

# Corso di Architettura degli Elaboratori e Laboratorio (M-Z)

## Circuiti integrati

*Nino Cauli*

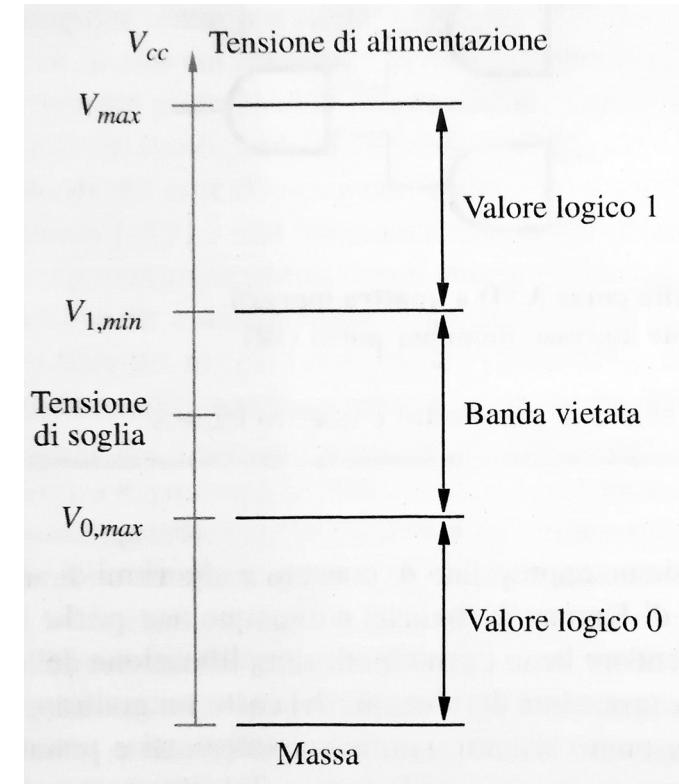


UNIVERSITÀ  
degli STUDI  
di CATANIA

Dipartimento di Matematica e Informatica

# Rappresentazione variabili binarie

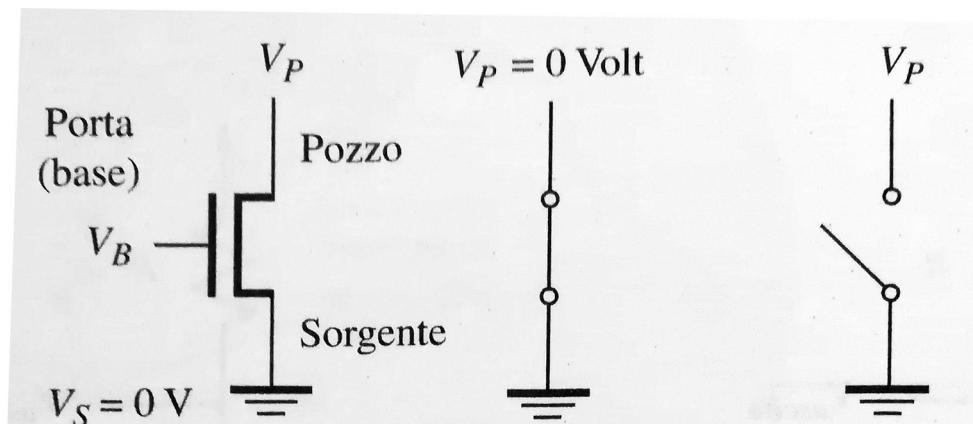
- Nei **circuiti elettronici**, per rappresentare i valori 0 e 1 delle variabili binarie, normalmente si usano valori di **tensione elettrica (voltaggio)**
- Per discretizzare il valore della tensione (grandezza continua), si usa la **soglia di separazione**
- Tutti i valori di tensione superiori alla **tensione di soglia** rappresentano il valore 1 mentre quelli inferiori il valore 0
- Per evitare l'incertezza data dal rumore del circuito, tutti i **valori prossimi alla tensione di soglia** non vengono presi in considerazione (**banda vietata**)



- I **transistori** sono delle componenti elettroniche che possono svolgere la funzione di **interruttori**
- A seconda della tensione ricevuta in ingresso possono trovarsi in stato di **conduzione o interdizione**
- La tecnologia più comunemente usata è il **transistore a metallo-ossido-semiconduttore (MOS)**
- Valori tipici di tensione per tecnologia **MOS**:
  - $V_{cc} = 5$  Volt,  $V_{soglia} = 2.5$  Volt
  - $V_{cc} = 3.3$  Volt,  $V_{soglia} = 1.5$  Volt

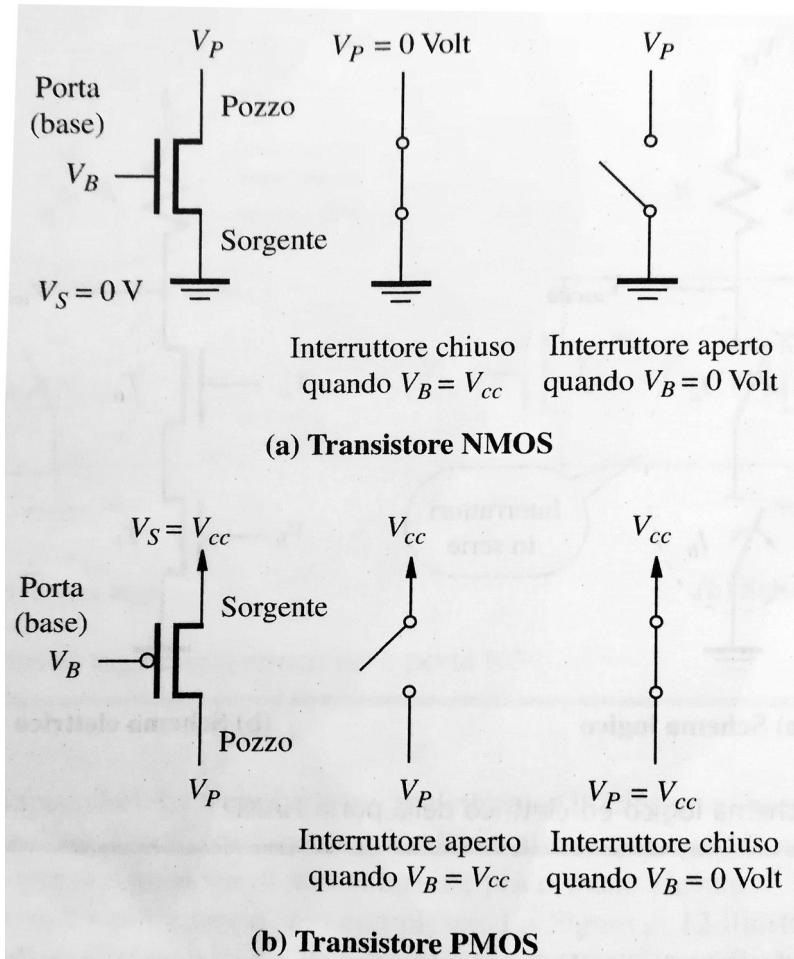
# Transistori MOS

- I transistori MOS hanno 3 collegamenti: **Base (Porta), Pozzo e Sorgente**
- A seconda della tensione in ingresso nella Base il transistore collegherà o meno la Sorgente al Pozzo
- Se il transistore è in **stato di conduzione** la tensione nel Pozzo diventerà uguale alla tensione nella Sorgente



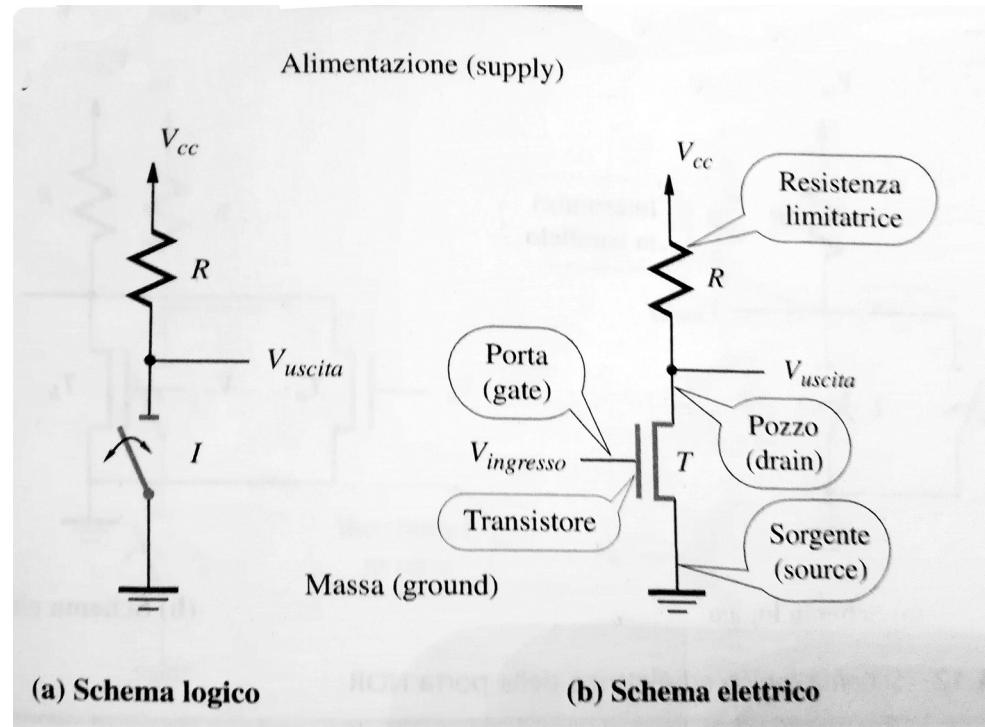
# Transistori NMOS e PMOS

- Esistono 2 tipi di transistori MOS: **NMOS – PMOS**
- Nei transistori NMOS:
  - **Tensione di base alta = conduzione**
  - **Tensione di base bassa = interdizione**
  - **Sorgente collegata alla massa**
- Nei transistori PMOS:
  - **Tensione di base alta = interdizione**
  - **Tensione di base bassa = conduzione**
  - **Sorgente collegata all'alimentazione**



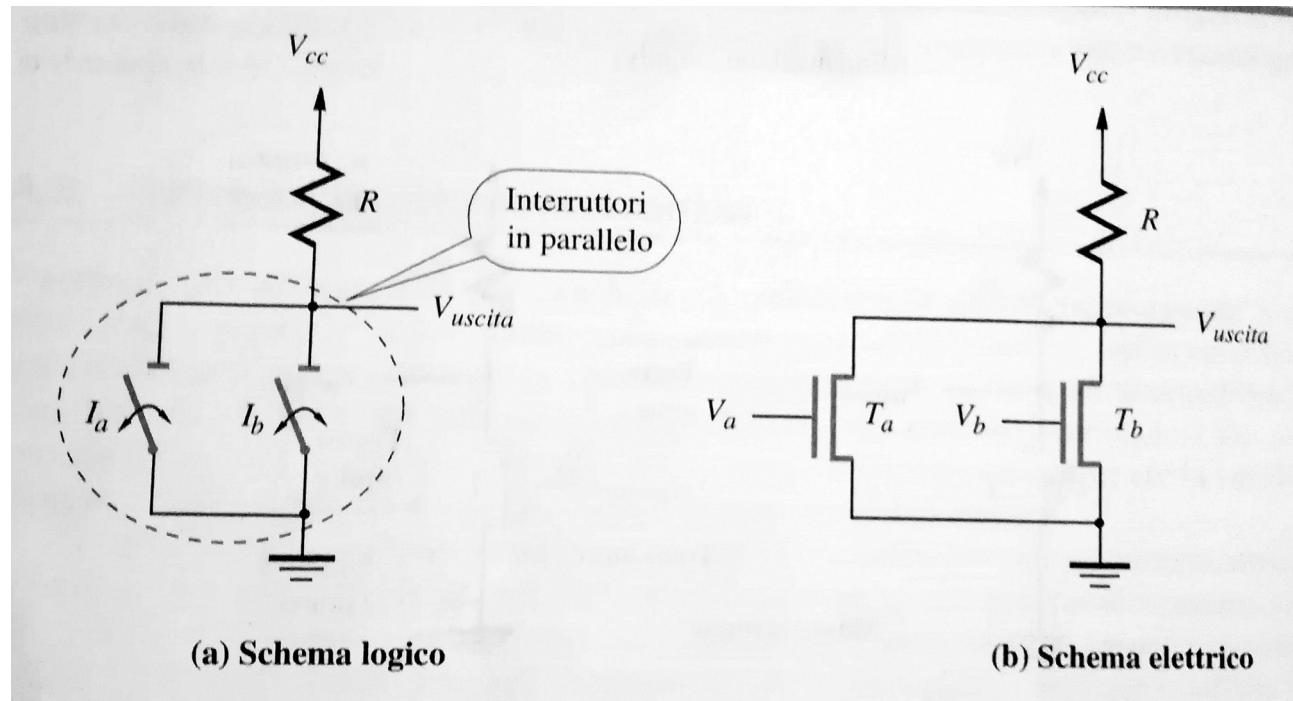
# Circuito NOT

- Si ottiene una **porta NOT** con un transistore NMOS collegando:
  - Sorgente alla massa
  - Pozzo all'alimentazione tramite una resistenza
- Per una tensione di ingresso alla base a “1” si ottiene una tensione di uscita nel pozzo a “0” e viceversa



