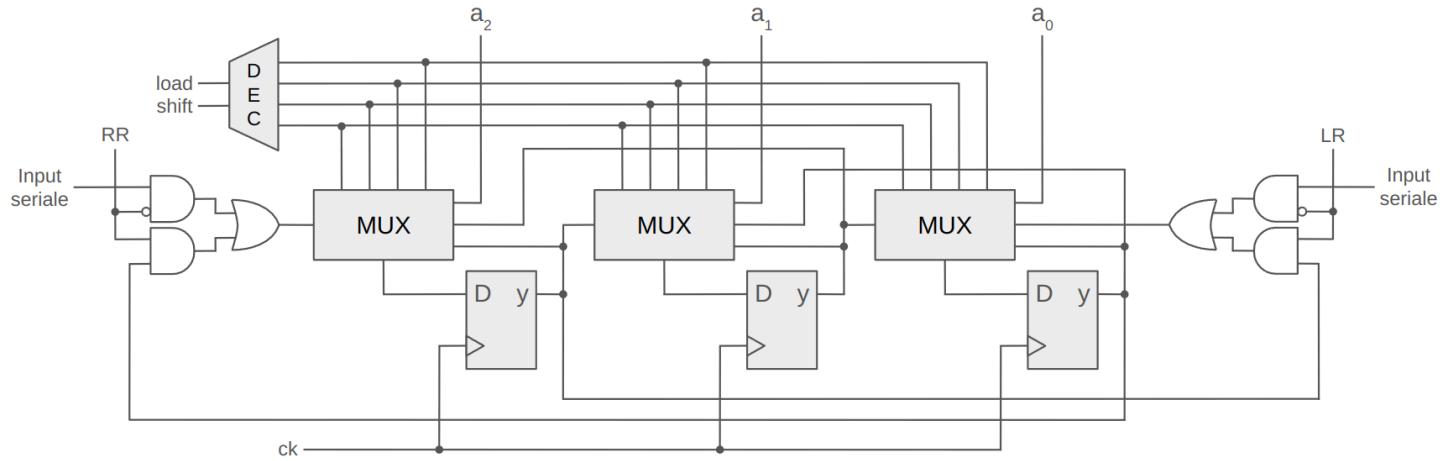
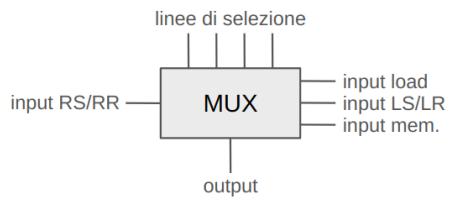
#### Registro universale:

- **caricamento parallelo;**
- **scorrimento** a destra e a sinistra;
- **rotazione** a destra e a sinistra;
- **memorizzazione.**





load	shift	mem.	LS	load	RS
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



### Registri contatori:

- **sincroni**: stesso clock;
- **asincroni**: o a cascata.

### Contatore sincrono modulo 8 (modulo $2^3$ ):

Automa

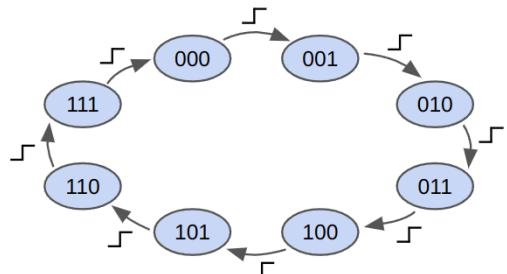


Tavola stati futuri

$y_2$	$y_1$	$y_0$	$Y_2$	$Y_1$	$Y_0$	$j_2$	$k_2$	$j_1$	$k_1$	$j_0$	$k_0$
0	0	0	0	0	1	0	$\delta$	0	$\delta$	1	$\delta$
0	0	1	0	1	0	0	$\delta$	1	$\delta$	$\delta$	1
0	1	0	0	1	1	0	$\delta$	$\delta$	0	1	$\delta$
0	1	1	1	0	0	1	$\delta$	$\delta$	1	$\delta$	1
1	0	0	1	0	1	$\delta$	0	0	$\delta$	1	$\delta$
1	0	1	1	1	0	$\delta$	0	1	$\delta$	$\delta$	1
1	1	0	1	1	1	$\delta$	0	$\delta$	0	1	$\delta$
1	1	1	0	0	0	$\delta$	1	$\delta$	1	$\delta$	1

