

Elettronica dei Sistemi Digitali e Laboratorio di Elettronica

Corso di Laurea in Informatica e Tecnologie Fisiche Innovative
Anno Accademico 2008-2009

- Periodo didattico : I trimestre dal 22 Settembre al 29 Novembre 2007
- Aula lezioni di teoria : F4
- Laboratorio di Elettronica : F3
- Titolare del corso: dott. Mirco Andreotti
- Email : mandreot@fe.infn.it
- Tel. Uff. : 0532/974328
- Studio : stanza Dip. Di Fisica C228

Orario: Mar 9:00 - 11:00 (aula F4) INFO + TFI
Mer 9:30 - 13:30 (Lab F3) INFO
Gio 9:00 - 13:00 (Lab F3) TFI

Testi consigliati:

- Stampe e/o fotocopie delle dispense
(disponibili online all'indirizzo <http://df.unife.it/u/mandreot>
oppure dal sistema di gestione delle dispense dal sito UNIFE)
- P. Horowitz, W. Hill "The Art of Electronics", Cambridge University Press, New York (1980)
- R. Giometti, F. Frascari "Elettronica, La Logica", Calderini, Bologna (1990)
- J. Millman, "Circuiti e sistemi microelettronici", Bollati Boringhieri, Torino (1985)

Svolgimento del corso:

- Lezioni teoriche *con qualche avanzato elemento di matematica per un approccio approfondito all'algebra di Boole*
- lezioni di laboratorio pratico
- lezioni di laboratorio con simulazione
- progetto da realizzare in gruppo

Modalità d'esame:

- se presenze in laboratorio > 50%
 - Test a risposta multipla per l'ammissione all'orale
 - Orale sul progetto finale e programma del corso (il numero di domande varia da esame a esame)
- se presenze in laboratorio < 50%
 - Pratico + Test + Orale
- in appelli ufficiali
- fuori appello, previo accordo con la commissione

Frequenza:

- non c'è obbligo di frequenza

Programma del corso

Elettronica Digitale

1. Strumenti matematici per l'elettronica digitale

- Introduzione alle grandezze fisiche che interessano l'elettronica
- Introduzione ai sistemi elettronici digitali
- La logica dei sistemi elettronici digitali
- Sistemi di numerazione
- Algebra di Boole + **trattazione matematica matriciale**

2. Elettronica digitale combinatoria

- Operatori elementari: porte logiche NOT, AND, OR, NAND, NOR, EXOR, EXNOR; porte con bit di abilitazione ed inibizione (ENABLE, INHIBIT); l'universalità delle porte logiche NAND e NOR.
- Cenni sui circuiti integrati: gruppi, famiglie e caratteristiche.
- **introduzione al simulatore circuitmaker**
- Studio di circuiti logici combinatori.
- Comparatori digitali, MUX, DEMUX
- Convertitore BCD 7 segmenti

Programma del corso

Elettronica Digitale

3. Elettronica digitale sequenziale

- Studio di circuiti logici sequenziali
- Celle di memoria, FLIP-FLOP S-R, FLIP-FLOP J-K, FLIP-FLOP J-K Master-Slavel, FLIP-FLOP Delay, FLIP-FLOP Toggle
- Contatori asincroni, contatori sincroni, registri a scorrimento (Shift Register)
- Funzionamento SISO, SIPO, PIPO, PISO

4. Applicazioni di elettronica digitale

- Comparatori a più bit
- Sommatore e sottrattori

5. Tipi di circuiti integrati e applicazioni

- I transistor: circuiti digitali con uscite Totem-Pole, Open-Collector, Three-State
- Bus dati per la comunicazione di dati fra sistemi diversi
- Studio dell'unità aritmetico-logica (ALU)
- Contatori di impulsi a 3 cifre

Programma del corso

Cenni di Elettronica Analogica

1. Strumenti matematici e fisici per l'elettronica analogica

- Funzioni periodiche
- Sviluppo in serie di Fourier
- Alcune forme d'onda particolari
- Grandezze fondamentali dell'elettronica analogica

2. Dispositivi elettrici fondamentali e risoluzioni delle reti elettriche

- Resistenze, condensatori, induttanze
- Leggi e teoremi per l'elettronica analogica
- Funzionamento delle reti elettriche in regime sinusoidale

3. Trattazione di particolari applicazioni di interesse pratico

- Partitore di tensione
- Convertitore Digitale-Analogico
- Circuiti R-C e C-R e loro utilizzo come filtri
- Partitori capacitivi e compensati

Esperienze di Laboratorio Parte I

Programma del corso

- D-1
 - Operazione con le porte logiche elementari
 - Verifica del teorema di De Morgan
 - Flusso di segnali digitali (gate)

- D-2
 - Realizzazione di un True/Invert
 - Realizzazione di EXOR (EXNOR) con sole porte NAND (NOR)
 - Funzione di uguaglianza
 - Comparatore digitale a un bit

- D-3
 - MUX (Multiplexer)
 - DeMUX (DeMultiplexer)



Esperienze di Laboratorio Parte II

- D-4
 - Operazioni con i FLIP-FLOP SR
 - Operazioni con FLIP-FLOP con ENABLE
 - Master-Slave non trasparente
 - Master-Slave Toggle

- D-5
 - Operazioni con FLIP-FLOP JK
 - Realizzazione di un contatore binario
 - Realizzazione di un registro a scorrimento
 - Funzioni SISO, SIPO, PIPO, PISO

Programma del corso

Esperienze di Laboratorio Parte III

- D-6
 - Realizzazione di un comparatore a più bit
 - Realizzazione di un sommatore

Esperienze di Laboratorio Parte IV

- D-7
 - Utilizzo dei dispositivi Totem-Pole, Open-Collector e Tristate
 - Comunicazione tramite bus tristate
- D-8
 - Trasmissione dati da tastiera
- D-9
 - Utilizzo dell'unità aritmetico-logica (ALU)
- D-10
 - Realizzazione di un contatore di impulsi a 3 cifre.



Laboratorio di Elettronica Elettronica Digitale

Parte I

Corso di Laurea in TFI
Anno Accademico 2007-2008

•ELETTRONICA

perché ci interessa?

SISTEMI TIPICI

sistema informatico
impianti HIFI

SEGNALI

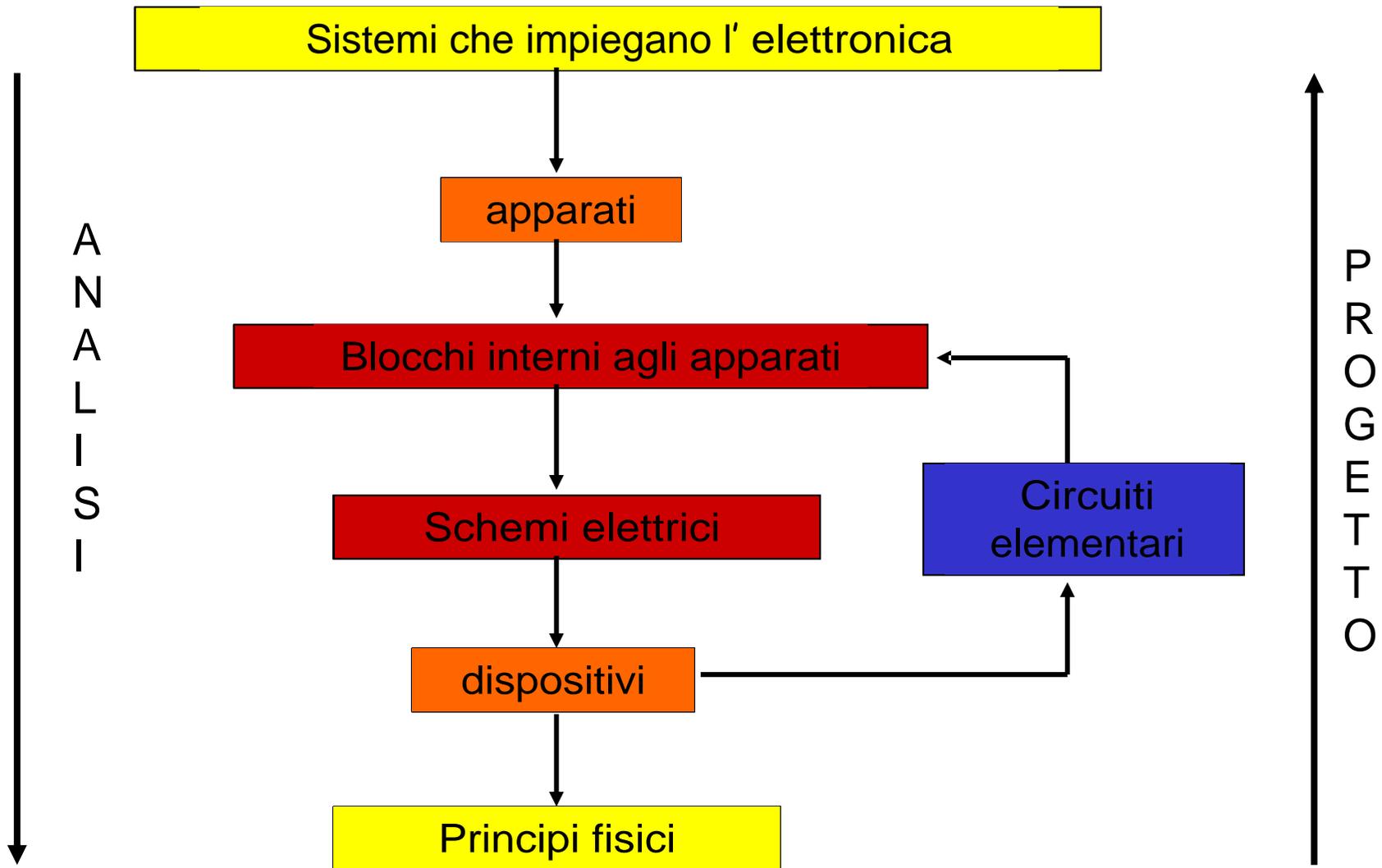
analogici
digitali

•APPROCCIO SISTEMISTICO

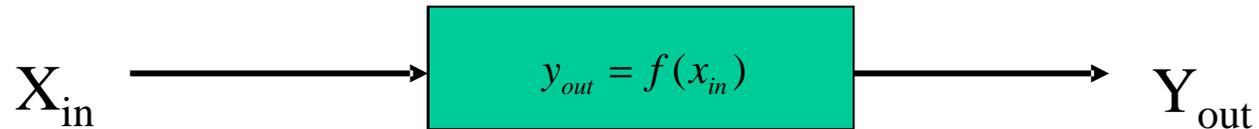
sistemi →→ sistema per la misura della velocità del suono
apparati →→ Oscilloscopio
blocchi funzionali →→ amplificatore, trigger di Schmidt, alimentatori...
schemi circuitali →→ molto complessi
componenti →→ molto pochi e ricorrenti:
circ. integrati e componenti

Riduce tutto, a qualunque livello, al concetto di blocco funzionale:

radio, TV, stereo, strumentazione varia.....



IL BLOCCO FUNZIONALE



Proprietà generali dei blocchi:

È completamente determinato dalla funzione che lega le variabili di ingresso e di uscita

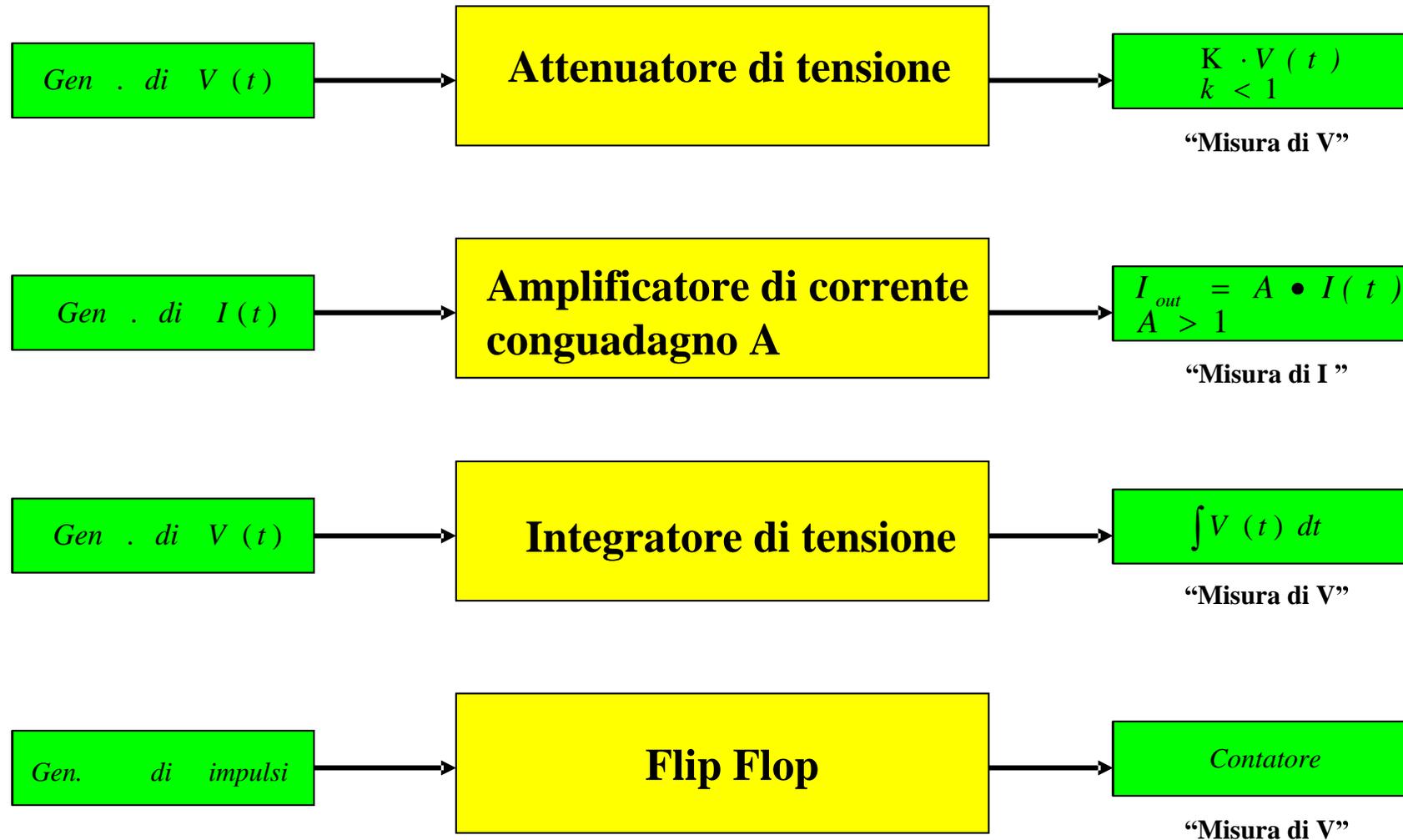
Possono essere:

Digitali : elettronica digitale
Analogici : elettronica analogica
Di conversione : A/D & D/A

Possono anche essere:

Lineari e non lineari

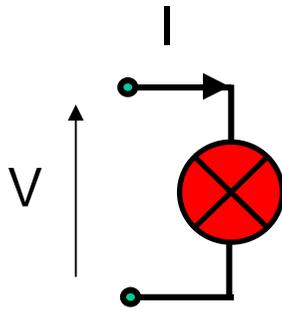
ESEMPI



Segnali Digitali/Analogici

Segnale  Segnale elettrico caratterizzato da certe grandezze:

- Tensione V (V) (differenza di potenziale elettrico)
- corrente I (A) (movimento di elettroni)
- tempo (s)



Segnali Digitali/Analogici

- ✓ Valori della tensione in funzione del tempo:
 - ❑ tensione continua di una pila (in funzione del tempo è una retta)
 - ❑ tensione alternata della rete domestica (sinusoide)

- ✓ segnale analogico: può assumere tutti i valori di tensione

- ✓ segnale digitale: può assumere solo due valori di tensione
 - i 2 valori dipendono dalla famiglia di segnali che si usano
 - e dalla logica

- due livelli di tensione: V_{HIGH} (H)/ V_{LOW} (L)

- famiglie logiche:

TTL, HTL, ECL,
MOS, CMOS....

- logica

positiva: $H \rightarrow T/1$, $L \rightarrow F/0$

negativa: $L \rightarrow T/1$, $H \rightarrow F/0$

- funzioni logiche

le stesse per tutte



Family	Basic gate	Fanout	Pd (mW/gate)	Noise immunity	Prop. delay (ns/gate)	Clock (MHz)
TTL	NAND	10	10	VG	10	35
TTL-H	NAND	10	22	VG	6	50
TTL-L	NAND	20	1	VG	33	3
TTL-LS	NAND	20	2	VG	9.5	45
TTL-S	NAND	10	19	VG	3	125
TTL-AS	NAND	40	10	VG	1.5	175
TTL-ALS	NAND	20	1	VG	4	50
ECL 10K	OR-NOR	25	40-55	P	2	>60
ECL100K	OR-NOR	??	40-55	P	0.75	600
MOS	NAND	20	0.2-10	G	300	2
74C	NOR/NAND	50	0.01/1	VG	70	10
74HC	NOR/NAND	20	0.0025/0.6	VG	18	60
74HCT	NOR/NAND	20	0.0025/0.6	VG	18	60
74AC	NOR/NAND	50	0.005/0.75	VG	5.25	100
74ACT	NOR/NAND	50	0.005/0.75	VG	4.75	100

PERCHE' APPROCCIO SISTEMISTICO?

- sistema = blocco
- blocco di natura elettronica

PERCHE' PRIMA L' ELETTRONICA DIGITALE?

- più facile
- non richiede nozioni preliminari
- candidato ideale al tipo di approccio
- due soli stati (variabili di ingresso: tensioni)

fisici: H,L

logici: T,F; 1,0; ... sec. i gusti

0 e 1 sono i simboli usati nel sistema di numerazione binario!

→ sistemi di numerazione

Sistemi di Numerazione

- Numero
 - concetto non primitivo
 - serve per “quantificare” la realtà
- Sistema di numerazione:
 - insieme finito di simboli
 - simboli organizzati in sequenze secondo “regole”

I sistema più antico è (forse) quello egizio ed ha circa 5000 anni. E' di tipo decimale con simboli ripetuti per i multipli di una stessa quantità.

1		100	
2		231	  
10		1000	
11		1101	  
20			

- **I Sumeri avevano l' unità numerica fondamentale che corrisponde al nostro 60**

nostro sistema di misura degli angoli?

Non hanno lo zero ed i simboli sono posizionali

- lo zero introdotto forse nella civiltà indiana e poi arriva in Europa portato dagli Arabi
- I simboli da noi usati oggi (indo-Arabi) risalgono al X secolo
- I numeri frazionari arrivano solo nel XVI secolo
- Il punto decimale viene introdotto verso la metà del XVII secolo

I sistemi di numerazioni usati sono caratterizzati da:

Posizionale: sistema in cui il valore associato ad ogni simbolo dipende dalla sua posizione nella “stringa”

Basi: numero di simboli usati nella numerazione

Peso: il fattore per cui il simbolo (numero) deve essere moltiplicato per potere essere confrontato con gli altri simboli (numeri): **potenza ad esponente variabile della base del sistema di numerazione**

Vediamo alcuni esempi →

Sistema decimale:

- è in base 10
- 10 simboli: 0-9
- è posizionale

Esempio: 4518,23

4	5	1	8,	2	3
10^3	10^2	10^1	10^0 ,	10^{-1}	10^{-2}

Che significa:

4000+500+10+8+0,2+0,03

Sistema binario:

- è in base 2
- 2 simboli: 0,1
- è posizionale

Esempio: 1001,01

1	0	0	1,	0	1
2^3	2^2	2^1	2^0 ,	2^{-1}	2^{-2}

Che significa:

8+0+0+1+0+0.25

In generale in base “R”:

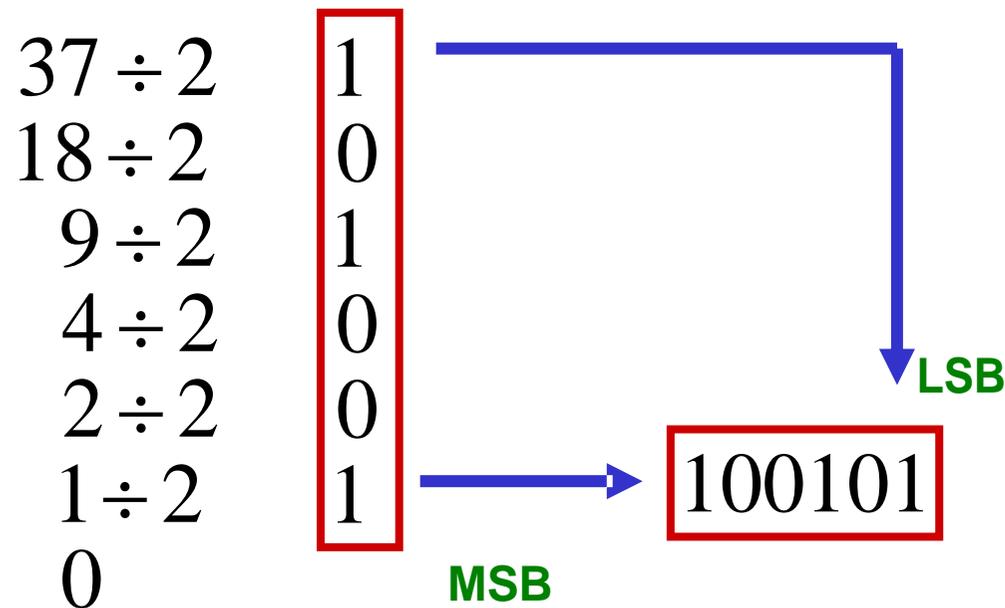
$$N = C_{n-1} \cdot R^{n-1} + C_{n-2} \cdot R^{n-2} + \dots + C_0 \cdot R^0 + C_{-1} \cdot R^{-1} + \dots$$

Come si passa da un sistema all' altro?

✓ **Binario → Decimale** $(1100101)_2 =$

$$1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (101)_{10}$$

✓ **Decimale → Binario** $(37)_{10}$



Sistema ottale:

- è in base 8
- 8 simboli: 0-7
- è posizionale

Esempio: (514,23)₈

5	1	4,	2	3
8^2	8^1	8^0 ,	8^{-1}	8^{-2}

Esempio: (456)₈

1	0	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---

Sistema esadecimale:

- è in base 16
- 16 simboli: 0-9,A,B,C,D,E,F
- è posizionale

Esempio: (3AFF9)₁₆

3	10	15	15	9
16^4	16^3	16^2	16^1	16^0

Esempio: (B7F)₁₆

1	0	1	1	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---

Il sistema binario

- è in base 2
- 2 simboli: 0,1
- è posizionale

Regole pratiche:

Somma	$0 + 0 = 0$
	$0 + 1 = 1$
	$1 + 1 = 10$ (0 con riporto di 1)
Sottrazione	$0 - 0 = 0$
	$1 - 0 = 1$
	$1 - 1 = 0$
	<u>$0 - 1 = 1$ con richiamo di 1</u>

Alcuni esempi:

$$1101 + 1011010 = ?$$

$$10 + 1011 + 111 + 1010 = ?$$

$$11011 - 01101 = ?$$

$$11011001 - 10101010 = ?$$

} la sottrazione è complicata

→ introduciamo la **COMPLEMENTAZIONE**

Complementazione

- Decimale -- a 9 dato xxx il suo complemento a 9 è 999-xxx
- a 10 dato xxx il suo complemento a 10 è 1000-xxx
 (complemento a 9 + 1)

- Binario -- a 1 dato xxx il suo complemento a 1 è 111-xxx
- a 2 dato xxx il suo complemento a 2 è 1000-xxx
 (complemento a 1 + 1)

→ **anzichè eseguire una sottrazione
utilizziamo un'alternativa:**

- Decimale

1) eseguiamo il complemento a 10 del minuendo

2) se sottraendo $>$ minuendo: a) sommiamo al sottraendo 1)
b) scartiamo l'eventuale riporto

se sottraendo $<$ minuendo: a) sommiamo al sottraendo 1)
b) eseguiamo il complemento a 10 del risultato
c) ne cambiamo il segno

- Binario

1) eseguiamo il complemento a 2 del minuendo

2) se sottraendo $>$ minuendo: a) sommiamo al sottraendo 1)
b) scartiamo l'eventuale riporto

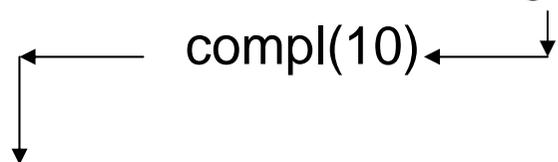
se sottraendo $<$ minuendo: a) sommiamo al sottraendo 1)
b) eseguiamo il complemento a 2 del risultato
c) ne cambiamo il segno

- **Esempio Decimale**
sottrendo > minuendo

$$\begin{array}{r} 457 - \\ \underline{129} = \end{array} \rightarrow \text{compl}(10) \rightarrow \begin{array}{r} 457 + \\ \underline{871} = \\ \cancel{1418} \end{array}$$

sottraendo < minuendo

$$\begin{array}{r} 129 - \\ \underline{547} = \end{array} \rightarrow \text{compl}(10) \rightarrow \begin{array}{r} 129 + \\ \underline{453} = \\ 582 \end{array}$$



$$418 \rightarrow \text{CHS} \rightarrow -418$$

- **Esempio binario**
sottrendo > minuendo

$$\begin{array}{r} 1100 - \\ \underline{11} = \end{array} \rightarrow \text{compl}(2) \rightarrow \begin{array}{r} 1100 + \\ \underline{1101} = \\ \cancel{11001} \end{array}$$

sottraendo < minuendo

$$\begin{array}{r} 101 - \\ \underline{11011} = \end{array} \rightarrow \text{compl}(2) \rightarrow \begin{array}{r} 101 + \\ \underline{00101} = \\ 01010 \end{array}$$



$$10110 \rightarrow \text{CHS} \rightarrow -10110$$

- **Esempio Decimale**
sottraendo > minuendo

$$\begin{array}{r}
 457 - \\
 \underline{129} =
 \end{array}
 \rightarrow \text{compl}(10) \rightarrow
 \begin{array}{r}
 457 + \\
 \underline{871} = \\
 \cancel{418}
 \end{array}
 \rightarrow$$

$$\begin{aligned}
 xxx - yyy &= \\
 &= xxx - 1000 + 1000 - yyy = \\
 &= xxx - 1000 + yyy_{10} = \\
 &= [xxx + yyy_{10}] - 1000
 \end{aligned}$$

sottraendo < minuendo

$$\begin{array}{r}
 129 - \\
 \underline{547} =
 \end{array}
 \rightarrow \text{compl}(10) \rightarrow
 \begin{array}{r}
 129 + \\
 \underline{453} = \\
 582
 \end{array}$$

compl(10) ←

418 → CHS → -418

$$\begin{aligned}
 xxx - yyy &= xxx - 1000 + 1000 - yyy = \\
 &= xxx - 1000 + yyy_{10} = \\
 &= [xxx + yyy_{10}] - 1000 = \\
 &= (-1) \cdot \{1000 - [xxx + yyy_{10}]\} = \\
 &= (-1) \cdot [xxx + yyy_{10}]_{10}
 \end{aligned}$$

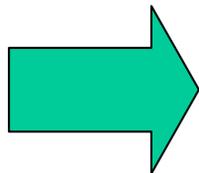
- L'esempio binario è del tutto analogo a quello decimale

9999 → 1111
10000 → 10000

-Perchè tutta questa confusione con complementi a 1 o a 2 e poi cambi di segno etc etc?

