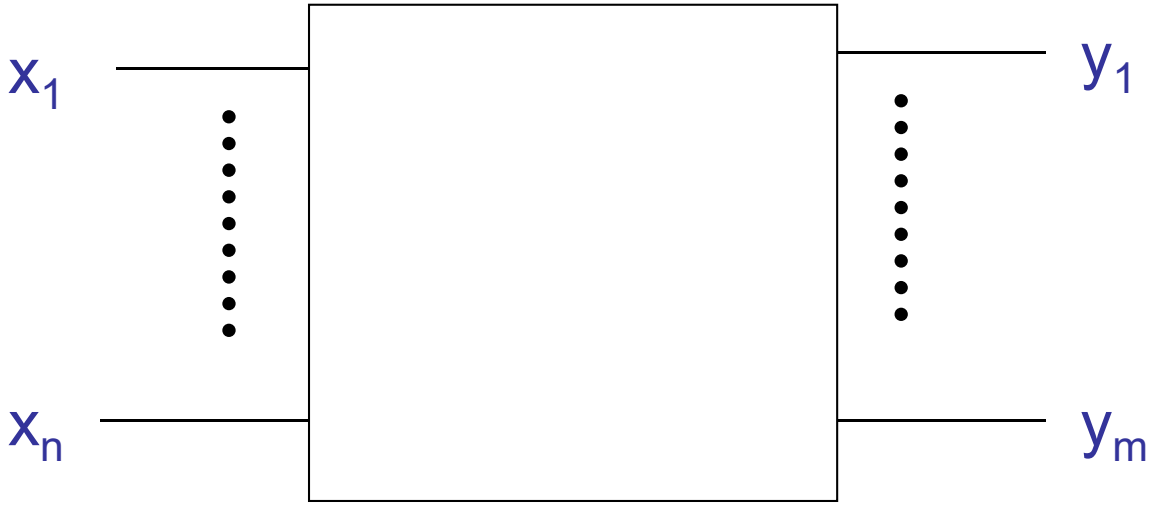
Macchine combinatorie (1)

Reti logiche con *n* ingressi $x_1, x_2, \ldots, x_n$ e *m* uscite $y_1, y_2, \ldots, y_m$ che realizzano la corrispondenza:

$$y_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$\ldots\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n)$$



Macchine combinatorie (2)

In una macchina combinatoria i valori delle uscite dipendono esclusivamente dai valori degli ingressi

In una macchina combinatoria ideale tale dipendenza è istantanea

In una macchina reale c'è sempre un ritardo tra l'istante in cui c'è una variazione in uno degli ingressi e l'istante in cui l'effetto di questa variazione si manifesta sulle uscite

Sintesi di funzioni logiche

- Possiamo sintetizzare una funzione logica a partire dalla sua tabella di verità nella forma *somma di prodotti* o *prodotti di somme.*

x_1	x_2	x_3	f_1	f_2
0	0	0	1	1
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

- Somma di prodotti:
  - Individuare le righe della tabella di verità a valore 1
  - Per ognuna di esse, rappresentare un termine come il prodotto delle variabili in input, prese come x_i se $x_i = 1$, altrimenti (NOT x_i) se $x_i = 0$.
  - Sommare tutti i termini così ottenuti

Minimizzazione delle funzioni logiche

- Costo di una funzione logica: numero di gates + numero di input per ogni porta
- Minimizzare: rendere minimo il costo della rete logica associata ad una funzione logica
- Possiamo minimizzare effettuando operazioni algebriche, tenendo a mente le regole di logica binaria viste in precedenza:
  - Distributività, Idempotenza, Complemento, …

$$f = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3$$

$$f = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3}$$

$$f = \overline{x_1} \cdot \overline{x_2}(\overline{x_3} + x_3) + x_1 \cdot \overline{x_2}(\overline{x_3} + x_3) + \overline{x_1} \cdot \overline{x_3}(\overline{x_2} + x_2)$$

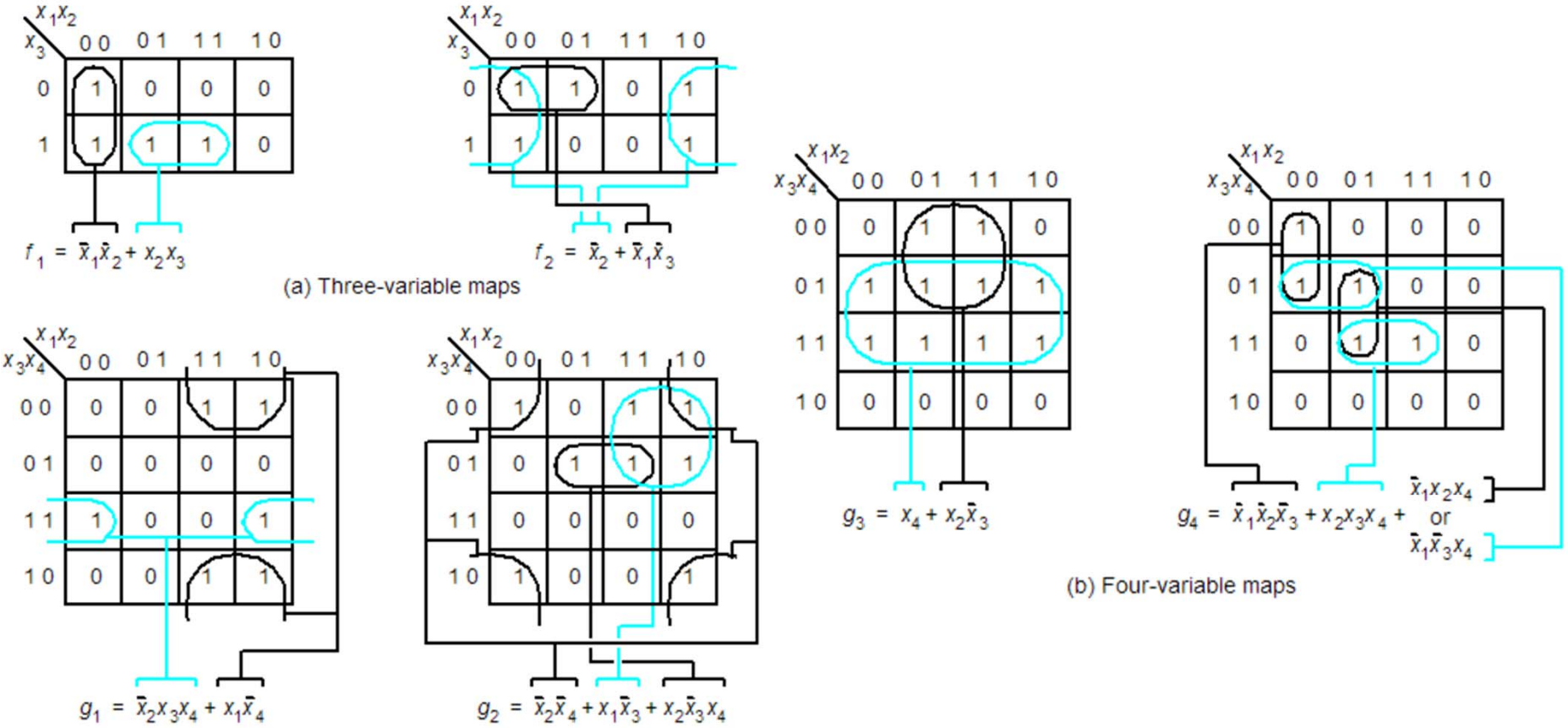
$$= \overline{x_1} \cdot \overline{x_2} + x_1 \cdot \overline{x_2} + \overline{x_1} \cdot \overline{x_3} = \overline{x_2} + \overline{x_1} \cdot \overline{x_3}$$

Minimizzazione con le mappe di Karnaugh (1/2)

- Tecnica geometrica per derivare rapidamente la rappresentazione minima di una funzione logica
- Si riportano su due assi le variabili in input (con una variazione alla volta) e si indica all'interno dei quadrati della mappa il valore della funzione logica corrispondente
- La funzione logica può essere agevolmente rappresentata fino a 4 variabili di input (cioè basta una sola mappa di Karnaugh piana)
- Si raccolgono i quadrati in gruppi di 2^k e si riportano i termini prodotto, indicando le sole variabili che non cambiano.

Minimizzazione con le mappe di Karnaugh (2/2)

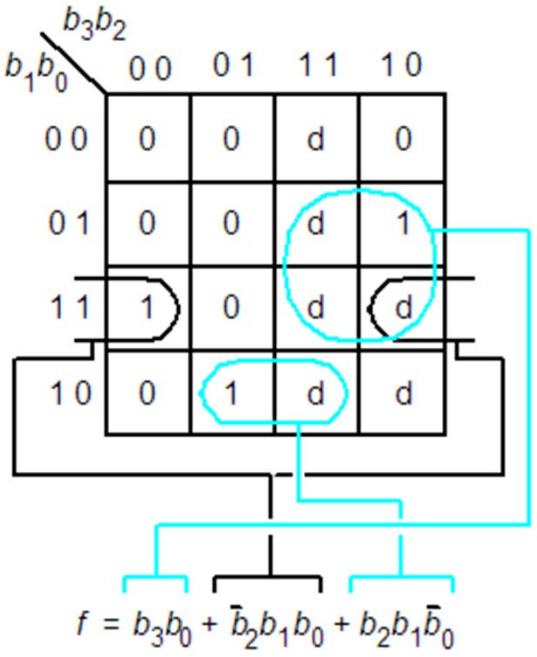
- Esempi:



Variabili Don't Care

- *Don't Care*: variabili che possono assumere indifferentemente valore 0 o 1. Sono indicate con d nella tavola di verità.
- Esempio: BCD a 4 bit

Decimal digit represented	Binary coding $b_3\ b_2\ b_1\ b_0$	f
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	0
5	0 1 0 1	0
6	0 1 1 0	1
7	0 1 1 1	0
8	1 0 0 0	0
9	1 0 0 1	1
unused	1 0 1 0	d
	1 0 1 1	d
	1 1 0 0	d
	1 1 0 1	d
	1 1 1 0	d
	1 1 1 1	d



Esercizio

- Trovare l'espressione delle seguenti funzioni logiche nella forma somma di prodotti
- Minimizzarla mediante tavola di Karnaugh
- Verificare il risultato ottenuto con LogiSim

x_1	x_2	x_3	f_1	f_2	f_3	f_4
0	0	0	1	1	d	0
0	0	1	1	1	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	d
1	0	0	1	0	d	d
1	0	1	0	0	0	d
1	1	0	1	0	1	1
1	1	1	1	1	1	0

Figure PA.1 Logic functions for Problem A.