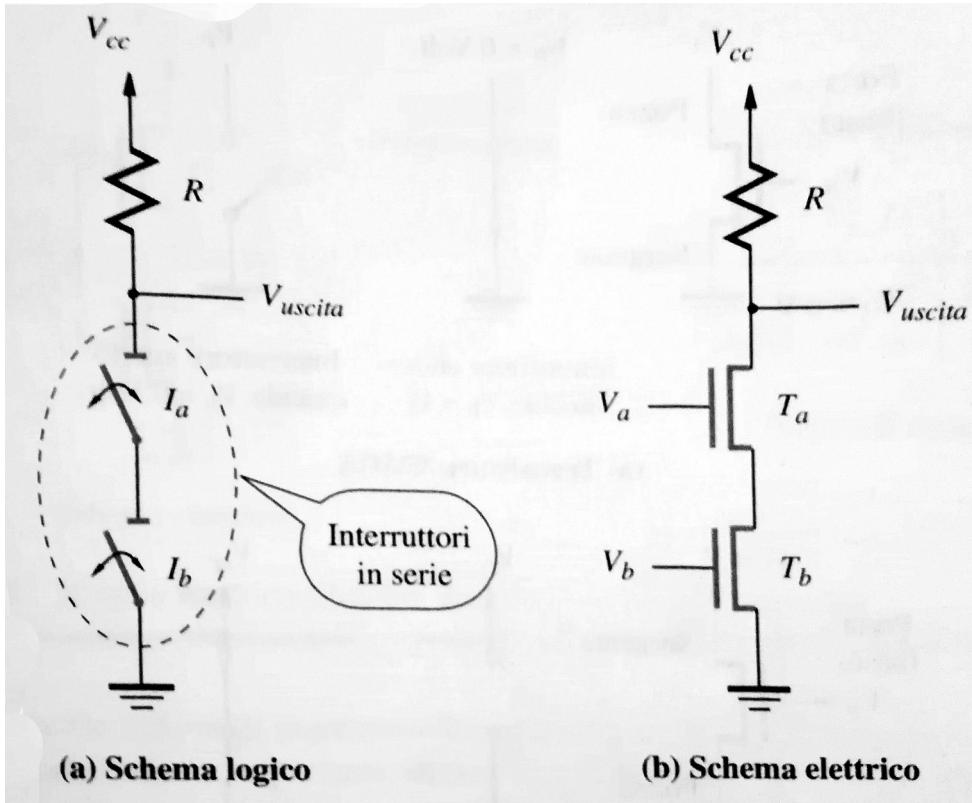
# Circuito NOR

- Collegando due transistori **NMOS** in parallelo si ottiene una **porta NOR**
- Solo se entrambi i transistori sono in interdizione la tensione in uscita sarà “1”

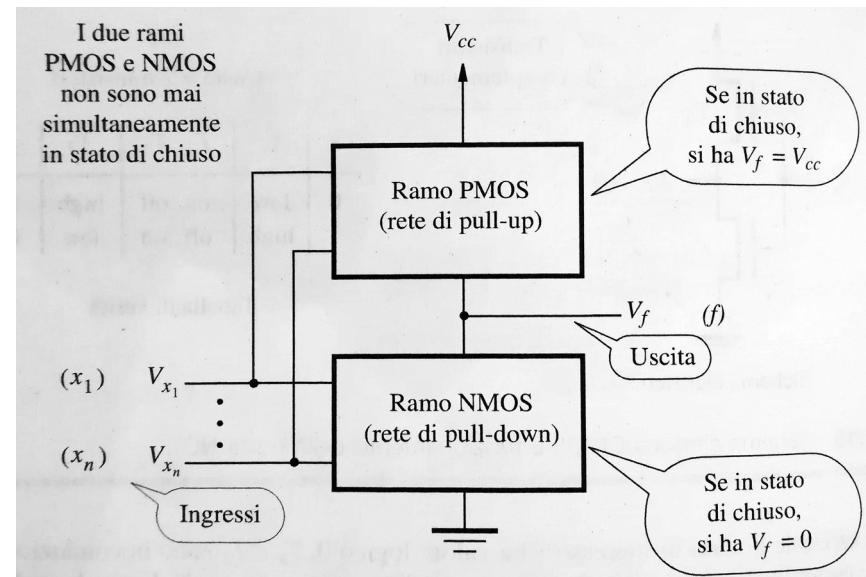


# Circuito NAND

- Collegando due transistori **NMOS in serie** si ottiene una **porta NAND**
- Solo se entrambi i transistori sono in conduzione ( $V_a = V_b = "1"$ ) la tensione in uscita sarà “0”



- Transistori NMOS hanno il problema di **consumare molta energia in stato di conduzione** dovuto alla resistenza
- Il problema si risolve con la tecnologia **MOS Complementare (CMOS)**
- La tecnologia CMOS consiste in un circuito composto da **un ramo di transistor NMOS collegato in serie ad uno di PMOS**
- Il comportamento dei due rami è **complementare** e in stato stabile non c'è mai continuità tra massa e alimentazione



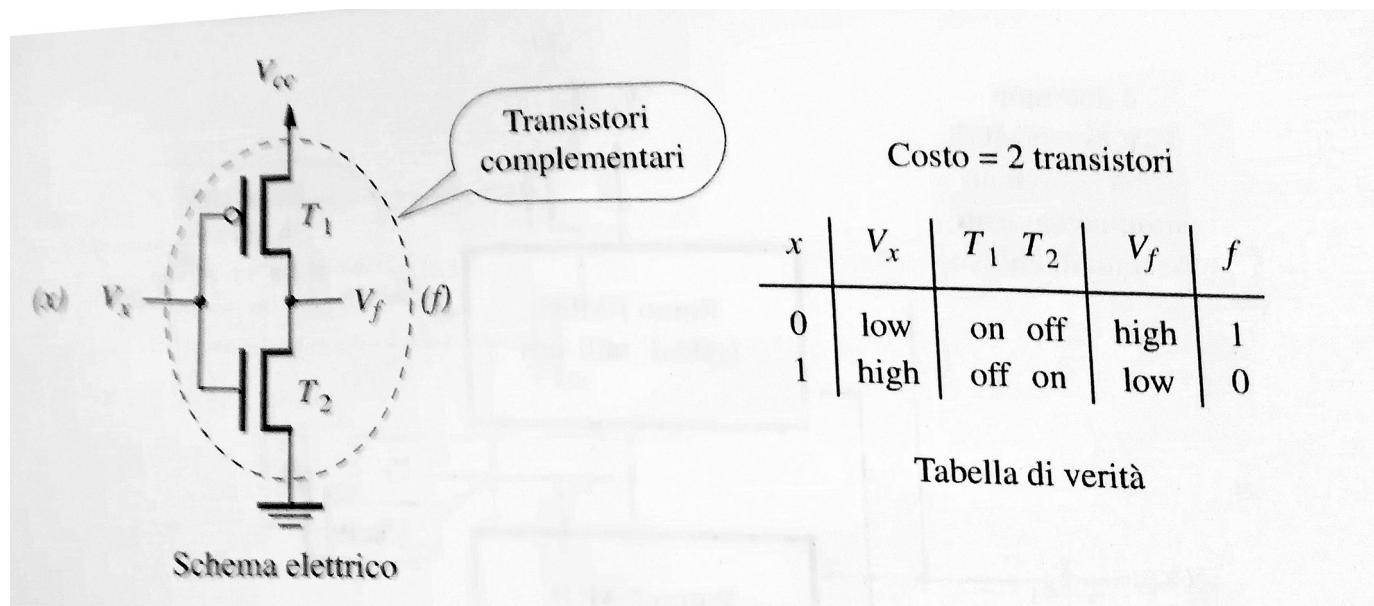
# Vantaggi tecnologia CMOS



- **Consumo di potenza ridotto** (consumo solo in fase di commutazione)
- Potenza elettrica dissipata proporzionale alla frequenza di commutazione
- Transistori MOS hanno **dimensioni molto ridotte** (componenti con miliardi di transistori integrati)
- **Piccole dimensioni = alta frequenza massima di commutazione** (nell'ordine dei GigaHertz)

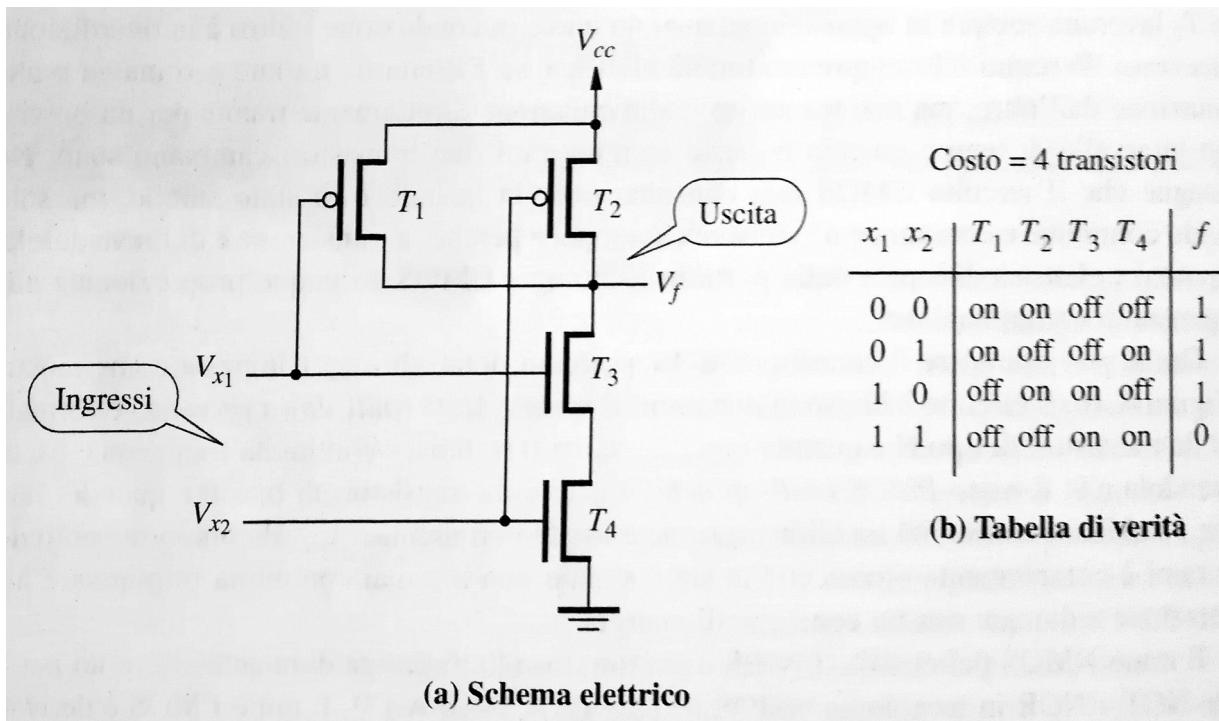
# Circuito CMOS porta NOT

- Una porta NOT è realizzata da un transistore NMOS collegato in serie ad uno PMOS che condividono la stessa tensione di ingresso alla base
- Quando un transistore è in stato di interdizione l'altro è in stato di conduzione