→ è un modo per avere sempre operazioni con risultati positivi
→ situazione meglio gestibile nella realizzazione pratica di circuiti elettronici per la realizzazione delle operazioni.

→ ne studieremo i dettagli più avanti



Un primo progetto per l'esame finale

Significato matematico della complementazione in generale (decimale)

$$\begin{aligned}xxx - yyy &= \\&= xxx - 1000 + \underline{1000} - yyy = \\&= xxx - 1000 + yyy_{10} = \\&= [xxx + yyy_{10}] - 1000\end{aligned}$$

$$\begin{aligned}xxx - yyy &= \\&= xxx - 999 + \underline{999} - yyy = \\&= xxx - 999 + yyy_9 = \\&= [xxx + yyy_9] - 999 = \\&= (-1) \cdot [xxx + yyy_9]_9 = \\&= (-1) \cdot [xxx + yyy_9]_{10} + 1 = \dots\end{aligned}$$

$$\begin{aligned}xxxx - yy &= \\&= xxxx - 10000 + \underline{10000} - yy = xxx - 10000 + yy_{10} \\&= xxxx - 9999 + \underline{9999} - yy = xxx - 9999 + yy_9\end{aligned}$$

Per seguire le regole di prima il complemento si effettua guardando al membro con maggior numero di cifre

Significato matematico della complementazione in generale (binario)

$$\begin{aligned}xxx - yyy &= \\&= xxx - 1000 + \underline{1000} - yyy = \\&= xxx - 1000 + yyy_2 = \\&= [xxx + yyy_2] - 1000\end{aligned}$$

$$\begin{aligned}xxx - yyy &= \\&= xxx - 111 + \underline{111} - yyy = \\&= xxx - 111 + yyy_1 = \\&= [xxx + yyy_1] - 111 = \\&= (-1) \cdot [xxx + yyy_1]_1 = \\&= (-1) \cdot [xxx + yyy_1]_2 + 1 = \dots\end{aligned}$$

$$\begin{aligned}xxxx - yy &= \\&= xxxx - 10000 + \underline{10000} - yy = xxx - 10000 + yy_2 \\&= xxxx - 1111 + \underline{1111} - yy = xxx - 1111 + yy_1\end{aligned}$$

Per seguire le regole di prima il complemento si effettua guardando al membro con maggior numero di cifre

Qualche trucchetto per la complementazione

→ in generale è più facile complementare a 9(1) piuttosto che a 10(2)

→ se serve complementare a 10(2), ma ci riesce più facile complementare a 9(1) allora:

$$xxx_{10} = 1000 - xxx = 999 - xxx + 1$$

$$xxx_2 = 1000 - xxx = 111 - xxx + 1$$

→ complementiamo a 9 (o 1) quindi sommiamo 1 e otteniamo il complemento a 10 (o 2).

→ nella differenza si complementa al membro con maggior numero di cifre per avere più semplicità di calcolo:

$$\begin{aligned} xxxx - yy &= xxxx - 100 + 100 - yy = \\ &= [xxxx - yy_{10(2)}] - 100 = zzzz - 100 \end{aligned}$$

→ zzzz-100 risulta semplice ma implica sempre una sottrazione mentre dalle regolette precedenti eliminavamo solo il riporto, come si avrebbe in questo caso:

$$\begin{aligned} xxxx - yy &= xxxx - 10000 + 10000 - yy = \\ &= [xxxx - yy_{10(2)}] - 10000 = 1zzzz - 10000 \end{aligned}$$

→ in questo modo il riporto c'è sempre →

il riporto c'è sempre →

$$\begin{aligned} & xxxx < 10000 ; \quad yy < 10000 ; \quad xxxx > yy \\ \Rightarrow & xxxx - yy < 10000 \\ \Rightarrow & xxxx - yy = xxxx - 10000 + 10000 - yy = \\ = & \underbrace{[10000 + (xxxx - yy)]}_{\text{Compreso tra 10000 e 11111 (o in decimale 19999)}} - 10000 \end{aligned}$$

Compreso tra 10000 e 11111 (o in decimale 19999)

Sarà oggetto di altre discussioni più approfondite

... riprendiamo con l'elettronica ...

PERCHE' L' ELETTRONICA DIGITALE?

- più facile (dell'analogica)
- non richiede nozioni preliminari
- candidato ideale al tipo di approccio
- due soli stati (variabili di ingresso: tensioni)

fisici: H,L

logici: T,F; 1,0; ... sec. i gusti

✓ la logica usata:

ALGEBRA DI BOOLE

- costanti: 0,1; T,F; H,L.....
- variabili: x,y,z.... ma ognuna ha 2 soli valori!
- funzioni: $f(x,y,.....)$...come sopra
- solo 3 operazioni (fondamentali):

NOT ($_$, $\bar{_}$) agisce solo su 1 var, cost. o funzione

AND (x , $*$, \cdot) agisce su 2 o più var, cost. o funzioni

OR ($+$) agisce su 2 o più var, cost. o funzioni

Simboli usati

\overline{X} → NOT

$A + B$ → + → OR logico e non somma algebrica

$A \cdot B$ → · → AND logico e non prodotto algebrico

✓ saranno esplicitamente indicati i casi in cui i simboli + e · vengano essere usati come operazioni aritmetiche anzichè logiche

Operatori logici

Somma (OR) $A+B+C+\dots = 0$ se tutte le var = 0
 $A+B+C+\dots = 1$ se almeno una var = 1

Prodotto (AND) $A \cdot B \cdot C \cdot \dots = 0$ se almeno una var = 0
 $A \cdot B \cdot C \cdot \dots = 1$ se tutte le var = 1

Complemento (NOT) $A = 0 \rightarrow \bar{A} = 1$
 $A = 1 \rightarrow \bar{A} = 0$

Postulati

1) $A = 0 \text{ o } A = 1$

2) $0 \cdot 0 = 0$

3) $1 \cdot 1 = 1$

4) $1 \cdot 0 = 0 \cdot 1 = 0$

5) $0 + 0 = 0$

6) $1 + 0 = 0 + 1 = 1$

7) $1 + 1 = 1$

Proprietà & Teoremi

commutativa 1) $A + B = B + A$ $A \cdot B = B \cdot A$

associativa 2) $(A + B) + C = A + (B + C)$

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

distributiva 3) $A \cdot (B + C) = A \cdot B + A \cdot C$



$A + (B \cdot C) = (A + B) \cdot (A + C)$
Non vale per l'algebra dei numeri reali

idempotenza 4) $A + A = A$ $A \cdot A = A$

involutione 5) $\overline{\overline{A}} = A$

applicazione \Rightarrow 6) $A + (A \cdot B) = A$

$$A \cdot (A + B) = A$$

Proprietà & Teoremi

dimostrazione

$$3b) \quad A + (B \cdot C) = (A + B) \cdot (A + C)$$

$$(A + B) \cdot (A + C) =$$

$$A \cdot A + A \cdot C + A \cdot B + B \cdot C =$$

$$A \cdot (1 + C + B) + B \cdot C = A + B \cdot C$$

dimostrazione

$$6a) \quad A + (A \cdot B) = A$$

$$6b) \quad A \cdot (A + B) = A$$

$$6a) \quad A + (A \cdot B) = (A + A) \cdot (A + B) =$$

$$6b) \quad A \cdot (A + B) = A \cdot A + A \cdot B =$$

$$A + A \cdot B = A \cdot (1 + B) = A$$

identità	7)	$0 + A = 1 \cdot A = A$
dominanza	8)	$1 + A = 1 \quad 0 \cdot A = 0$
complementazione	9)	$\overline{\overline{A}} + A = 1 \quad \overline{\overline{A}} \cdot A = 0$
assorbimento	10)	$A + \overline{A} B = A + B$ $A \cdot (\overline{A} + B) = A \cdot B$

Teorema di: **DE MORGAN**

$$11) \quad \overline{(A + B)} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

Teorema di: **DE MORGAN**

$$\begin{array}{l} 1) \overline{A + B} = \overline{A} \cdot \overline{B} \\ 2) \overline{A \cdot B} = \overline{A} + \overline{B} \end{array}$$

Dimostrazione algebrica della 1

- Ricordiamo la proprietà $X \cdot \overline{X} = 0$
- ponendo $X = \overline{A} \cdot \overline{B} \Rightarrow \overline{A} \cdot \overline{B} \cdot (\overline{\overline{A} \cdot \overline{B}}) = 0$
- se è vera l'uguaglianza 1 del teorema di De Morgan allora facendo la seguente sostituzione, la precedente uguaglianza non deve cambiare: $\overline{(\overline{A} \cdot \overline{B})} = \overline{\overline{A + B}} = A + B$
- Verifichiamo quindi che $\overline{A} \cdot \overline{B} \cdot (A + B) = 0$

$$\overline{A} \cdot \overline{B} \cdot (A + B) = \underbrace{\overline{A} \cdot A}_{=0} \cdot \overline{B} + \overline{A} \cdot \underbrace{\overline{B} \cdot B}_{=0} = 0 + 0 = 0$$

Teorema di: **DE MORGAN**

$$\begin{array}{l} 1) \quad \overline{A + B} = \overline{A} \cdot \overline{B} \\ 2) \quad \overline{A \cdot B} = \overline{A} + \overline{B} \end{array}$$

Dimostrazione algebrica della 2

- Ricordiamo la proprietà $X \cdot \overline{X} = 0$
- ponendo $X = \overline{A} + \overline{B} \Rightarrow (\overline{A} + \overline{B}) \cdot \overline{(\overline{A} + \overline{B})} = 0$
- se è vera l'uguaglianza 2 del teorema di De Morgan allora facendo la seguente sostituzione, la precedente uguaglianza non deve cambiare: $\overline{(\overline{A} + \overline{B})} = \overline{\overline{A \cdot B}} = A \cdot B$
- Verifichiamo quindi che $(\overline{A} + \overline{B}) \cdot (A \cdot B) = 0$

$$(\overline{A} + \overline{B}) \cdot (A \cdot B) = \underbrace{\overline{A} \cdot A}_{=0} \cdot B + A \cdot \underbrace{\overline{B} \cdot B}_{=0} = 0 + 0 = 0$$

Significato del Teorema di **DE MORGAN**

$$\begin{array}{l} 1) \overline{A + B} = \overline{A} \cdot \overline{B} \\ 2) \overline{A \cdot B} = \overline{A} + \overline{B} \end{array}$$

✓ In generale ogni funzione booleana è una qualsiasi combinazione di operazioni logiche di base (AND, OR e NOT) fra un certo numero di variabili:

$$f(A, B, C \dots; +, \cdot, \overline{}) = A \cdot (B + \overline{C}) + \overline{A} \cdot \overline{C} \dots$$

1) negando la prima uguaglianza del teorema otteniamo:

$$\overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}} \Rightarrow A + B = \overline{\overline{A} \cdot \overline{B}}$$

Questa uguaglianza dice che ogni OR logico può essere ottenuto con un'opportuna combinazione di AND e NOT

➔ In una funzione ogni OR può essere sostituito con l'opportuna combinazione di AND e NOT

➔ Ogni funzione può essere espressa in termini di 2 sole operazioni logiche, cioè AND e NOT, anziché delle 3 di base.

2) negando la seconda uguaglianza del teorema otteniamo:

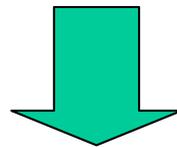
$$\overline{\overline{A \cdot B}} = \overline{\overline{A + B}} \Rightarrow A \cdot B = \overline{\overline{A + B}}$$

$$\begin{array}{l} 1) \overline{A + B} = \overline{A} \cdot \overline{B} \\ 2) \overline{A \cdot B} = \overline{A} + \overline{B} \end{array}$$

Questa uguaglianza dice che ogni AND logico può essere ottenuto con un'opportuna combinazione di OR e NOT

→ In una funzione ogni AND può essere sostituito con l'opportuna combinazione di OR e NOT

→ Ogni funzione può essere espressa in termini di 2 sole operazioni logiche, cioè OR e NOT, anziché delle 3 di base.



La 1 dice che

→ Ogni funzione può essere espressa in termini di 2 sole operazioni logiche, cioè AND e NOT, anziché delle 3 di base.

La 2 dice che

→ Ogni funzione può essere espressa in termini di 2 sole operazioni logiche, cioè OR e NOT, anziché delle 3 di base.



$$1) \overline{A + B} = \overline{A} \cdot \overline{B}$$
$$2) \overline{A \cdot B} = \overline{A} + \overline{B}$$

Il Teorema di De Morgan dice che:

Ogni funzione booleana può essere espressa in termini di 2 sole operazioni logiche:

AND e NOT $f(+, ;, -) = f'(\cdot, -)$

OPPURE

OR e NOT $f(+, ;, -) = f'(+, -)$

anziché delle 3 di base AND, OR e NOT.

Vedremo in termini di circuiti il vantaggio pratico di questo teorema fondamentale.

FUNZIONI LOGICHE:

si rappresentano in **tabelle della verità o con espressioni algebriche**



2^n combinazioni

Tutte le possibili combinazioni fra le variabili.

Date n variabili che possono assumere solo 2 valori, il numero totale di possibili combinazioni è 2^n

A	B	F(A,B)
0	0	F_1
0	1	F_2
1	0	F_3
1	1	F_4

Valori assunti dalla funzione in corrispondenza della particolare combinazione

FUNZIONI LOGICHE

- si rappresentano con **tabelle di verità**

$$A + B = A + \bar{A}B$$

A	B	$A + B$	$\bar{A}B$	$A + \bar{A}B$
0	0	0	0	0
0	1	1	1	1
1	0	1	0	1
1	1	1	0	1

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

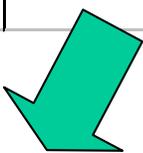
A	B	C	$B \cdot C$	$A + (B \cdot C)$	$A + B$	$A + C$	$(A + B) \cdot (A + C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Sviluppo delle funzioni in minterm e maxterm

Definiamo:

A	B	minterm	maxterm	F
0	0	$\bar{A} \cdot \bar{B}$	$A + B$	F_1
0	1	$\bar{A} \cdot B$	$A + \bar{B}$	F_2
1	0	$A \cdot \bar{B}$	$\bar{A} + B$	F_3
1	1	$A \cdot B$	$\bar{A} + \bar{B}$	F_4

} 2^n combinazioni


 $m_i = 1$


 $M_i = 0$

minterm (m_i) = prodotto fra i valori che devono assumere le variabili (negate o non) affinché il risultato sia 1

maxterm (M_i) = somma fra i valori che devono assumere le variabili (negate o non) affinché il risultato sia 0

Sviluppo delle funzioni in minterm e maxterm

Per il Teorema di De Morgan: $\begin{cases} m_i = \overline{M_i} \\ M_i = \overline{m_i} \end{cases} \quad F = \sum_{i=1}^{2^n} F_i \cdot m_i = \prod_{i=1}^{2^n} (F_i + M_i)$

Sviluppo di una funzione:

con somme



somma, su tutte le combinazioni delle variabili, dei prodotti fra il valore della funzione e il minterm

con prodotti



prodotto, su tutte le combinazioni delle variabili, delle somme fra il valore della funzione e il maxterm

$$F = \sum_{i=1}^{2^n} F_i \cdot m_i \quad = \quad F = \prod_{i=1}^{2^n} (F_i + M_i)$$

Sviluppo delle funzioni in minterm e maxterm

$$F = \sum_{i=1}^{2^n} F_i \cdot m_i$$

in questa sommatoria
contribuiscono solo
i termini in cui $F_i=1$



$$F = \sum_{\substack{i=1 \\ F_i=1}}^{2^n} F_i \cdot m_i$$

perchè $0 \cdot m_i = 0$

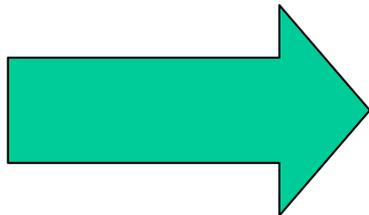
$$F = \prod_{i=1}^{2^n} (F_i + M_i)$$

in questa produttoria
contribuiscono solo
i termini in cui $F_i=0$



$$F = \prod_{\substack{i=1 \\ F_i=0}}^{2^n} (F_i + M_i)$$

perchè $1 + M_i = 1$



$$F = \sum_{i=1}^{2^n} F_i \cdot m_i = \prod_{i=1}^{2^n} (F_i + M_i)$$

Sviluppo delle funzioni in minterm e maxterm

Dimostriamo la seguente uguaglianza: $F = \sum_{i=1}^{2^n} F_i \cdot m_i = \prod_{i=1}^{2^n} (F_i + M_i)$

✓ Ammettiamo vera la prima (in seguito dimostreremo che è vera): $F = \sum_{i=1}^{2^n} F_i \cdot m_i$

✓ siccome è vera la prima allora sarà anche vero che: $\overline{F} = \sum_{i=1}^{2^n} \overline{F}_i \cdot m_i$

✓ Verifichiamo quindi, tramite le precedenti due relazioni, che la produttoria è uguale a F:

$$\begin{aligned} F &= \prod_{i=1}^{2^n} (F_i + M_i) = \prod_{i=1}^{2^n} \overline{\overline{F}_i \cdot \overline{M}_i} = \overline{\sum_{i=1}^{2^n} \overline{F}_i \cdot \overline{M}_i} = \\ &= \overline{\sum_{i=1}^{2^n} \overline{F}_i \cdot m_i} = \overline{\overline{F}} = F \end{aligned} \quad \longrightarrow \quad F = \sum_{i=1}^{2^n} F_i \cdot m_i = \prod_{i=1}^{2^n} (F_i + M_i)$$

Sviluppo delle funzioni in minterm e maxterm

- serve per semplificare le funzioni (semplificabili)
- serve per ricavare l'espressione di un funzione della quale ne conosciamo solo la tavola della verità, esempio:

A	B	F	m	M
0	0	0	$\bar{A} \cdot \bar{B}$	$A + B$
0	1	1	$\bar{A} \cdot B$	$A + \bar{B}$
1	0	0	$A \cdot \bar{B}$	$\bar{A} + B$
1	1	0	$A \cdot B$	$\bar{A} + \bar{B}$

con minterm

$$F = \bar{A} \cdot B$$

con Maxterm

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

Forma “normale” di una funzione “combinatoria”

Somme di prodotti

$$X = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + ABCD$$

$$\overline{X} = \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + ABC\overline{D} + ABCD + ABCD$$

A	B	C	D	X	\overline{X}	minterm
0	0	0	0	1	0	$\overline{A}\overline{B}\overline{C}\overline{D}$
0	0	0	1	0	1	$\overline{A}\overline{B}\overline{C}D$
0	0	1	0	0	1	$\overline{A}\overline{B}C\overline{D}$
0	0	1	1	1	0	$\overline{A}\overline{B}CD$
0	1	0	0	1	0	$\overline{A}B\overline{C}\overline{D}$
0	1	0	1	0	1	$\overline{A}B\overline{C}D$
0	1	1	0	0	1	$\overline{A}BC\overline{D}$
0	1	1	1	0	1	$\overline{A}BCD$
1	0	0	0	1	0	$A\overline{B}\overline{C}\overline{D}$
1	0	0	1	1	0	$A\overline{B}\overline{C}D$
1	0	1	0	1	0	$A\overline{B}C\overline{D}$
1	0	1	1	0	1	$A\overline{B}CD$
1	1	0	0	0	1	$AB\overline{C}\overline{D}$
1	1	0	1	0	1	$AB\overline{C}D$
1	1	1	0	0	1	$ABC\overline{D}$
1	1	1	1	1	0	$ABCD$

$$X = (A + B + C + \overline{D}) \cdot (A + B + \overline{C} + D) \cdot (A + \overline{B} + C + \overline{D}) \cdot (A + \overline{B} + \overline{C} + D) \cdot (\overline{A} + B + C + \overline{D}) \cdot (\overline{A} + B + \overline{C} + D) \cdot (\overline{A} + \overline{B} + C + D)$$

Prodotti di Somme (si Prendono gli Zeri!!!!)

Ottenuta come?

Esempi di semplificazione:

$$F_1 = \bar{a}b + a\bar{c} + \bar{a}c + a\bar{b} = \bar{a}b + a\bar{c} + \bar{b}c$$

$$\begin{aligned} F_1 &= \bar{a}b + a\bar{c} + \bar{a}c(b + \bar{b}) + a\bar{b}(c + \bar{c}) = \\ &\bar{a}b + a\bar{c} + \bar{a}cb + a\bar{c}\bar{b} + a\bar{b}c + a\bar{b}\bar{c} = \\ &\bar{a}b(1 + c) + a\bar{c}(1 + \bar{b}) + \bar{b}c(a + \bar{a}) = \end{aligned}$$

$$F_2 = \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} = \bar{a}\bar{c} + \bar{a}b + a\bar{b}\bar{c}$$

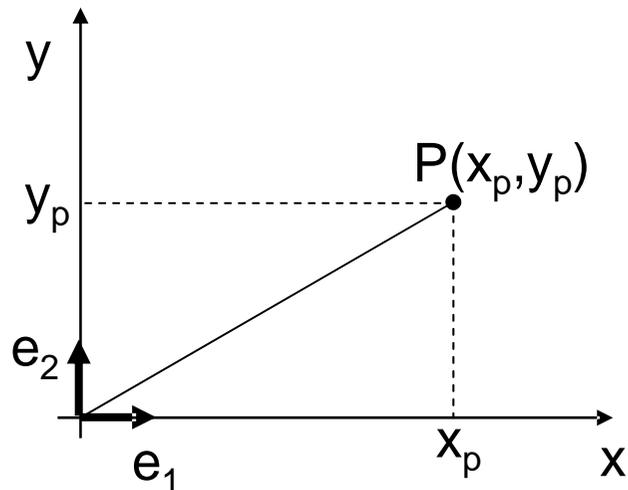
$$\begin{aligned} F_2 &= \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} = \\ &\bar{a}\bar{c}(b + \bar{b}) + \bar{a}bc + a\bar{b}\bar{c} = \\ &\bar{a}\bar{c}(1 + b) + \bar{a}bc + a\bar{b}\bar{c} = \\ &\bar{a}\bar{c} + \bar{a}\bar{c}b + \bar{a}bc + a\bar{b}\bar{c} = \\ &\bar{a}\bar{c} + \bar{a}b(c + \bar{c}) + a\bar{b}\bar{c} \end{aligned}$$

Significato dello sviluppo in minterm – spazi vettoriali

- per capire il significato dello sviluppo in minterm dobbiamo aprire un parentesi sulle basi negli spazi vettoriali
- base di uno spazio vettoriale: gruppo di vettori (di base) linearmente indipendenti con i quali si possono costruire tutti gli altri vettori dello spazio:

$$\sum_i \alpha_i \cdot e_i = 0 \Leftrightarrow \alpha_i = 0 \quad \forall i$$

- esempio nel piano

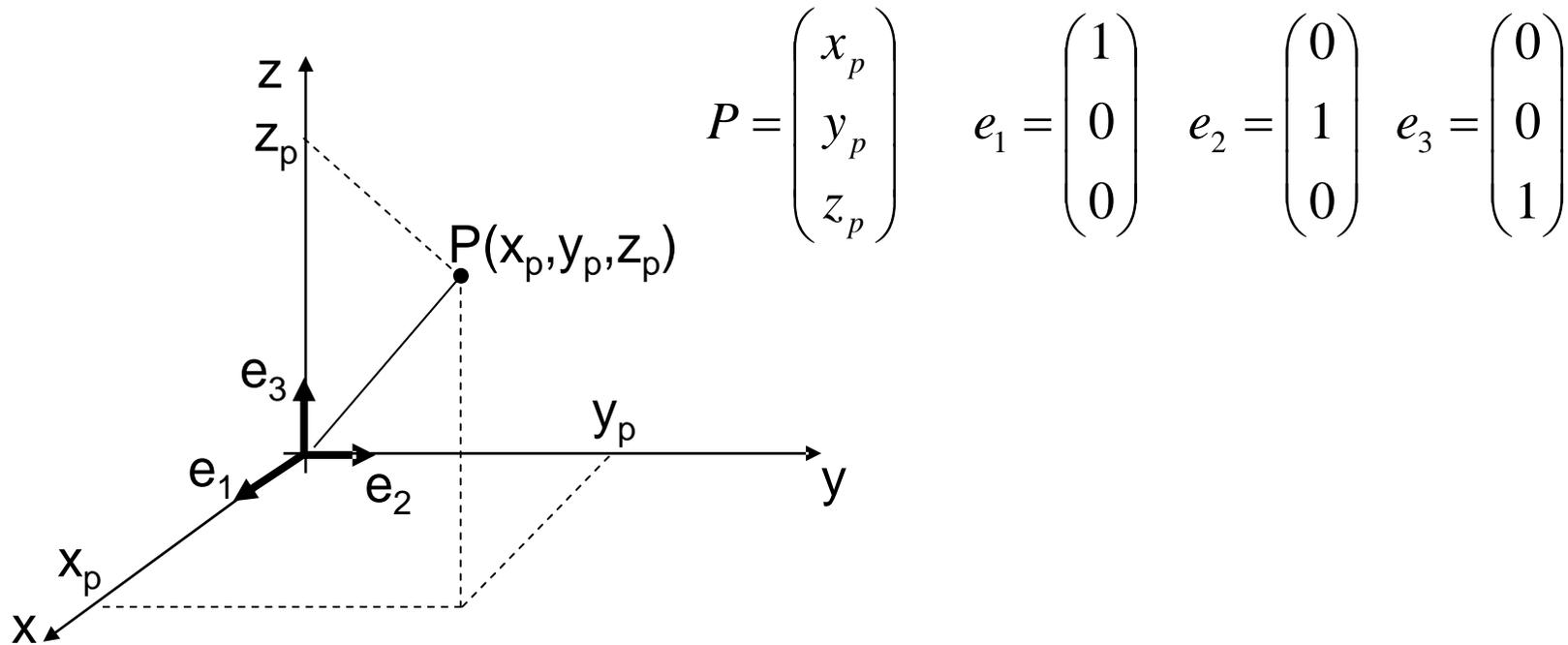


$$P = \begin{pmatrix} x_p \\ y_p \end{pmatrix} \quad e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\begin{aligned} P &= x_p e_1 + y_p e_2 = x_p \begin{pmatrix} 1 \\ 0 \end{pmatrix} + y_p \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \\ &= \begin{pmatrix} x_p \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ y_p \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} \end{aligned}$$

Significato dello sviluppo in minterm – spazi vettoriali

➤ esempio nel spazio



$$P = x_p e_1 + y_p e_2 + z_p e_3 = x_p \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + y_p \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + z_p \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x_p \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ y_p \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ z_p \end{pmatrix} = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix}$$

Significato dello sviluppo in minterm – spazi vettoriali

➤ piano 2-dim / spazio 3-dim / spazio N-dim

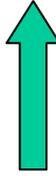
$$P = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{pmatrix} \quad e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \dots \quad e_n = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

$$P = \sum_{i=1}^n p_i e_i = p_1 e_1 + p_2 e_2 + p_3 e_3 + \dots + p_n e_n$$

Significato dello sviluppo in minterm – spazi vettoriali

➤ riprendiamo lo sviluppo in minterm per una funzione di 2 variabili booleane

	A	B	F	$\bar{A} \cdot \bar{B}$	$\bar{A} \cdot B$	$A \cdot \bar{B}$	$A \cdot B$
2^n combinazioni Dove n è il numero di variabili	0	0	F_1	1	0	0	0
	0	1	F_2	0	1	0	0
	1	0	F_3	0	0	1	0
	1	1	F_4	0	0	0	1



2^n vettori di base $e_1 \ e_2 \ e_3 \ \dots \ e_{2^n}$

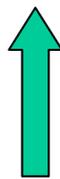
$$F = \sum_{i=1}^{2^n} F_i m_i = F_1 \cdot \bar{A} \cdot \bar{B} + F_2 \cdot \bar{A} \cdot B + F_3 \cdot A \cdot \bar{B} + F_4 \cdot A \cdot B$$

Significato dello sviluppo in minterm – spazi vettoriali

➤ funzione di 3 variabili booleane – **Da verificare**

A	B	C	F	$\bar{A} \cdot \bar{B} \cdot \bar{C}$	$\bar{A} \cdot \bar{B} \cdot C$	$\bar{A} \cdot B \cdot \bar{C}$	$\bar{A} \cdot B \cdot C$	$A \cdot \bar{B} \cdot \bar{C}$	$A \cdot \bar{B} \cdot C$	$A \cdot B \cdot \bar{C}$	$A \cdot B \cdot C$
0	0	0	F_1	1	0	0	0	0	0	0	0
0	0	1	F_2	0	1	0	0	0	0	0	0
0	1	0	F_3	0	0	1	0	0	0	0	0
0	1	1	F_4	0	0	0	1	0	0	0	0
1	0	0	F_5	0	0	0	0	1	0	0	0
1	0	1	F_6	0	0	0	0	0	1	0	0
1	1	0	F_7	0	0	0	0	0	0	1	0
1	1	1	F_8	0	0	0	0	0	0	0	1

2^n
combinazioni



Vettore F di
dimensione 2^n

2^n vettori di base $e_1 e_2 e_3 \dots e_{2^n}$

$$F = \sum_{i=1}^{2^n} F_i m_i$$

Significato dello sviluppo in minterm – spazi vettoriali

➤ funzione di n variabili booleane

✓ 2^n possibili combinazioni delle variabili

✓ quanti sono i possibili minterm che si possono costruire con n variabili?