Espressioni booleane

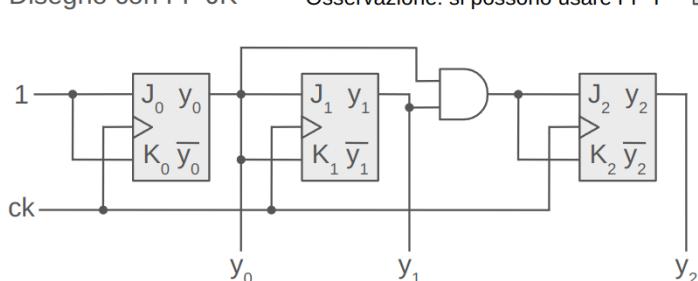
$j_2$	$y_2, y_0$	00	01	11	10
0	$y_2$	0	0	1	0
1	$\delta$	$\delta$	$\delta$	$\delta$	$\delta$

$$j_2 = k_2 = y_1 y_0$$

$$j_1 = k_1 = y_0$$

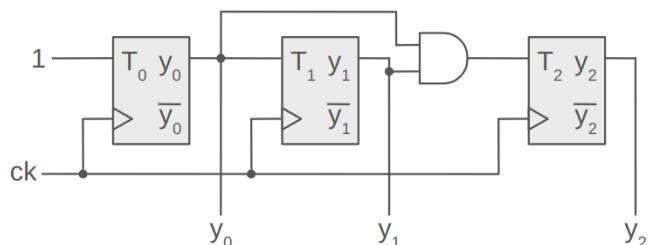
$$j_0 = k_0 = 1$$

Disegno con FF JK



Osservazione: si possono usare FF T

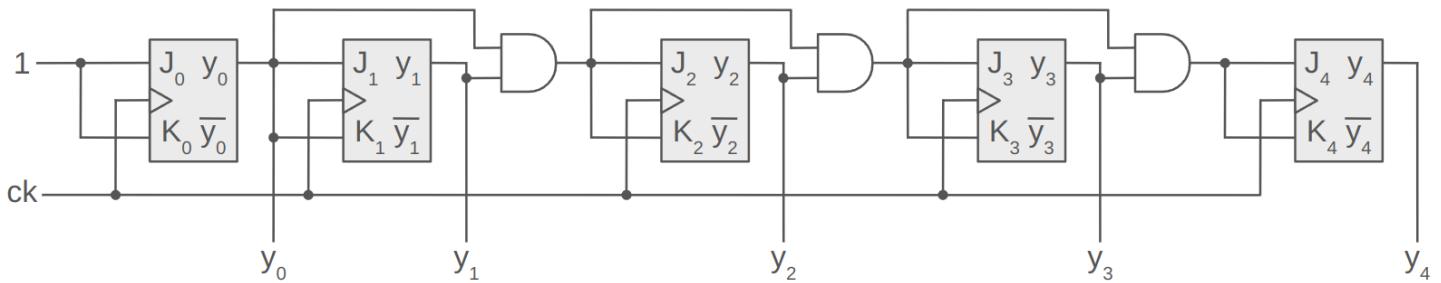
Disegno con FF T



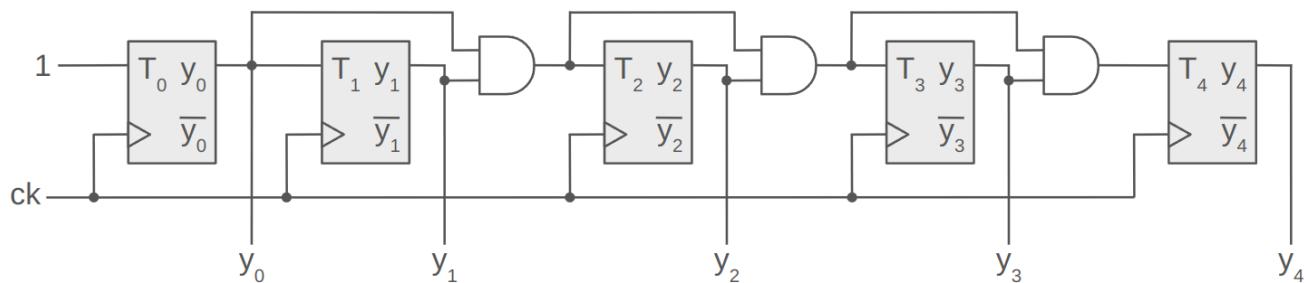
### Contatore sincrono modulo 32 ( $2^5$ ):

- osservazione:** il contatore sincrono esegue un'operazione AND su tutti i bit fino a  $y_{n-2}$ .