NAND e NOR gates

- Implementazione elettronica più vantaggiosa

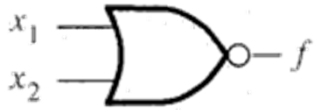
x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

$f = x_1 \uparrow x_2 = \overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$

$f = x_1 \downarrow x_2 = \overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

(a) NAND

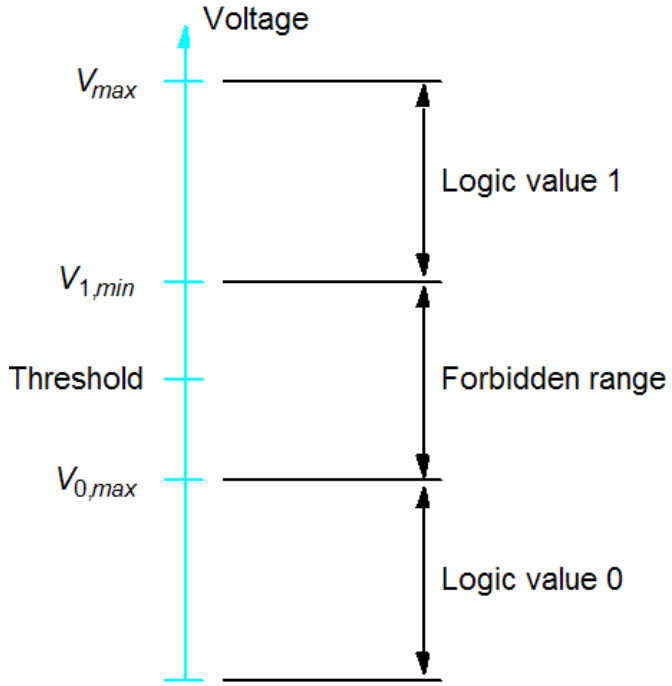


(b) NOR

- Una NAND (o NOR) è un insieme funzionalmente completo
  - Posso rappresentare una funzione somma di prodotti con sole NAND
- Non è applicabile la proprietà associativa

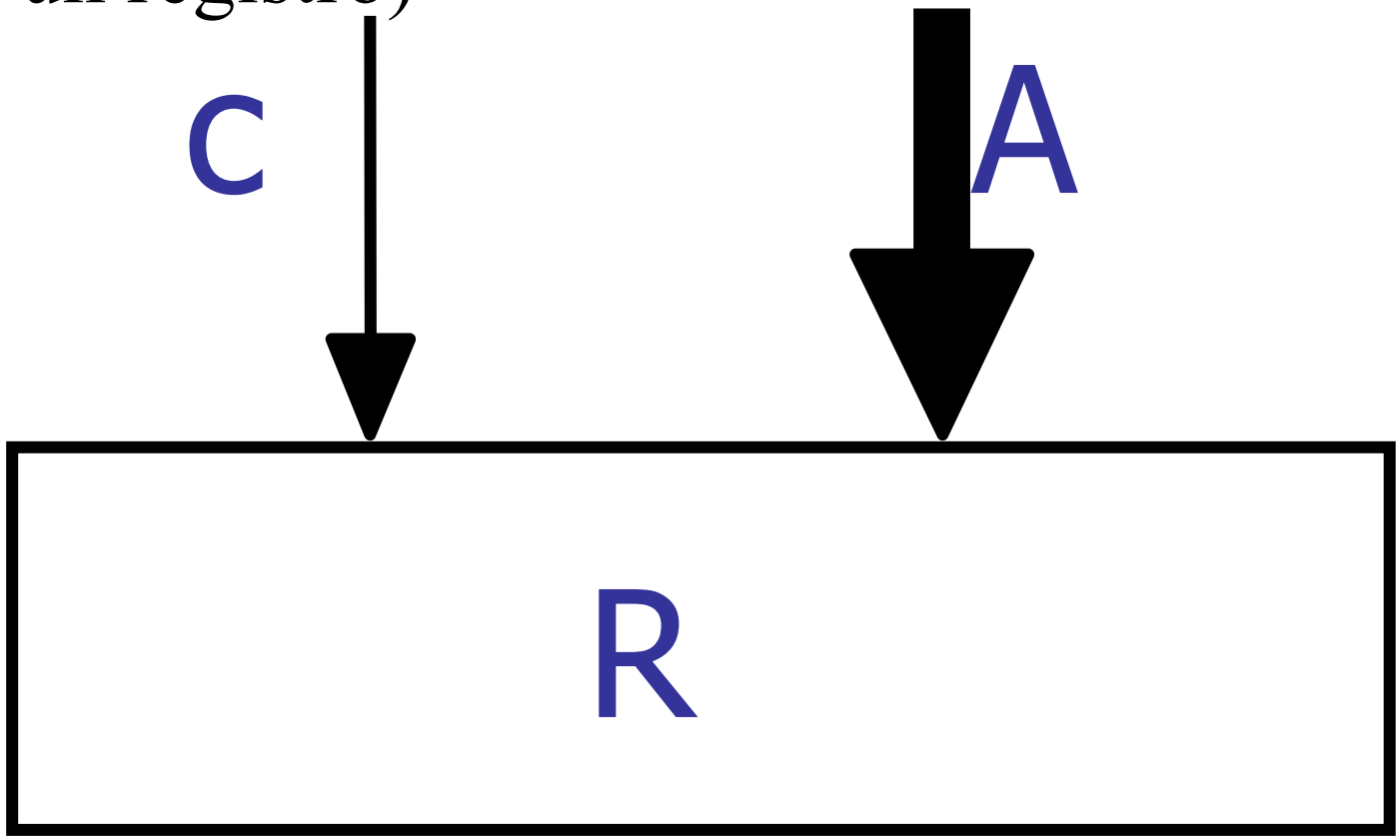
Implementazione reale delle porte logiche

- I valori logici sono rappresentati mediante correnti o tensioni
  - I due valori sono separati da una soglia (threshold)
  - In realtà vi è una regione interdetta ai valori logici (forbidden range) per contemplare eventuali disturbi elettrici (noise)



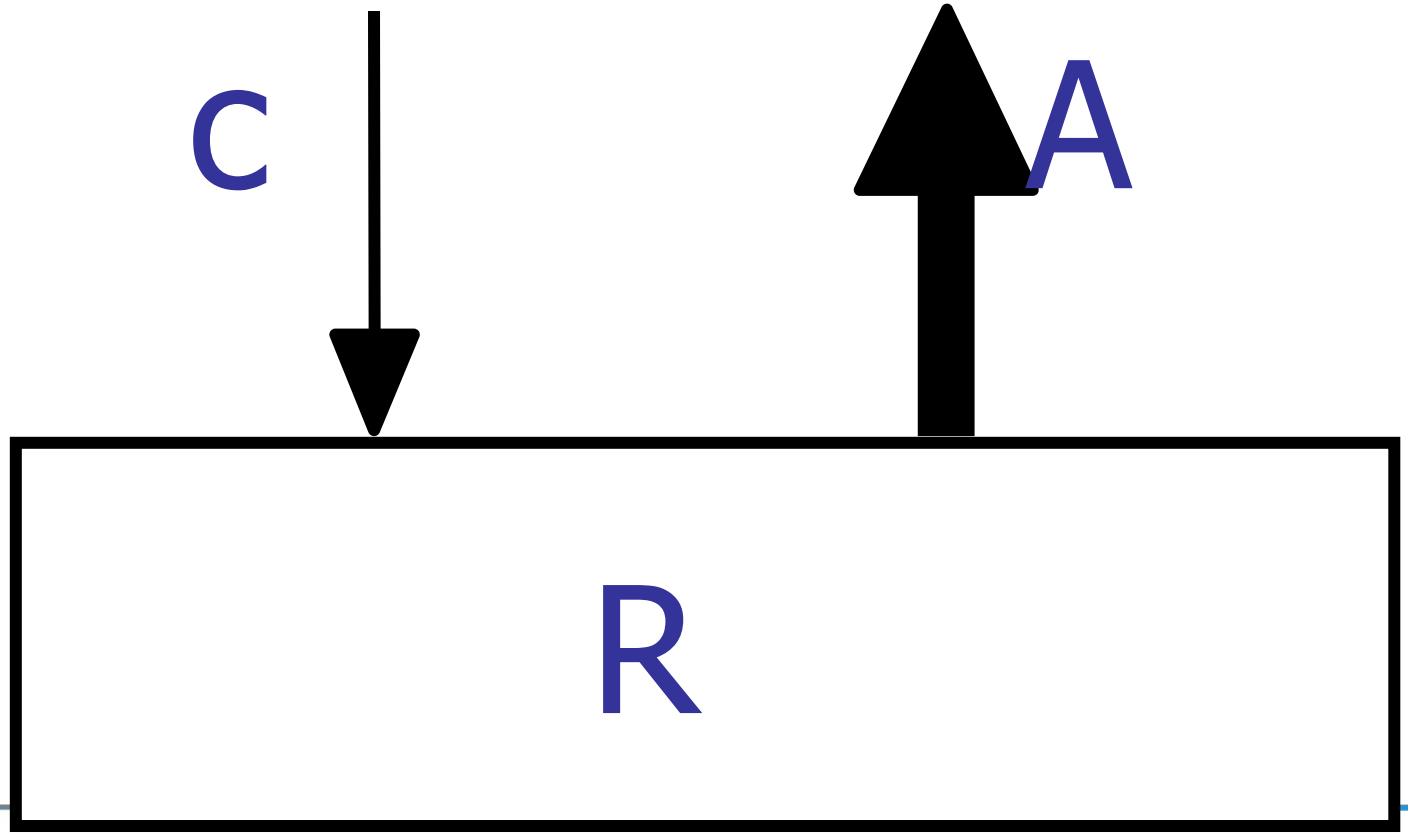
Trasferimento dati su bus unico -1/2

- Trasferimento da bus a registro (caricamento di un registro)

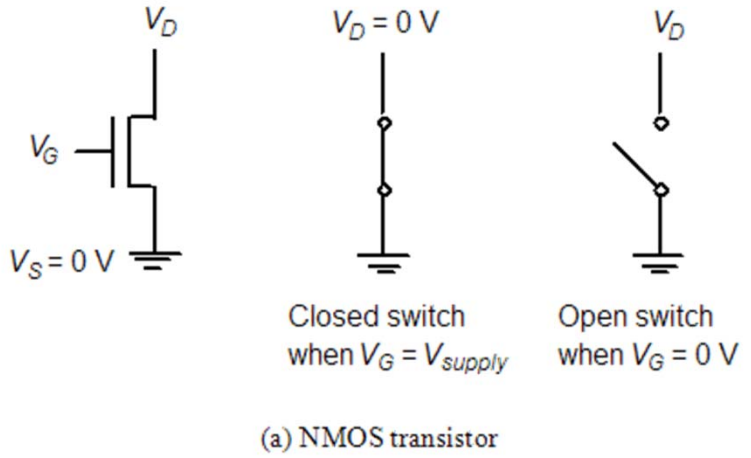


Trasferimento dati su bus unico -2/2

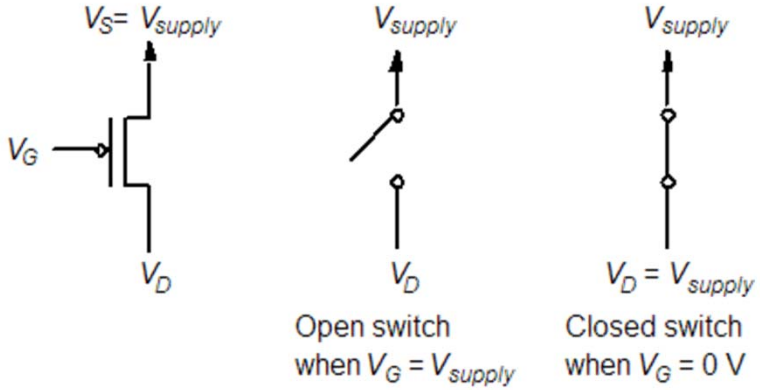
- Trasferimento da registro a bus (caricamento da un registro)