- sono tutti i possibili prodotti delle n variabili prese nei 2 possibili stati (non-negato e negato) $\rightarrow A \cdot B \cdot C \cdot D \cdot E \dots$,

Cioè è uguale al numero delle

- possibili combinazioni di n variabili prese in 2 stati (non-negato e negato) $\rightarrow ABCDE\dots$

$$\Rightarrow 2^n$$

Modo elementare per determinare il numero delle possibili combinazioni

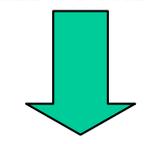
- ✓ quanti sono i possibili minterm che si possono costruire con n variabili? Contiamoli!
- ✓ (si potrebbe semplicemente dire che essendo n oggetti, l quali possono assumere x valori, allora eseguendo tutti i prodotti si ottiene che le combinazioni totali sono x^n)

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	...	<i>n</i>	
↪	<u>invertiamo</u>				↵	
<i>n</i>	...	<i>D</i>	<i>C</i>	<i>B</i>	<i>A</i>	
↕	<u>incrementiamo</u>				↕	
0	...	0	0	0	0	→ 0
0	...	0	0	0	1	→ 1
0	...	0	0	1	0	→ 2
0	...	0	0	1	1	→ 3
⋮	⋮	⋮	⋮	⋮	⋮	
0	⋮	1	1	0	0	
⋮	...	⋮	⋮	⋮	⋮	
1	...	1	1	1	1	
2^{n-1}	...	2^3	2^2	2^1	2^0	

$$\begin{array}{r} 1 \quad \dots \quad 1 \quad 1 \quad 1 \quad 1 \quad + \\ \hline 1 \quad 0 \quad \dots \quad 0 \quad 0 \quad 0 \quad 0 \quad = \end{array} \quad \begin{array}{l} \text{Per tenere conto della} \\ \text{combinazione } 0000\dots 0 \\ = \\ 2^n \end{array}$$

$$\Rightarrow \sum_{i=0}^{n-1} 2^i + 1 = 2^n$$

✓ le combinazioni che corrispondono ai numeri da 1 a 2^n sono in totale 2^n



✓ le combinazioni che corrispondono ai numeri da 0 a $\sum_{i=0}^{n-1} 2^i$ sono in totale 2^n

2^n minterm

Significato dello sviluppo in minterm – spazi vettoriali

- I minterm hanno 2^{n-1} componenti = 0 e una sola =1

A	B	C	F	$\bar{A} \cdot \bar{B} \cdot \bar{C}$	$\bar{A} \cdot \bar{B} \cdot C$	$\bar{A} \cdot B \cdot \bar{C}$	$\bar{A} \cdot B \cdot C$	$A \cdot \bar{B} \cdot \bar{C}$	$A \cdot \bar{B} \cdot C$	$A \cdot B \cdot \bar{C}$	$A \cdot B \cdot C$
0	0	0	F_1	1	0	0	0	0	0	0	0
0	0	1	F_2	0	1	0	0	0	0	0	0
0	1	0	F_3	0	0	1	0	0	0	0	0
0	1	1	F_4	0	0	0	1	0	0	0	0
1	0	0	F_5	0	0	0	0	1	0	0	0
1	0	1	F_6	0	0	0	0	0	1	0	0
1	1	0	F_7	0	0	0	0	0	0	1	0
1	1	1	F_8	0	0	0	0	0	0	0	1

- il minterm è quel particolare prodotto delle variabili negate/non il cui risultato vale 1 per una particolare combinazione
- quindi ogni altra combinazione corrispondente al prodotto indicato darà come risultato 0, perché tutte le altre combinazioni sono diverse da quella considerata
- quindi sicuramente una o più delle variabili prese in quella combinazione e con quel prodotto sarà/saranno = 0, cioè il prodotto =0

Significato dello sviluppo in minterm – spazi vettoriali

- ✓ date n variabili $\rightarrow 2^n$ possibili combinazioni
- ✓ ad ogni combinazione delle var \rightarrow un termine di una funzione
 - ✓ una funzione \rightarrow espressa da 2^n termini
 \rightarrow vettore 2^n -dimensionale

A	B	F	$\bar{A} \cdot \bar{B}$	$\bar{A} \cdot B$	$A \cdot \bar{B}$	$A \cdot B$
0	0	F_1	1	0	0	0
0	1	F_2	0	1	0	0
1	0	F_3	0	0	1	0
1	1	F_4	0	0	0	1

- ✓ con n var possiamo costruire 2^n minterm
 - ✓ ogni minterm è una funzione \rightarrow vettore 2^n -dimensionale
 - ✓ ha $2^n - 1$ componenti nulle e 1 uguale a 1 \rightarrow ottimo candidato come elemento di una base
 - ✓ sono tutti diversi \rightarrow sono linearmente indipendenti
- ✓ I minterm costituiscono la base canonica di un ipotetico spazio vettoriale (reticolo nel caso nostro di vettori booleani) 2^n -dimensionale

✓ Sviluppo di una funzione in minterm \leftrightarrow rappresentazione di un vettore nella base canonica

✓ Lo spazio vettoriale in questione è in realtà un reticolo in quanto tutti gli elementi hanno come componenti solo 0 oppure 1.

FUNZIONI LOGICHE:

si rappresentano in **tabelle della verità o con espressioni algebriche**



2^n combinazioni

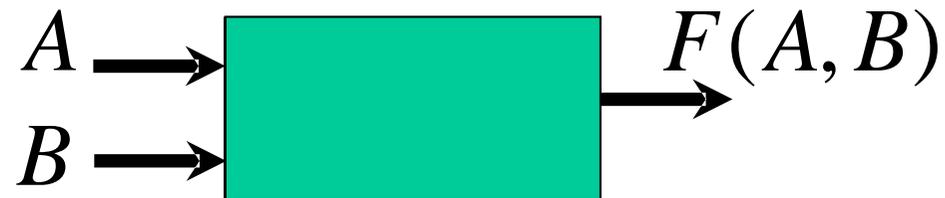
Tutte le possibili combinazioni fra le variabili.

Date n variabili che possono assumere solo 2 valori, il numero totale di possibili combinazioni è 2^n

A	B	F(A,B)
0	0	F_1
0	1	F_2
1	0	F_3
1	1	F_4

Valori assunti dalla funzione in corrispondenza della particolare combinazione

FUNZIONI LOGICHE: eseguite da circuiti digitali



✓ I circuiti digitali dovranno eseguire in determinate combinazioni le operazioni logiche di base:

Operatori logici

➤ **Somma (OR)** $A+B+C+\dots = 0$ se tutte le var = 0
 $A+B+C+\dots = 1$ se almeno una var = 1

➤ **Prodotto (AND)** $A \cdot B \cdot C \cdot \dots = 0$ se almeno una var = 0
 $A \cdot B \cdot C \cdot \dots = 1$ se tutte le var = 1

➤ **Complemento (NOT)** $A = 0 \rightarrow \bar{A} = 1$
 $A = 1 \rightarrow \bar{A} = 0$

Vediamo la rappresentazione grafica di questi operatori

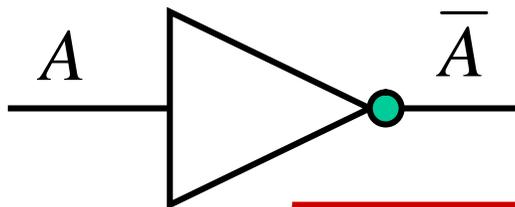
La rappresentazione grafica degli operatori logici è uno standard e comodità di progettazione

Le operazioni ed i simboli in elettronica (digitale)

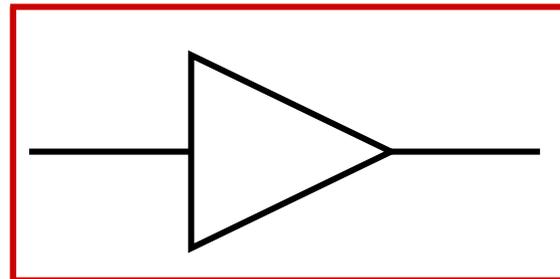
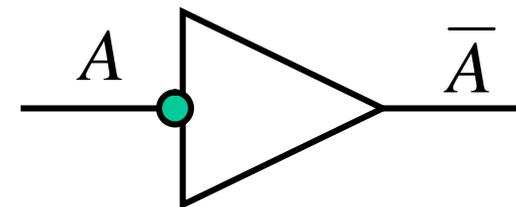
1) NOT



A	\bar{A}
0	1
1	0



oppure

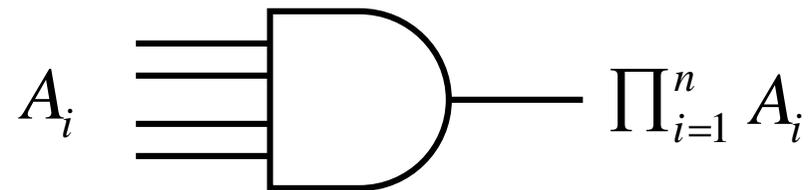
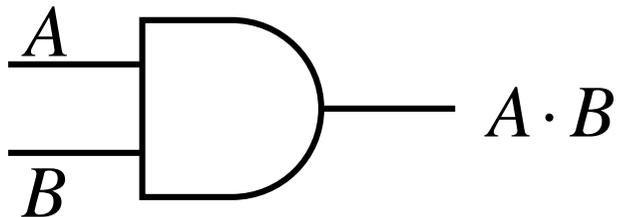


Si chiama BUFFER

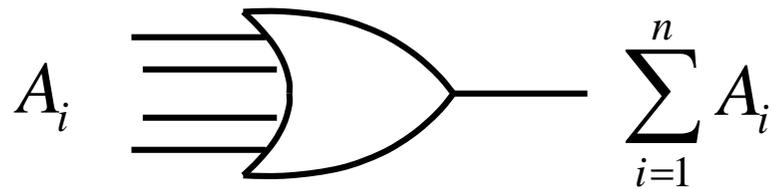
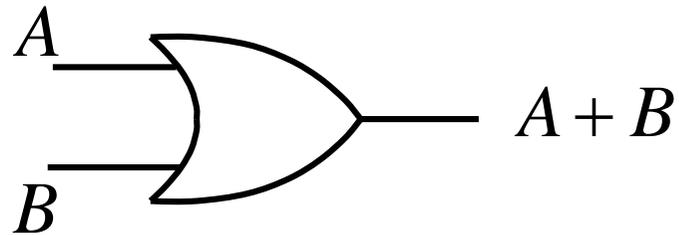
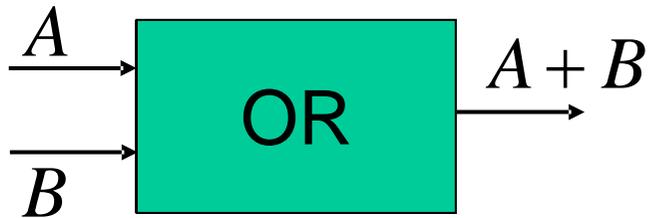
2) AND



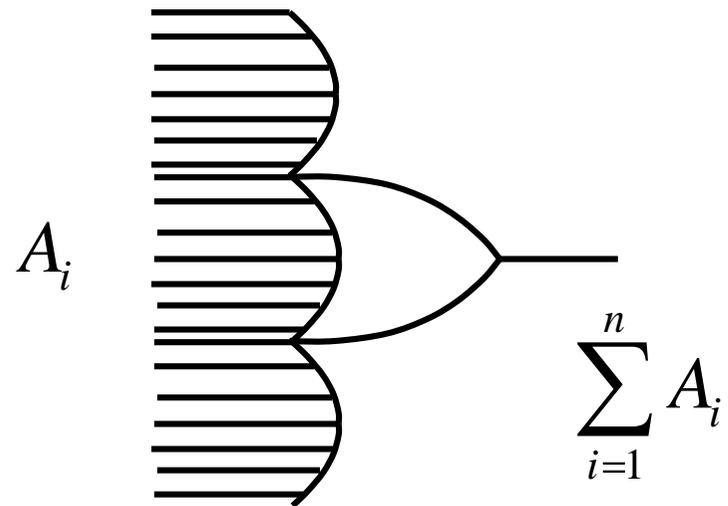
A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1



3) OR (inclusivo)



A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

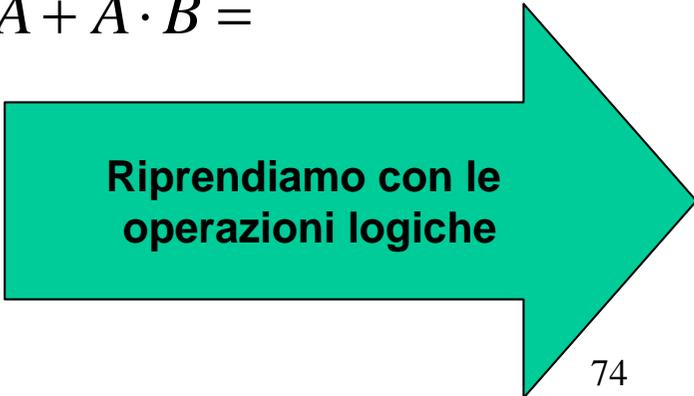


Applicazione di sviluppo in minterm dell'OR

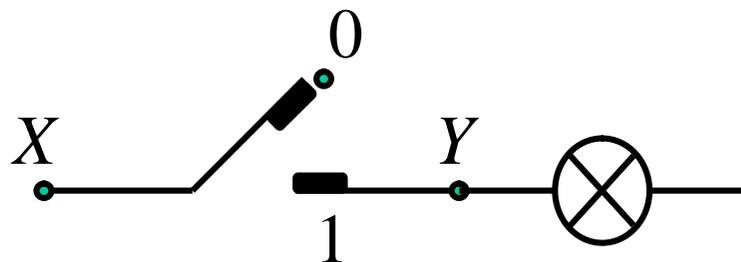
A	B	$A + B$	m
0	0	0	$\overline{A} \cdot \overline{B}$
0	1	1	$A \cdot \overline{B}$
1	0	1	$\overline{A} \cdot B$
1	1	1	$A \cdot B$

A	B	$A + B$	$\overline{A} \cdot \overline{B}$	$\overline{A} \cdot B$	$A \cdot \overline{B}$	$A \cdot B$
0	0	0	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

$$\begin{aligned}
 F &= \sum_{i=1}^{2^n} F_i \cdot m_i = 0 \cdot \overline{A} \cdot \overline{B} + 1 \cdot A \cdot \overline{B} + 1 \cdot \overline{A} \cdot B + 1 \cdot A \cdot B = \left(= F = \sum_{\substack{i=1 \\ F_i=1}}^{2^n} F_i \cdot m_i \right) \\
 &= A \cdot \overline{B} + \overline{A} \cdot B + A \cdot B = A \cdot (\overline{B} + B) + \overline{A} \cdot B = A + \overline{A} \cdot B = \\
 &= (A + \overline{A}) \cdot (A + B) = A + B
 \end{aligned}$$



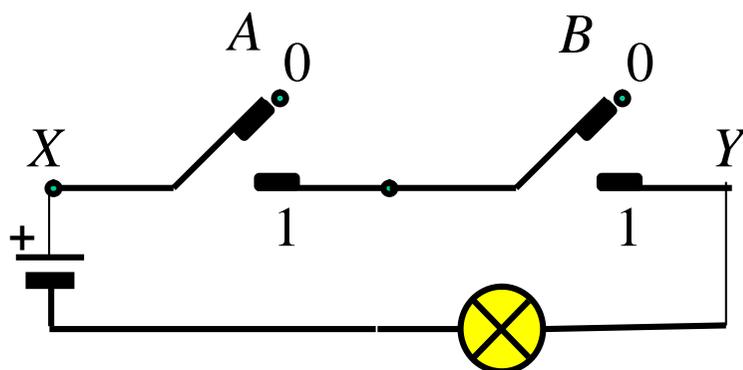
Come li possiamo interpretare?...la preistoria



L' interruttore è un dispositivo a due posizioni (0,1), una delle quali determina la chiusura del contatto elettrico fra i punti X e Y mentre l'altra lascia sconnessi i due punti.

0 → interruttore aperto
1 → interruttore chiuso

2 interruttori in serie → AND logico



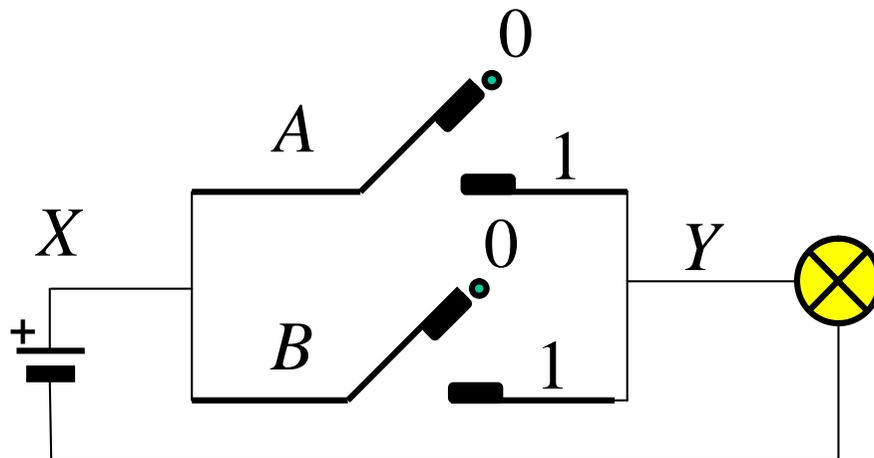
A	B	⊗
0	0	0
0	1	0
1	0	0
1	1	1

spento

acceso

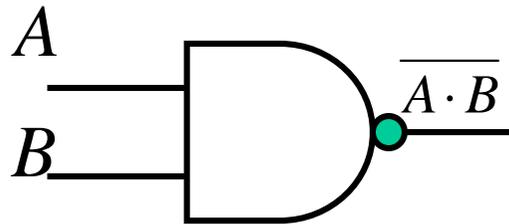
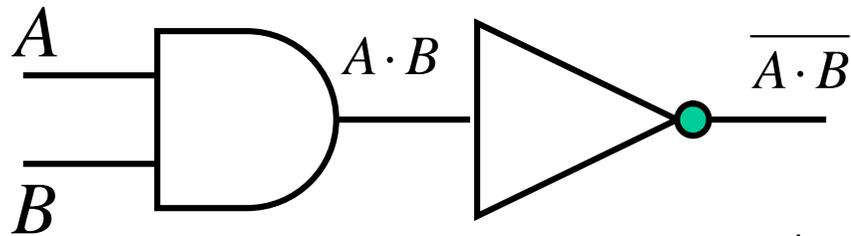
Come li possiamo interpretare?...la preistoria

2 interruttori in parallelo → OR logico



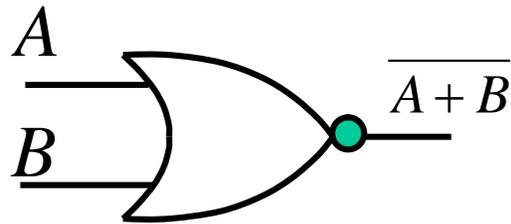
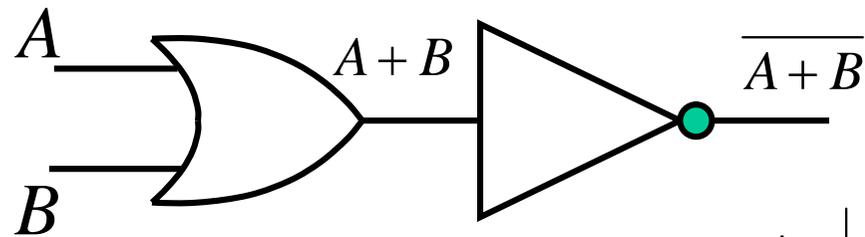
A	B	\otimes	
0	0	0	spento
0	1	1	} acceso
1	0	1	
1	1	1	

4) NAND (porta universale)



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

4) NOR (porta universale)

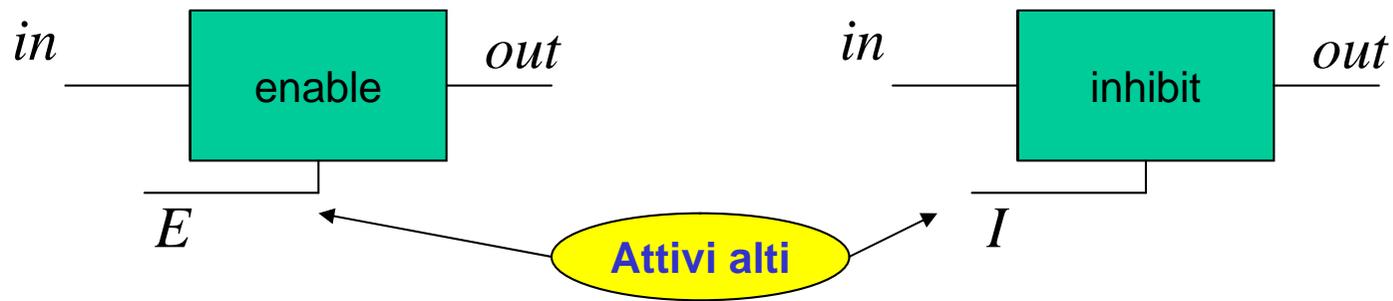


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

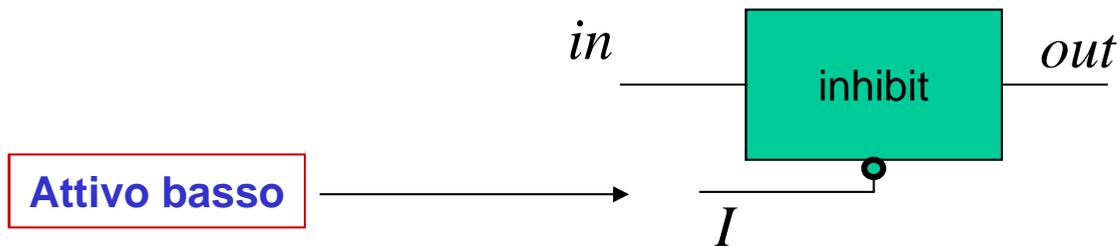
... Ma esistono altri circuiti “non fondamentali”

Enable gate (strobe):

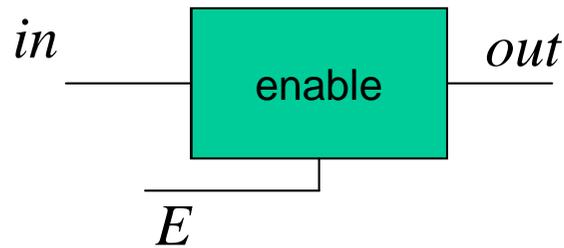
Inhibit gate:



Un comando attivo svolge la funzione relativa al nome del blocco!!