Disegno con FF JK



Disegno con FF T



### Contatore sincrono alla rovescia:

Tavola stati futuri

$y_2$	$y_1$	$y_0$	$Y_2$	$Y_1$	$Y_0$	$j_2$	$k_2$	$j_1$	$k_1$	$j_0$	$k_0$
0	0	0	1	1	1	1	$\delta$	1	$\delta$	1	$\delta$
0	0	1	0	0	0	0	$\delta$	0	$\delta$	$\delta$	1
0	1	0	0	0	1	0	$\delta$	$\delta$	1	1	$\delta$
0	1	1	0	1	0	0	$\delta$	$\delta$	0	$\delta$	1
1	0	0	0	1	1	$\delta$	1	1	$\delta$	1	$\delta$
1	0	1	1	0	0	$\delta$	0	0	$\delta$	$\delta$	1
1	1	0	1	0	1	$\delta$	0	$\delta$	1	1	$\delta$
1	1	1	1	1	0	$\delta$	0	$\delta$	0	$\delta$	1

Espressioni booleane

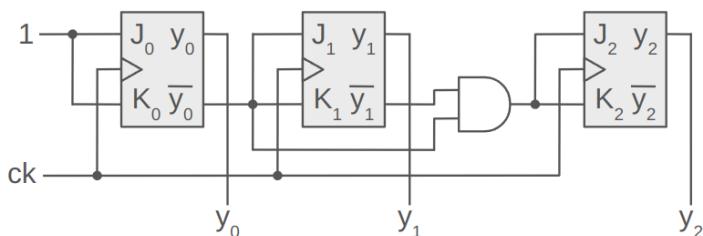
$y_2$	$y_1$	00	01	11	10
0	0	1	0	0	0
1	$\delta$	$\delta$	$\delta$	$\delta$	$\delta$