Transistor NMOS e PMOS

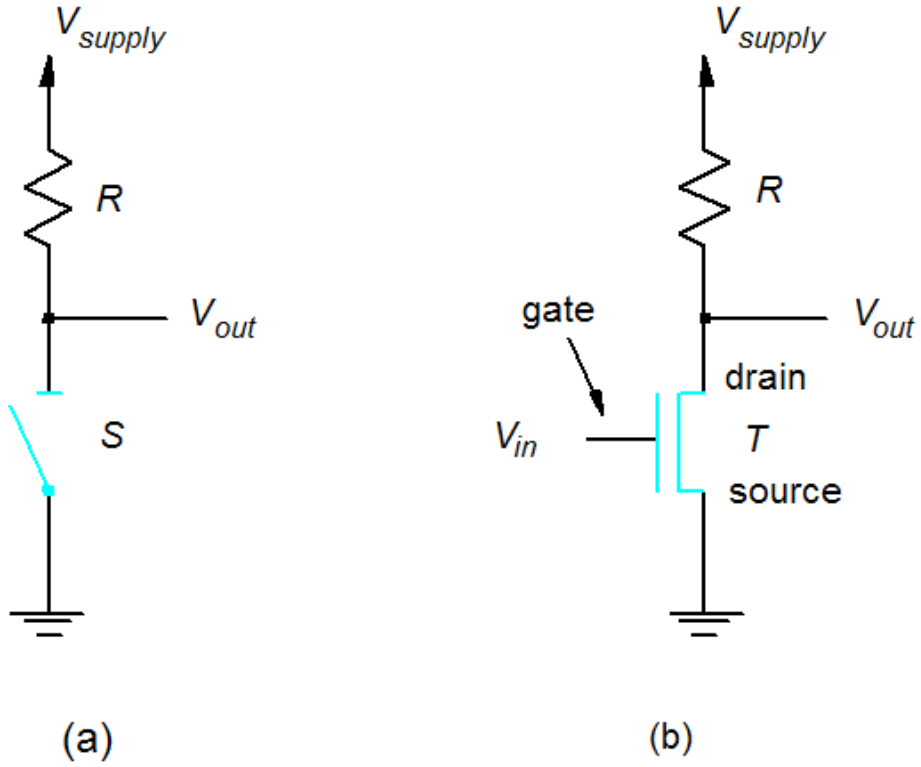


(a) NMOS transistor



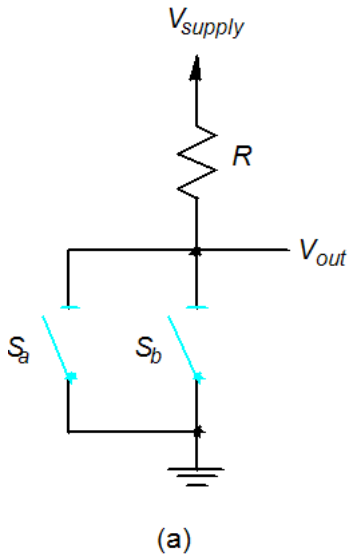
(b) PMOS transistor

Circuito Invertitore

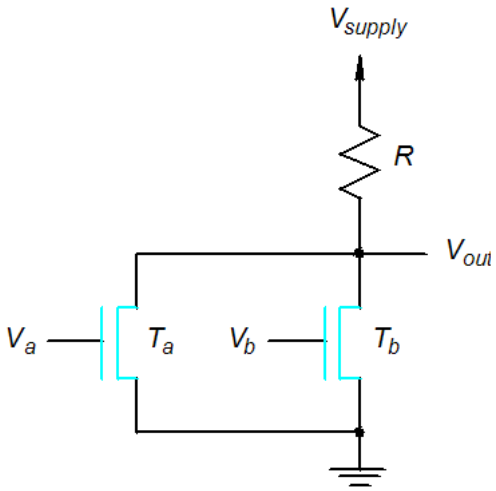


Se Vin=0 allora Vout=Vsupply
Se Vin=Vsupply allora Vout=0

Circuiti per NAND e NOR

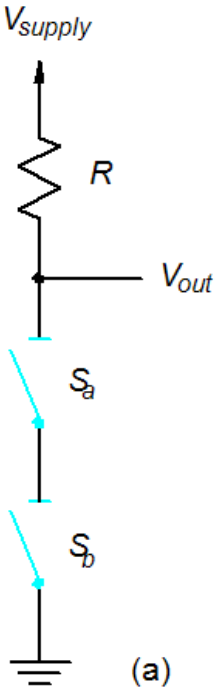


(a)

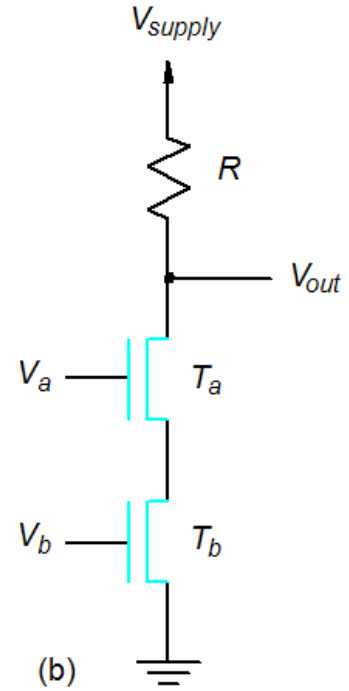


(b)

NOR



(a)

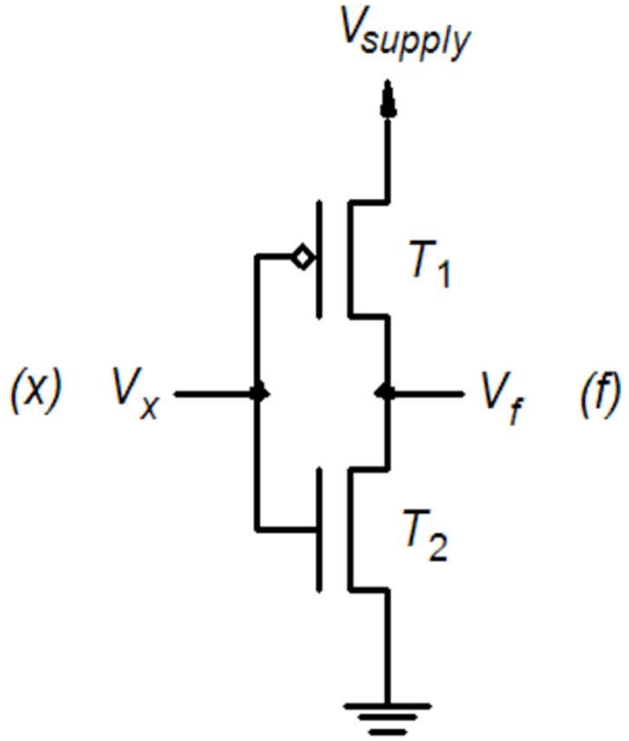


(b)

NAND

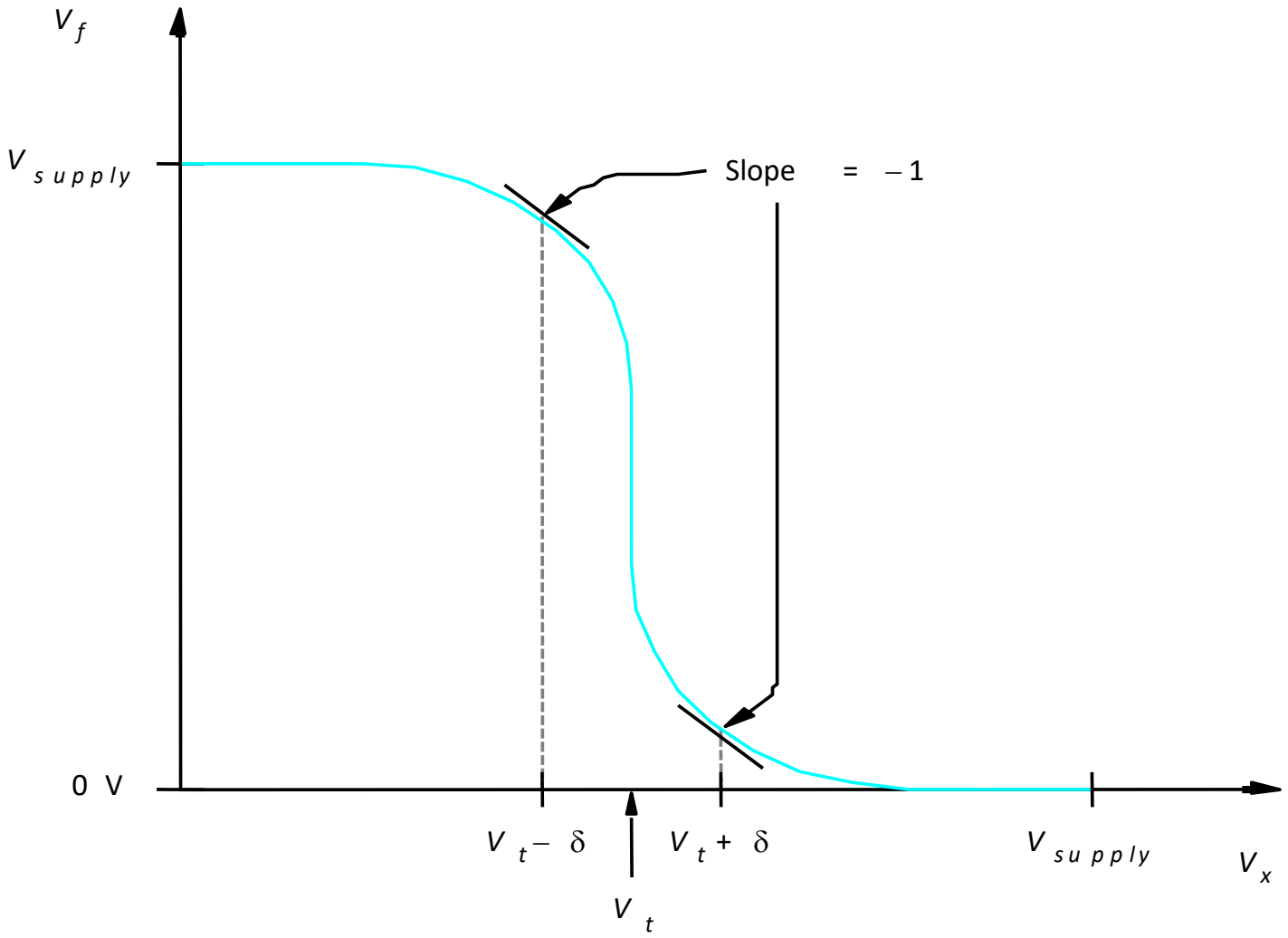
Invertitore CMOS

Problema energetico: quando il percorso verso terra è chiuso, la corrente passa attraverso il resistore, che dissipa energia sotto forma di calore.
Soluzione: sostituire il resistore con un transistor PMOS. In questo modo non c'è mai un percorso chiuso per la corrente verso terra, tranne in fase di transizione degli stati (nota: la potenza dissipata dipende dal numero di transizioni di stato in questo caso)