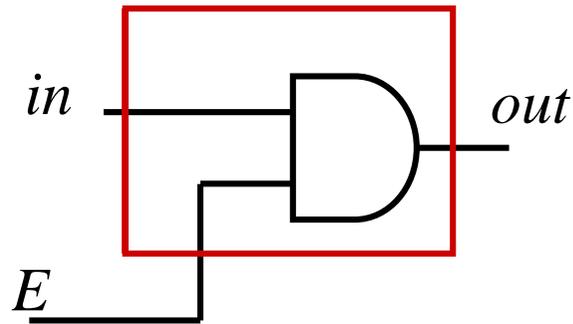
Enable gate (strobe):



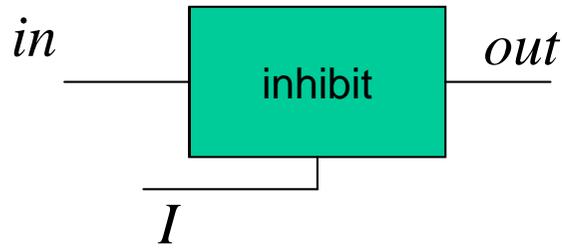
E	In	Out
0	x	0
1	x	x

$$\Rightarrow Out = E \cdot In$$

E	In	Out
0	0	0
0	1	0
1	0	0
1	1	1

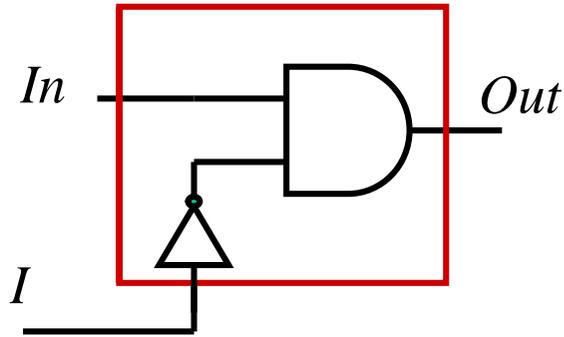


Inhibit gate:

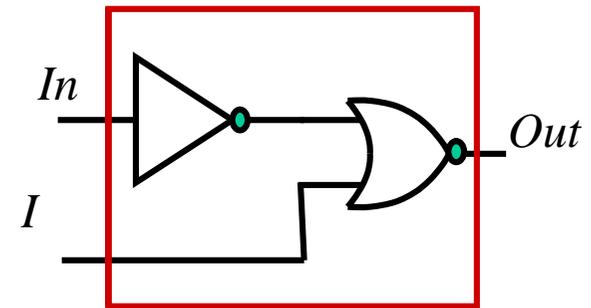


I	In	Out
0	0	0
0	x	x
1	x	0
1	1	0

$\Rightarrow Out = \bar{I} \cdot In$

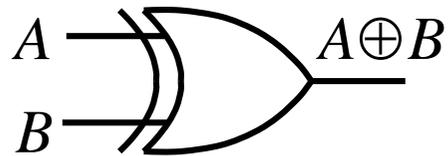


.....e con De Morgan:



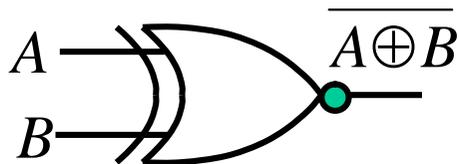
Altre funzioni di due variabili:

- OR esclusivo XOR o EXOR (si differenzia dall'OR inclusivo perché esclude tutte le combinazioni in cui le due variabili sono uguali)



A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

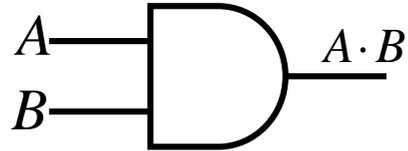
- NOR esclusivo XNOR o EXNOR (si differenzia dal NOR inclusivo perché esclude tutte le combinazioni in cui le due variabili sono diverse)



A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

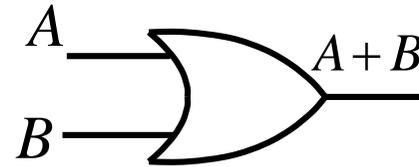
Riepilogo delle funzioni viste

AND



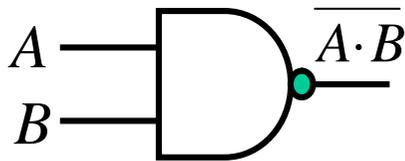
A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

OR



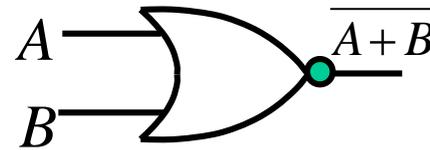
A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

NAND



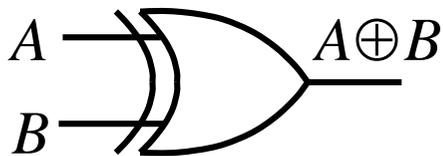
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR



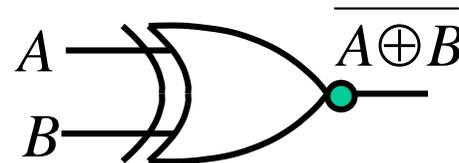
A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

EXOR



A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

EXNOR



A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Dimostriamo: Teorema di DE MORGAN

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

Dimostriamo in 2 modi diversi

1. Algebricamente (già visto)
2. Con le tabelle della verità

Dimostrazione 2 Teorema di De Morgan
Tavole della verità

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

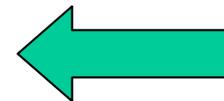
A	B	X = A · B
0	0	0
0	1	0
1	0	0
1	1	1



Cambiamo i nomi:

$$\overline{A} = C \quad \overline{B} = D \quad \overline{X} = Y$$

$$Y = C + D$$



C	D	Y
1	1	1
1	0	1
0	1	1
0	0	0

$$Y = C + D = \overline{A} + \overline{B}$$

$$Y = \overline{X} = \overline{A \cdot B} \Rightarrow \overline{A \cdot B} = \overline{A} + \overline{B}$$

$$\overline{Y} = \overline{C + D}$$

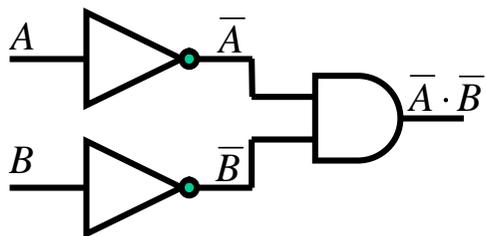
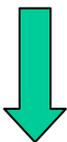
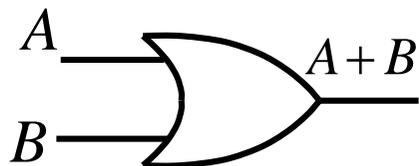
$$\overline{Y} = X = A \cdot B = \overline{C} \cdot \overline{D} \Rightarrow \overline{C + D} = \overline{C} \cdot \overline{D}$$

Teorema di De Morgan → concetto di dualità

Se uno schema logico (elettronico) realizza una certa funzione, per ottenerne il complemento basta scambiare le AND con le OR (o viceversa) e complementare le variabili di ingresso

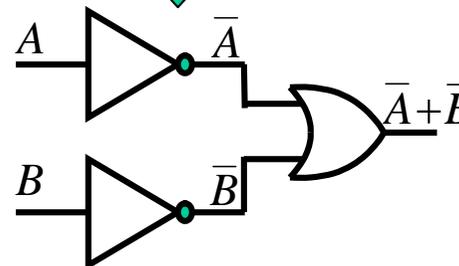
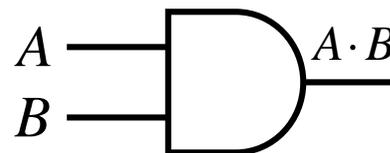
$$F = A + B$$

$$\overline{F} = \overline{A + B} = \overline{A} \cdot \overline{B}$$



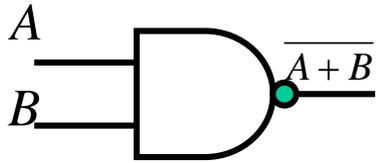
$$F = A \cdot B$$

$$\overline{F} = \overline{A \cdot B} = \overline{A} + \overline{B}$$



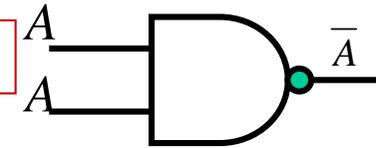
Concetto di porte universali.....cosa vuol dire?

1. NAND

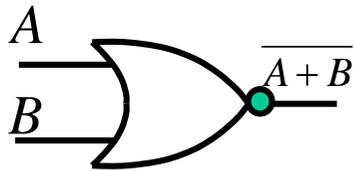


A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

Cosa sono?

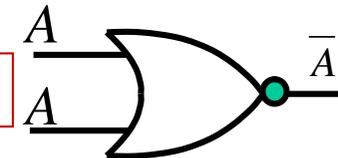


2. NOR



A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Cosa sono?



I
N
V
E
R
T
I
T
O
R
I

Porte universali

Dal teorema di De Morgan:

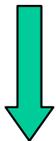
$$A + B = \overline{\overline{A} \cdot \overline{B}} \quad \longrightarrow \text{OR con solo AND e NOT \(\rightarrow\) NAND}$$

$$A \cdot B = \overline{\overline{A} + \overline{B}} \quad \longrightarrow \text{AND con solo OR e NOT \(\rightarrow\) NOR}$$

Da NAND e NOR come NOT:

$$\overline{A \cdot A} = \overline{A} \quad \longrightarrow \text{NOT con NAND}$$

$$\overline{A + A} = \overline{A} \quad \longrightarrow \text{NOT con NOR}$$



$$\overline{\overline{A \cdot B}} = A \cdot B \quad \longrightarrow \text{AND con NAND}$$

$$\overline{\overline{A + B}} = A + B \quad \longrightarrow \text{OR con NOR}$$

AND OR e NOT \(\rightarrow\) NAND
AND OR e NOT \(\rightarrow\) NOR

Ogni circuito può essere
realizzato con solo porte
NAND oppure solo NOR



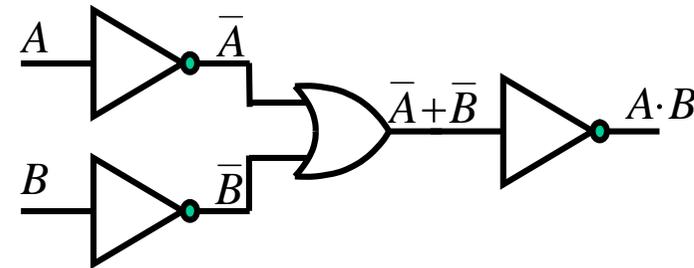
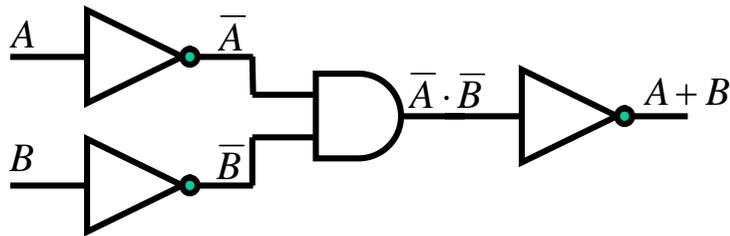
NAND e NOR
Porte universali

Porte universali

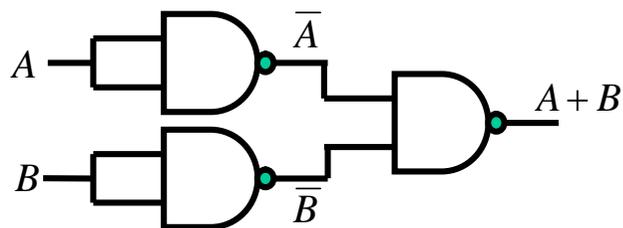
Dal teorema di De Morgan:

$$A + B = \overline{\overline{A} \cdot \overline{B}} \quad A \cdot B = \overline{\overline{A} + \overline{B}}$$

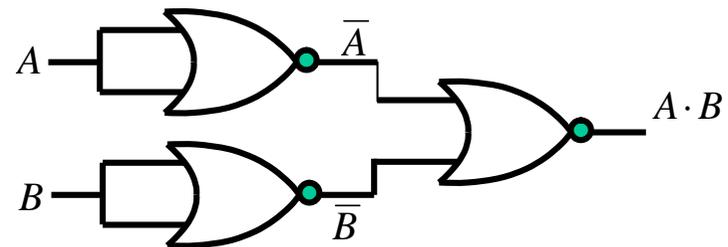
È sufficiente realizzare il circuito corrispondente al primo membro e verificare che la sua tavola della verità sia uguale a quella del secondo membro.



Or con solo porte NAND



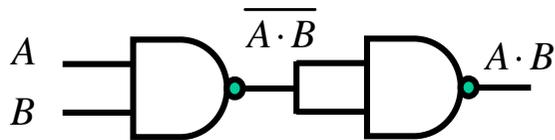
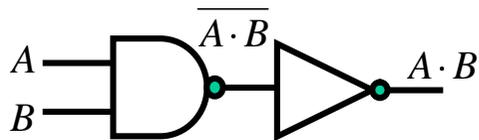
AND con solo porte NOR



Porte universali

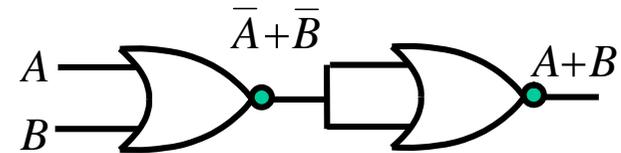
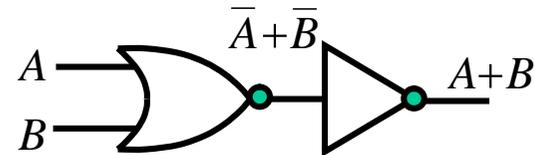
And con solo porte NAND

$$\overline{\overline{A \cdot B}} = A \cdot B$$



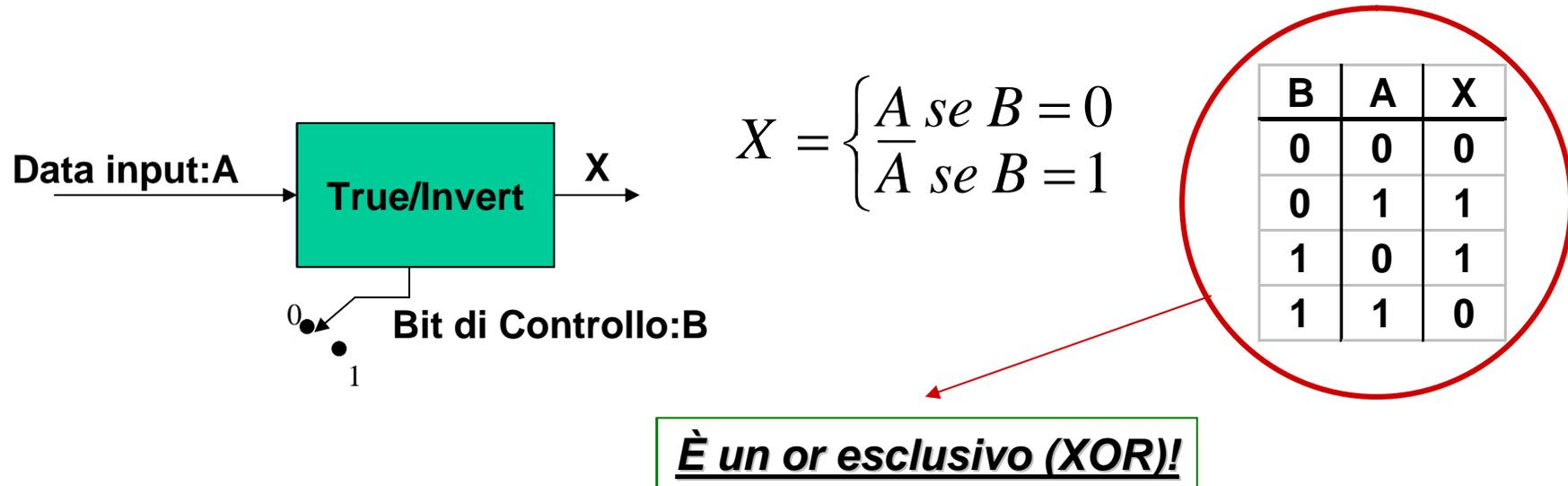
Or con solo porte NOR

$$\overline{\overline{A + B}} = A + B$$



Ogni circuito logico può essere costruito con solo porte NAND oppure con solo porte NOR

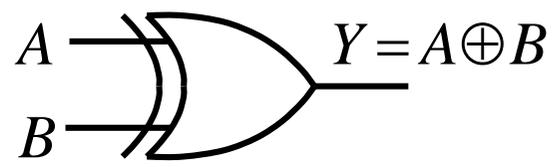
La Funzione True/Invert



Usando la variabile B come bit di controllo in un XOR otteniamo un blocco che esegue la funzione True/Invert:

- se B=0 il blocco riporta in uscita la variabile d'ingresso (True)
- se B=1 il blocco riporta in uscita la variabile d'ingresso negata (Invert)

XOR

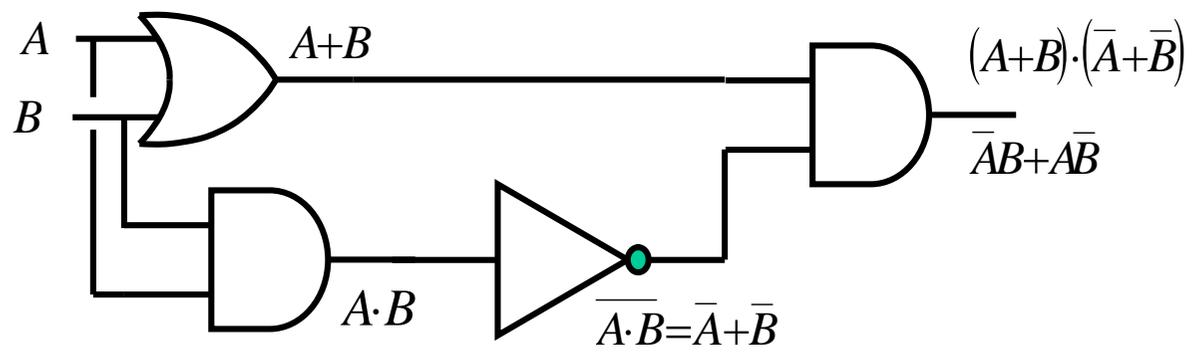


B	A	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

E' uguale alla OR tranne che vale zero se $A=B=1$. Quindi:

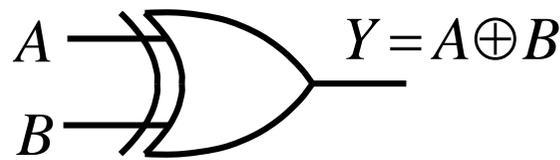
è vera se : è vera $(A + B)$
ed è falsa $A \cdot B$

$$XOR(A, B) = (A + B) \cdot \overline{A \cdot B}$$



4 porte di base (non universali)

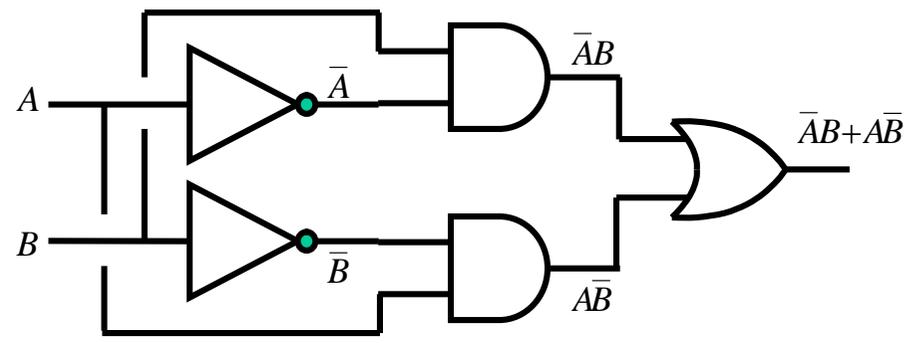
XOR



B	A	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

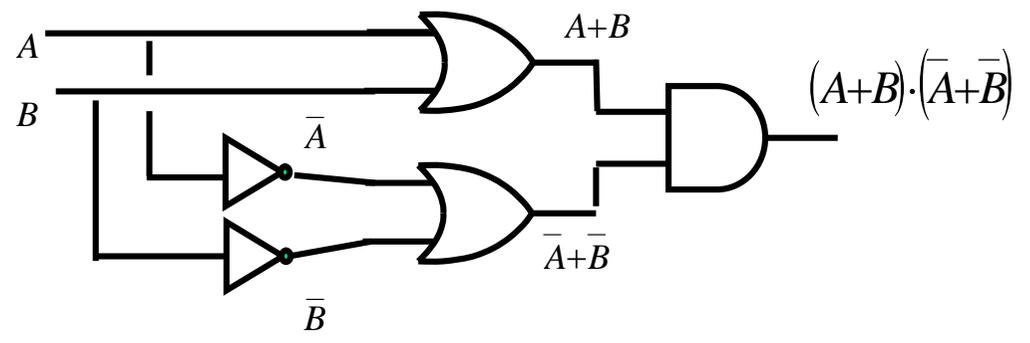
Sviluppo minterm

$$A \oplus B = \bar{A}B + A\bar{B}$$



Sviluppo maxterm

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



Ma sono 5 porte! E si può migliorare.....con: De Morgan

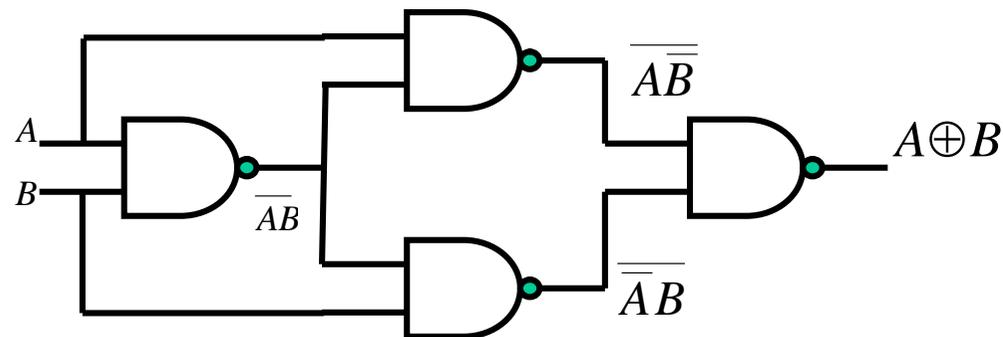
Con De Morgan e qualche trucco... otteniamo XOR con 4 porte

$$A \oplus B = \bar{A}B + A\bar{B} = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}}$$

$$\overline{\bar{A}B} = \overline{\bar{A}B + B\bar{B}} = \overline{B(\bar{A} + \bar{B})} = \overline{B\bar{A}\bar{B}}$$

$$\overline{A\bar{B}} = \overline{A\bar{B} + A\bar{A}} = \overline{A(\bar{B} + \bar{A})} = \overline{A\bar{A}\bar{B}}$$

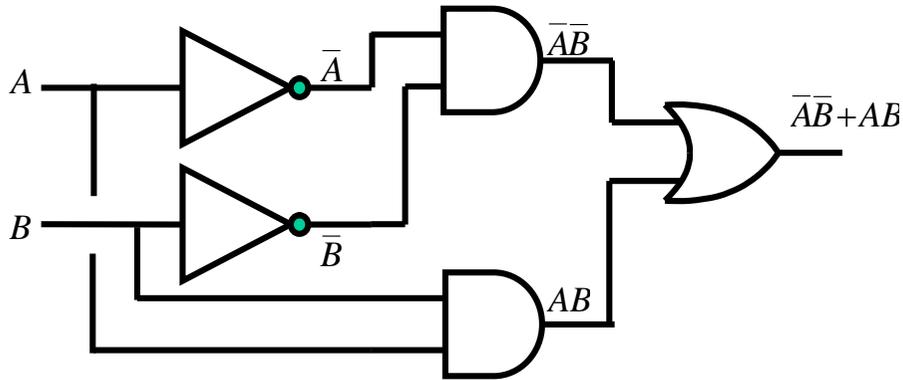
$$\Rightarrow A \oplus B = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}}$$



**Si poteva arrivare a questo risultato con qualche metodo sistematico?
La risposta non è banale, vedremo...**

XNOR

B	A	$A \oplus B$
0	0	1
0	1	0
1	0	0
1	1	1

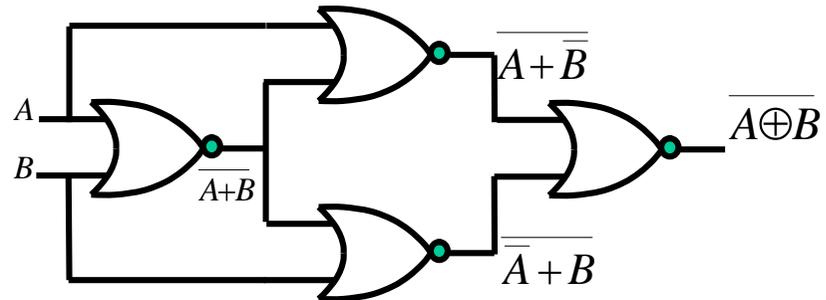


Ma sono 5 porte! E si può migliorare.....con: **De Morgan**

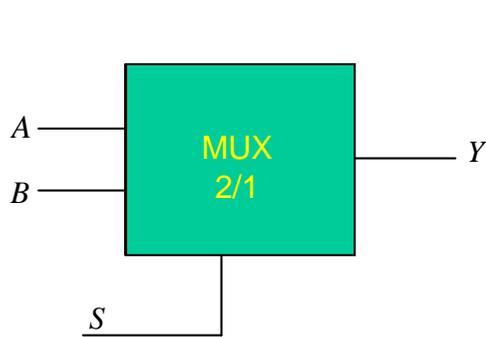
La funzione vale 1 solo se $A=B$ (funzione di eguaglianza e complemento della XOR). Per De Morgan il complemento si ottiene: scambiando le AND con le OR e complementando le variabili di ingresso. In questo caso però l'ultima operazione di complementazione non modifica la tavola della verità:

$$XNOR(A, B) = XNOR(\bar{A}, \bar{B})$$

per cui è sufficiente scambiare le AND con le OR:



Multiplexers (smistatore): smista l'ingresso selezionato in uscita



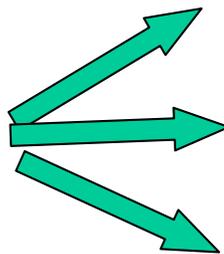
S	A	B	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



S	A	B	Y
0	x	x	B
1	X	X	A

S = bit di selezione,
la variabile d'uscita sarà uguale
all'ingresso A o B come deciso
dal bit S

S= selettore

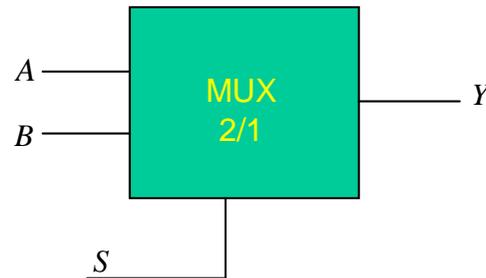


1 bit se le var da smistare sono 2 → S=0,1

2 bit se le var da smistare sono da 2 a 4
→ S₁S₂ = 00, 01, 10, 11

In generale per smistare N variabili serve un
selettore che possa assumere N combinazioni
→ $N=2^{\text{nbit}}$ → $\text{nbit} = \log_2 N$