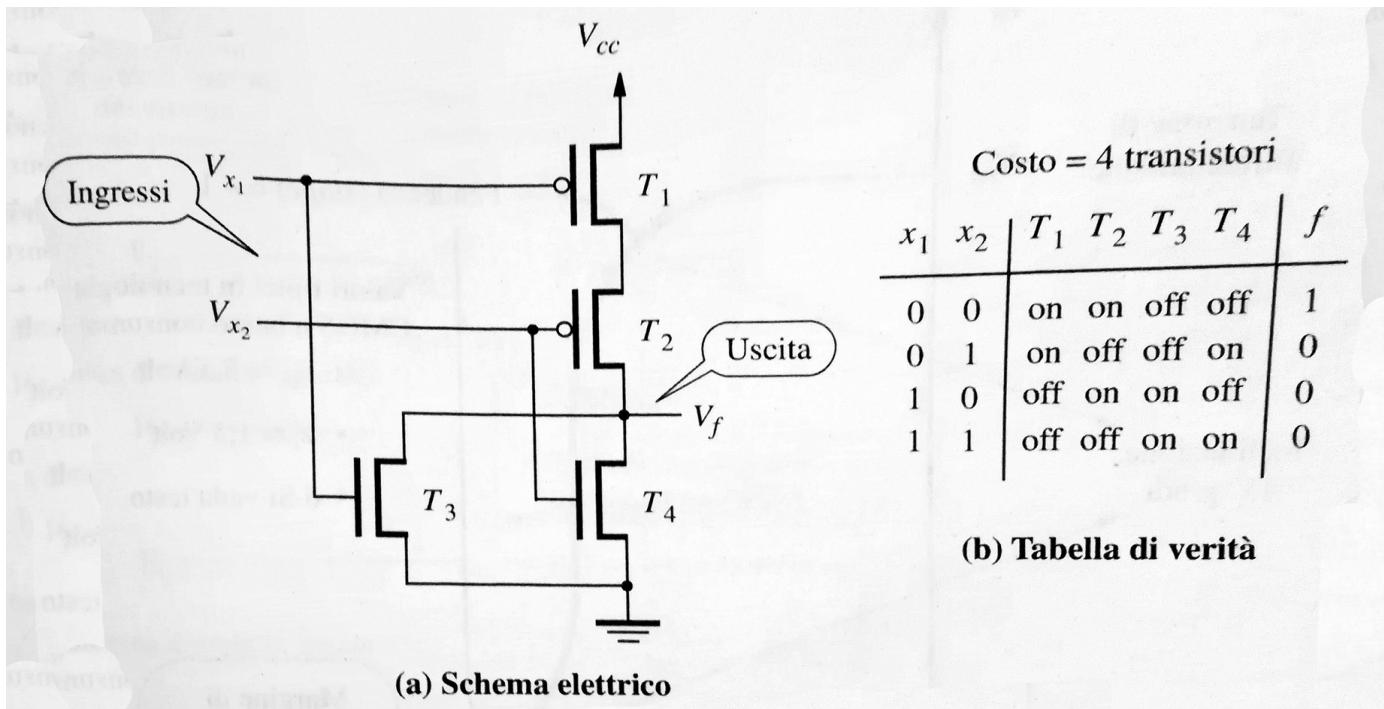
# Circuito CMOS porta NAND

- Una porta **NAND** è realizzata da un circuito CMOS dove il ramo **NMOS** presenta **2 transistori in serie** (come visto prima) e quello **PMOS** **due in parallelo**



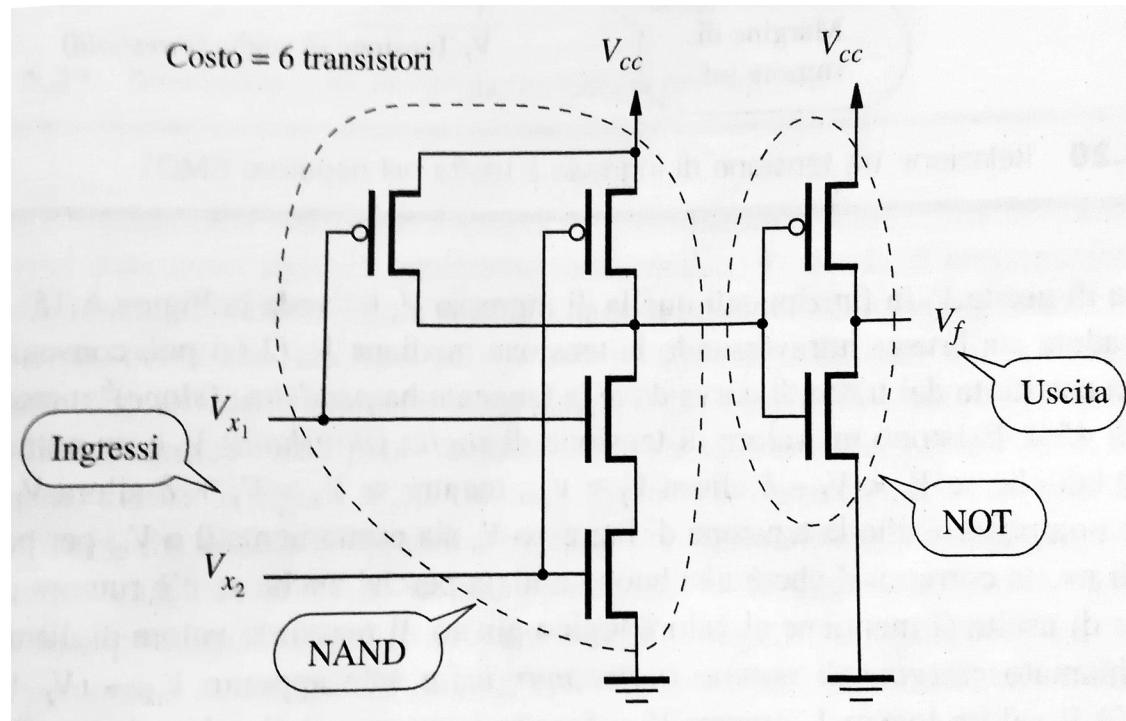
# Circuito CMOS porta NOR

- Una porta **NOR** è realizzata da un circuito CMOS dove il ramo **NMOS presenta 2 transistor in parallelo** (come visto prima) e quello **PMOS due in serie**



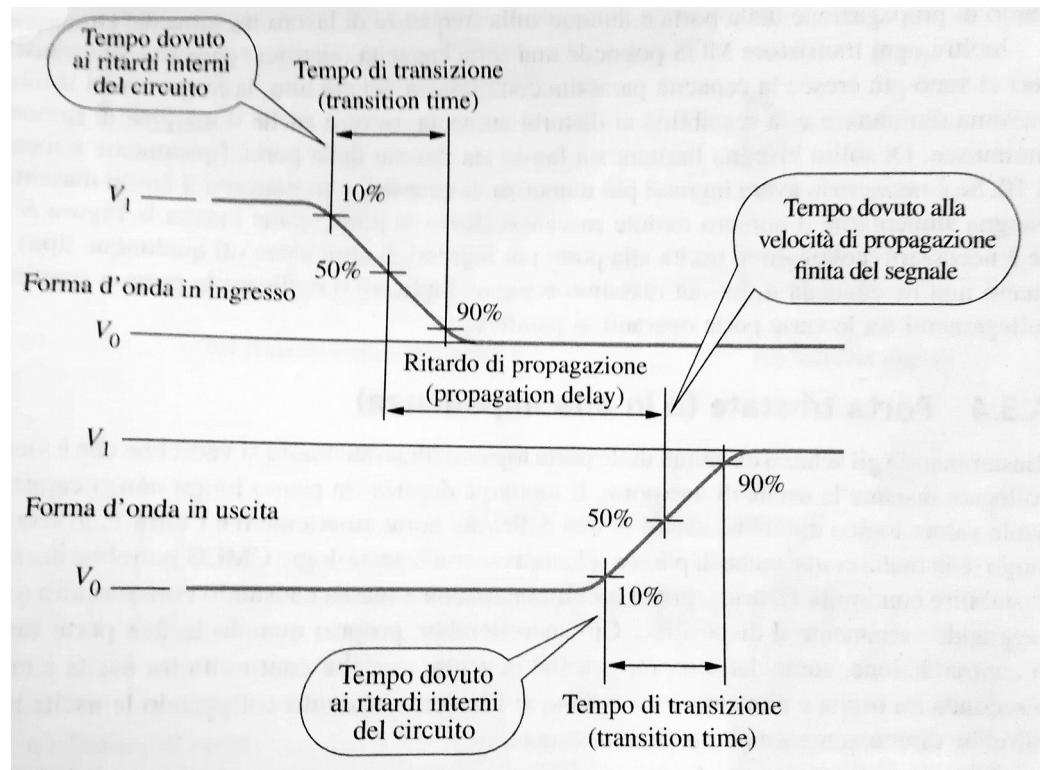
# Circuito CMOS porta AND

- Una porta AND è realizzata collegando una porta NOT all'uscita di una porta NAND
- Il costo di una porta **AND** è di 6 transistori (**la porta NAND ha costo 4**)



# Ritardi in un circuito

- Il **tempo di transizione** è il tempo impiegato da un segnale per transitare di livello
- Il **ritardo di propagazione** è il tempo che impiega l'uscita di un circuito ad adattarsi ai nuovi valori di input
- Il ritardo di propagazione del percorso più lento che collega ingresso e uscita si dice **critico**
- La **frequenza di lavoro** di un circuito sono le volte che esso commuta in un determinato tempo



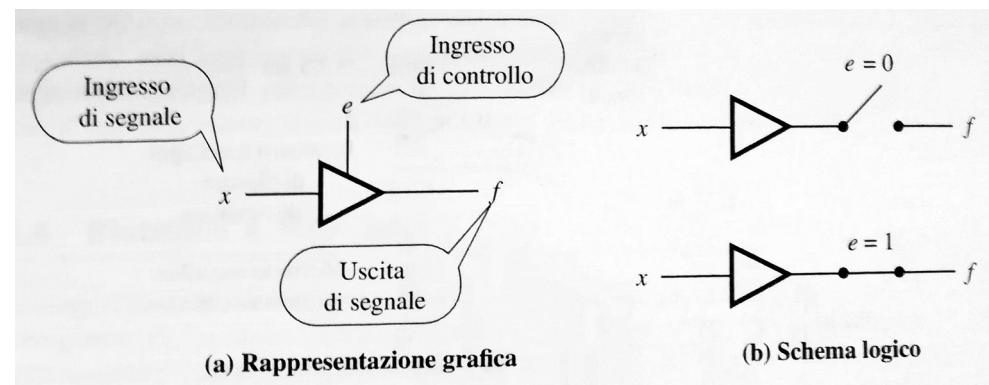
# Fan-in e fan-out

- Il numero di ingressi di una porta logica è chiamato **fan-in**
- Il numero di ingressi paralleli a cui può essere collegata l'uscita di una porta logica è chiamato **fan-out**
- Fan-in e fan-out **elevati** incidono negativamente sul **ritardo di propagazione** e sul **margine di rumore**
- Tipicamente si limitano il fan-in e fan-out a 10 per porta

# Porta tri-state

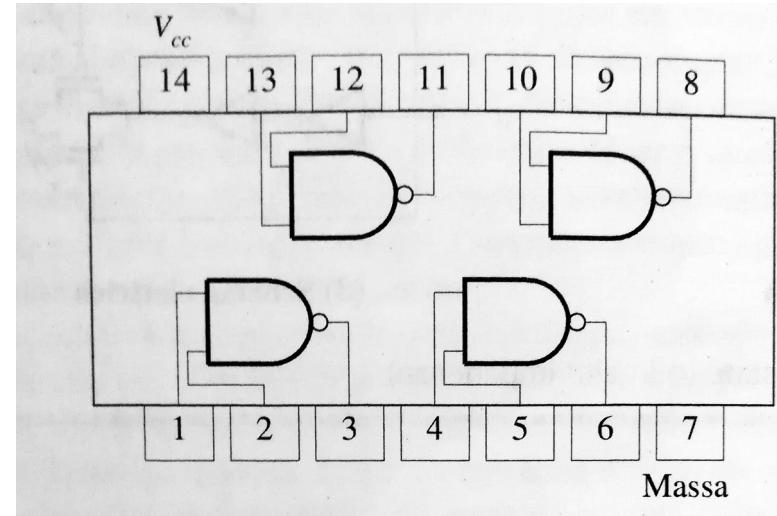
- Non si possono collegare più uscite a uno stesso ingresso (possibile cortocircuito e impossibilità di distinguere i valori di ingresso)
- Bisogna essere in grado di attivare un segnale di ingresso alla volta
- Le porte tri-state hanno due ingressi (**segnale** e **abilitazione**) e un'uscita a tre stati (**0**, **1** e **Z** (alta impedenza))
- Quando l'ingresso di abilitazione è
  - **1: uscita = segnale di ingresso**
  - **0: uscita = Z**

e	x	f
0	0	z
0	1	z
1	0	0
1	1	1



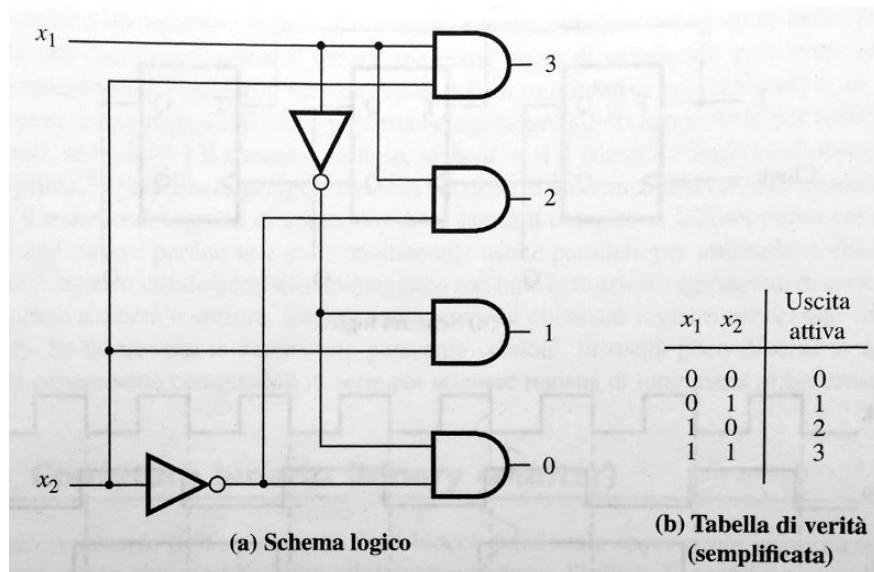
# Circuiti integrati

- Le realizzazioni circuituali di porte logiche sono raggruppate in **circuiti integrati**
- I **circuiti integrati** sono piastrine in silicio incapsulate in un **invólucro protettivo** dotato di **morsetti (pin)** esterni
- Esistono 4 tipi di circuiti integrati a seconda della **scala di integrazione**:
  - SSI (piccola)**: poche porte logiche
  - MSI (media)**: addizionatore, sottrattore, singoli registri, multiplatore, etc.
  - LSI (grande)**: ALU, banco di registri, piccoli processori
  - VLSI (molto grande)**: memorie molto capaci, processori potenti