x	V_x	T_1	T_2	V_f	f
0	low	on	off	high	1
1	high	off	on	low	0

Caratteristica di trasferimento del CMOS



Circuiti CMOS

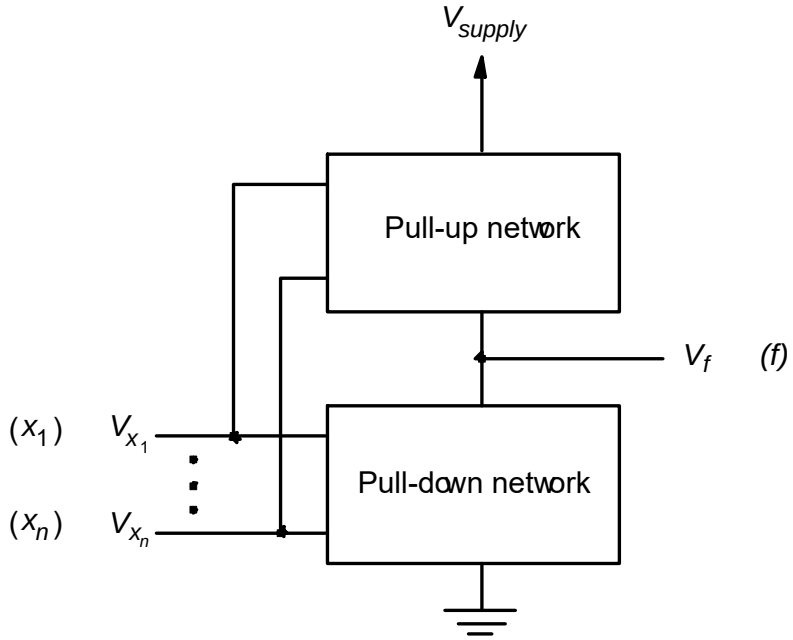
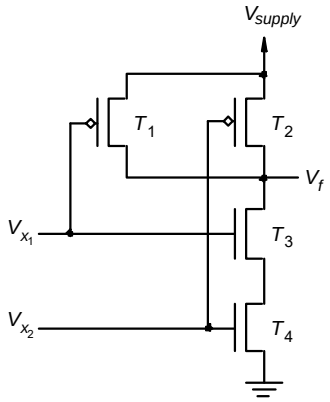


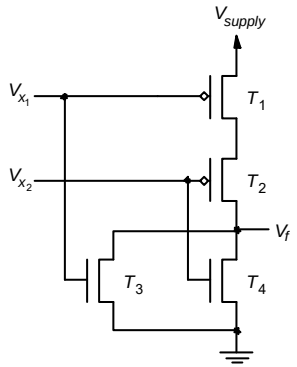
Figure A.16. Structure of a CMOS circuit.



(a) Circuit

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(b) Truth table and transistor states



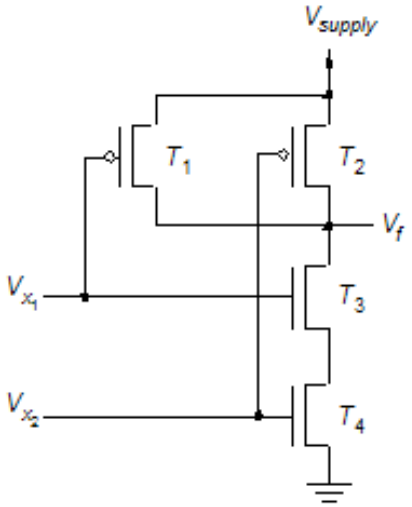
(a) Circuit

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	0
1	0	off	on	on	off	0
1	1	off	off	on	on	0

(b) Truth table and transistor states

Figure A.18. CMOS realization of a NOR gate.

NAND vs AND

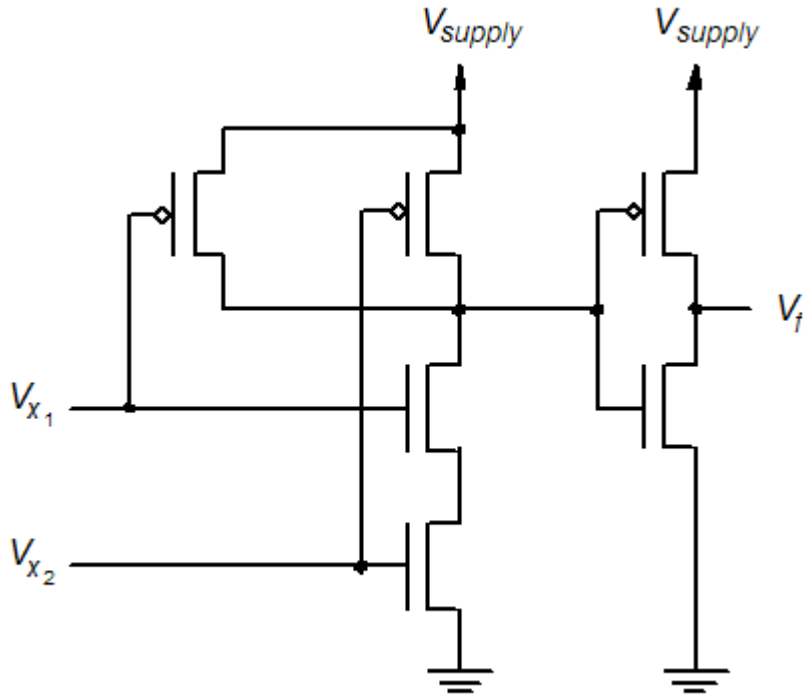


(a) Circuit

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(b) Truth table and transistor states

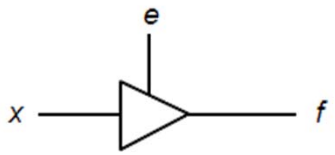
NAND



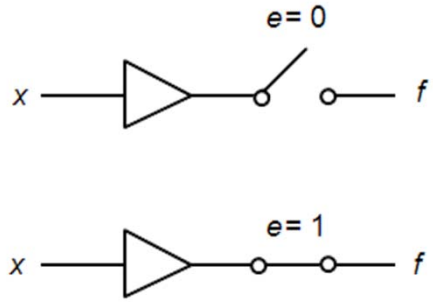
AND

Buffer a tre stati

- Si introduce un terzo stato (high-impedance) per disconnettere elettricamente un dispositivo dal circuito
- È la tecnica più diffusa per realizzare il collegamento di più registri "sorgenti" verso un bus comune



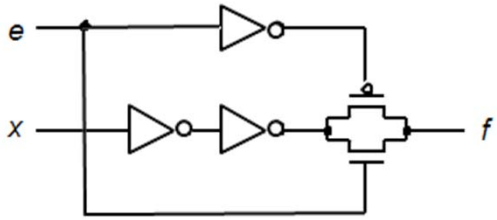
(a) Symbol



(b) Equivalent circuit

e	x	f
0	0	Z
0	1	Z
1	0	0
1	1	1

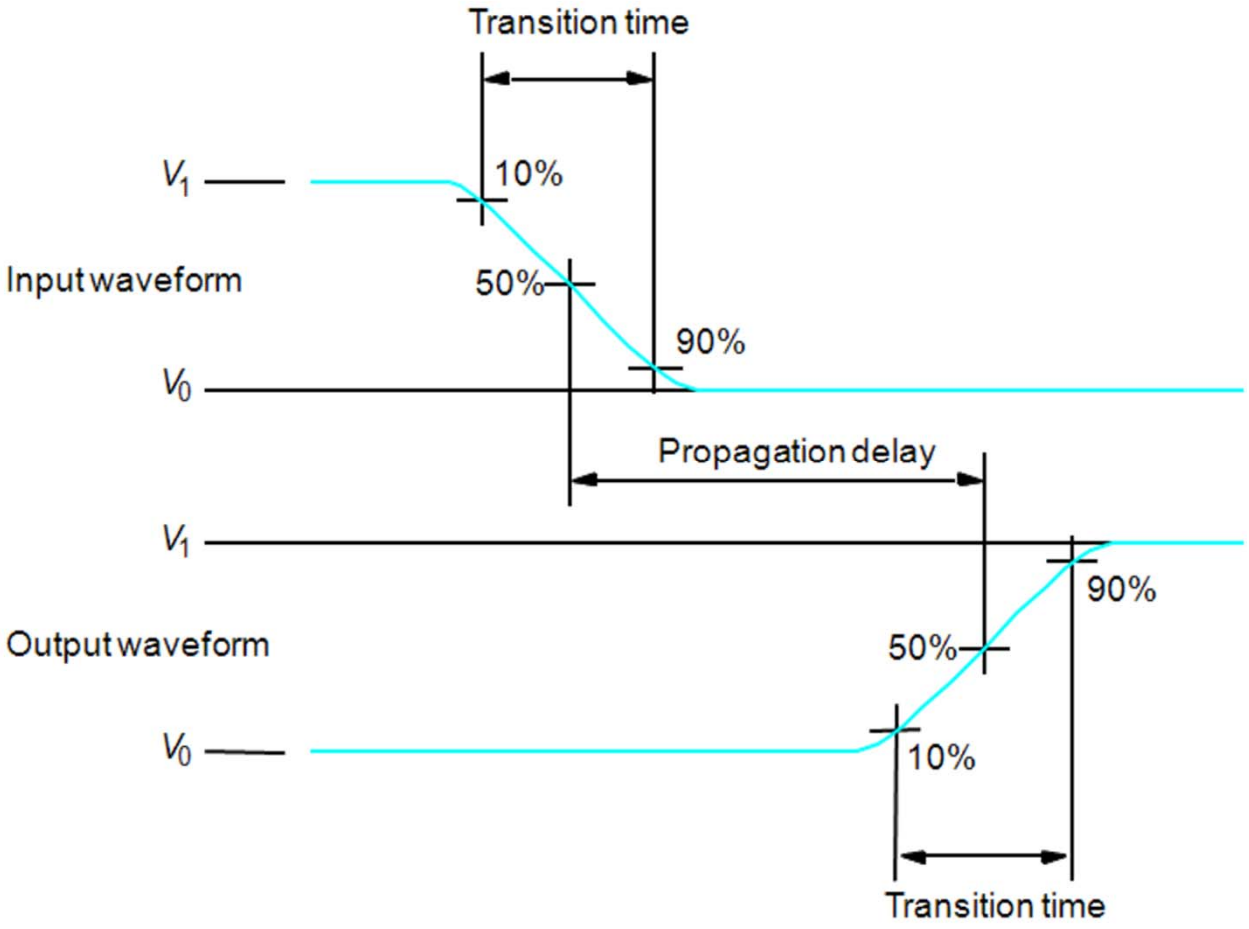
(c) Truth table



(d) Implementation

Ritardo di propagazione

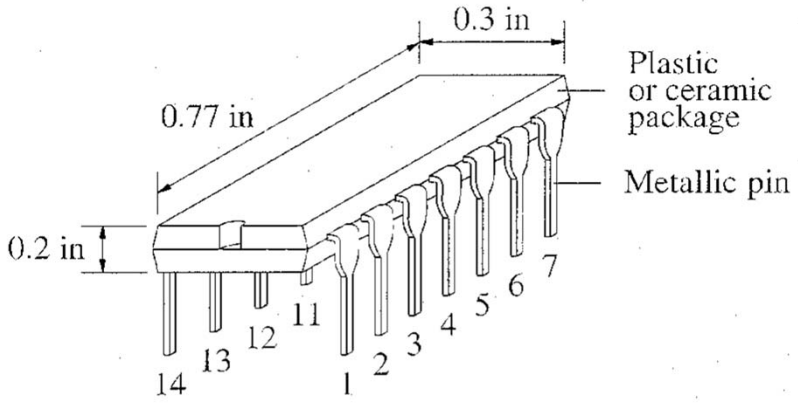
Maggiore è il ritardo di propagazione, minore è la velocità massima del circuito logico