Multiplexers a due ingressi

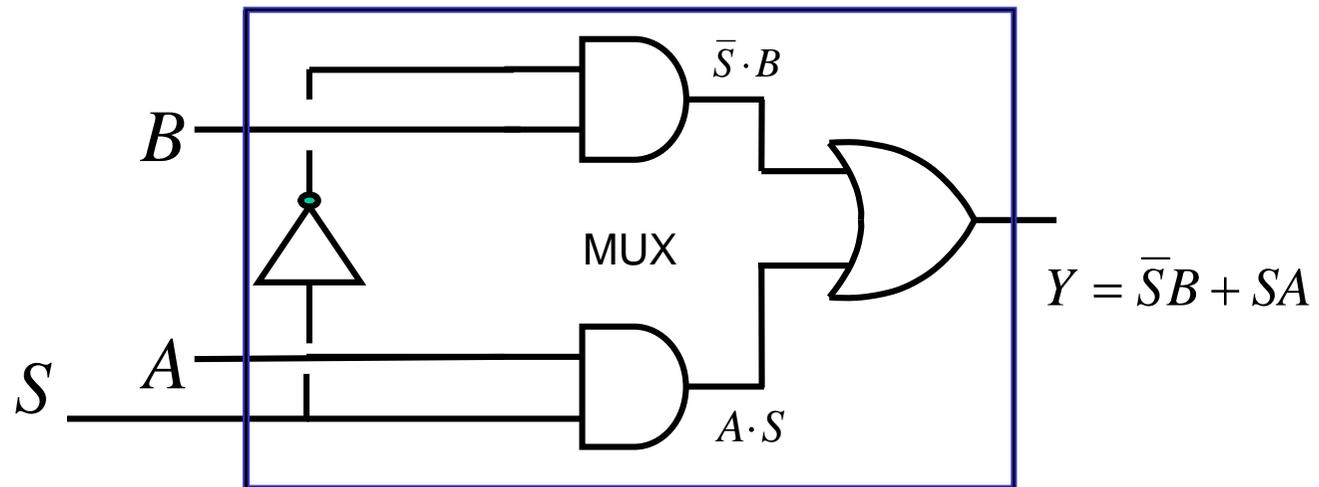


S	A	B	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

COME PREVEDIBILE ABBIAMO UNA SOLA FUNZIONE LOGICA:

$$Y = \bar{A}B\bar{S} + AB\bar{S} + \bar{A}BS + ABS$$

$$Y = \bar{S}B + SA$$



DeMultiplexers (smistatore): smista l'ingresso nell'uscita selezionata

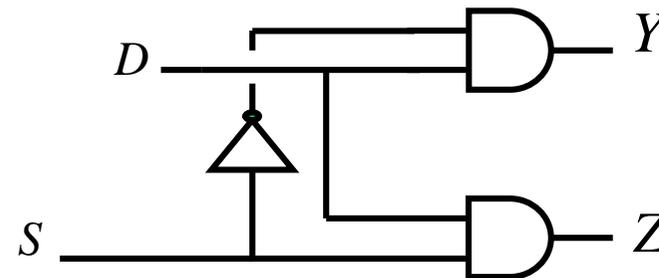


Numero di uscite da gestire e numero di bit di selezione:
analogo al multiplexer

COME PREVEDIBILE ABBIAMO
DUE FUNZIONI LOGICHE:

$$Z = SD$$

$$Y = \bar{S}D$$



Comparatore



A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

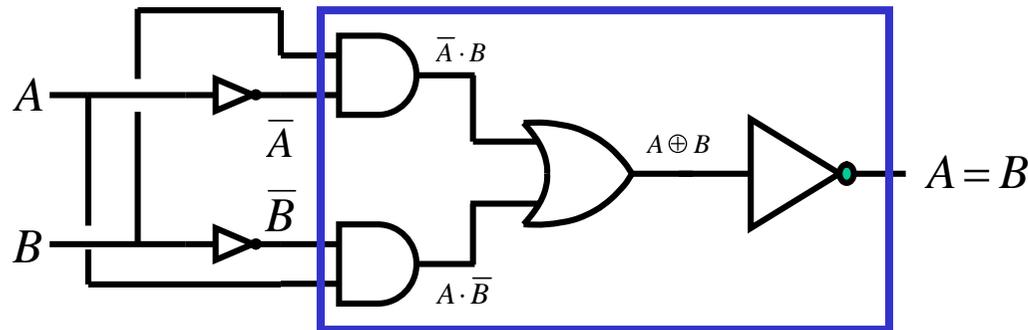
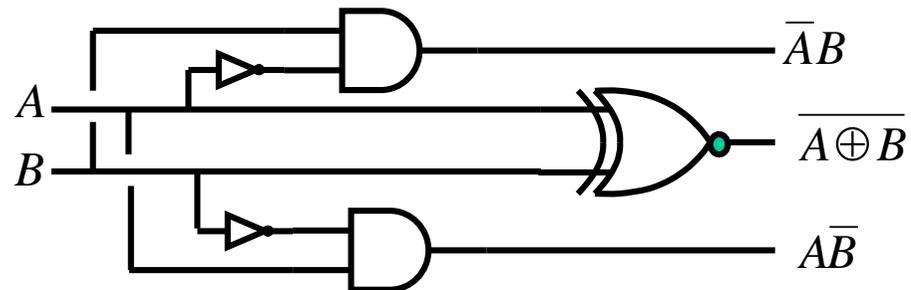
Partiamo dal caso più semplice a 1 bit:

COME PREVEDIBILE ABBIAMO TRE FUNZIONI LOGICHE:

$$A < B = \bar{A} \cdot B$$

$$A > B = A \cdot \bar{B}$$

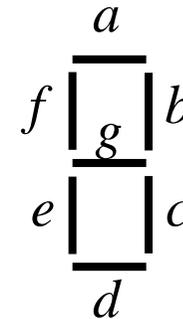
$$A = B = \bar{A} \cdot \bar{B} + AB$$



Convertitore BCD-7 segmenti

BCD: binary coded decimal

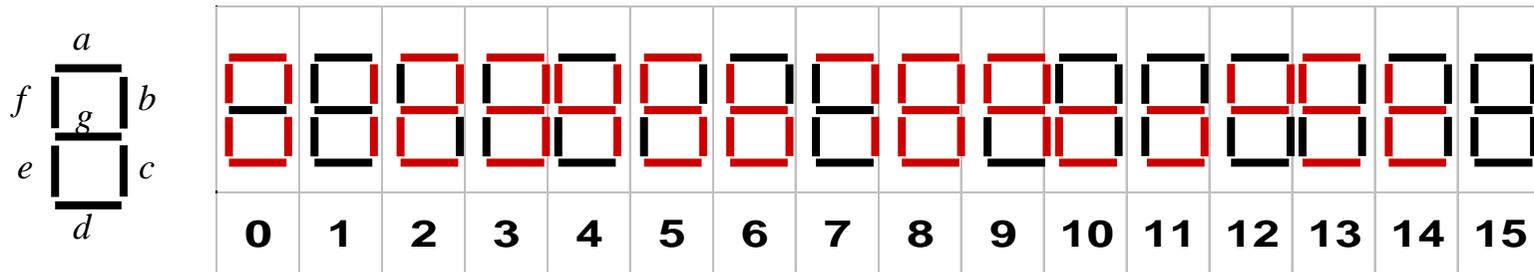
7 segmenti: è un tipo di display



-- Dobbiamo rappresentare i numeri da 0 a 9:
di quanti bit abbiamo bisogno?

$$N_{\text{bit}} = \log_2 10 = 3.32 = 4 \text{ bit}$$

-- Ogni segmento è una funzione dei 4 bit →



'46A, '47A, 'LS47 FUNCTION TABLE (T1)

DECIMAL OR FUNCTION	INPUTS						$\overline{\text{BI}}/\overline{\text{RBO}}^\dagger$	OUTPUTS							NOTE
	$\overline{\text{LT}}$	$\overline{\text{RBI}}$	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	ON	ON	ON	ON	ON	ON	OFF	
1	H	X	L	L	L	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	H	X	L	L	H	L	H	ON	ON	OFF	ON	ON	OFF	ON	
3	H	X	L	L	H	H	H	ON	ON	ON	ON	OFF	OFF	ON	
4	H	X	L	H	L	L	H	OFF	ON	ON	OFF	OFF	ON	ON	
5	H	X	L	H	L	H	H	ON	OFF	ON	ON	OFF	ON	ON	
6	H	X	L	H	H	L	H	OFF	OFF	ON	ON	ON	ON	ON	
7	H	X	L	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF	
8	H	X	H	L	L	L	H	ON	ON	ON	ON	ON	ON	ON	
9	H	X	H	L	L	H	H	ON	ON	ON	OFF	OFF	ON	ON	
10	H	X	H	L	H	L	H	OFF	OFF	OFF	ON	ON	OFF	ON	
11	H	X	H	L	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON	
12	H	X	H	H	L	L	H	OFF	ON	OFF	OFF	OFF	ON	ON	
13	H	X	H	H	L	H	H	ON	OFF	OFF	ON	OFF	ON	ON	
14	H	X	H	H	H	L	H	OFF	OFF	OFF	ON	ON	ON	ON	
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
BI	X	X	X	X	X	X	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	H	L	L	L	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	L	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON	4

H = high level, L = low level, X = irrelevant

NOTES: 1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.

2. When a low logic level is applied directly to the blanking input ($\overline{\text{BI}}$), all segment outputs are off regardless of the level of any other input.

3. When ripple-blanking input ($\overline{\text{RBI}}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output ($\overline{\text{RBO}}$) goes to a low level (response condition).

4. When the blanking input/ripple blanking output ($\overline{\text{BI}}/\overline{\text{RBO}}$) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

$^\dagger \overline{\text{BI}}/\overline{\text{RBO}}$ is wire AND logic serving as blanking input ($\overline{\text{BI}}$) and/or ripple-blanking output ($\overline{\text{RBO}}$).

Consideriamo il segmento e

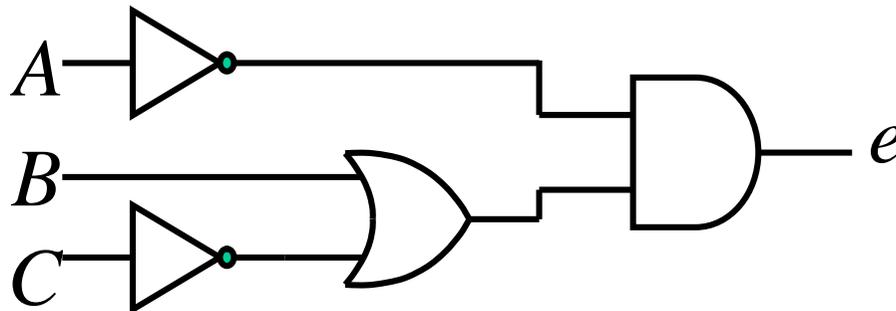
$$e = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD$$

....e semplifichiamo la funzione!

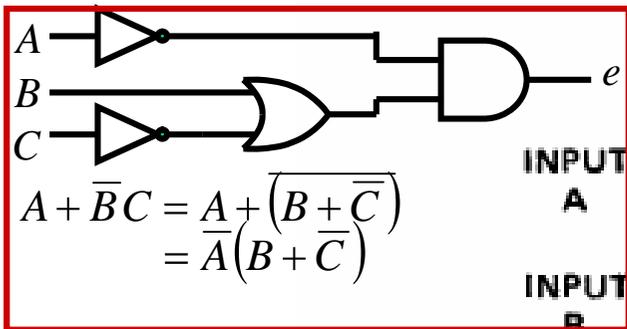
$$\begin{aligned} e &= \overline{A}\overline{C}\overline{D}(B + \overline{B}) + \overline{A}BC(D + \overline{D}) + \overline{A}\overline{C}D(B + \overline{B}) \\ &= \overline{A}\overline{C}\overline{D} + \overline{A}BC + \overline{A}\overline{C}D = \overline{A}\overline{C}(\overline{D} + D) + \overline{A}BC \end{aligned}$$

....poi si ricorre ad un trucco:

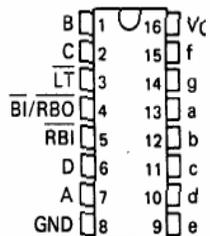
$$\begin{aligned} e &= \overline{A}\overline{C} + \overline{A}BC = \overline{A}\overline{C}(1 + B) + \overline{A}BC \\ &= \overline{A}\overline{C} + \overline{A}\overline{C}B + \overline{A}BC = \overline{A}\overline{C} + \overline{A}B = \overline{A}(\overline{C} + B) \end{aligned}$$



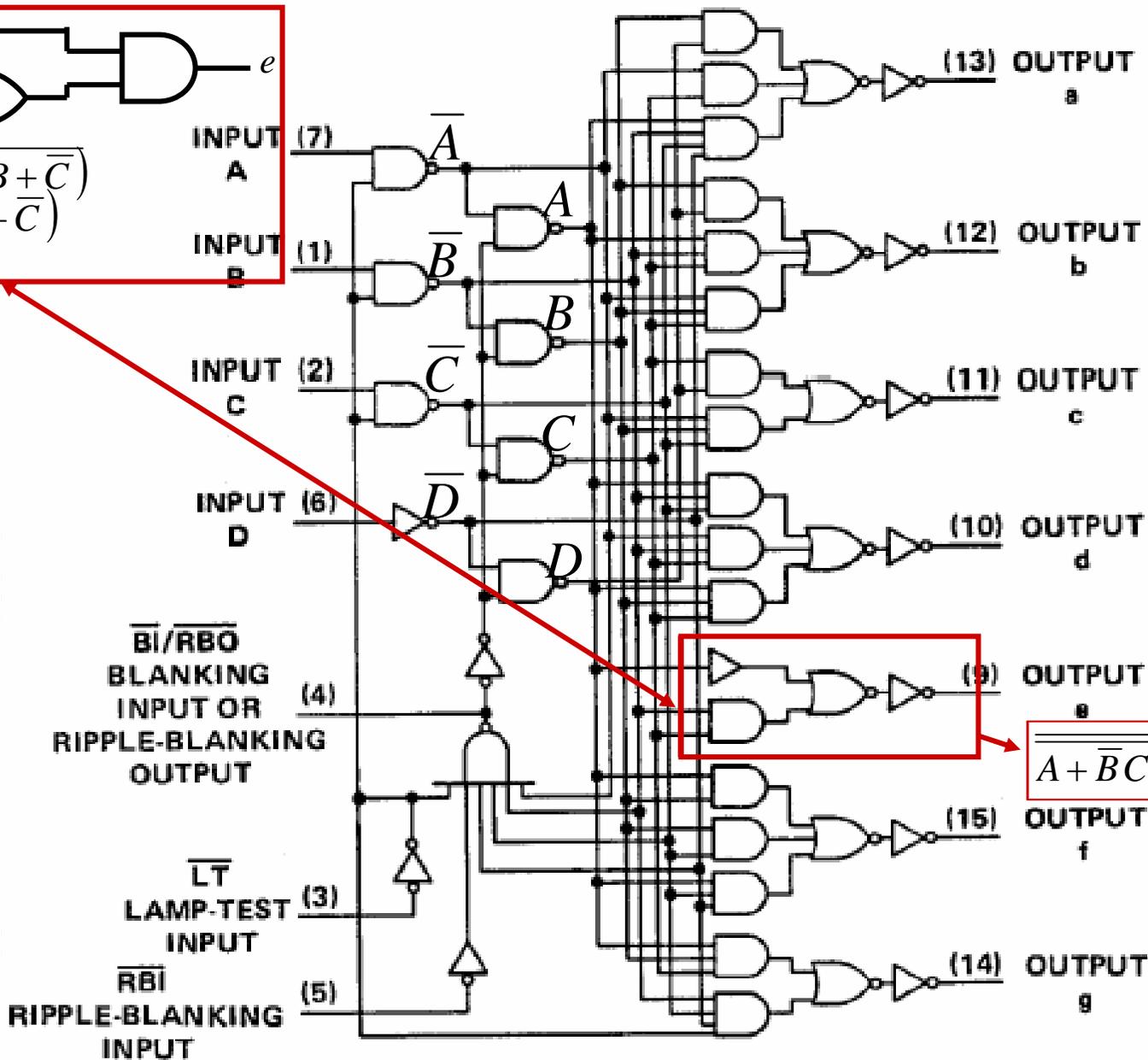
'46A, '47A, 'LS47



SN5446A, SN5447A, SN54LS4
 SN54LS48 ... J PACK
 SN7446A, SN7447A
 SN7448 ... N PACKA
 SN74LS47, SN74LS48 ... D OF
 (TOP VIEW)

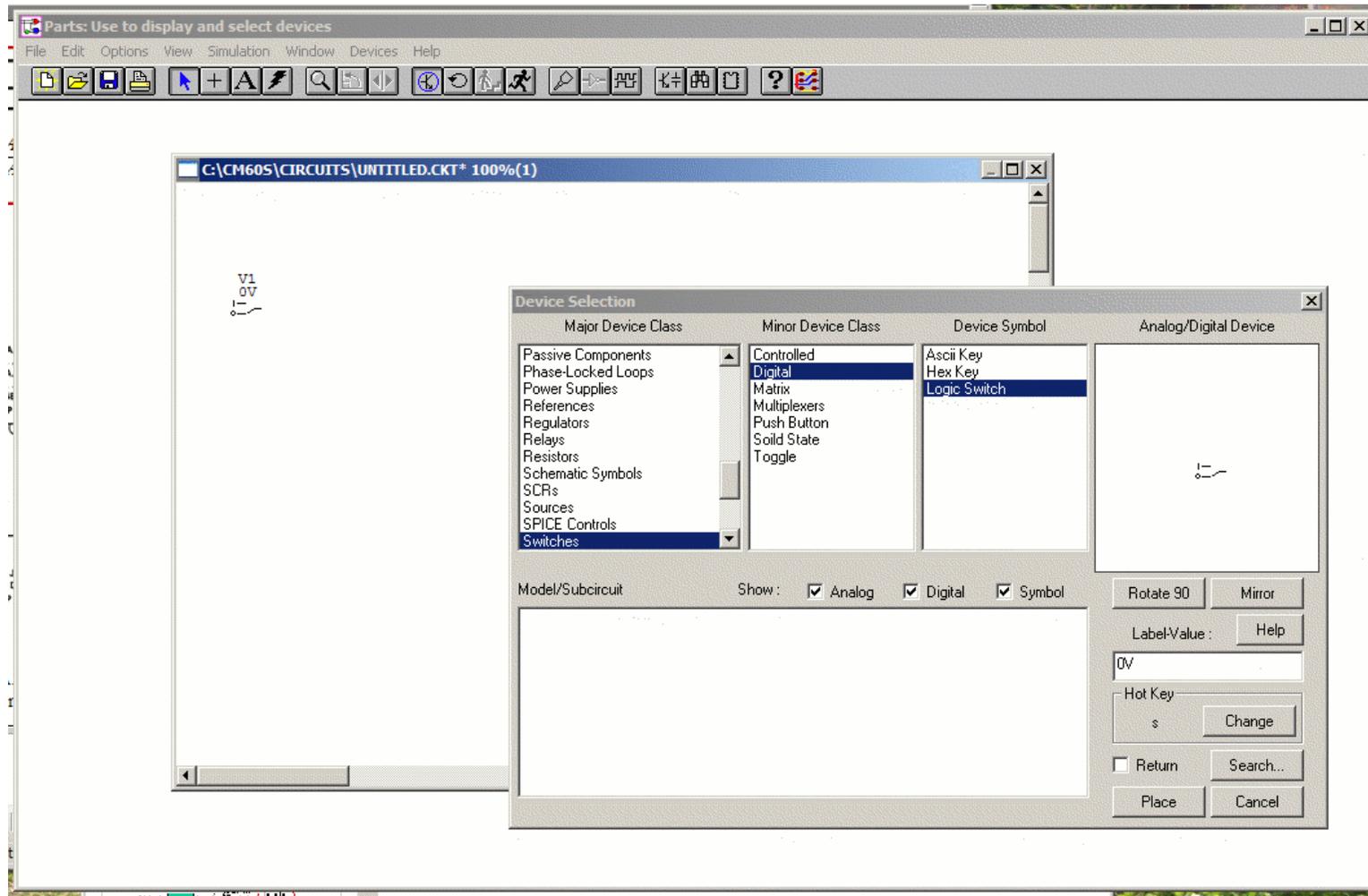


SN54LS49 ... J OR W PA
 SN74LS49 ... D OR N PA
 (TOP VIEW)

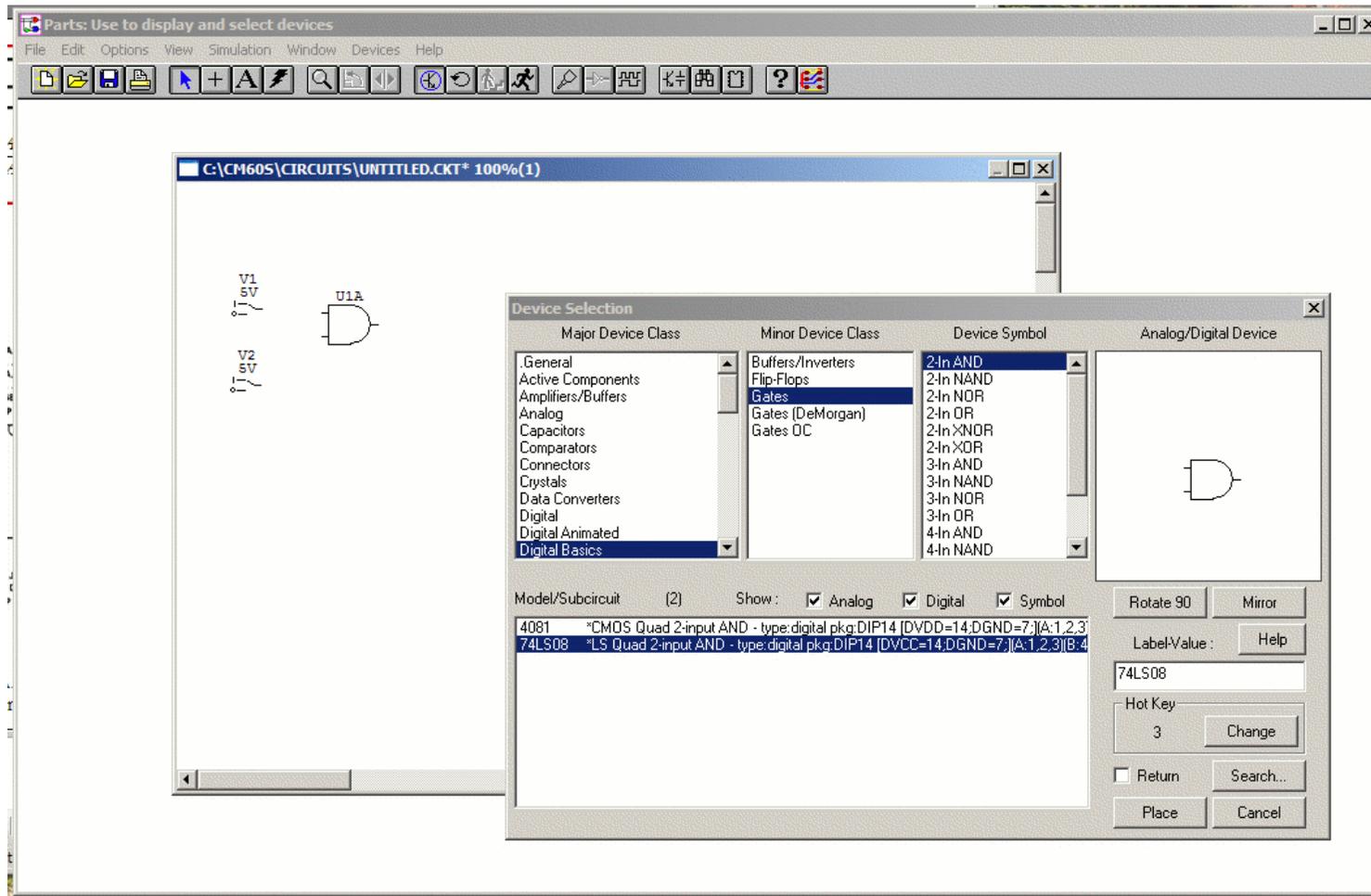


$$\overline{A + BC} = \overline{A} + \overline{BC}$$

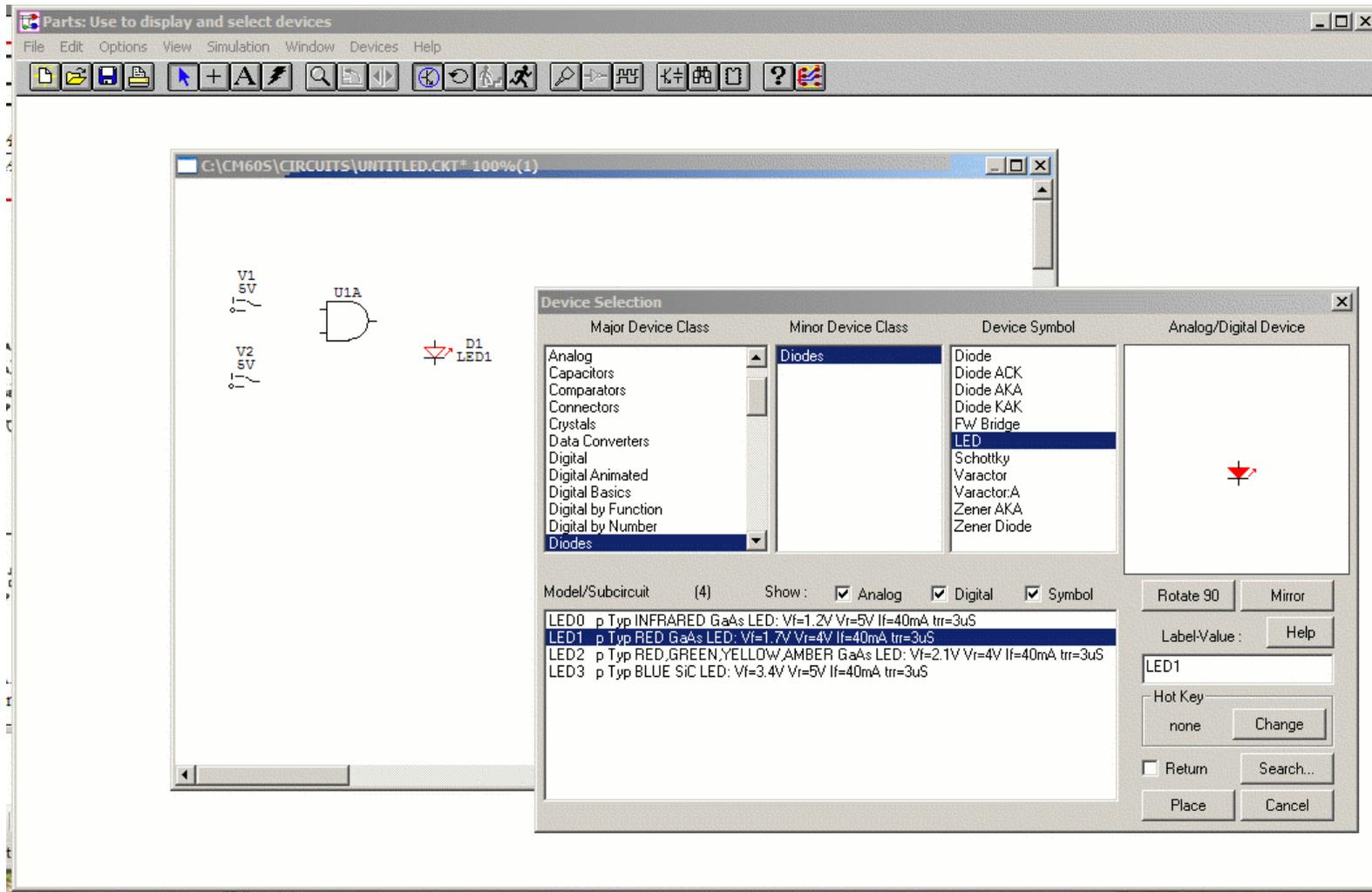
Introduzione a CircuitMaker – qualche immagine



Introduzione a CircuitMaker – qualche immagine



Introduzione a CircuitMaker – qualche immagine



Attività di Laboratorio
ELETRONICA DEI
SISTEMI DIGITALI
Parte I

Corso di Laurea in Informatica e TFI
Anno Accademico 2007-2008

Scopo del Laboratorio di **Elettronica Digitale**

Circuiti: Analogici, Logici
sottosistemi a componenti: Discreti, Integrati

Realizzazioni di prototipi: uso delle breadboard

Circuiti Integrati: chip monolitici

Due grossi gruppi:

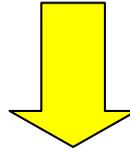
- gruppo bipolare: cariche positive e negative
- gruppo unipolare: cariche di una sola polarità

I gruppi si dividono in famiglie

Le famiglie sono caratterizzate da caratteristiche salienti
All' interno delle famiglie i CI sono tutti compatibili tra loro

- Stessi livelli
- Stesse alimentazioni
- Potenze compatibili

Tra famiglie diverse i CI sono (in generale) incompatibili tra loro



Circuiti di interfaccia

I principali parametri che caratterizzano le famiglie:

1. Ritardo di propagazione:maggiore nei circuiti unipolari
2. Dissipazione di potenza:inferiore negli unipolari
3. Capacità di pilotaggio (fan-out):maggiore negli unipolari
4. Immunità al rumore:migliore negli unipolari
5. Capacità di una porta (fan-in): equivalente
6. Densità di integrazione:maggiore negli unipolari

La scelta va fatta in base alle caratteristiche/necessità di progetto

Famiglie Bipolari:

1. RTL:obsoleta
2. DTL:obsoleta
3. HTL:
4. TTL standard:
5. TTL a bassa dissipazione:
6. TTL high speed:
7. TT Schottky:
8. ECL:
9. I²L:

Famiglie unipolari:

1. P-MOS H.V.:
2. P-MOS L.V.:
3. N-MOS:
4. C-MOS:

Scale di Integrazione:

1. **S**(mall)**S**(cale)**I**(ntegration):
12 porte (50 transistor equivalenti)
2. **M**(edium)**S**(cale)**I**(tegration):
12-100 porte (50-500 trs equivalenti)
3. **L**(arge)**S**(cale)**I**(ntegration):
100-1000 porte (500-4000 trs equivalenti)
4. **V**(ery)**L**(arge)**S**(cale)**I**(ntegration):
>1000 porte ($\geq 10^7$ trs eq. per il
PENTIUM INTEL nel 2002)

TI LOGIC DEVICE NOMENCLATURE

Are you confused by the device names and numbers of logic products? Do you need to know if a part has bus hold or series damping resistors? What kind of package the part is available in? The table below takes the mystery out of what all the letters and numbers in TI standard catalog logic devices represent. Take your favorite TI logic part number and find out what it can do.