# Decodificatore (decoder)

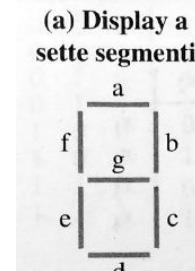
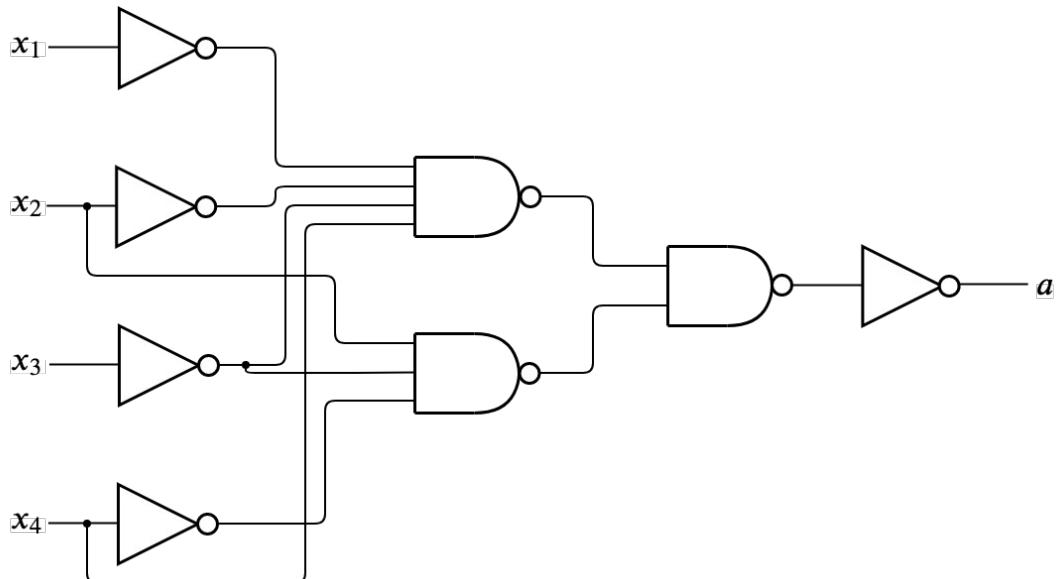
- Il **decodificatore** è un blocco funzionale combinatorio in grado di decodificare un codice binario in ingresso
- Il decodificatore base possiede  **$n$  ingressi** e  **$2^n$  uscite** e attiva la linea di uscita corrispondente al numero binario in ingresso



# Decoder più complesso, esempio

## Decoder per decodificare una cifra decimale per pilotare un display a sette segmenti

- Ogni uscita rappresenta un segmento (1 acceso, 0 spento)
- Esempio implementazione con porte NAND e NOT

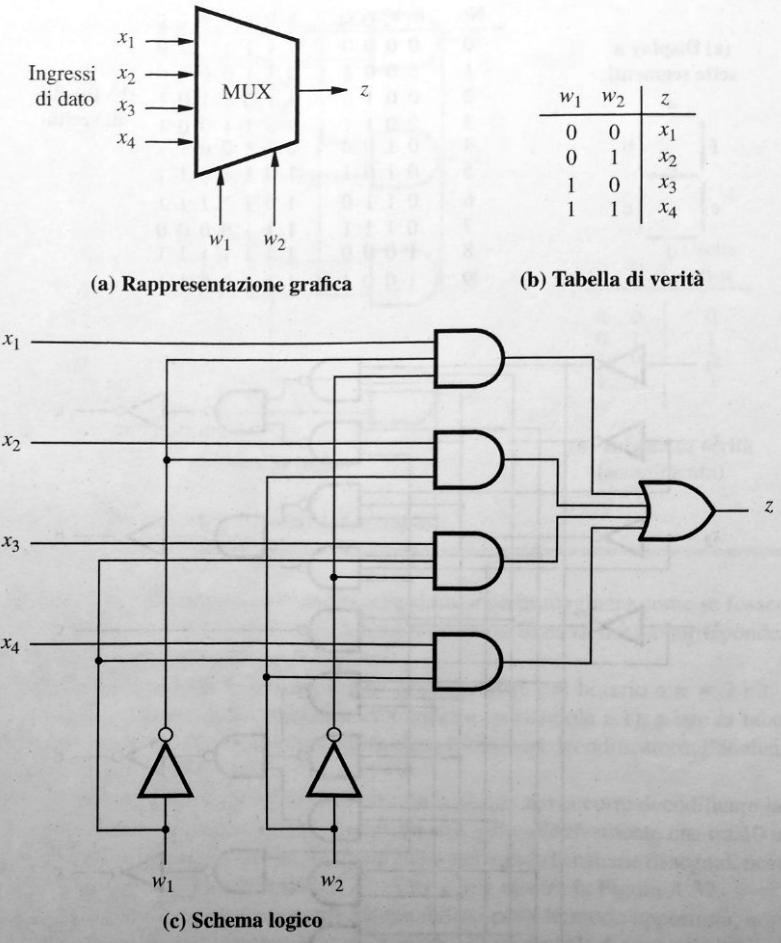


(b) Tabella di verità

No.	$x_1$	$x_2$	$x_3$	$x_4$	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	1	1	1

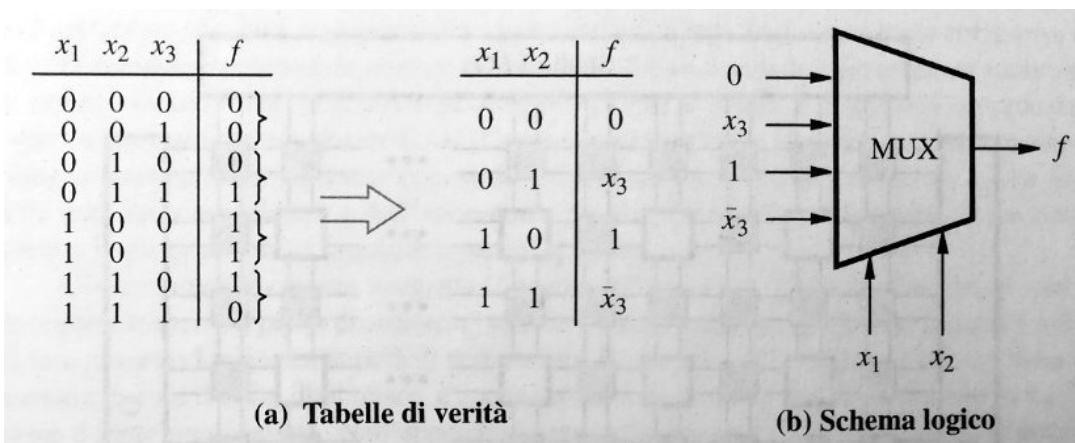
# Multiplatore (multiplexer)

- Il **multiplatore** è un circuito logico in grado di selezionare uno dei suoi “**ingressi dato**” da convogliare nella sua uscita
- Il multiplatore ha  **$n$  ingressi di selezione**,  **$2^n$  ingressi dato** e un uscita
- L'ingresso dato è selezionato dalla configurazione degli  $n$  bit di selezione
- Realizzabile come somma di prodotti degli ingressi



# Multiplatore (multiplexer)

- Il **multiplatore** può essere usato per la **sintesi di funzioni combinatorie**
- Una funzione logica a 3 variabili si può rappresentare con un multiplatore a 2 ingressi di selezione
- Si compatta la funzione a 3 variabili in una funzione a 2 variabili raggruppando 2 a 2 le righe e mettendo la terza variabile in evidenza nella colonna di uscita



# Addizionatore ad 1 bit

Per **SOMMARE** numeri binari ad 1 bit:

$0 +$ $0 =$ <hr style="width: 100%; border: 0; border-top: 1px solid black; margin: 5px 0;"/> <span style="font-size: 2em;">0</span>	$1 +$ $0 =$ <hr style="width: 100%; border: 0; border-top: 1px solid black; margin: 5px 0;"/> <span style="font-size: 2em;">1</span>	$0 +$ $1 =$ <hr style="width: 100%; border: 0; border-top: 1px solid black; margin: 5px 0;"/> <span style="font-size: 2em;">1</span>	$1 +$ $1 =$ <hr style="width: 100%; border: 0; border-top: 1px solid black; margin: 5px 0;"/> <span style="font-size: 2em;">1 0</span>
<span style="font-size: 1.5em;">Addendi da un bit</span>		<span style="font-size: 1.5em;">Riporto in uscita</span> <span style="font-size: 1.5em;">Somma (1 bit)</span>	

**Figura 1.4** - Addizione di numeri a un bit

Il **RIPORTO IN USCITA** della cifre precedente viene assegnato come **RIPORTO IN ENTRATA** alla successiva

# Addizionatore ad 1 bit

- Un addizionatore tra due singoli bit può essere espresso da 2 funzioni logiche a tre ingressi (i due bit da sommare più il riporto in ingresso):
  - La prima calcola la somma tra i bit ed il riporto in ingresso
  - La seconda calcola il riporto in uscita
- Dalla tabella di verità si ricavano le espressioni logiche per somma e riporto in uscita

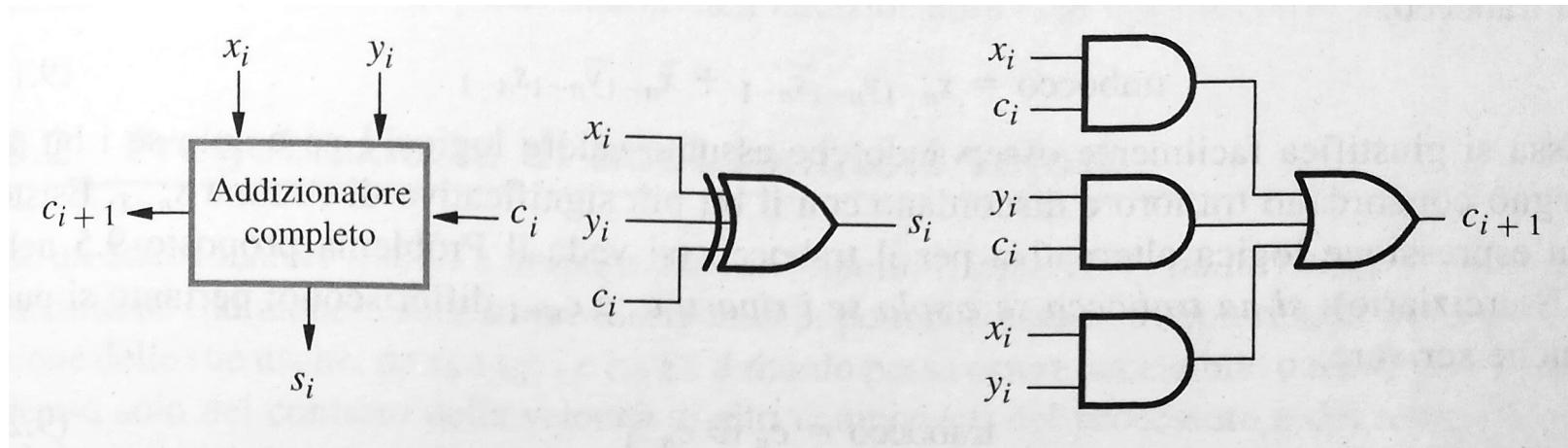
$x_i$	$y_i$	Riporto in ingresso $c_i$	Somma $s_i$	Riporto in uscita $c_{i+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$s_i = \bar{x}_i \bar{y}_i r_i + \bar{x}_i y_i \bar{c}_i + x_i \bar{y}_i \bar{c}_i + x_i y_i c_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i c_i + y_i c_i + x_i y_i$$