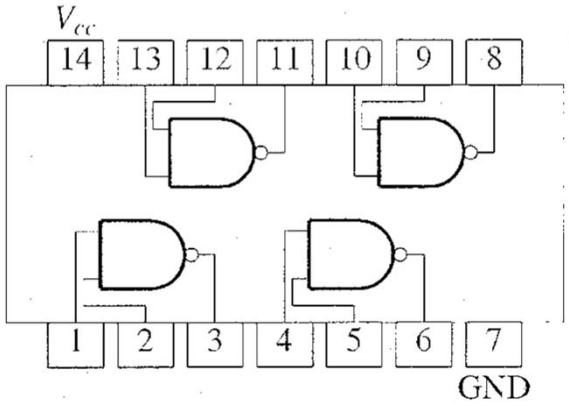
Fan-In e Fan-Out

- Fan-In: numero di ingressi alle porte logiche di un circuito
  - Maggiore è il Fan-In, maggiore è il ritardo di propagazione del circuito logico
  - Nel caso di CMOS un nuovo ingresso implica un nuovo PMOS e un nuovo NMOS
- Fan-Out: numero di uscite che pilotano ulteriori ingressi
- Solitamente il Fan-In e il Fan-Out sono in numero limitato per non far decrescere la velocità del circuito.
  - Per successivi ingressi si adopera un nuovo gate

Package di Circuiti Integrati (IC)



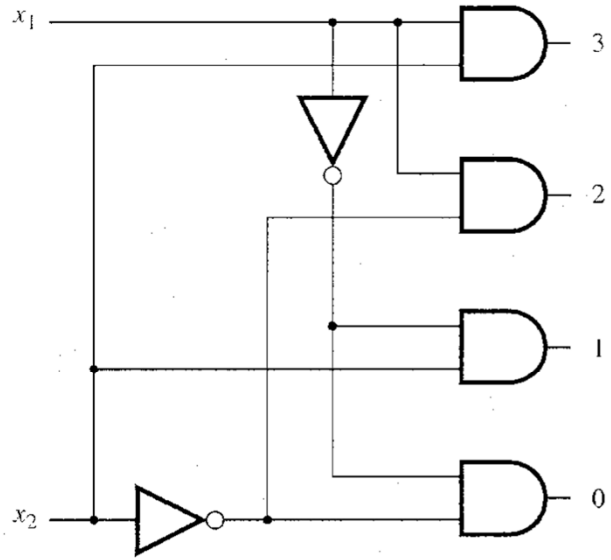
(a) Physical appearance



(b) Schematic of an integrated circuit providing four 2-input NAND gates

Decoder

- Traduce un input di n bit (input codificato), in uno stato di uscita (output decodificato)
  - Es.: n bit input -> 2^n stati di uscita



x_1	x_2	Active output
0	0	0
0	1	1
1	0	2
1	1	3

Figure A.36. A two-input to four-output decoder.

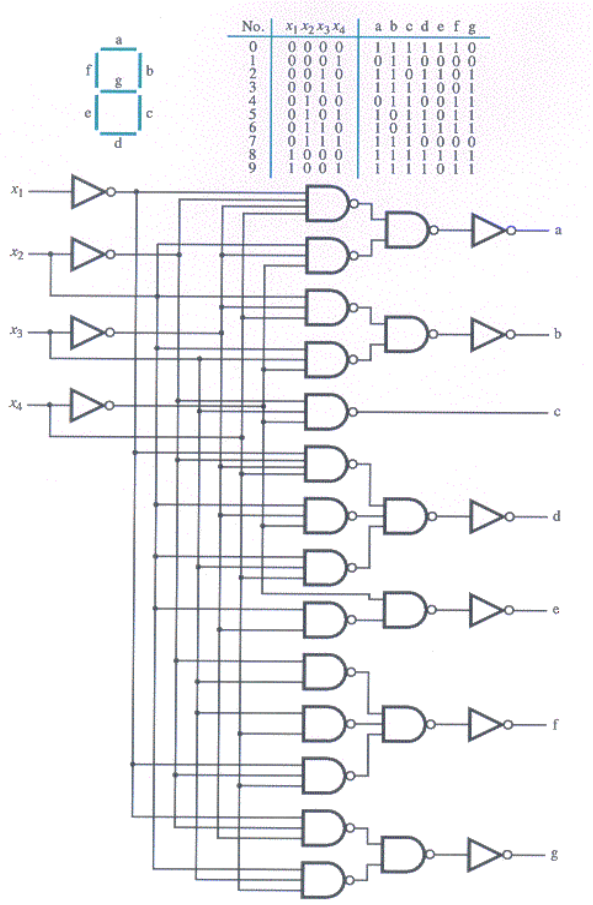
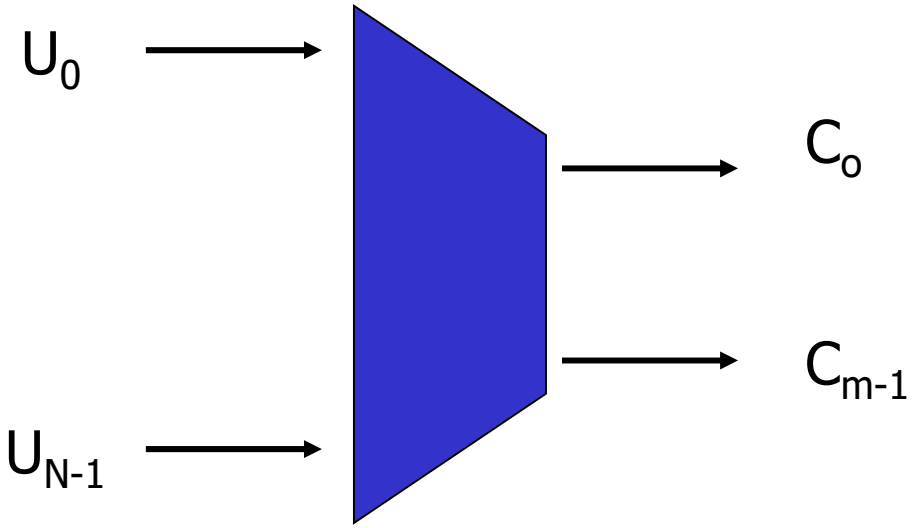


Figure A.37. A BCD to seven-segment display decoder.

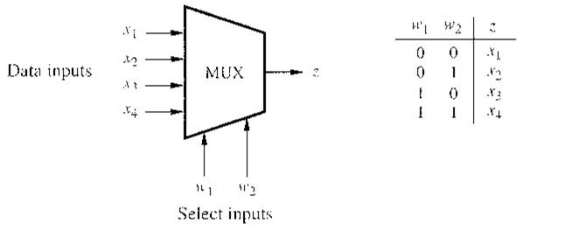
Codificatore (Encoder)

- Un codificatore è una macchina che riceve in ingresso una rappresentazione decodificata (linee U_0,.., U_{N-1}) e fornisce in uscita la parola codice associata



Multiplexer

- Circuito che sceglie uno degli input da portare in output
  - Gli input di selezione sono detti appunto “select”
  - Es.: per 2^k input servono k select



w_1	w_2	z
0	0	x_1
0	1	x_2
1	0	x_3
1	1	x_4

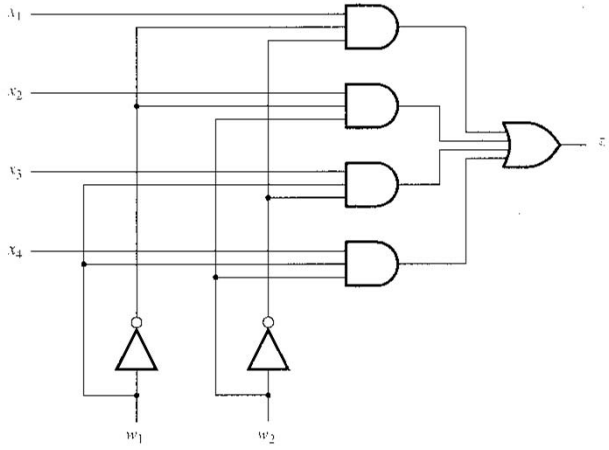


Figure A.38. A four-input multiplexer.

Può essere utilizzato per implementare una qualunque funzione:

e.g.
4 input mux -> f(x1,x2,x3)

8 input mux-> F(x1,x2,x3,x4)
…

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

⇒

x_1	x_2	f
0	0	0
0	1	x_3
1	0	1
1	1	$\bar{x}_3$

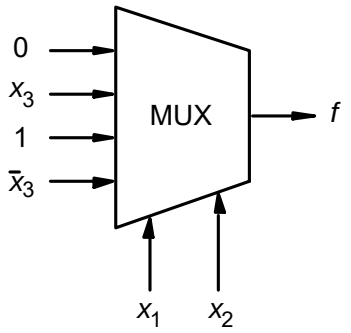
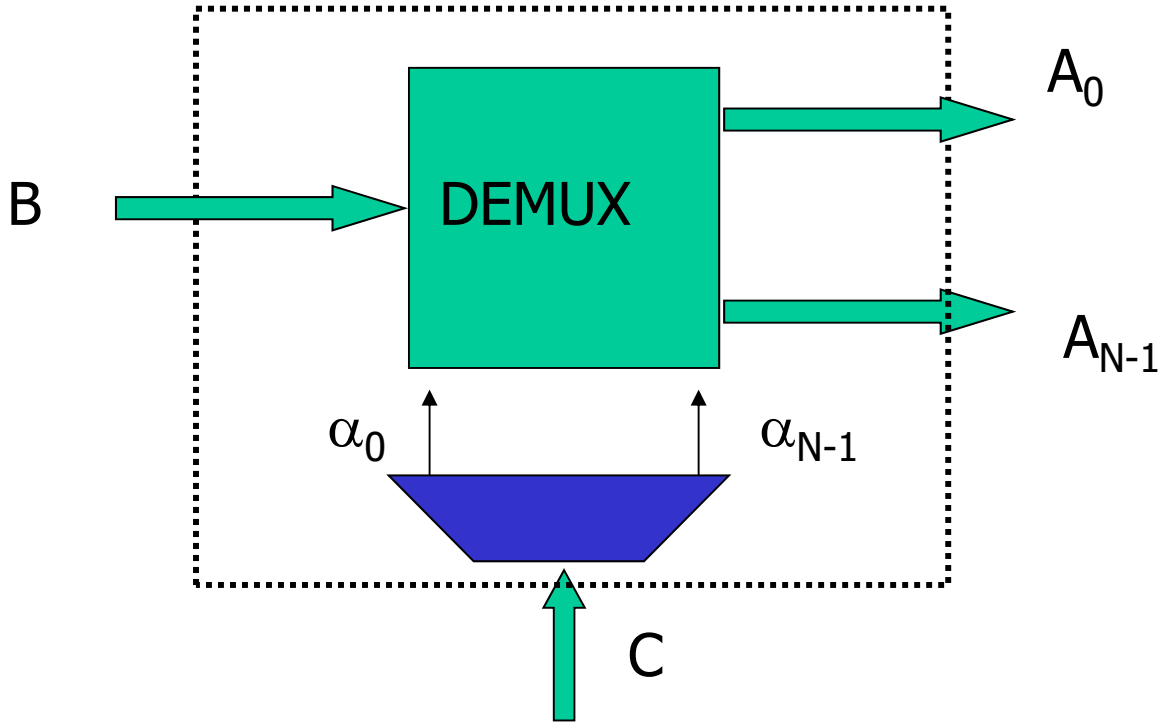