SN	74	LVC	H	16	2	244	A	DGG	R
1	2	3	4	5	6	7	8	9	10

1. Standard Prefix

- Example: SNJ - Conforms to MIL-PRF-38535 (QML)

2. Temperature Range

- 54 - Military
- 74 - Commercial

3. Family

4. Special Features

- Blank = No Special Features
- C - Configurable Vcc (LVCC)
- D - [Level-Shifting Diode](#) (CBTD)
- H - [Bus Hold](#) (ALVCH)
- K - Undershoot-Protection Circuitry (CBTK)
- R - [Damping Resistor on Inputs/Outputs](#) (LVCR)
- S - Schottky Clamping Diode (CBTS)
- Z - Power-Up 3-State (LVCZ)

5. Bit Width

- Blank = Gates, MSI, and Octals
- 1G - Single Gate
- 8 - Octal IEEE 1149.1 ([JTAG](#))
- 16 - [Widebus™](#) (16, 18, and 20 bit)
- 18 - Widebus IEEE 1149.1 ([JTAG](#))
- 32 - [Widebus+™](#) (32 and 36 bit)

6. Options

- Blank = No Options
- 2 - [Series-Damping Resistor on Outputs](#)
- 4 - Level Shifter
- 25 - 25-ohm Line Driver

7. Function

- 244 - Noninverting Buffer/Driver
- 374 - D-Type Flip-Flop
- 573 - D-Type Transparent Latch
- 640 - Inverting Transceiver

8. Device Revision

- Blank = No Revision
- Letter Designator A-Z

9. Packages

- D, DW - Small-Outline Integrated Circuit (SOIC)
- DB, DL - Shrink Small-Outline Package (SSOP)
- DBB, DGV - Thin Very Small-Outline Package (TVSOP)
- DBQ - Quarter-Size Outline Package (QSOP)
- DBV, DCK - Small-Outline Transistor Package (SOT)
- DGG, PW - Thin Shrink Small-Outline Package (TSSOP)
- FK - Leadless Ceramic Chip Carrier (LCCC)
- FN - Plastic Leaded Chip Carrier (PLCC)
- GB - Ceramic Pin Grid Array (CPGA)
- GKE, GKF - MicroStar BGA™ Low-Profile Fine-Pitch Ball Grid Array (LFBGA)
- GQL, GQN - MicroStar Junior BGA Very-Thin-Profile Fine-Pitch Ball Grid Array (VFBGA)
- HFP, HS, HT, HV - Ceramic Quad Flat Package (CQFP)
- J, JT - Ceramic Dual-In-Line Package (CDIP)
- N, NP, NT - Plastic Dual-In-Line Package (PDIP)
- NS, PS - Small Outline Package (SOP)
- PAG, PAH, PCA, PCB, PM, PN, PZ - Thin Quad Flat Package (TQFP)
- PH, PQ, RC - Quad Flat Package (QFP)
- W, WA, WD - Ceramic Flat Package (CFP)

10. Tape and Reel

All new or changed devices in the **DB** and **PW** package types include the **R** designation for reeled product. Existing products designated as **LE** presently maintain that designation, but will be converted to **R** in the future.

Nomenclature Examples:

- For an Existing Device - SN74LVTxxxDB**LE**
- For a New or Changed Device - SN74LVTxxxADB**R**
- **LE** - Left Embossed (valid for DB and PW packages only)
- **R** - Standard (valid for all surface-mount packages except existing DB and PW devices)

Family Comparison

The table below compares some typical characteristics of several popular logic families available in the market today. The following sections provide brief explanations of the various parameters.

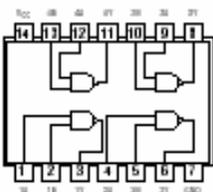
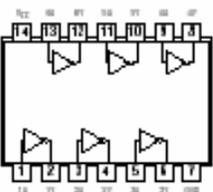
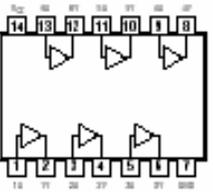
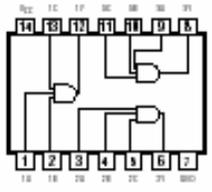
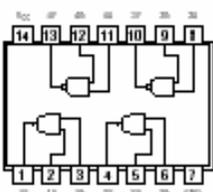
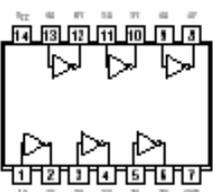
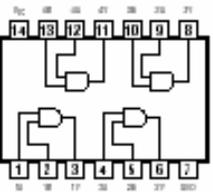
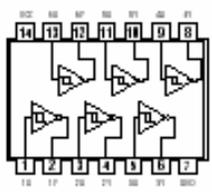
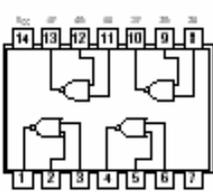
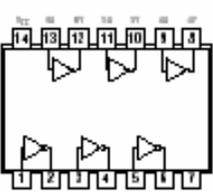
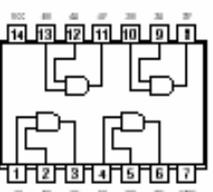
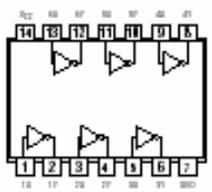
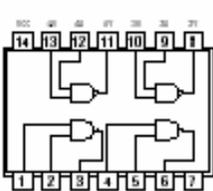
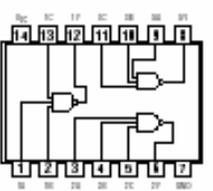
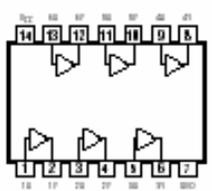
Typical Commercial Parameter (0°C to +70°C)	Logic Families												
	TTL				CMOS					ECL			
	LS	ALS	ABT	FAST	MG	HC	FACT	LVC	LCX	10H	100K	ECL in PS(3)	E-Lite
Speed Gate Prop Delay (ns)	9	7	2.7	3	65	8	5	3.3	3.5	1	0.75	0.33	0.22
Flip-Flop Toggle Rate (MHz)	33	45	200	125	4	45	160	200	200	330	400	1,000	2800
Output Edge Rate (ns)	6	3	3	2	50	4	2	3.7	3.6	1	0.7	0.5	0.25
Power Consumption Per Gate (mW)													
Quiescent	5	1.2	0.005	12.5	0.0006	0.003	0.0001	0.003	1E-04	25	50	25	73
Operating (1 MHz)	5	1.2	1.0	12.5	0.04	0.6	0.6	0.8	0.3	25	50	25	73
Supply Voltage (V)	+4.5 to +5.5	+4.5 to +5.5	+4.5 to +5.5	+4.5 to +5.5	+3 to +18	+2 to +6	+1.2 to +3.6	+2 to +3.6	+2 to +6	-4.5 to -5.5	-4.2 to -4.8	-4.2 to -5.5	-4.2 to -5.5
Output Drive (mA)	8	8	32/64	20	1	4	24	24	24	50 ohm load	50 ohm load	50 ohm load	50 ohm load

Pin Assignments

<p>00 QUADRUPLE 2-INPUT POSITIVE-NAND GATES positive logic: $V = \overline{A \cdot B}$</p> <p>See page 133</p>	<p>04 HEX INVERTERS positive logic: $V = \overline{X}$</p> <p>See page 133</p>
<p>01 QUADRUPLE 2-INPUT POSITIVE-NAND GATES WITH OPEN-COLLECTOR OUTPUTS positive logic: $V = \overline{A \cdot B}$</p> <p>See page 133</p>	<p>U04 HEX INVERTERS positive logic: $V = \overline{X}$</p> <p>See page 133</p>
<p>02 QUADRUPLE 2-INPUT POSITIVE-NOR GATES positive logic: $V = \overline{A + B}$</p> <p>See page 131</p>	<p>05 HEX INVERTERS WITH OPEN-COLLECTOR OUTPUTS positive logic: $V = \overline{X}$</p> <p>See page 133</p>
<p>03 QUADRUPLE 2-INPUT POSITIVE-NAND GATES WITH OPEN-COLLECTOR OUTPUTS positive logic: $V = \overline{A \cdot B}$</p> <p>See page 133</p>	<p>06 HEX INVERTER BUFFER/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS positive logic: $V = \overline{X}$</p> <p>See page 133</p>

Pin Assignments

<p>112 DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR</p> <p>See page 135</p>	<p>124 DUAL VOLTAGE-CONTROLLED OSCILLATORS WITH ENABLE INPUTS</p> <p>See page 135</p>
<p>121 MONOSTABLE MULTIVIBRATOR</p> <p>See page 135</p>	<p>125 QUADRUPLE BUS BUFFER GATES WITH THREE-STATE OUTPUTS positive logic: $V = A$</p> <p>See page 135</p>
<p>122 RETRIGGERABLE MONOSTABLE MULTIVIBRATORS WITH CLEAR</p> <p>See page 137</p>	<p>126 QUADRUPLE BUS BUFFER GATES WITH THREE-STATE OUTPUTS positive logic: $V = A$</p> <p>See page 137</p>
<p>123 DUAL RETRIGGERABLE MONOSTABLE MULTIVIBRATORS WITH CLEAR</p> <p>See page 135</p>	<p>128 SN54128...75-Q LINE DRIVER SN74128...50-Q LINE DRIVER positive logic: $V = A = B$</p> <p>See page 135</p>

<p>00 QUADRUPLE 2-INPUT POSITIVE-NAND GATES positive logic: $Y = \overline{A \cdot B}$</p>  <p>See page 139</p>	<p>04 HEX INVERTERS positive logic: $Y = \overline{A}$</p>  <p>See page 140</p>	<p>07 HEX BUFFERS/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS positive logic: $Y = \overline{A}$</p>  <p>See page 145</p>	<p>11 TRIPLE 3-INPUT POSITIVE-AND GATES positive logic: $Y = A \cdot B \cdot C$</p>  <p>See page 140</p>
<p>01 QUADRUPLE 2-INPUT POSITIVE-NAND GATES WITH OPEN-COLLECTOR OUTPUTS positive logic: $Y = \overline{A \cdot B}$</p>  <p>See page 140</p>	<p>U04 HEX INVERTERS positive logic: $Y = \overline{A}$</p>  <p>See page 146</p>	<p>08 QUADRUPLE 2-INPUT POSITIVE-AND GATES positive logic: $Y = A \cdot B$</p>  <p>See page 145</p>	<p>14 HEX SCHMITT-TRIGGER INVERTERS positive logic: $Y = \overline{A}$</p>  <p>See page 150</p>
<p>02 QUADRUPLE 2-INPUT POSITIVE-NOR GATES positive logic: $Y = \overline{A + B}$</p>  <p>See page 141</p>	<p>05 HEX INVERTERS WITH OPEN-COLLECTOR OUTPUTS positive logic: $Y = \overline{A}$</p>  <p>See page 146</p>	<p>09 QUADRUPLE 2-INPUT POSITIVE-AND GATES WITH OPEN-COLLECTOR OUTPUTS positive logic: $Y = A \cdot B$</p>  <p>See page 147</p>	<p>16 HEX INVERTER BUFFERS/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS positive logic: $Y = \overline{A}$</p>  <p>See page 151</p>
<p>03 QUADRUPLE 2-INPUT POSITIVE-NAND GATES WITH OPEN-COLLECTOR OUTPUTS positive logic: $Y = \overline{A \cdot B}$</p>  <p>See page 142</p>	<p>06 HEX INVERTER BUFFERS/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS positive logic: $Y = \overline{A}$</p>  <p>See page 145</p>	<p>10 TRIPLE 3-INPUT POSITIVE-NAND GATES positive logic: $Y = \overline{A \cdot B \cdot C}$</p>  <p>See page 148</p>	<p>17 HEX BUFFERS/DRIVERS WITH OPEN-COLLECTOR HIGH-VOLTAGE OUTPUTS positive logic: $Y = \overline{A}$</p>  <p>See page 151</p>

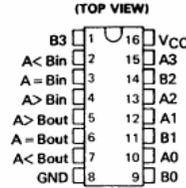
SN5485, SN54LS85, SN54S85
SN7485, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS
SDLS123 - MARCH 1974 - REVISED MARCH 1988

TYPE	TYPICAL POWER DISSIPATION	TYPICAL DELAY (4-BIT WORDS)
'85	275 mW	23 ns
'LS85	52 mW	24 ns
'S85	385 mW	11 ns

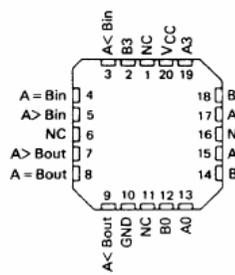
description

These four-bit magnitude comparators perform comparison of straight binary and straight BCD (8-4-2-1) codes. Three fully decoded decisions about two 4-bit words (A, B) are made and are externally available at three outputs. These devices are fully expandable to any number of bits without external gates. Words of greater length may be compared by connecting comparators in cascade. The A > B, A < B, and A = B outputs of a stage handling less-significant bits are connected to the corresponding A > B, A < B, and A = B inputs of the next stage handling more-significant bits. The stage handling the least-significant bits must have a high-level voltage applied to the A = B input. The cascading paths of the '85, 'LS85, and 'S85 are implemented with only a two-gate-level delay to reduce overall comparison times for long words. An alternate method of cascading which further reduces the comparison time is shown in the typical application data.

SN5485, SN54LS85, SN54S85 . . . J OR W PACKAGE
SN7485 . . . N PACKAGE
SN74LS85, SN74S85 . . . D OR N PACKAGE



SN54LS85, SN54S85 . . . FK PACKAGE



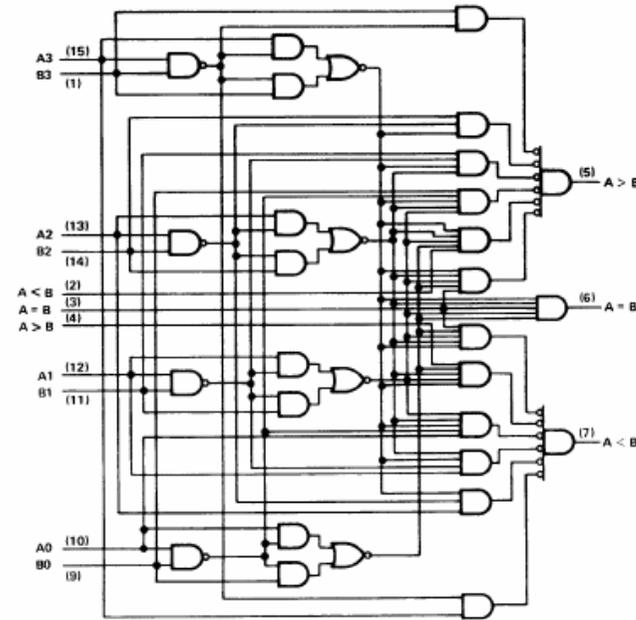
NC - No internal connection

FUNCTION TABLE

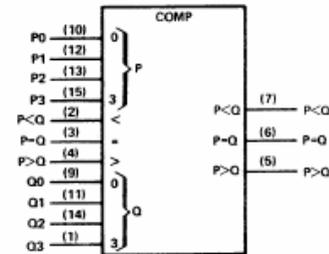
COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B2	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A2 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

SN5485, SN54LS85, SN54S85
SN7485, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS
SDLS123 - MARCH 1974 - REVISED MARCH 1988

logic diagrams (positive logic)



logic symbol†



†This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12. Pin numbers shown are for D, J, N, and W packages.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

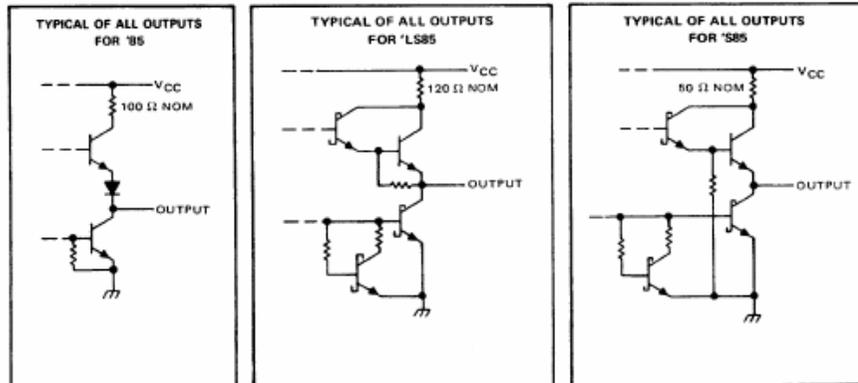
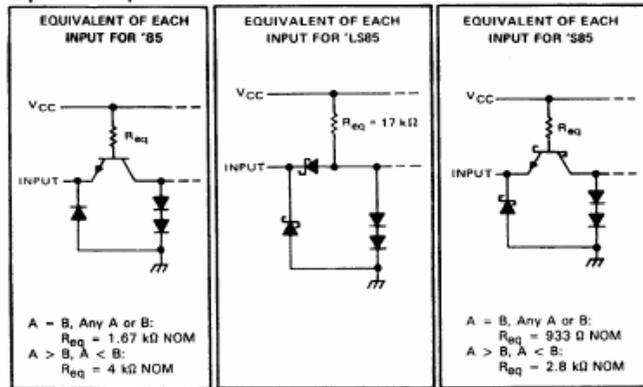


Copyright © 1988, Texas Instruments Incorporated



SN5485, SN54LS85, SN54S85
SN7485, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS
SOLS 123 - MARCH 1974 - REVISED MARCH 1985

schematics of inputs and outputs



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

	SN54 ¹ SN54S ¹	SN54LS ¹	SN74 ¹ SN74S ¹	SN74LS ¹	UNIT
Supply voltage, V_{CC} (see Note 1)	7	7	7	7	V
Input voltage	5.5	7	5.5	7	V
Intermitter voltage (see Note 2)	5.5		5.5		V
Operating free-air temperature range	-55 to 125		-0 to 70		°C
Storage temperature range	-65 to 150		-65 to 150		°C

NOTES: 1. Voltage values, except intermitter voltage, are with respect to network ground terminal.
2. This is the voltage between two emitters of a multiple-emitter input transistor. This rating applies to each A input in conjunction with its respective B input of the '85 and 'S85.

SN5485, SN54LS85, SN54S85
SN7485, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS
SOLS 123 - MARCH 1974 - REVISED MARCH 1985

recommended operating conditions

	SN5485			SN7485			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			-400			-400	μA
Low-level output current, I_{OL}			16			16	mA
Operating free-air temperature, T_A	-55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS ¹	MIN	TYP ²	MAX	UNIT	
V_{IH} High-level input voltage		2			V	
V_{IL} Low-level input voltage		0.8			V	
V_{IK} Input clamp voltage	$V_{CC} = \text{MIN.}$, $I_I = -12 \text{ mA}$	-1.5			V	
V_{OH} High-level output voltage	$V_{CC} = \text{MIN.}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = 0.8 \text{ V}$, $I_{OH} = -400 \mu\text{A}$	2.4	3.4		V	
V_{OL} Low-level output voltage	$V_{CC} = \text{MIN.}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = 0.8 \text{ V}$, $I_{OL} = 16 \text{ mA}$	0.2	0.4		V	
I_I Input current at maximum input voltage	$V_{CC} = \text{MAX.}$, $V_I = 5.5 \text{ V}$	1			mA	
I_{IH} High-level input current	A < B, A > B inputs	40			μA	
	all other inputs	120				
I_{IL} Low-level input current	A < B, A > B inputs	-1.8			mA	
	all other inputs	-4.8				
I_{OS} Short-circuit output current ³	$V_{CC} = \text{MAX.}$, $V_O = 0$	SN5485	-20	-55	mA	
		SN7485	-18	-55	mA	
I_{CC} Supply current	$V_{CC} = \text{MAX.}$, See Note 4	55			88	mA

¹For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

²All typical values are at $V_{CC} = 5 \text{ V}$, $T_A = 25^\circ\text{C}$.

³Not more than one output should be shorted at a time.

NOTE 4: I_{CC} is measured with outputs open, A = B grounded, and all other inputs at 4.5 V.

switching characteristics, $V_{CC} = 5 \text{ V}$, $T_A = 25^\circ\text{C}$

PARAMETER ¹	FROM INPUT	TO OUTPUT	NUMBER OF GATE LEVELS	TEST CONDITIONS	MIN	TYP	MAX	UNIT		
t_{PLH}	Any A or B data input	A < B, A > B	1	$C_L = 15 \mu\text{F}$, $R_L = 400 \Omega$, See Note 5	7			ns		
			2		12					
			3		17					
			4		23					
t_{PHL}	Any A or B data input	A < B, A > B	1		11					
			2		15					
			3		20					
			4		20					
t_{PLH}	A < B or A = B	A > B	1		7				11	ns
t_{PHL}	A < B or A = B	A > B	1		11				17	ns
t_{PLH}	A = B	A = B	2	13			20	ns		
t_{PHL}	A = B	A = B	2	11			17	ns		
t_{PLH}	A > B or A = B	A < B	1	7			11	ns		
t_{PHL}	A > B or A = B	A < B	1	11			17	ns		

¹ t_{PLH} = propagation delay time, low-to-high-level output

t_{PHL} = propagation delay time, high-to-low-level output

NOTE 5: Load circuits and voltage waveforms are shown in Section 1.

SN5485, SN54LS85, SN54S85
SN7485, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS
SDLS123 - MARCH 1974 - REVISED MARCH 1985

recommended operating conditions

	SN54LS85			SN74LS85			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				-400			μ A
Low-level output current, I_{OL}				4			8 mA
Operating free-air temperature, T_A	-55	125	0	70			$^{\circ}$ C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS [†]	SN54LS85			SN74LS85			UNIT
		MIN	TYP [‡]	MAX	MIN	TYP [‡]	MAX	
V_{IH} High-level input voltage		2			2			V
V_{IL} Low-level input voltage		0.7			0.7			V
V_{IK} Input clamp voltage	$V_{CC} = \text{MIN}$, $I_I = -18 \text{ mA}$	-1.5			-1.5			V
V_{OH} High-level output voltage	$V_{CC} = \text{MIN}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = V_{IL \text{ max}}$, $I_{OH} = -400 \mu\text{A}$	2.5	3.4		2.7	3.4		V
V_{OL} Low-level output voltage	$V_{CC} = \text{MIN}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = V_{IL \text{ max}}$, $I_{OL} = 4 \text{ mA}$	0.25	0.4		0.25	0.4		V
I_I Input current at maximum input voltage	A < B, A > B inputs				0.1			mA
	all other inputs				0.3			
I_{IH} High-level input current	A < B, A > B inputs				20			μ A
	all other inputs				60			
I_{IL} Low-level input current	A < B, A > B inputs				-0.4			mA
	all other inputs				-1.2			
I_{OS} Short-circuit output current [§]	$V_{CC} = \text{MAX}$	-20	-100		-20	-100		mA
I_{CC} Supply current	$V_{CC} = \text{MAX}$, See Note 4	10.4			20			mA

[†]For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
[‡]All typical values are at $V_{CC} = 5 \text{ V}$, $T_A = 25^{\circ}\text{C}$.
[§]Not more than one output should be shorted at a time, and duration of the short circuit should not exceed one second.
NOTE 4: I_{CC} is measured with outputs open, A = B grounded, and all other inputs at 4.5 V.

switching characteristics, $V_{CC} = 5 \text{ V}$, $T_A = 25^{\circ}\text{C}$

PARAMETER [†]	FROM INPUT	TO OUTPUT	NUMBER OF GATE LEVELS	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Any A or B data input	A < B, A > B	1	$C_L = 15 \text{ pF}$, $R_L = 2 \text{ k}\Omega$, See Note 5	14			ns
			2		19			
			3		24	36		
			4		27	45		
t_{PHL}	Any A or B data input	A < B, A > B	1		11			ns
			2		15			
			3		20	30		
			4		23	45		
t_{PLH}	A < B or A = B	A > B	1		14	22		ns
t_{PHL}	A < B or A = B	A > B	1		11	17		ns
t_{PLH}	A = B	A = B	2		13	20		ns
t_{PHL}	A = B	A = B	2		13	26		ns
t_{PLH}	A > B or A = B	A < B	1	14	22		ns	
t_{PHL}	A > B or A = B	A < B	1	11	17		ns	

[†] t_{PLH} = propagation delay time, low-to-high-level output
 t_{PHL} = propagation delay time, high-to-low-level output
NOTE 5: Load circuits and voltage waveforms are shown in Section 1.