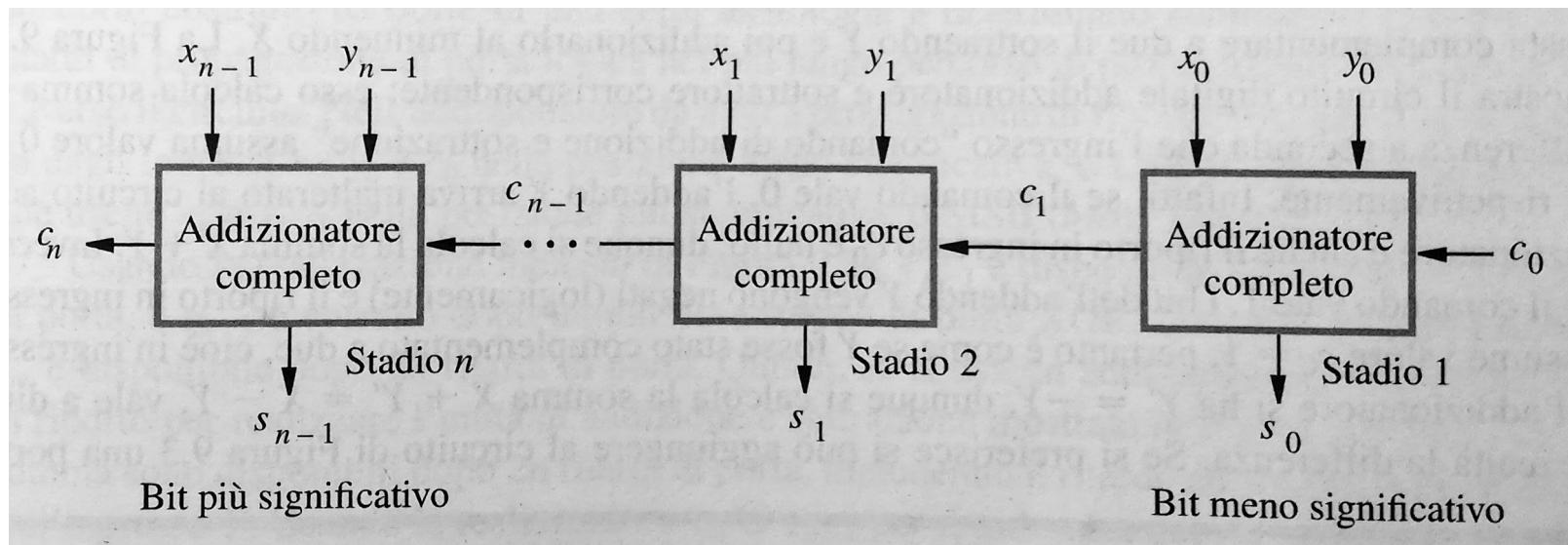
# Addizionatore completo (full adder)

- Unendo assieme in un singolo circuito le reti logiche per le funzioni di somma e riporto in uscita si ottiene l'addizionatore completo
- L'addizionatore completo prende in ingresso i due bit da sommare e il riporto in entrata e rende in uscita somma e riporto in uscita



# Addizionatore a propagazione di riporto

- Collegando una catena di  $n$  addizionatori completi in modo da propagare il riporto si ottiene un circuito in grado di sommare numeri binari di  $n$  bit
- Tale circuito è chiamato addizionatore a propagazione di riporto (ripple carry adder)



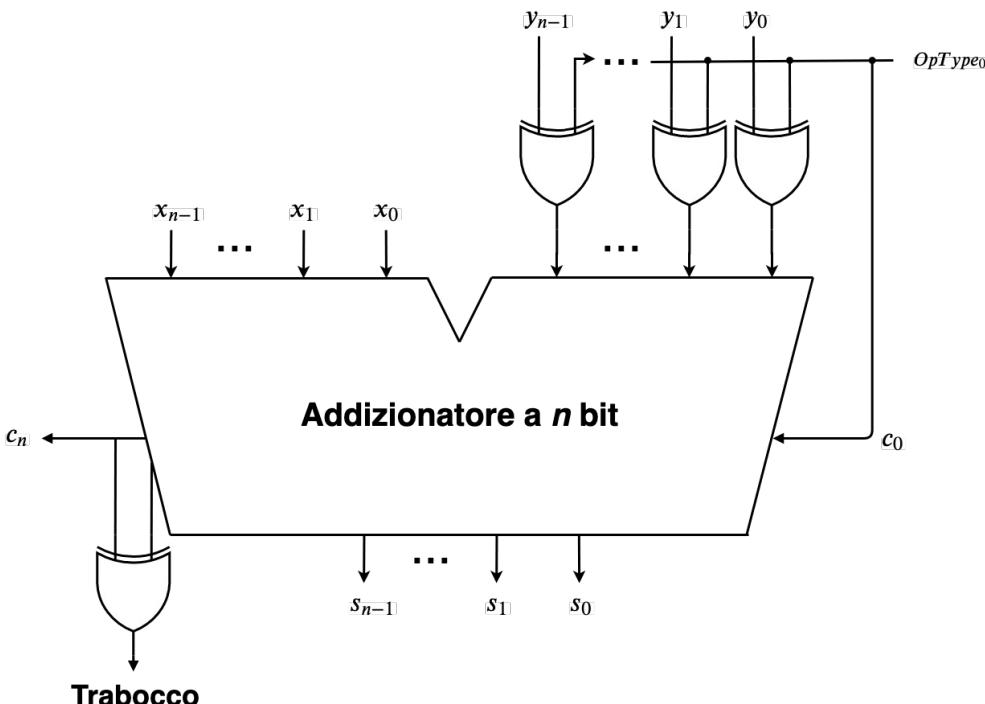
- L'addizione di due numeri in complemento a due corrisponde alla somma di due numeri binari naturali senza contare il riporto in uscita
- La sottrazione corrisponde ad un addizione complementando a due il sottraendo
- Bisogna però garantire che non avvenga trabocco
- Il calcolo del trabocco può essere espresso da una delle seguenti espressioni logiche:

$$\text{trabocco} = x_{n-1}y_{n-1}\bar{s}_{n-1} + \bar{x}_{n-1}\bar{y}_{n-1}s_{n-1}$$

$$\text{trabocco} = c_n \oplus c_{n-1}$$

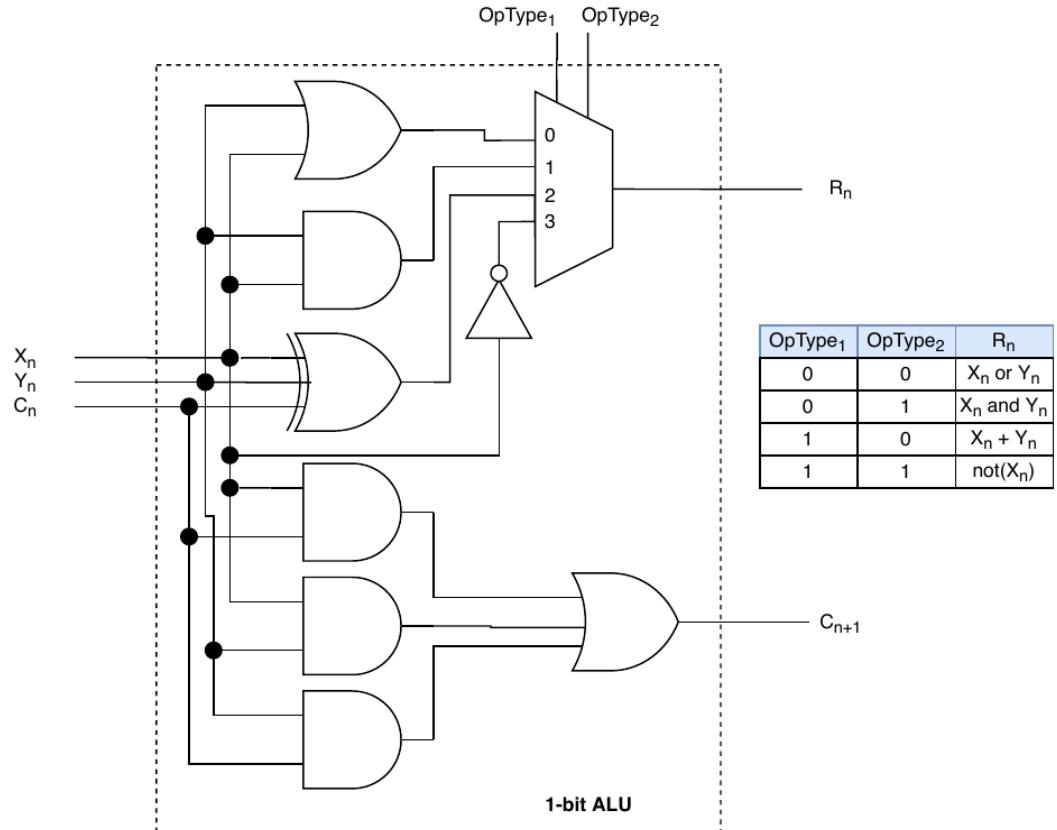
# Addizionatore algebrico a n bit

- Una unità logica per addizione e sottrazione può essere ottenuta usando un addizionatore a propagazione di riporto
- Si usa il bit  $OpType_0$  per complementare a due il sottraendo in caso di sottrazione
- Nel caso  $OpType_0 = 1$  si avrà un riporto in ingresso al bit meno significativo e il secondo addendo verrà complementato attraverso una catena di porte xor parallele ( $y_n \oplus OpType_0$ )
- Il trabocco viene calcolato come  $c_n \oplus c_{n-1}$

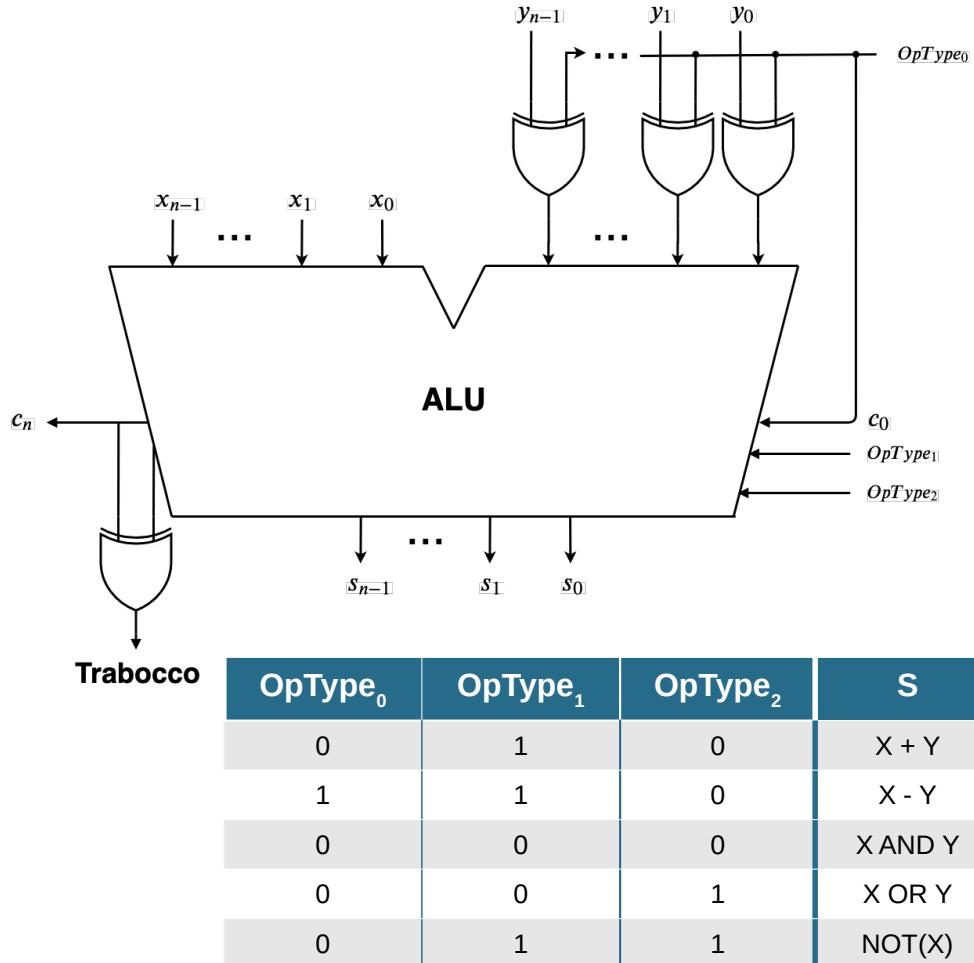


# ALU a 1 bit

- Estendiamo ora l'addizionatore completo includendo anche la possibilità di effettuare le seguenti operazioni logiche bitwise AND, OR e NOT
- Aggiungiamo dunque un multiplexer che consente di mandare in output, alternativamente, l'uscita del:
  - Sommatore:  $x_i + y_i$
  - Porta AND:  $x_i \text{ AND } y_i$
  - Porta OR:  $x_i \text{ OR } y_i$
  - Negazione:  $\neg x_i$



- Collegando in serie n ALU a 1 bit in un'unità logica simile all'addizionatore visto in precedenza otterremo una ALU a n bit
- I bit di controllo  $OpType_1$  e  $OpType_2$  servono a selezionare l'operazione da eseguire (addizione, AND, OR o NOT)
- $OpType_0$  serve a selezionare la sottrazione e complementare a 2 il sottraendo