Figure A.39. Multiplexer implementation of a logic functi

Demultiplexer Indirizzabile

- E’ un Demultiplexer Lineare i cui segnali di abilitazione sono collegati con le uscite di un decodificatore

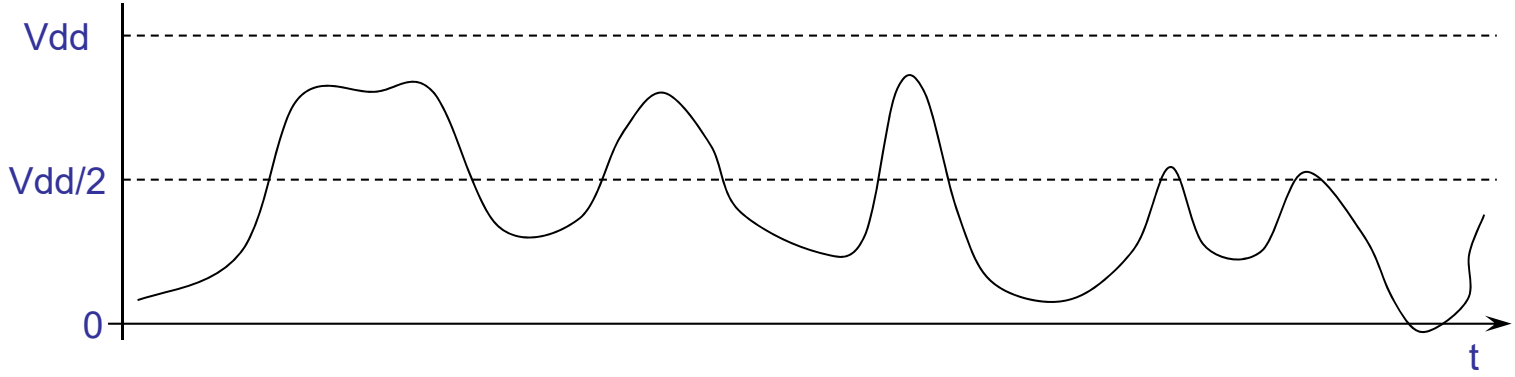


I circuiti digitali

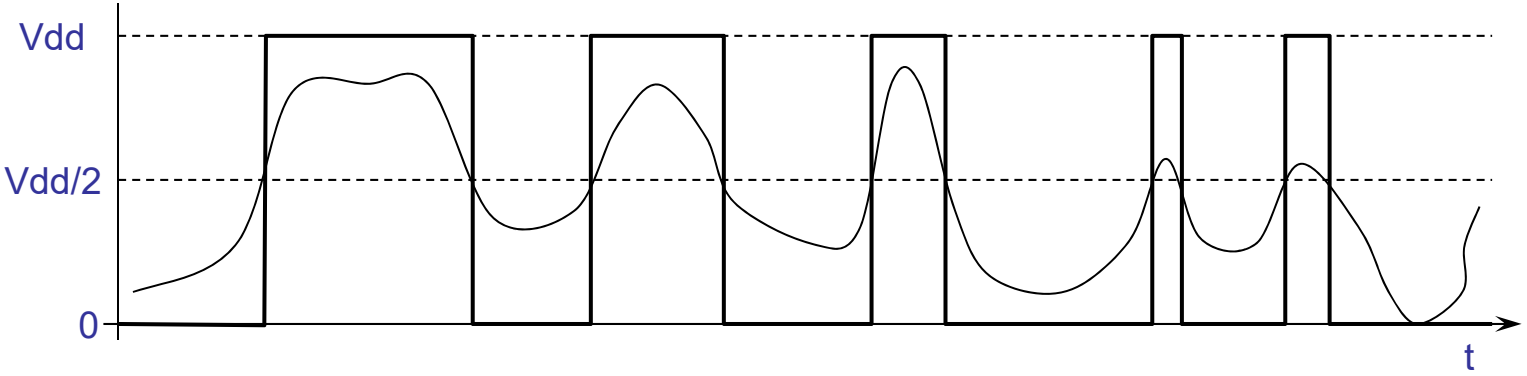
- I circuiti digitali possono essere classificati in due categorie
- Circuiti combinatori
  - Il valore delle uscite ad un determinato istante dipende unicamente dal valore degli ingressi in quello stesso istante.
- Circuiti sequenziali
  - Il valore delle uscite in un determinato istante dipende sia dal valore degli ingressi in quell'istante sia dal valore degli ingressi in istanti precedenti
  - Per definire il comportamento di un circuito sequenziale è necessario tenere conto della storia passata degli ingressi del circuito
- La definizione di circuito sequenziale implica due concetti:
  - Il concetto di tempo
  - Il concetto di stato

Il concetto di tempo

- Un segnale elettrico è una tensione variabile nel tempo

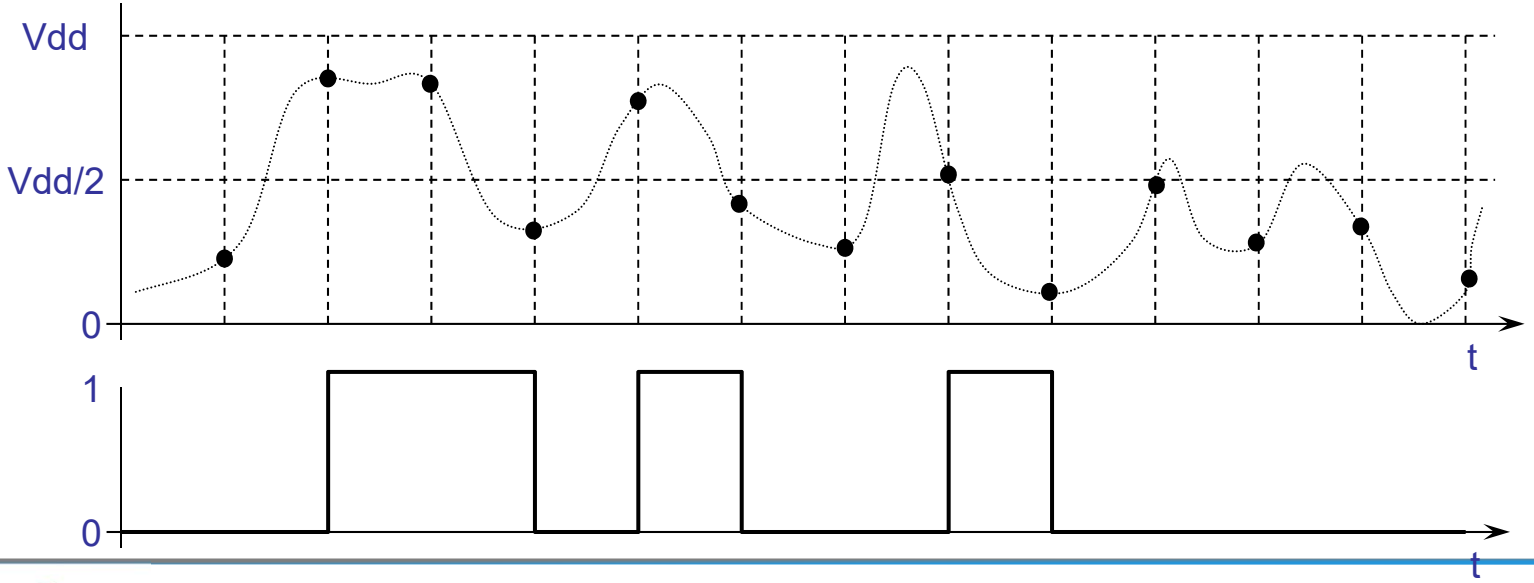


- I segnali binari sono rappresentati tipicamente mediante due livelli di tensione di un segnale elettrico.



Il concetto di tempo

- Il segnale binario è un segnale variabile con continuità
- In un intervallo di tempo $t=t_1-t_0$ il segnale assume infiniti valori, corrispondenti agli infiniti istanti tra t_0 e t_1
- Si ricorre al concetto di *tempo discreto* in cui il numero di istanti discreti in un intervallo $t=t_1-t_0$ è finito