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

A.A. 2008-09
1° trimestre

Dott. M. Andreotti

SN5485, SN54LS85, SN54S85
SN7485, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS
SDLS123 - MARCH 1974 - REVISED MARCH 1985

recommended operating conditions

	SN54S85			SN74S85			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				-1			-1 mA
Low-level output current, I_{OL}				20			20 mA
Operating free-air temperature, T_A	-55	125	0	70			$^{\circ}$ C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS [†]	MIN	TYP [‡]	MAX	UNIT	
						V_{IH} High-level input voltage
V_{IL} Low-level input voltage		0.8			V	
V_{IK} Input clamp voltage	$V_{CC} = \text{MIN}$, $I_I = -18 \text{ mA}$	-1.2			V	
V_{OH} High-level output voltage	$V_{CC} = \text{MIN}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = 0.8 \text{ V}$, $I_{OH} = -1 \text{ mA}$	SN54S85	2.5	3.4	V	
		SN74S85	2.7	3.4		
V_{OL} Low-level output voltage	$V_{CC} = \text{MIN}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = 0.8 \text{ V}$, $I_{OL} = 20 \text{ mA}$	0.5			V	
I_I Input current at maximum input voltage	$V_{CC} = \text{MAX}$, $V_I = 5.5 \text{ V}$	1			mA	
I_{IH} High-level input current	A < B, A > B inputs all other inputs	$V_{CC} = \text{MAX}$, $V_I = 2.7 \text{ V}$			50	μ A
					150	
I_{IL} Low-level input current	A < B, A > B inputs all other inputs	$V_{CC} = \text{MAX}$, $V_I = 0.5 \text{ V}$			-2	mA
					-6	
I_{CS} Short-circuit output current [§]	$V_{CC} = \text{MAX}$	-40		-100	mA	
I_{CC} Supply current	$V_{CC} = \text{MAX}$, See Note 4	73			115	mA
		SN54S85W	110			

[†]For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
[‡]All typical values are at $V_{CC} = 5 \text{ V}$, $T_A = 25^{\circ}\text{C}$.
[§]Not more than one output should be shorted at a time, and duration of the short circuit should not exceed one second.
NOTE 4: I_{CC} is measured with outputs open, A = B grounded, and all other inputs at 4.5 V.

switching characteristics, $V_{CC} = 5 \text{ V}$, $T_A = 25^{\circ}\text{C}$

PARAMETER [†]	FROM INPUT	TO OUTPUT	NUMBER OF GATE LEVELS	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Any A or B data input	A < B, A > B	1	$C_L = 15 \text{ pF}$, $R_L = 280 \Omega$, See Note 5	5			ns
			2		7.5			
			3		10.5	16		
			4		12	18		
t_{PHL}	Any A or B data input	A < B, A > B	1		5.5			ns
			2		7			
			3		11	16.5		
			4		11	16.5		
t_{PLH}	A < B or A = B	A > B	1		5	7.5		ns
t_{PHL}	A < B or A = B	A > B	1		5.5	8.5		ns
t_{PLH}	A = B	A = B	2		7	10.5		ns
t_{PHL}	A = B	A = B	2		6	7.5		ns
t_{PLH}	A > B or A = B	A < B	1	5	7.5		ns	
t_{PHL}	A > B or A = B	A < B	1	5.5	8.5		ns	

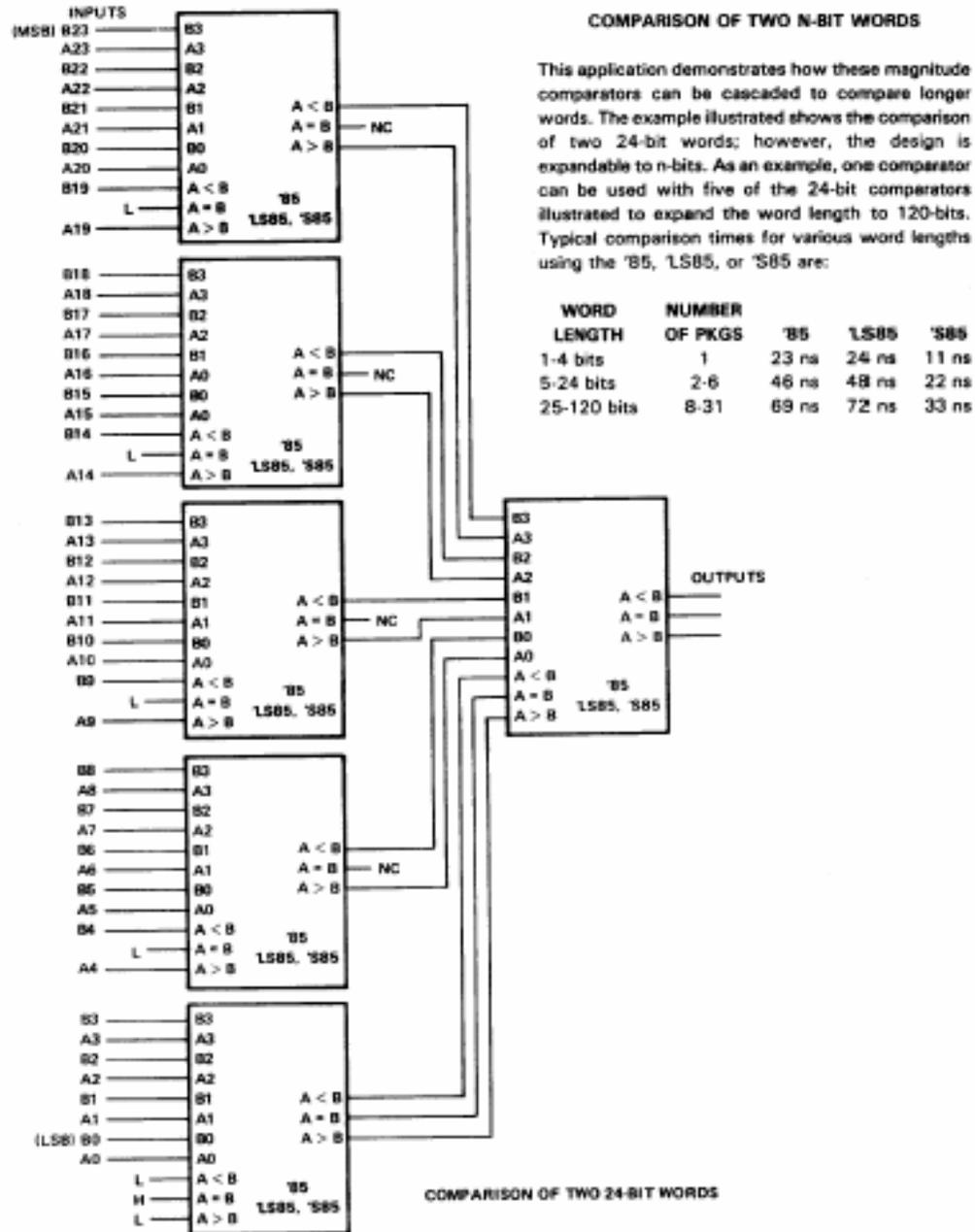
[†] t_{PLH} = propagation delay time, low-to-high-level output
 t_{PHL} = propagation delay time, high-to-low-level output
NOTE 5: Load circuits and voltage waveforms are shown in Section 1.



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

11 /

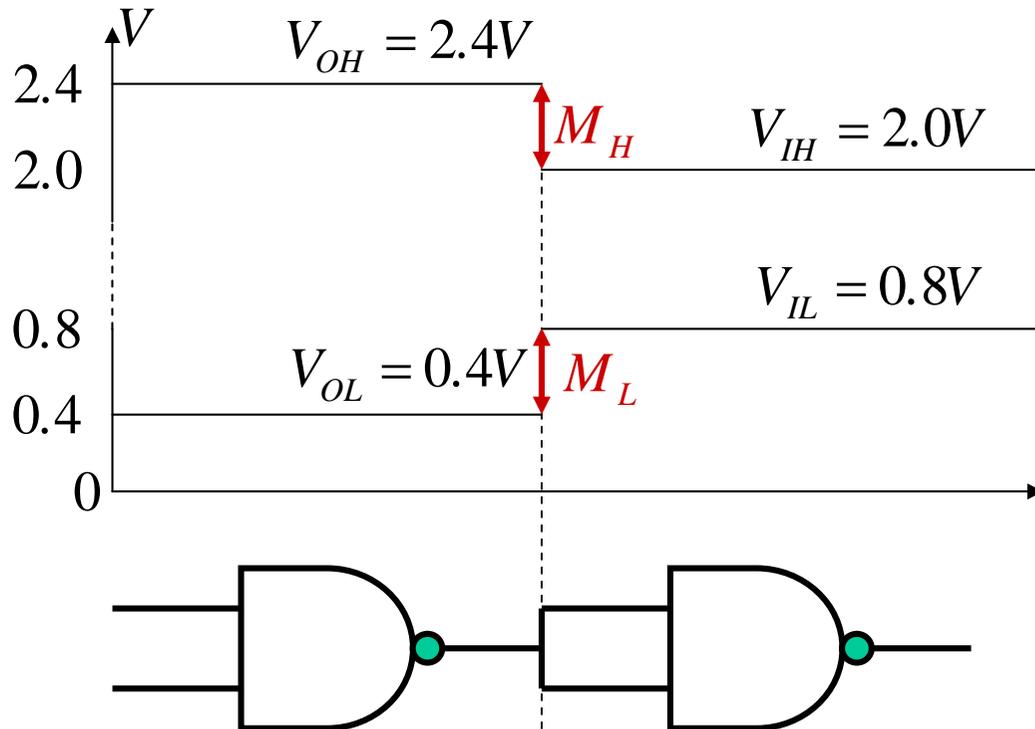
TYPICAL APPLICATION DATA



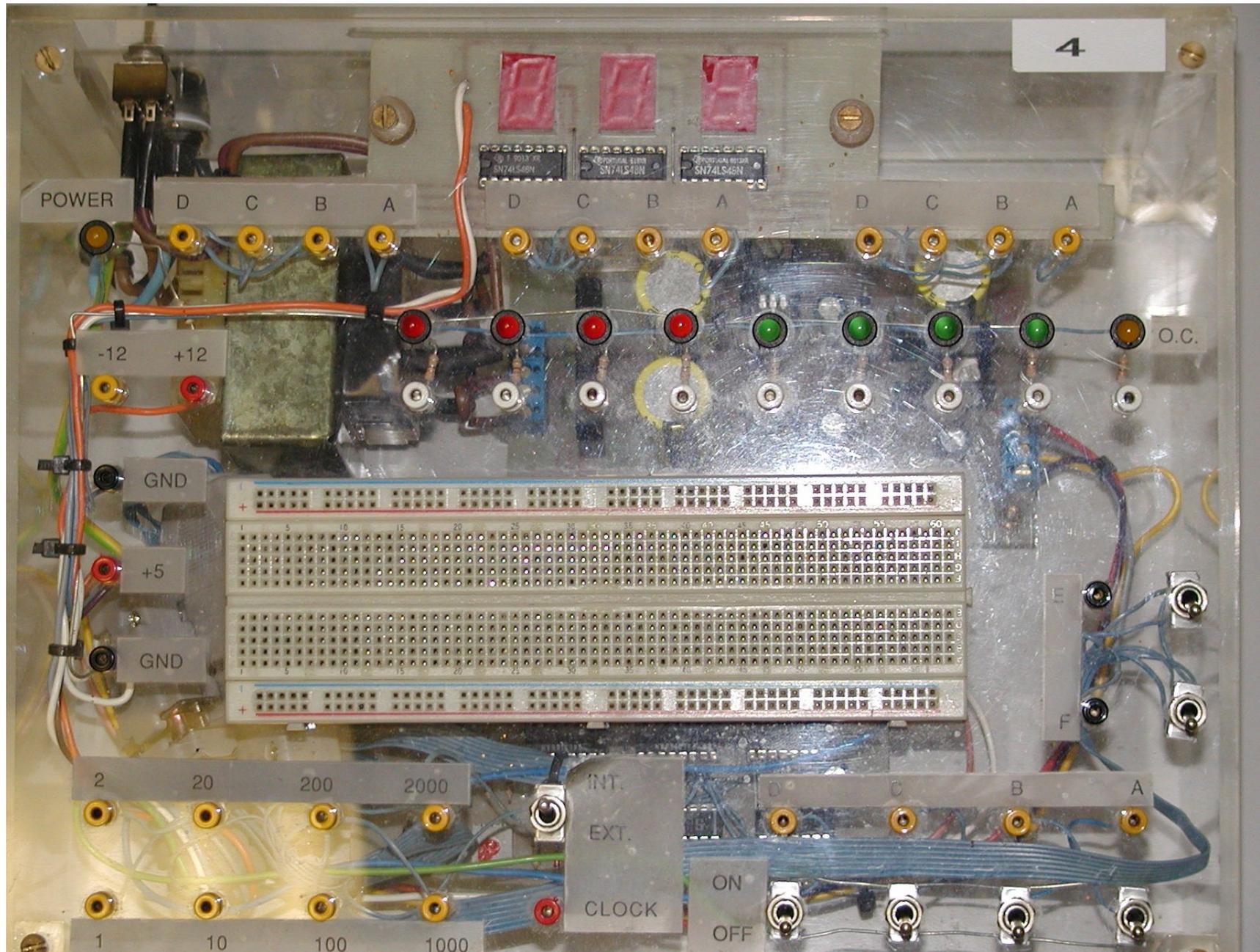
Noi useremo, in generale, integrati (C.I.) della famiglia **TTL (STANDARD?)**

Livelli di ingresso/uscita:

- H: +5 Volts
- L: 0 Volts



1. V_{OH} =minimo valore dello stato alto garantito in uscita
2. V_{OL} =massimo valore dello stato basso garantito in uscita
3. V_{IH} =minimo valore dello stato alto richiesto in ingresso
4. V_{IL} =massimo valore dello stato basso richiesto in ingresso
5. M_H =margine di rumore nello stato basso
6. M_L =margine di rumore nello stato alto



Uso del “laboratorio logico”

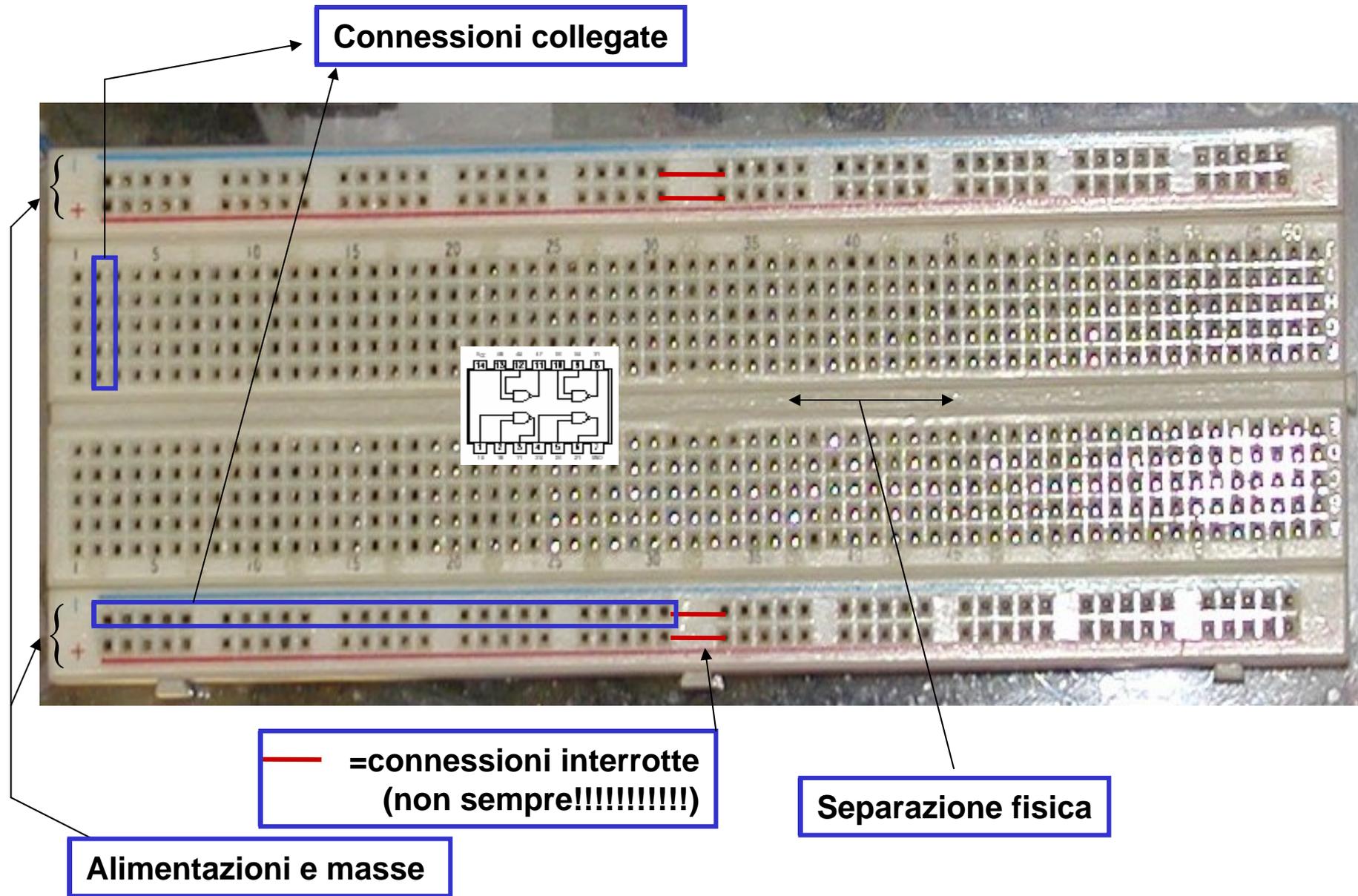
1. V_A = tensione riferita a massa
2. $V_B - V_A$ = tensione “fuori massa”

Le basette (“laboratorio logico”) sono dotate di:

- Breadboard
- Alimentazioni: +5V, +12V, -12V
- Clock: 1KHz, (2KHz), 10KHz, (20KHz), 100KHz, (200KHz), 1000KHz, (2000KHz)
- Commutatori (switch)
- Commutatori anti-rimbalzo
- Led (active high)
- Led o.c. (active low)
- Display a 7 segmenti

Importante:

Come si usa la breadboard?



Prime esperienze con le porte universali

1. Verifica delle tavole della verità:
 - Porte logiche fondamentali
 - NAND a due ingressi
 - NOR a due ingressi
2. Impiego delle porte NAND E NOR come:
 - Inverter
 - AND,OR con solo porte NAND,NOR
 - Enable, Inhibit
 - Mux, Demux
 - EX-OR con quattro NAND
 - EX-NOR con quattro NOR
 - Comparatore digitale a un bit
 - True/complement
3. Uso di integrati più complessi
 - Complementazione di un numero binario a 4 bit con un 7486 (EXOR come True/Complement)
 - Tavola della verità di una decodifica BCD-7segmenti

Esperienza D-1

- a) Per effettuare operazioni logiche elementari
- b) Per la verifica del **Teorema di De Morgan**
- c) Per controllare il flusso di segnali digitali

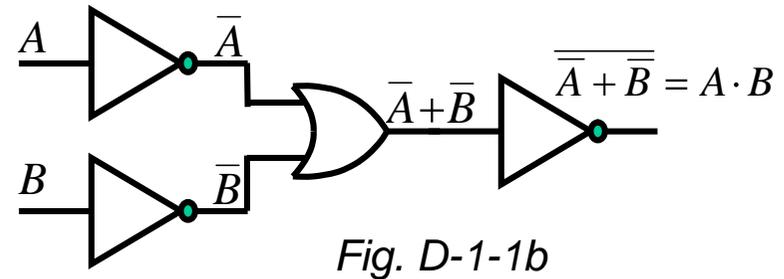
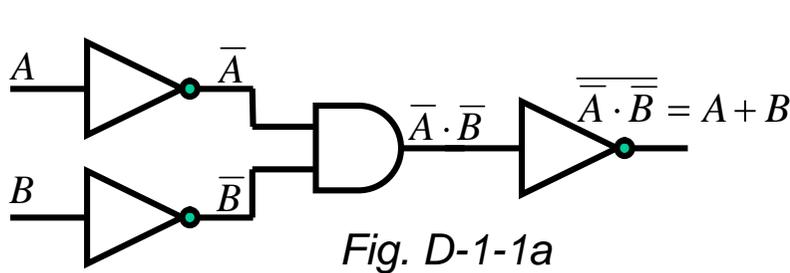
Preliminari alle singole prove:

- a) Comprende le prove
 1. Verifica delle tavole della verità di NAND (NOR) a due ingressi
 2. Impiego di NAND(NOR) come inverter
- b) Comprende le prove
 3. Uso di porte NAND (NOR) per realizzare AND(OR)
 4. Uso di porte NAND (NOR) per realizzare una porta OR(AND)
- c) Uso di gates per operazioni di:
 1. Enable, Inhibit

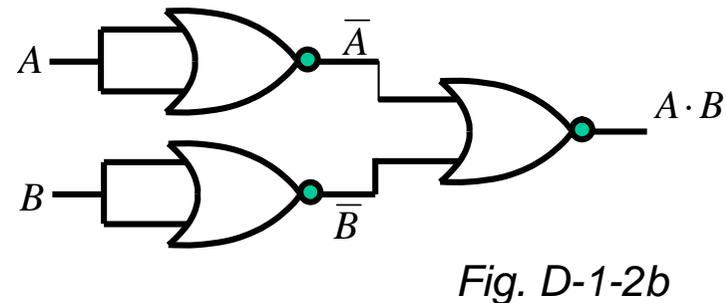
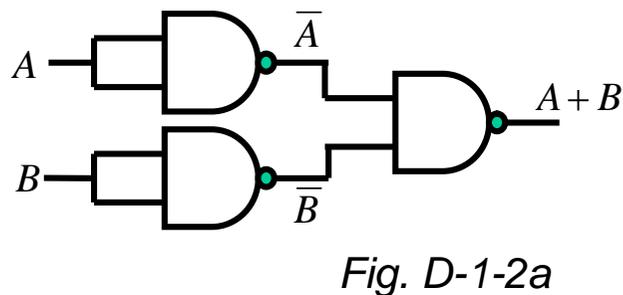
- a)è banale
- b) Avevamo già visto che dal Teorema di De Morgan:

$$\overline{\overline{A} \cdot \overline{B}} = A + B \qquad \overline{\overline{A} + \overline{B}} = A \cdot B$$

E' sufficiente realizzare il circuito corrispondente al primo membro e verificare che la sua tavola della verità sia uguale a quella del secondo membro.

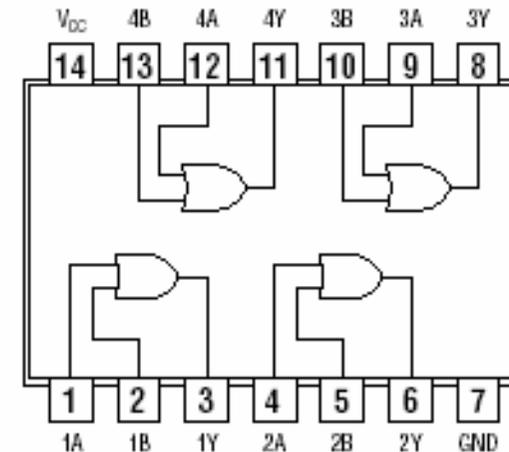


Modificare i circuiti in modo che contengano solo porte NAND o solo porte NOR



Materiale occorrente:

1. Laboratorio logico
2. IC: 7400, 7402, 7404, 7408, 7432
3. Manuale IC

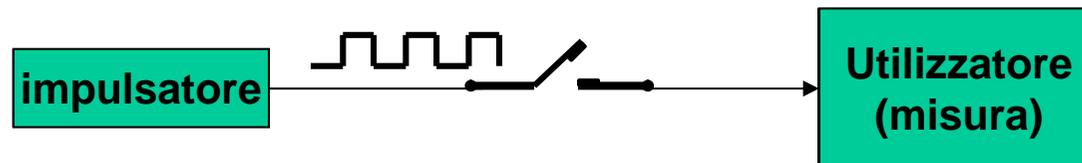


Come si procede:

1. Verificare le tabelle della verità di **TUTTE** le porte
2. Montare gli schemi di cui alle figure:
 - i. D-1-1a e D-1-1b
 - ii. D-1-2a e D-1-2b

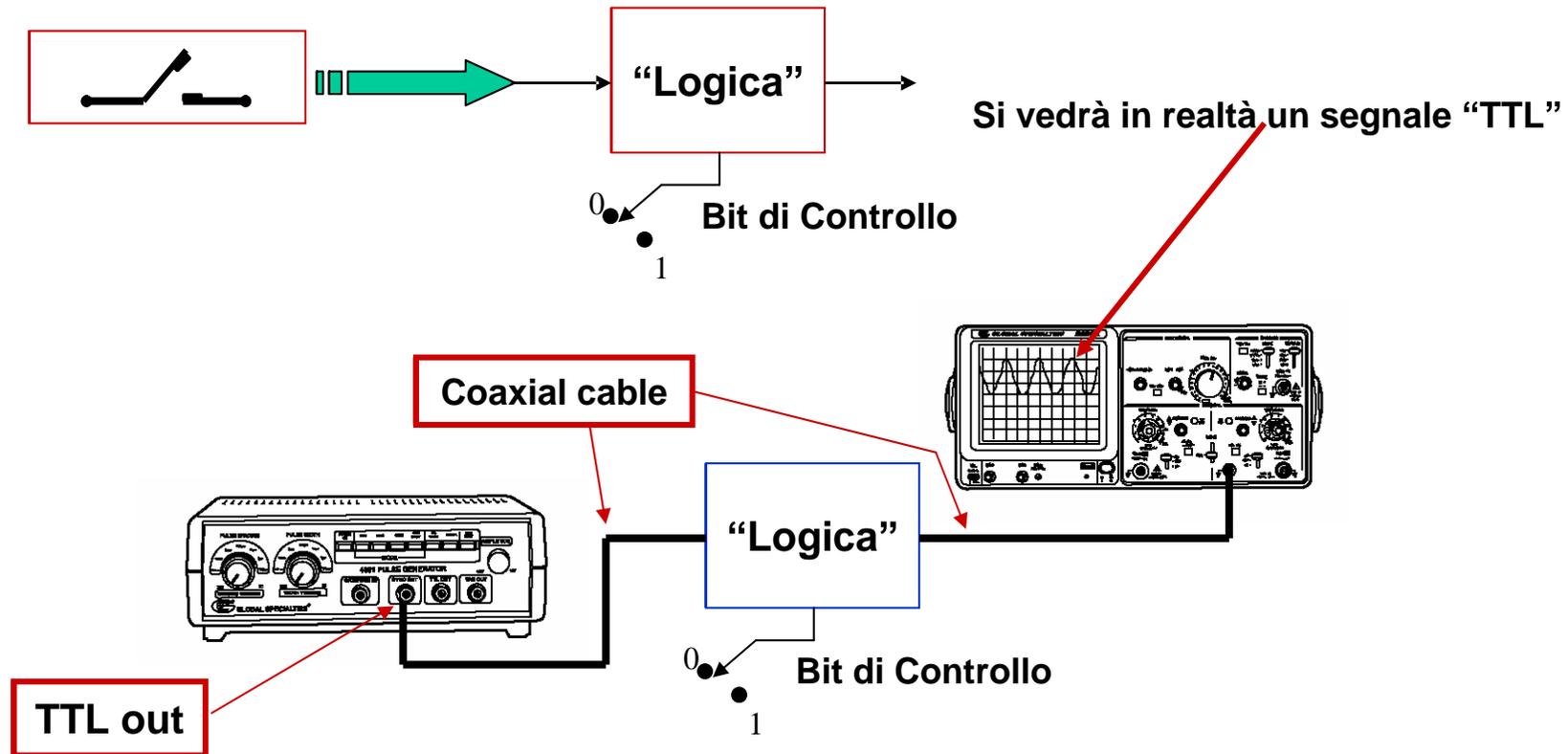
Usare il laboratorio logico che fornisce le alimentazioni per gli IC i segnali di ingresso (switch) ed i rivelatori di stato di uscita (led)

c) Operazioni di: Enable, Inhibit, True complement, Mux, Demux

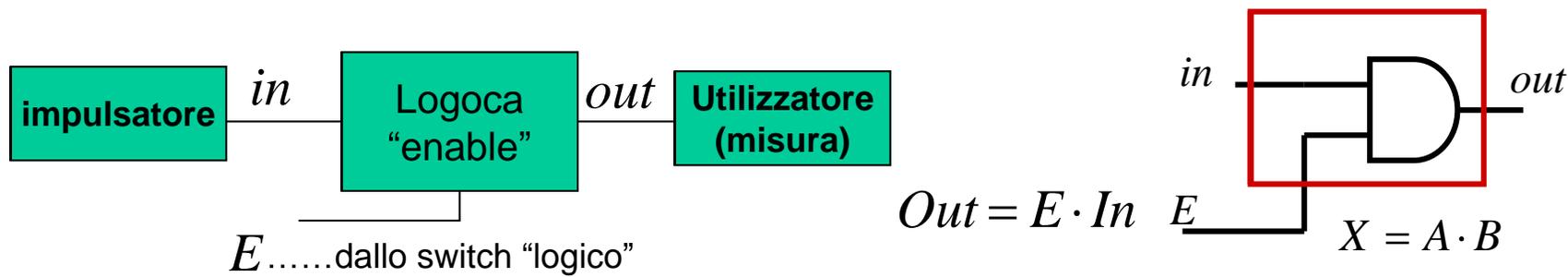


Usare preliminarmente gli switch manuali ed i led dopo avere costruito il blocco "logica" sulla breadboard, poi l' FG ed il CRO (vedi il seguito)