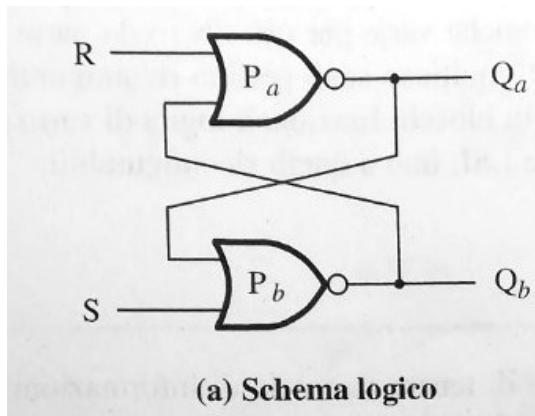


# Reti sequenziali

- Abbiamo visto come sia possibile, grazie alle reti combinatorie, realizzare varie unità funzionali (ALU, decodificatori, moltiplicatori, ecc.), ma per memorizzare informazione le reti combinatorie non bastano
- C'è bisogno di una rete logica le cui uscite non dipendano solo dall'input attuale, ma anche dai sui "stati" precedenti
- Queste reti sono chiamate reti sequenziali
- Le reti sequenziali sono reti logiche che presentano dei cicli

# Bistabile asincrono

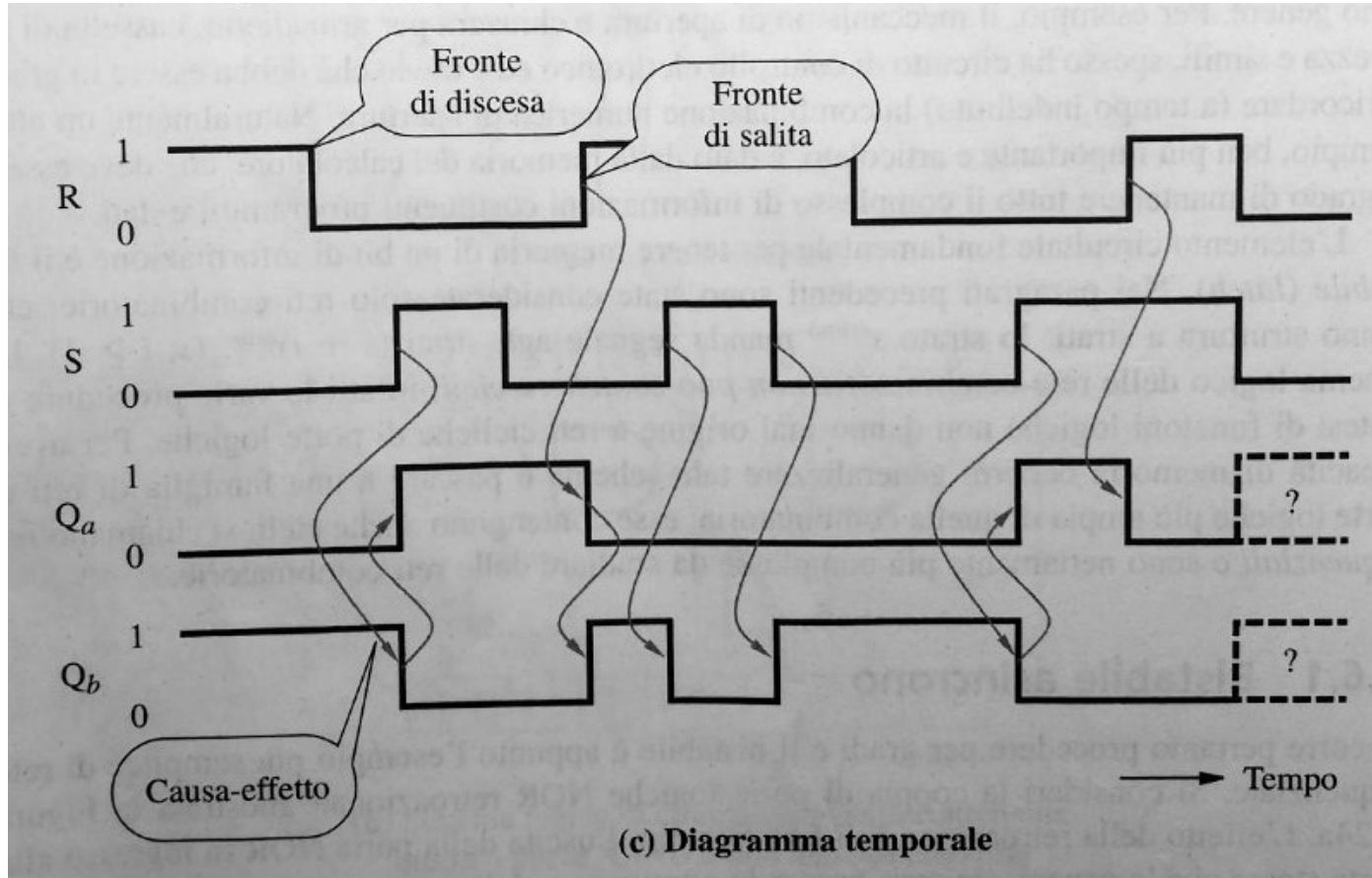
- Il bistabile è una rete sequenziale in grado di memorizzare 1 bit ( $Q_a$ )
- Tenendo gli input Set e Reset a 0 il bistabile mantiene la sua uscita precedente
- Mettendo a 1 Set o Reset, l'uscita  $Q_a$  si porta a 1 o a 0 rispettivamente
- Il caso di Set e Reset a 1 non viene usato per possibile ambiguità



S	R	$Q_a$	$Q_b$	O anche
0	0	0/1	1/0	$Q_a \quad Q_b$
0	1	0	1	
1	0	1	0	Resto idem
1	1	0	0	

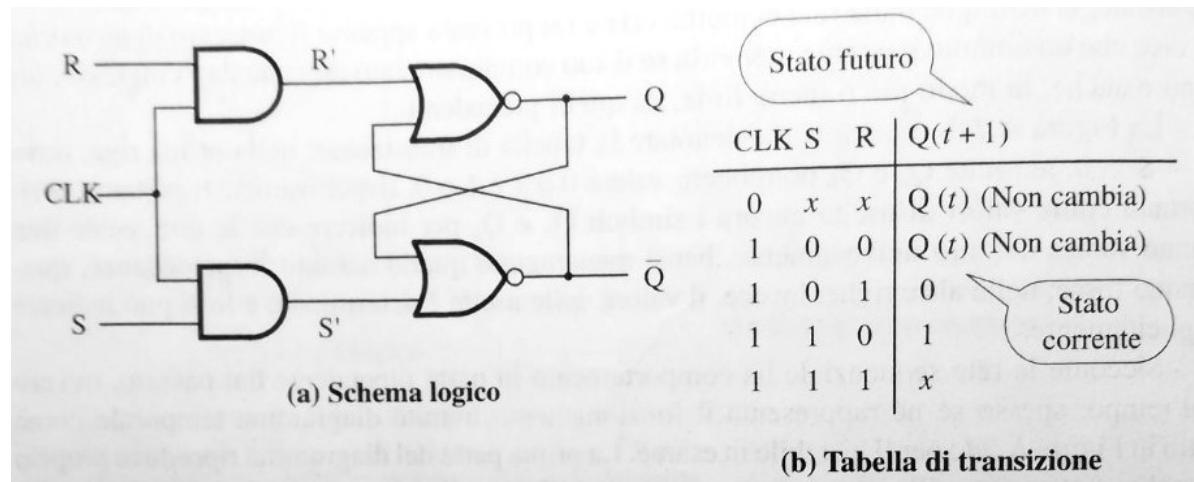
(b) Tabella di transizione

# Bistabile asincrono (diagramma temporale)

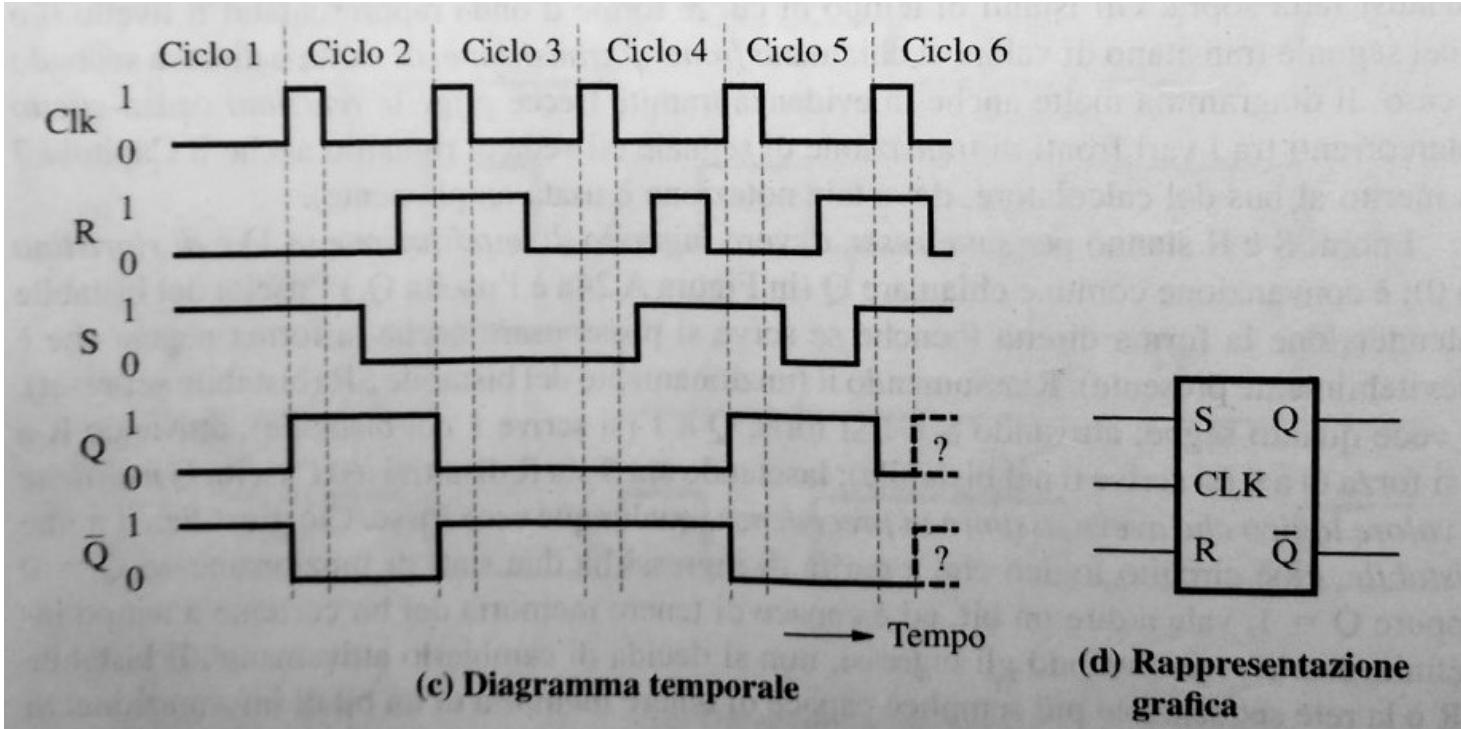


# Bistabile sincrono

- Il bistabile sincrono presenta un bit CLK di ingresso oltre Set e Reset
- Quando  $CLK = 0$  lo stato non cambia
- Quando  $CLK = 1$  si comporta come un bistabile asincrono
- Il funzionamento è legato ai cicli di clock