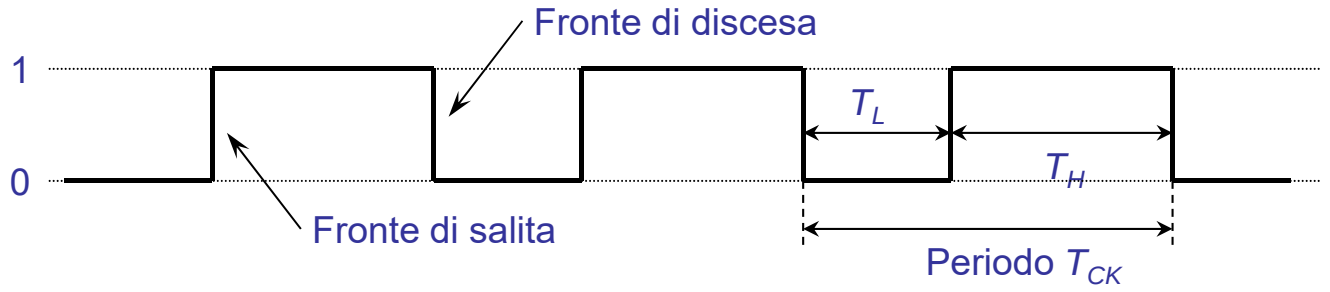


Il concetto di tempo

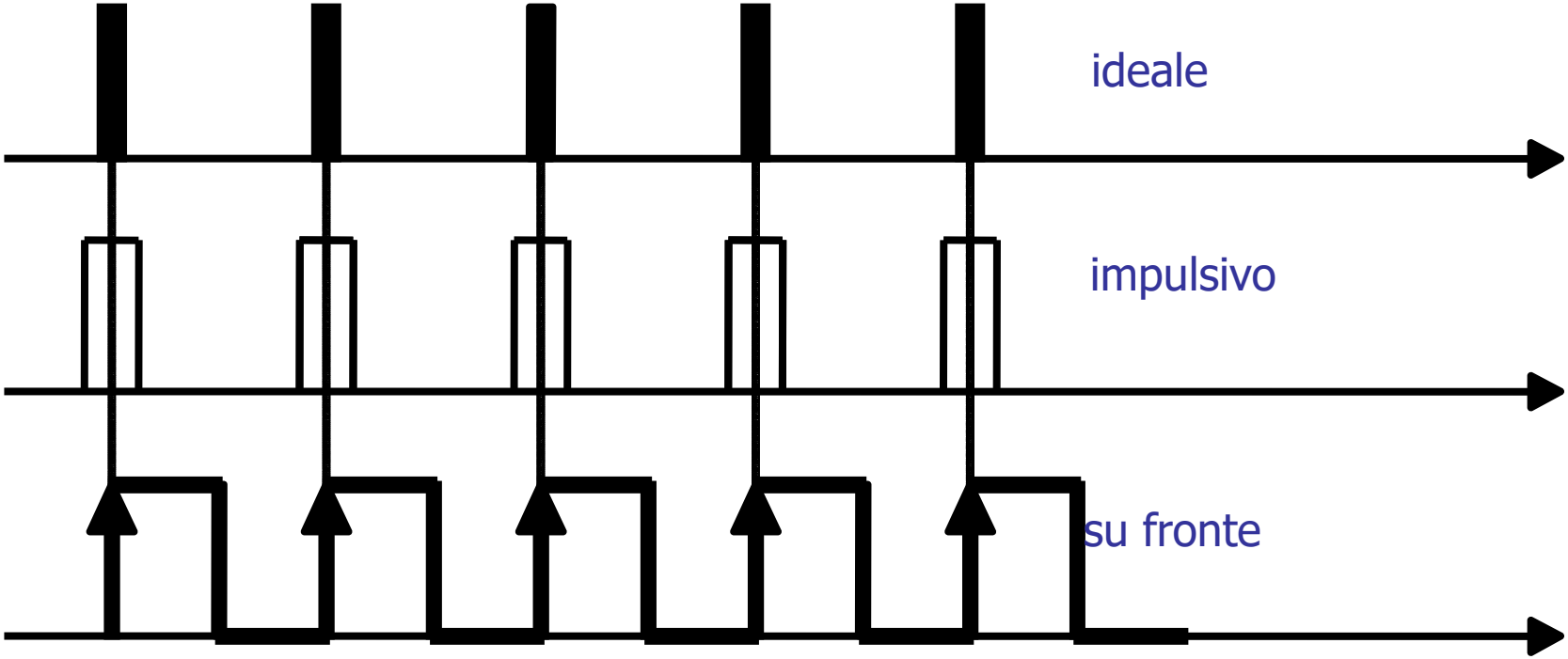
- Il valore del segnale elettrico viene letto o *campionato* in istanti determinati
- Gli istanti in cui deve essere *campionato* il segnale elettrico sono scanditi da un apposito segnale detto *clock*
- Un *clock* ha le seguenti caratteristiche:
  - E' un segnale binario
  - E' un segnale periodico

Il concetto di tempo

- Nel periodo T_{CK}, o *ciclo di clock*, il segnale assume:
  - Il valore logico 1 per un tempo T_H
  - Il valore logico 0 per un tempo T_L
- Il rapporto T_H / T_{CK} è detto *duty-cycle*
- Il passaggio dal valore 0 al valore 1 è detto *fronte di salita*
- Il passaggio dal valore 1 al valore 0 è detto *fronte di discesa*



Clock

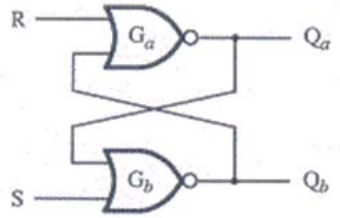


Elementi Bistabili

- Come detto, lo stato di un circuito deve essere memorizzato. A tale scopo si utilizzano degli elementi di memoria detti *bistabili.*
- Il termine *bistabili* deriva dal fatto che tali elementi possono assumere solo due valori: 0 e 1
- Esistono diversi tipi di bistabili che differiscono per il numero di ingressi e per il comportamento
- I flip-flop vengono utilizzati sia come elemento fondamentale per la costituzione dei registri sia come elemento ausiliario per la costruzione di macchine sequenziali più complesse.

SR latch

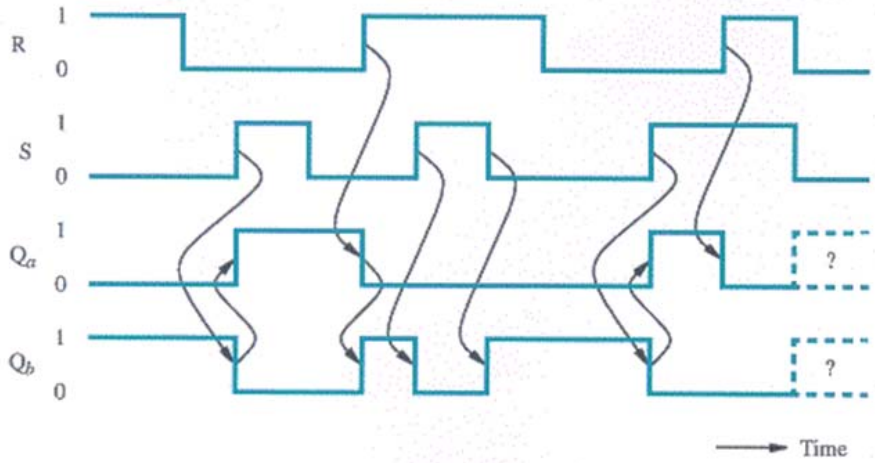
- Un *latch* è un dispositivo per memorizzare un dato binario.
- E' un circuito sequenziale



(a) Network

S	R	Q_a	Q_b	
0	0	0/1	1/0	(No change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

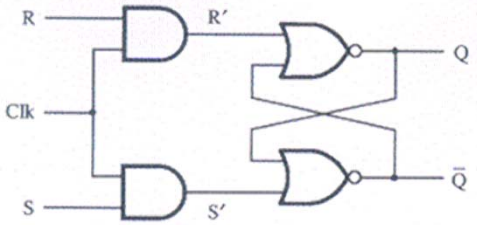
(b) Truth table



(c) Timing diagram

Gated SR latch

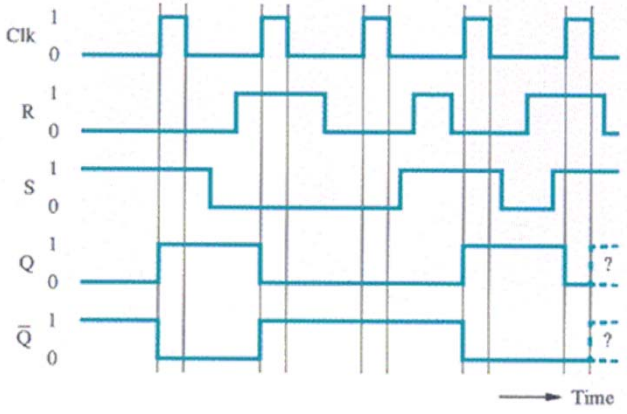
- Si introduce un input di controllo detto Clock (Clk)
  - Se Clk=1, il circuito segue il comportamento dettato da S e R. Se Clk=0 il circuito non cambia stato.
  - In presenza di abilitazione si parla di *gated latch*



(a) Circuit

Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x

(b) Truth table



(c) Timing diagram

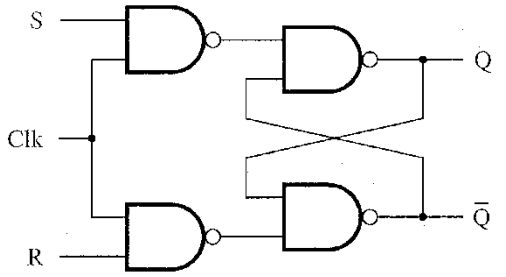
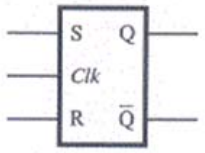


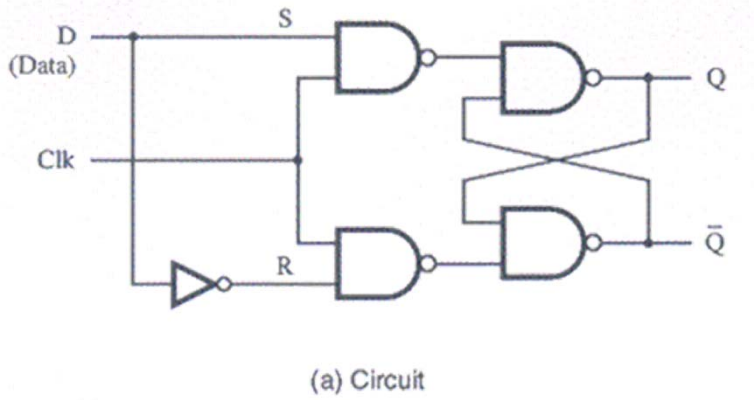
Figure A.26. Gated SR latch implemented with NAND gates.



(d) Graphical symbol

Gated D latch

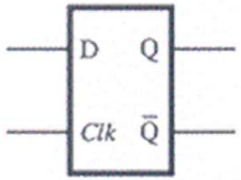
- S ed R sono comandati da un solo input, detto D (più il Clock).
- S ed R non varranno mai contemporaneamente 0,0 o 1,1
- L'uscita segue sempre il valore di D (se Clk=1).



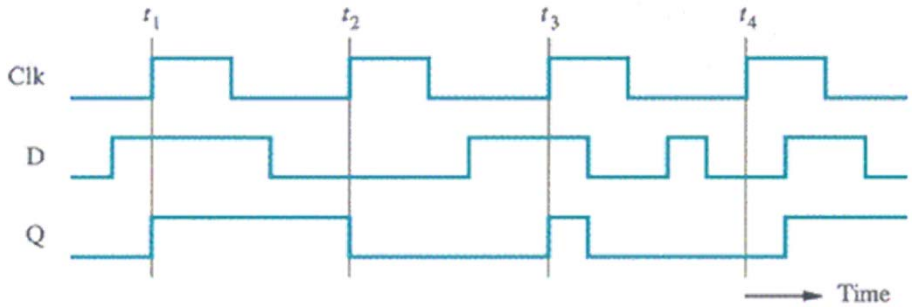
(a) Circuit

Clk	D	Q(t + 1)
0	x	Q(t)
1	0	0
1	1	1

(b) Truth table



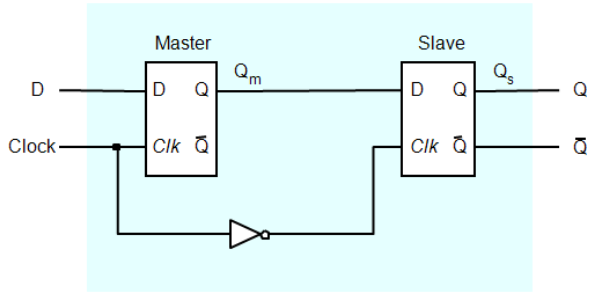
(c) Graphical symbol



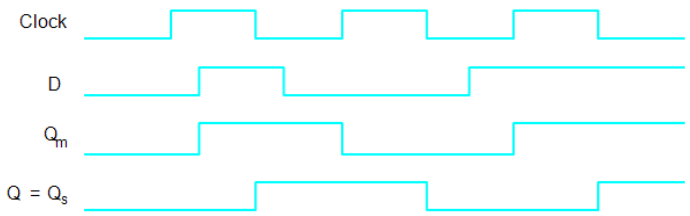
(d) Timing diagram

Flip-Flop D Master-Slave

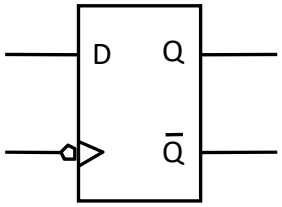
- Flip-Flop: elemento di memoria che cambia stato (cioè si abilita) sul fronte di un clock
- Vogliamo introdurre un ritardo tra l'arrivo di un nuovo input e l'uscita del circuito -> Circuito Master-Slave
- Se Clk=1 il Master segue D e lo Slave non cambia stato. Se Clk=0, Q=D.
- L'uscita cambia sul fronte di discesa del clock (fronte negativo) -> Inverto il clock (fronte positvo)