$$j_2 = k_2 = \bar{y}_1 \bar{y}_0$$

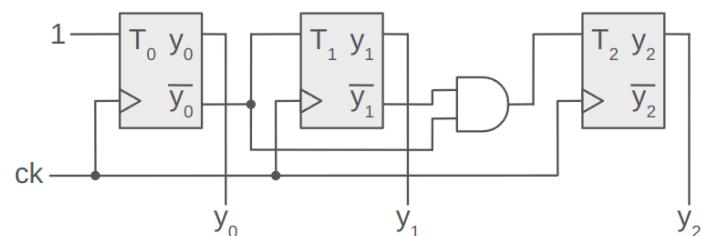
$$j_1 = k_1 = \bar{y}_0$$

$$j_0 = k_0 = 1$$

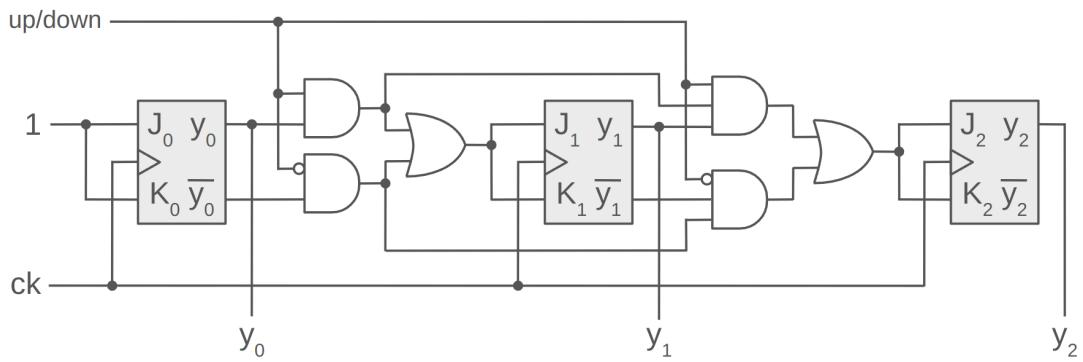
Disegno con FF JK



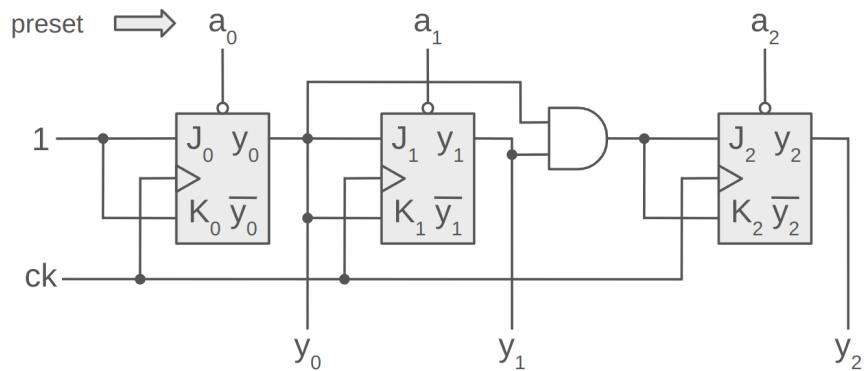
Disegno con FF T



### Contatore sincrono bidimensionale:

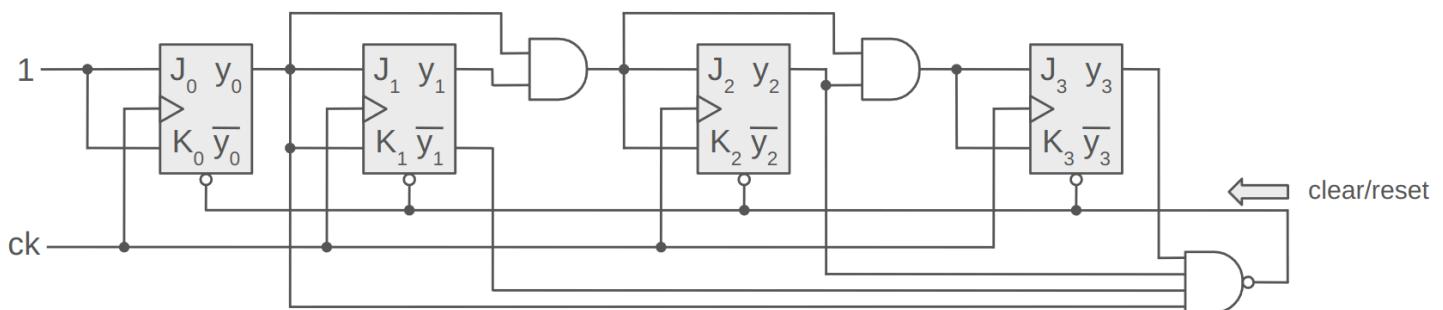


### Contatore sincrono preselezionabile:



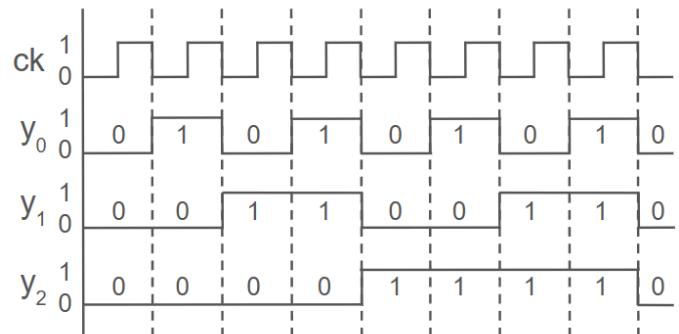
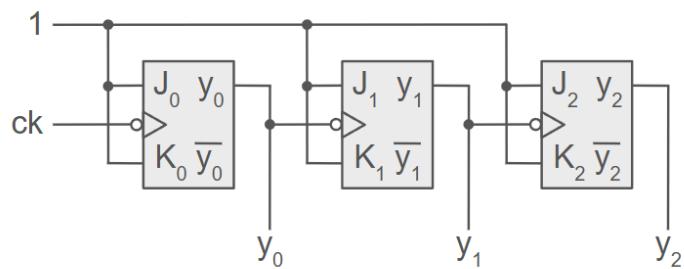
### Reset del contatore sincrono:

Contatore modulo 13 (da 0 a 12)



### Contatore a cascata (asincrono) modulo 8 (modulo $2^3$ ):

Contatore asincrono modulo 8



Contatore asincrono modulo 8 alla rovescia