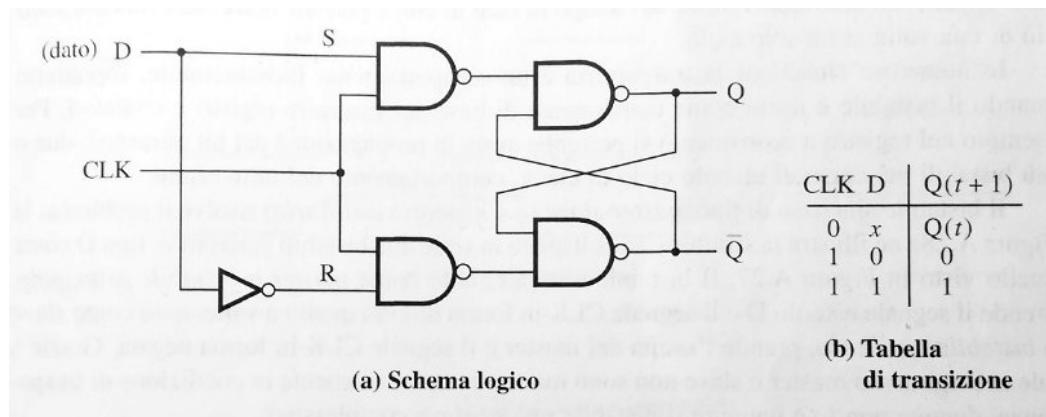


# Bistabile sincrono

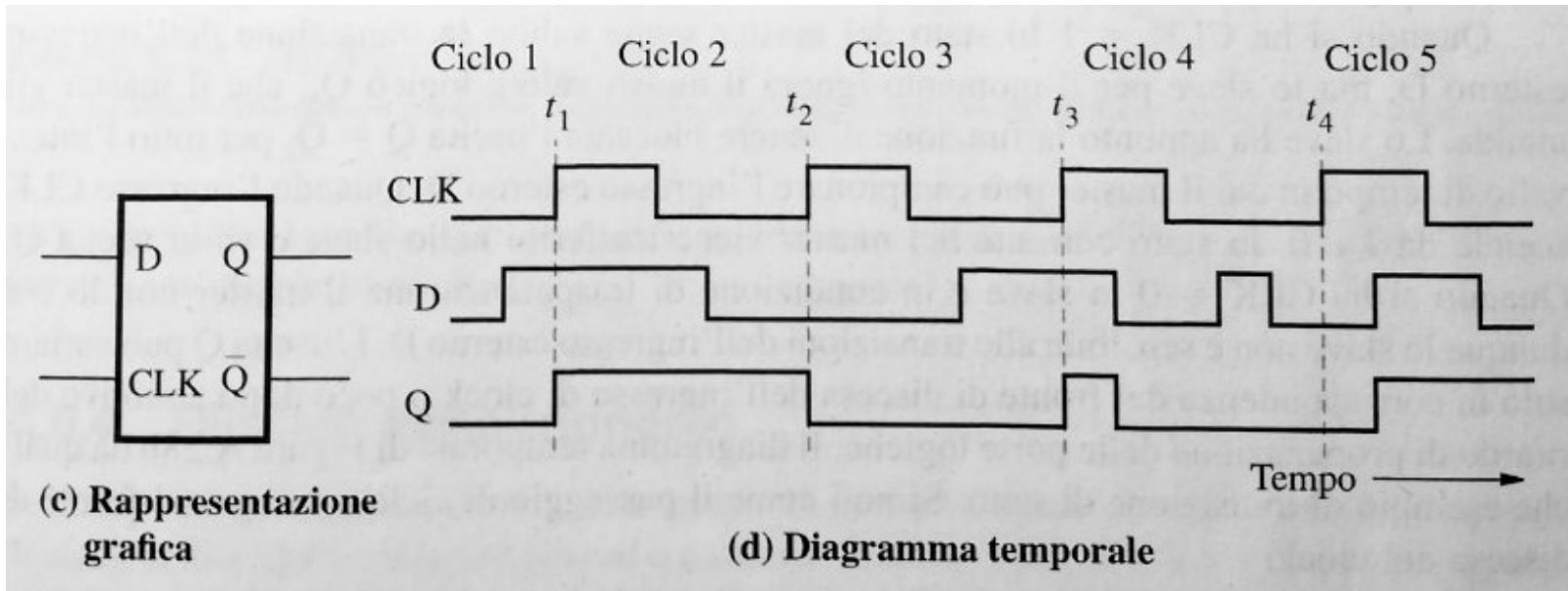


# Bistabile di tipo D

- Dato che Set e Reset si usano sempre con valore opposto, si può usare un singolo input D
- Il bistabile di tipo D memorizza il valore del suo singolo bit di input
- Nella figura è mostrata uno schema logico con soli NAND (equivalente alla versione con AND e NOR)



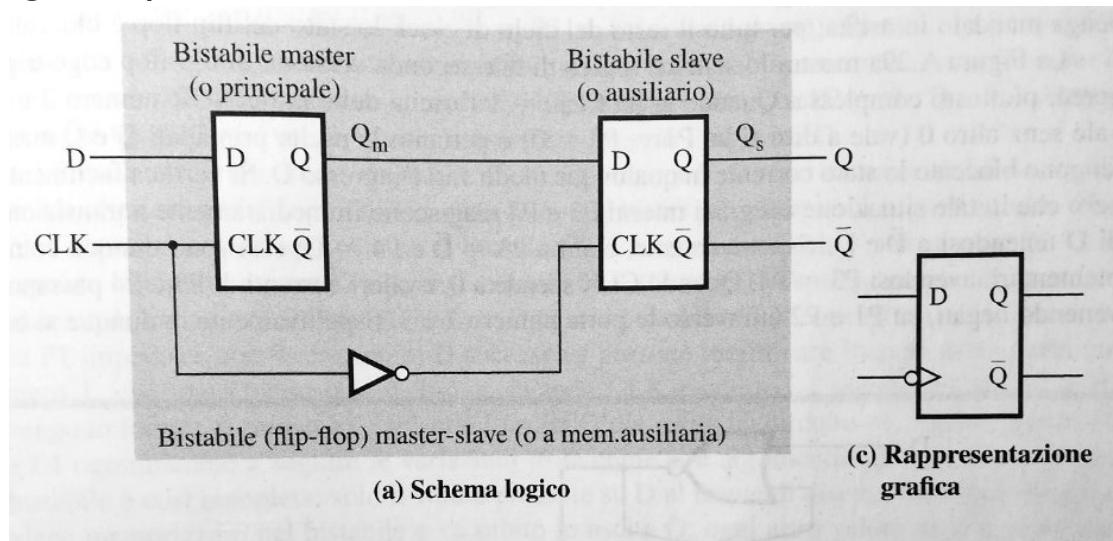
# Bistabile di tipo D



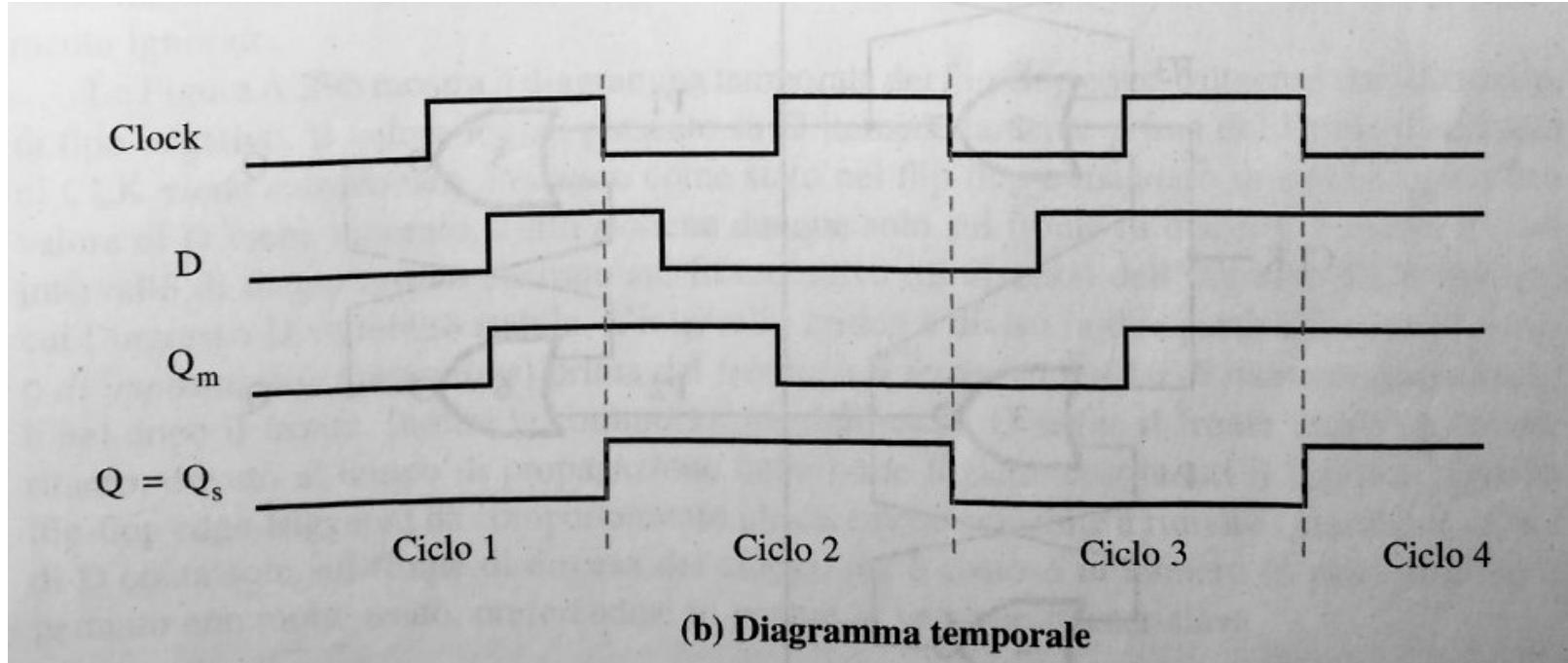
- Un bistabile sincrono può commutare varie volte quando l'ingresso di clock è alto (effetto chiamato trasparenza)
- Spesso la trasparenza è un comportamento indesiderabile

# Flip-flop master-slave

- Il Flip-Flop master-slave è una rete sequenziale che risolve il problema della trasparenza
- È formato da due bistabili D in serie (master e slave), il primo riceve il segnale di clock diretto e il secondo negato
- Come risultato questo tipo di flip-flop commuta solo durante il fronte di discesa del clock (edge-triggered negativo)

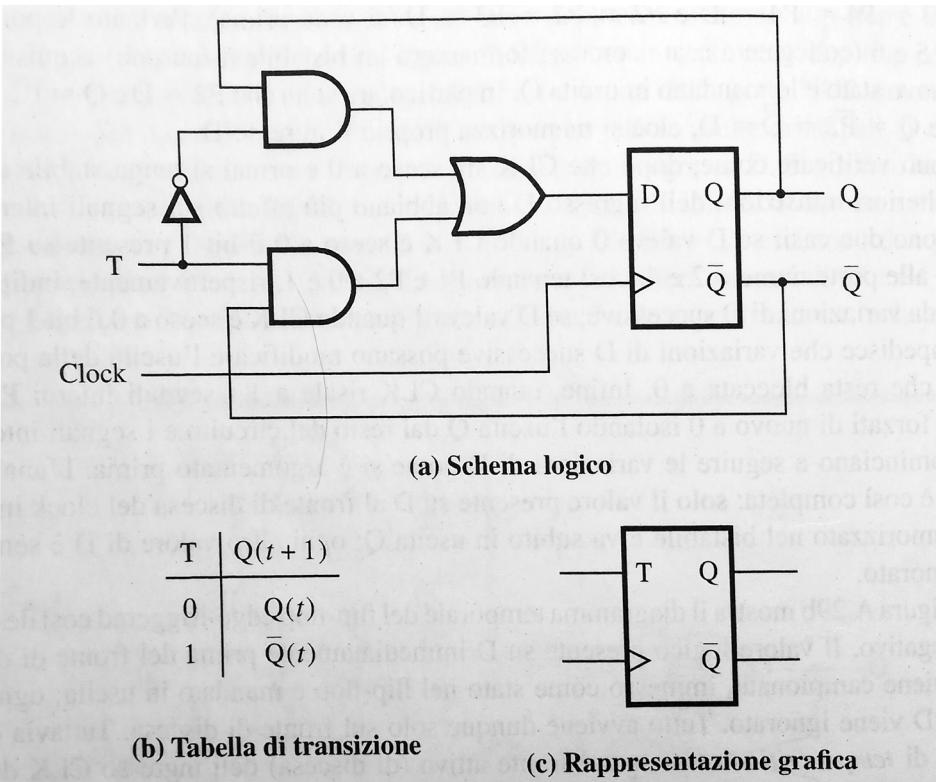


# Flip-flop master-slave



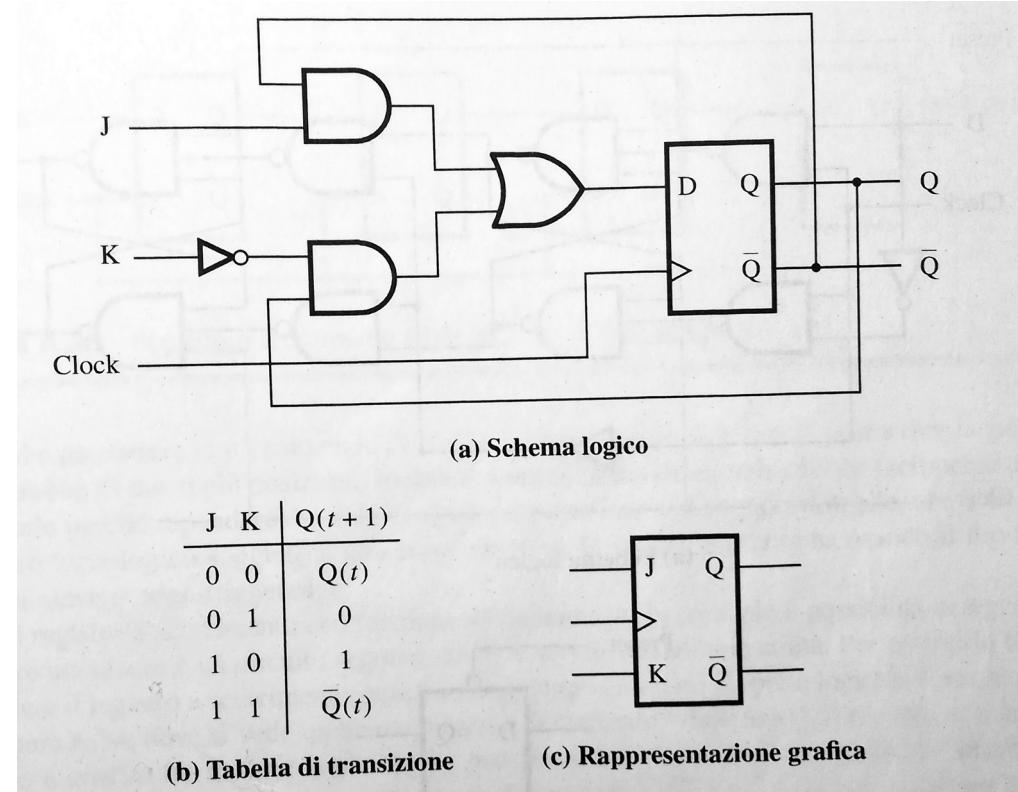
# Flip-flop tipo T

- Il flip-flop di tipo T commuta stato ogni ciclo di clock se il suo input T è a 1, altrimenti viene riconfermato
- A seconda del valore di T viene rimandata in ingresso ad un flip-flop di tipo D la sua uscita diretta o negata
- Utile per realizzare i contatori