(a) Circuit



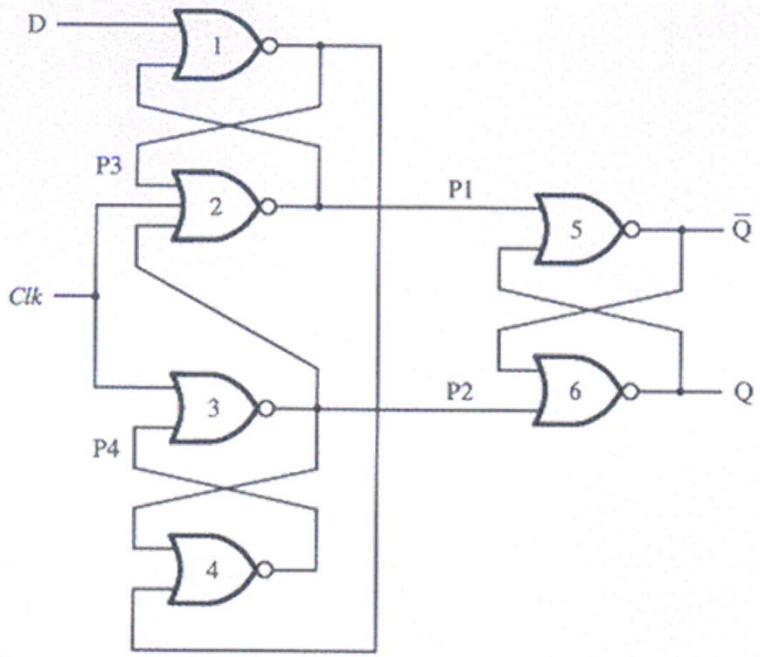
(b) Timing diagram



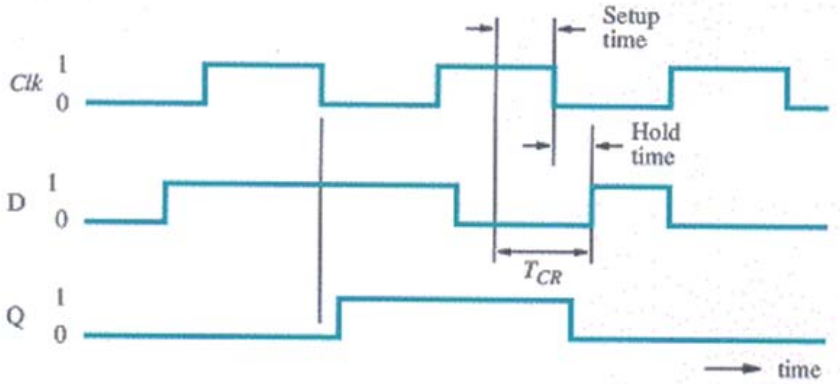
(c) Graphical symbol

Flip-Flop D edge triggered

- I Flip-Flop cambiano stato solo durante le transizioni del clock (fronte positivo o negativo). Il tempo di transizione deve essere breve.
- La tempistica deve essere ben definita. Tcritical = Hold Time + Setup Time
- Tcr è l'intervallo in cui D non deve cambiare il suo valore



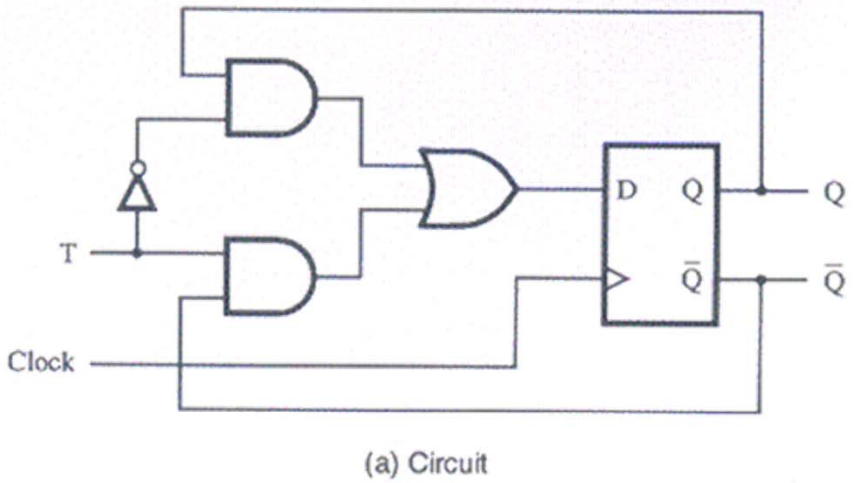
(a) Network



(b) Example of timing

Flip-Flop T

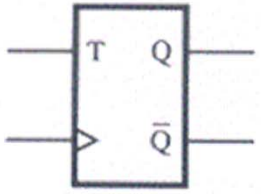
- Se Clk=1 e T=1, il flip-flop cambia (toggles) stato.
- E' usato per i circuiti contatori



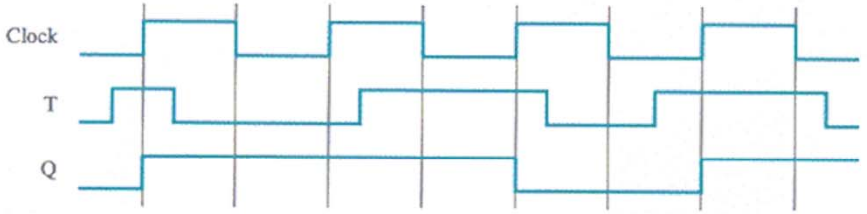
(a) Circuit

T	Q(t + 1)
0	Q(t)
1	$\bar{Q}(t)$

(b) Truth table



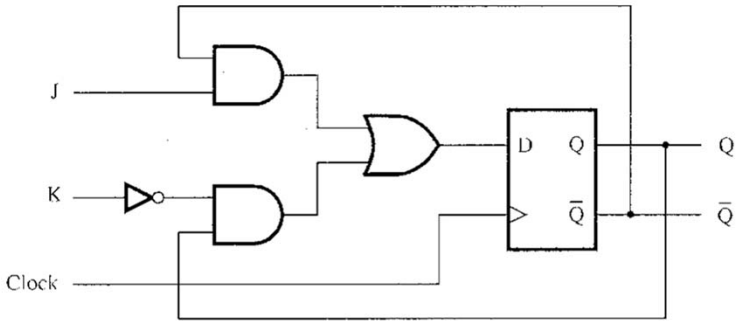
(c) Graphical symbol



(d) Timing diagram

Flip-Flop JK

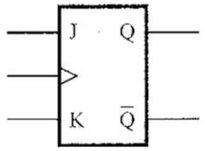
- Ha un comportamento simile all'SR e al T e quindi può essere utilizzato sia come memoria, sia per i counter



(a) Circuit

J	K	Q(t + 1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q̄(t)

(b) Truth table



(c) Graphical symbol

Figure A.31. JK flip-flop.

Flip-Flop con Preset e Clear

- Due input aggiuntivi che forzano lo stato del Flip-Flop a 0 o 1
  - Es. all'accensione del computer tutti i f-f dovrebbero essere a 0
  - Se P,C =1 il f-f è controllato da Clk e D. Se P=0,Q=1. Se C=0,Q=0.

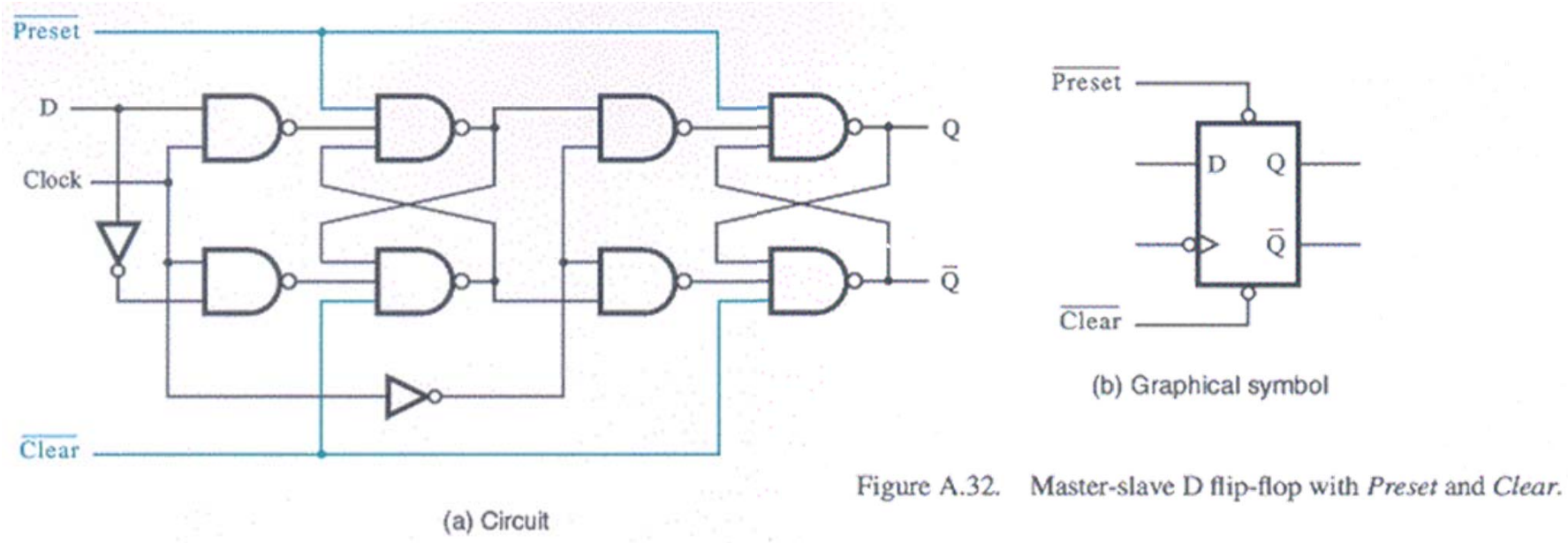


Figure A.32. Master-slave D flip-flop with *Preset* and *Clear*.