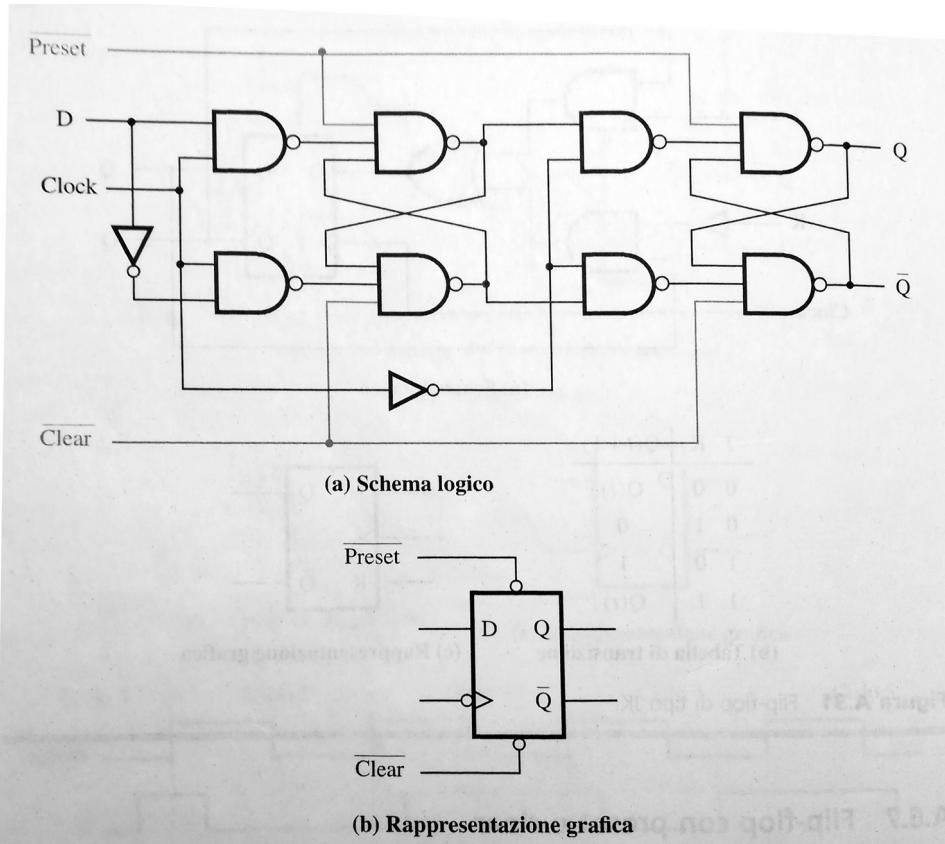
# Flip-flop tipo JK

- Il flip-flop di tipo JK unisce le funzionalità dei flip-flop di tipo D e T
- Se lo stesso bit viene duplicato negli ingressi J e K si comporta come un flip-flop T (1 commuta lo stato e 0 lo riconferma)
- Se J e K sono complementari si comporta come un filp-flop di tipo D, dove l'input J corrisponde a D



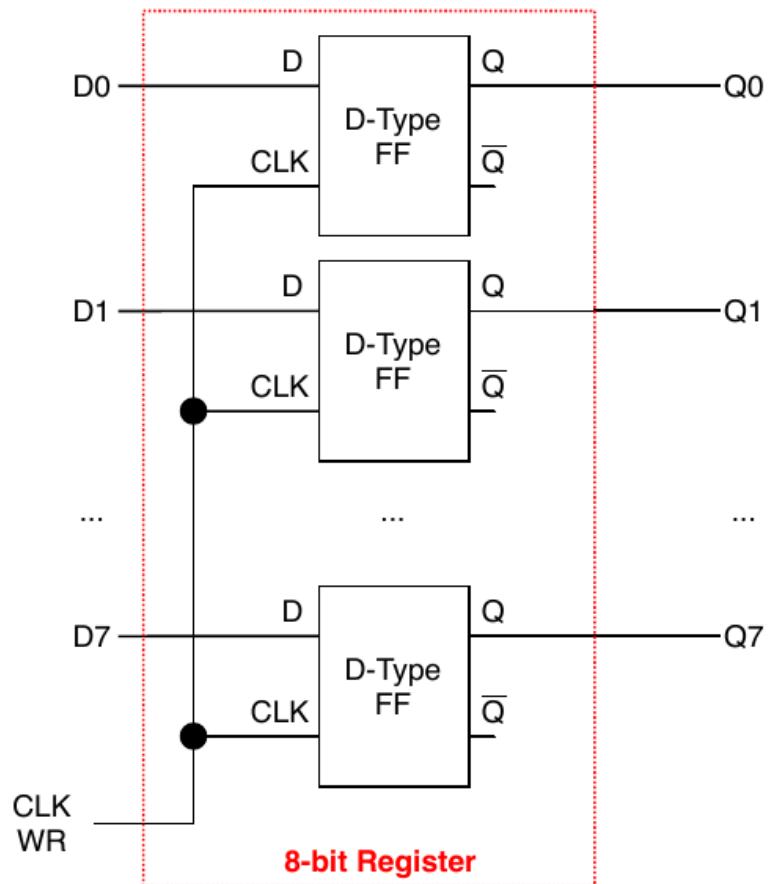
# Flip-flop con preset e clear

- Spesso è necessario inizializzare lo stato del flip-flop indipendentemente dai valori di ingresso e dello stato corrente
- Aggiungendo due ingressi (preset e clear) è possibile ottenere questa funzionalità
- Quando preset è attivo si forza lo stato a 1, quando clear è attivo si forza a 0
- Preset e clear sono attivi bassi e mai attivi nello stesso momento



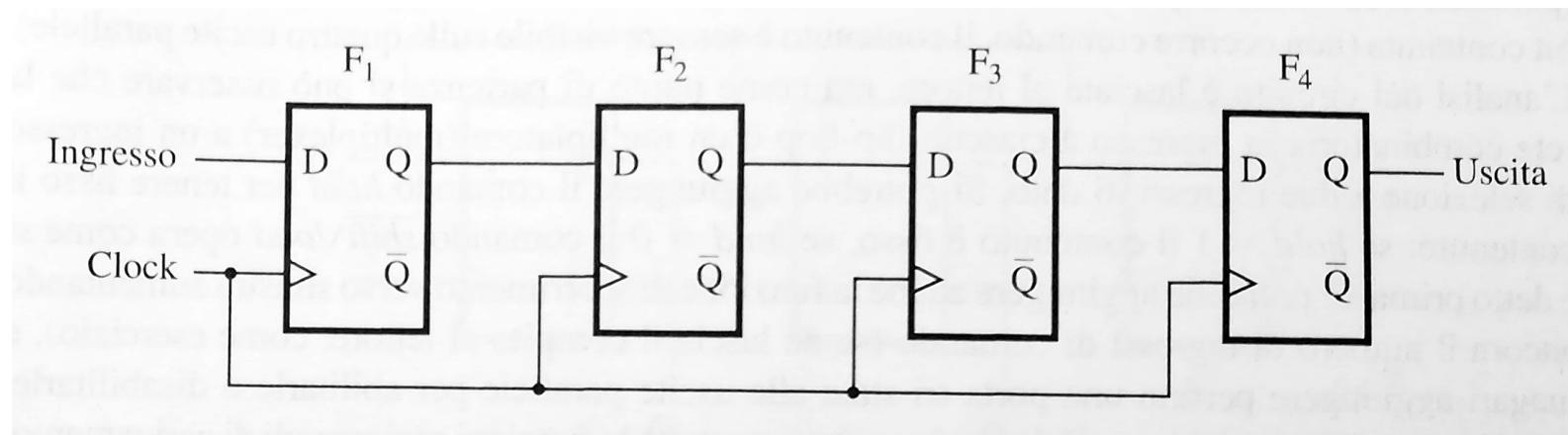
# Registri paralleli

- I registri sono dei circuiti elettronici in grado di immagazzinare una sequenza di n bit
- Finora abbiamo visto blocchi logici in grado di memorizzare 1 bit (flip-flop)
- Un registro è formato da un insieme di flip-flop collegati in parallelo ad uno stesso segnale di clock
- I registri paralleli sono dei registri in cui si può accedere contemporaneamente a tutti gli ingressi e le uscite dei flip-flop



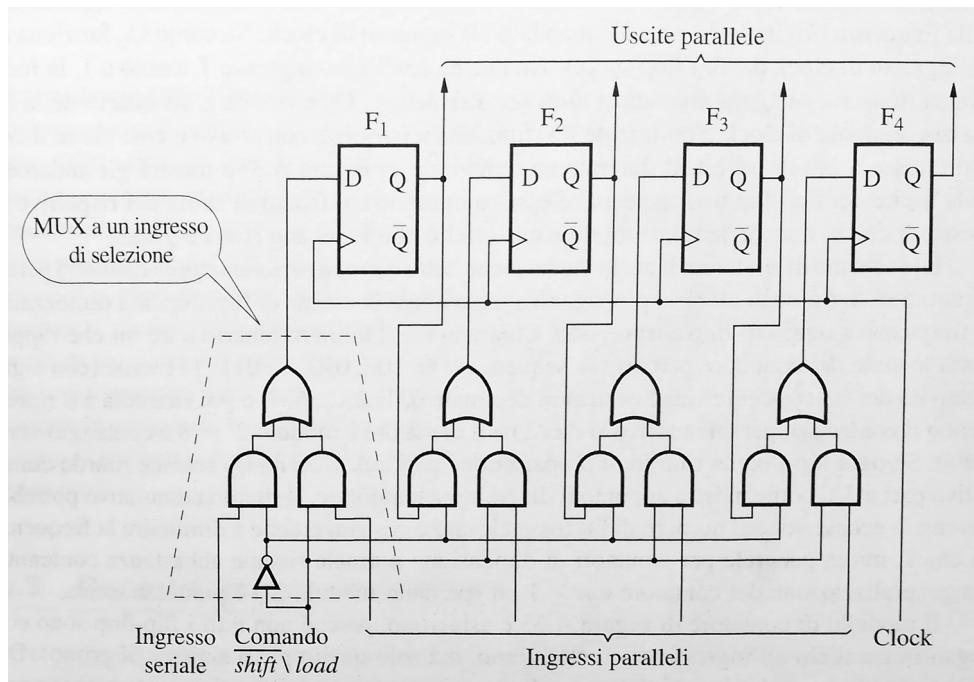
# Registro a scorrimento (shift register)

- Spesso è necessario essere in grado di memorizzare l'informazione serialmente, bit per bit
- I registri a scorrimento sono formati da n flip-flop collegati in serie attraverso le loro uscite/ingresso
- Ricevono 1 bit in ingresso solo al primo flip-flop e ogni ciclo di clock spostano il loro contenuto di una posizione
- L'uscita del registro corrisponde al bit di uscita dell'ultimo flip-flop



# Registro seriale-parallelo

- Il registro seriale parallelo unisce le funzionalità dei due registri visti finora
- Presenta un comando shift/load che permette di cambiare modalità di ingresso:
  - shift/load = 0: shift
  - shift/load = 1: load parallelo
- L'uscita può essere letta in parallelo
- La modalità viene selezionata attraverso n multipliatori a un ingresso di selezione (shift/load)



# Contatore

- Un contatore a  $n$  bit è in grado di percorrere la sequenza di numeri da 0 a  $2^{n-1}$  ciclicamente, incrementando di 1 unità ogni ciclo di clock
- È realizzato attraverso  $n$  flip-flop di tipo T, collegando in serie l'uscita  $\neg Q$  di ogni flip-flop all'ingresso di clock del flip-flop successivo
- Il numero attuale del conteggio è dato dalla stringa delle uscite Q (dal bit meno significativo al più significativo)
- Il contatore può anche essere usato per scalare la frequenza di un segnale di clock