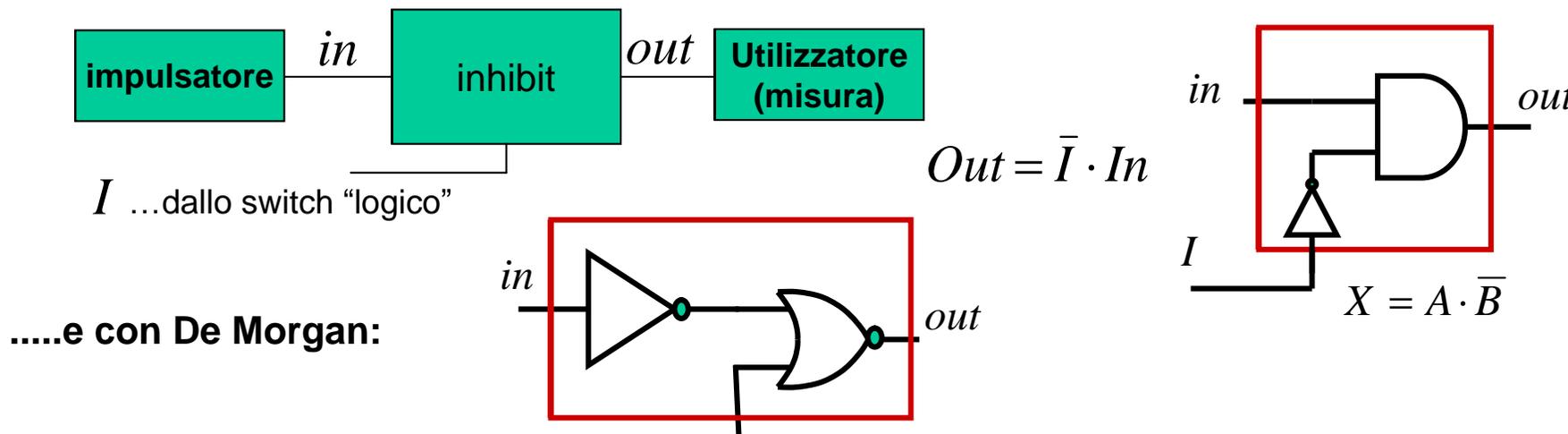
Cerchiamo di sostituire all' interruttore meccanico un meccanismo più sofisticato e pratico:



Enable gate (strobe):



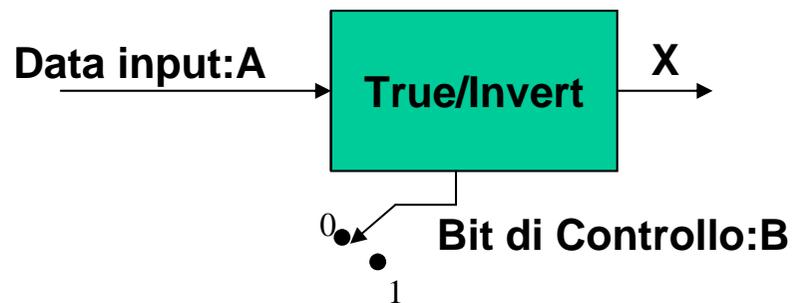
Inhibit gate:



Esperienza D-2

Comprende le prove:

- Realizzazione di un True/Complement (True/Invert)
- Realizzazione di un XOR(XNOR) con sole NAND e NOR
- Funzione di eguaglianza
- Comparatore digitale a 1 bit
- Mux
- Demux



$$X = \begin{cases} A & \text{se } B = 0 \\ \bar{A} & \text{se } B = 1 \end{cases}$$

B	A	X
0	0	0
0	1	1
1	0	1
1	1	0

È un or esclusivo!

EXOR

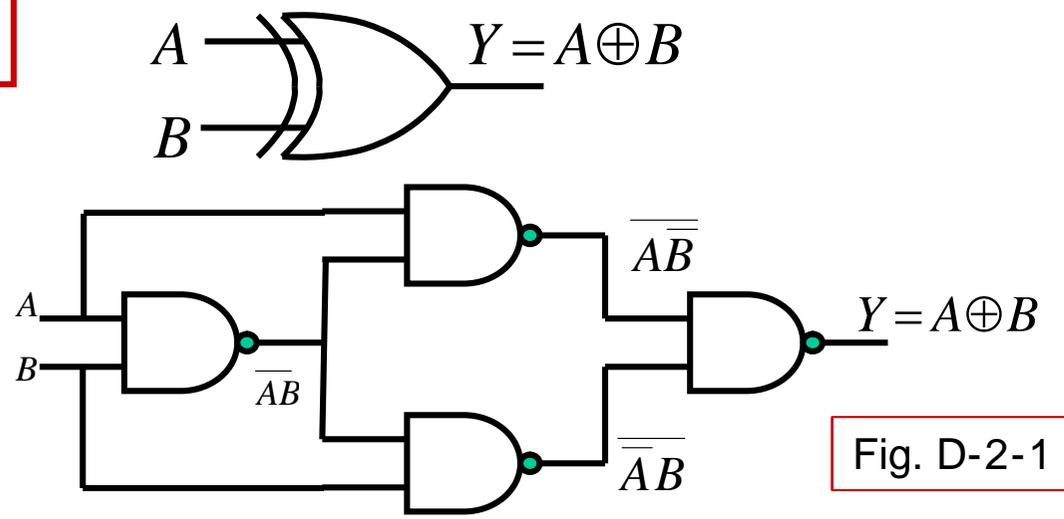


Fig. D-2-1

B	A	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

EXNOR

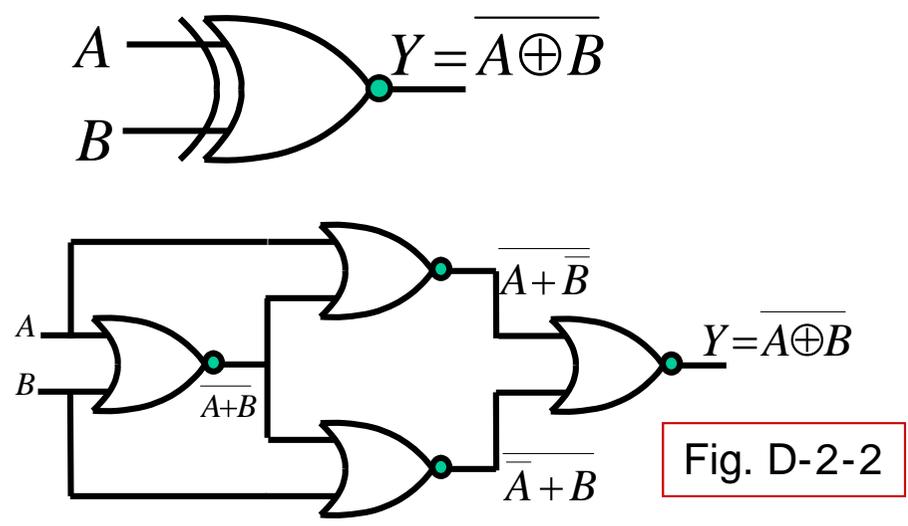


Fig. D-2-2

B	A	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Realizzazione Pratica:

1. usare un I.C. 7400 (quadrupla NAND a due ingressi) per realizzare lo schema (fig. D-2-1) e ricavarne la tavola della verità;

B	A	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Se si usa una delle variabili (es. B) come bit di controllo:

$$B = 0 \Rightarrow Out = A$$

$$B = 1 \Rightarrow Out = \overline{A}$$

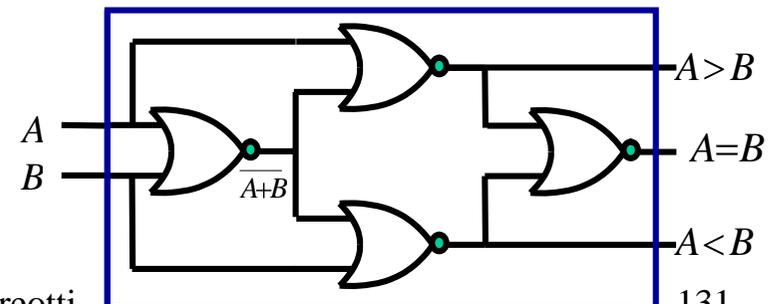
True/complement

2. usare un I.C. 7402 (quadrupla NOR a due ingressi) per realizzare lo schema (fig. D-2-2) e ricavarne la tavola della verità;

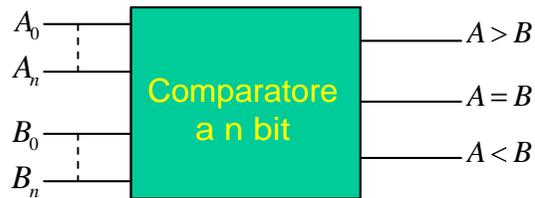
B	A	$A \oplus B$
0	0	1
0	1	0
1	0	0
1	1	1

- E' vera quando sono diversi gli ingressi
- la porta di uscita (NOR nella fig.D-2-2) è falsa quando o l'uno o l'altro dei due ingressi sono veri-> i due ingressi devono rappresentare $A > B$ e $A < B$:

Blocco comparatore
A 1 bit



Comparatore



A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

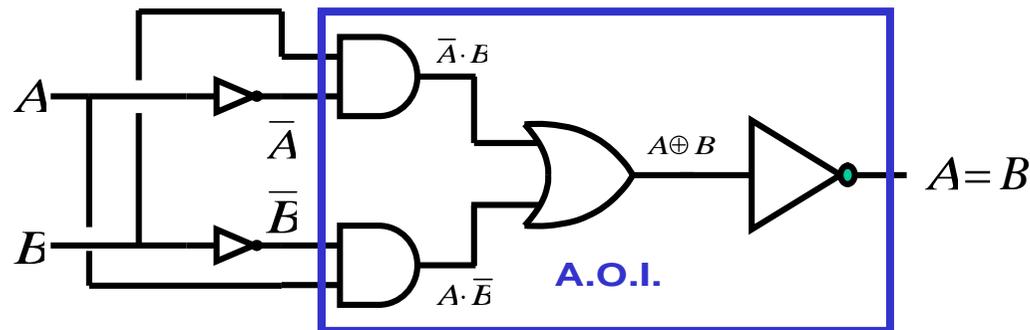
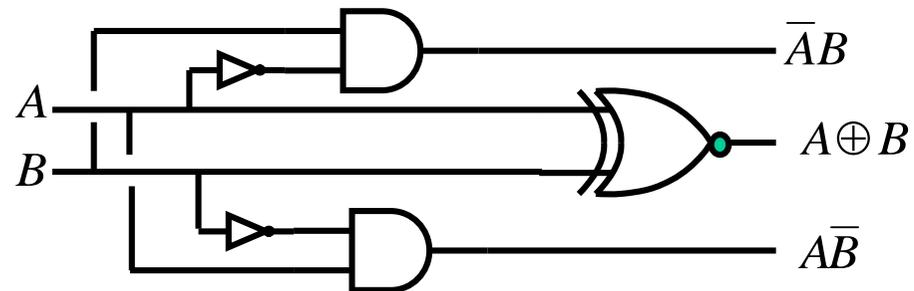
Partiamo dal caso più semplice a 1 bit:

COME PREVEDIBILE ABBIAMO TRE FUNZIONI LOGICHE:

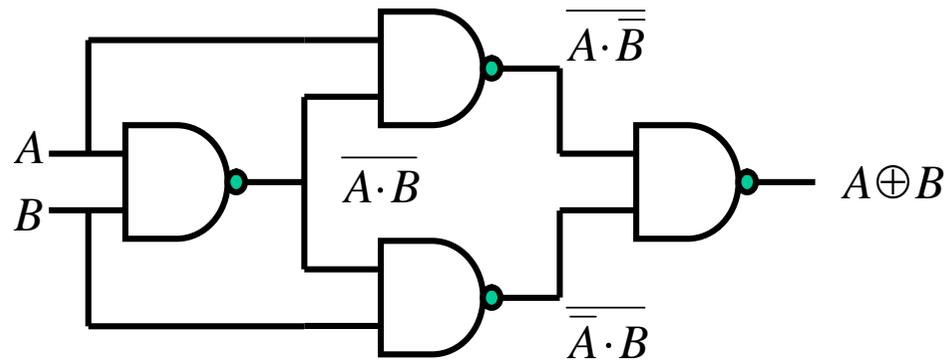
$$A < B = \bar{A} \cdot B$$

$$A > B = A \cdot \bar{B}$$

$$A = B = \bar{A} \cdot \bar{B} + AB$$

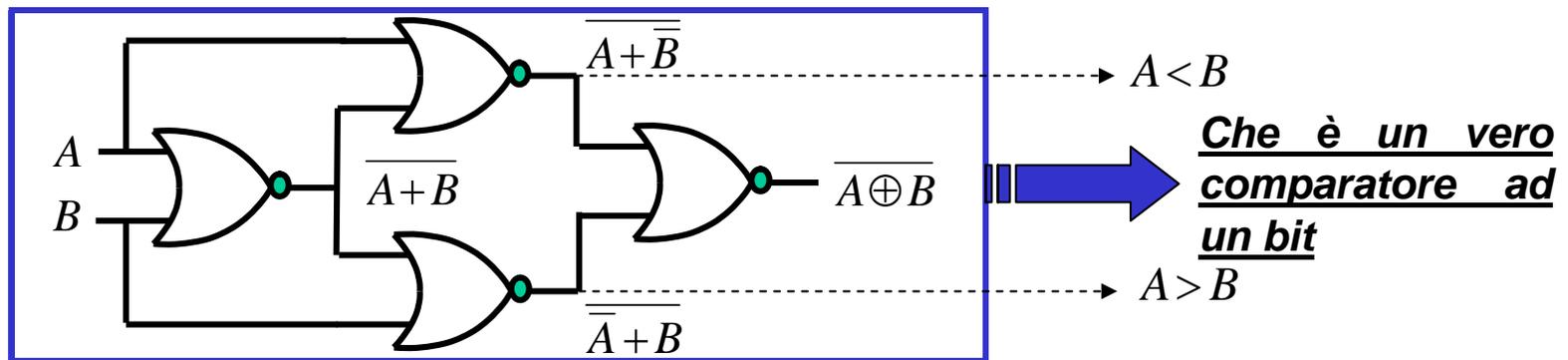


Avevamo già visto come fare un **EXOR** con 4 porte **NAND**

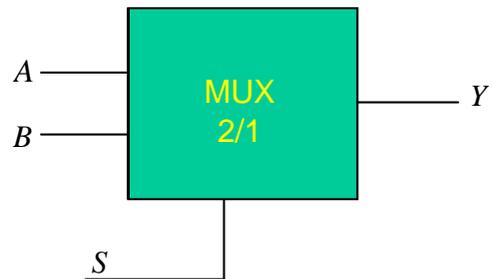


Ma è anche più immediato se si realizza mediante **NOR** infatti per **De Morgan**:

$$\overline{A+B} = \bar{A} \cdot \bar{B} \quad \overline{\bar{A}+B} = A \cdot \bar{B}$$



Multiplexers e



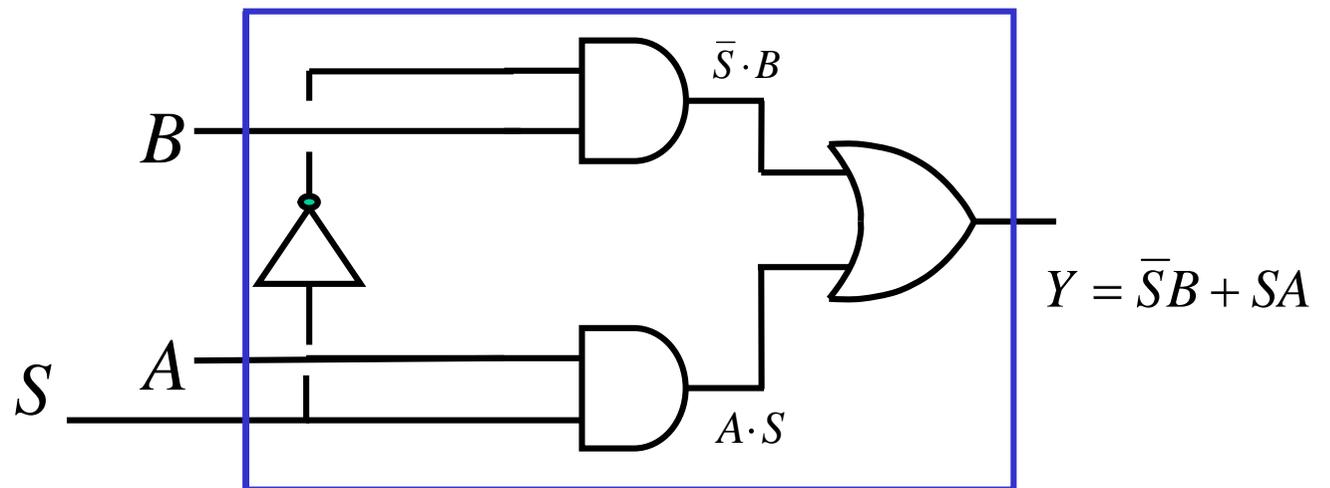
S	A	B	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

S	A	B	Y
0	x	x	B
1	x	x	A

COME PREVEDIBILE ABBIAMO UNA SOLA FUNZIONE LOGICA:

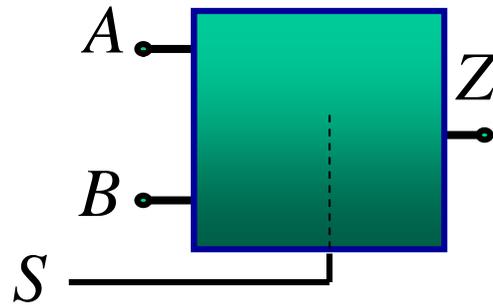
$$Y = \bar{A}B\bar{S} + AB\bar{S} + A\bar{B}S + ABS$$

$$Y = \bar{S}B + SA$$



Multiplexer (italiano: smistatore):

esempio: 2 su 1



S	B	A	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

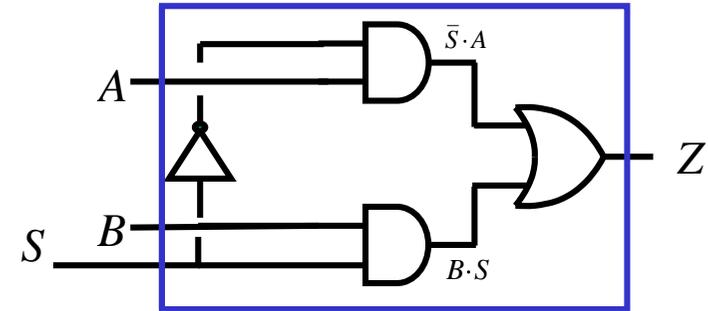
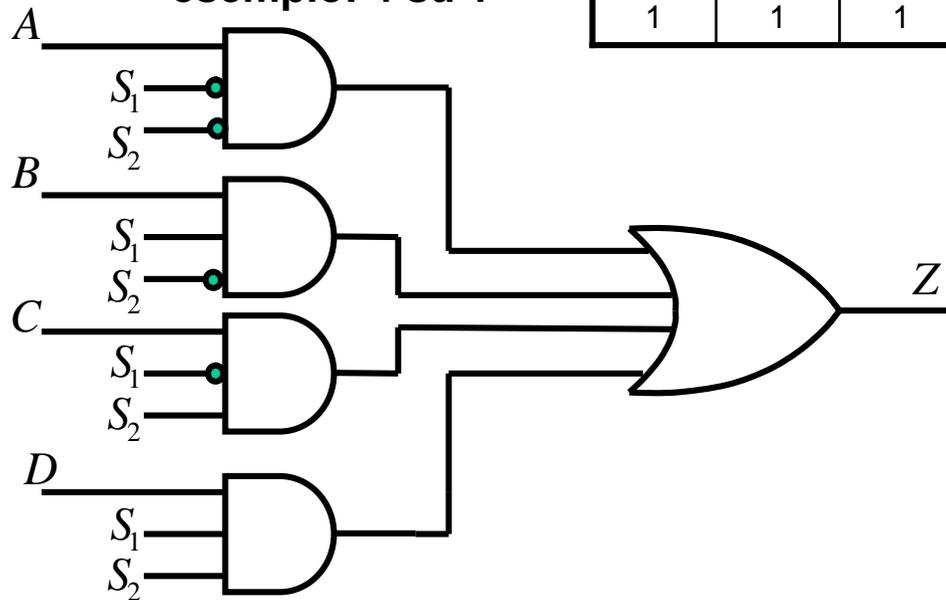


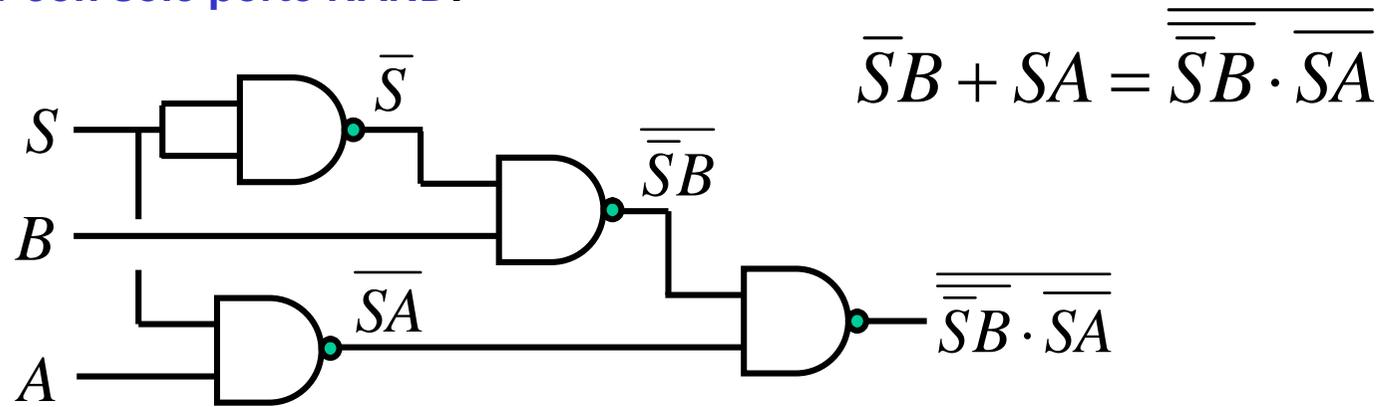
Fig. D-2-3

esempio: 4 su 1



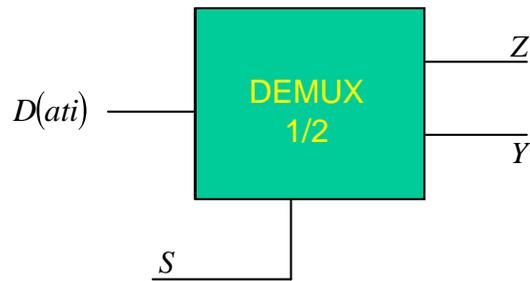
S1	S2	Z
0	0	A
0	1	C
1	0	B
1	1	D

Multiplexer con sole porte NAND:



Multiplexer con sole porte NOR: più di 4 → non è utile adesso

..... Demultiplexers



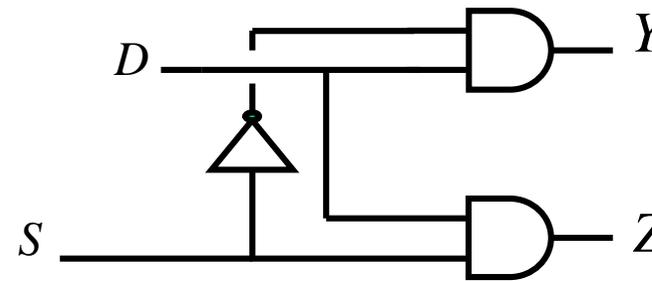
S	D	Z	Y
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

S	D	Z	Y
0	x	0	D
1	x	D	0

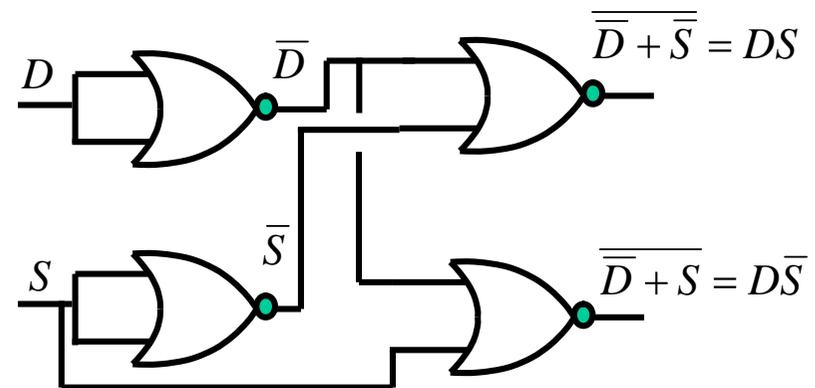
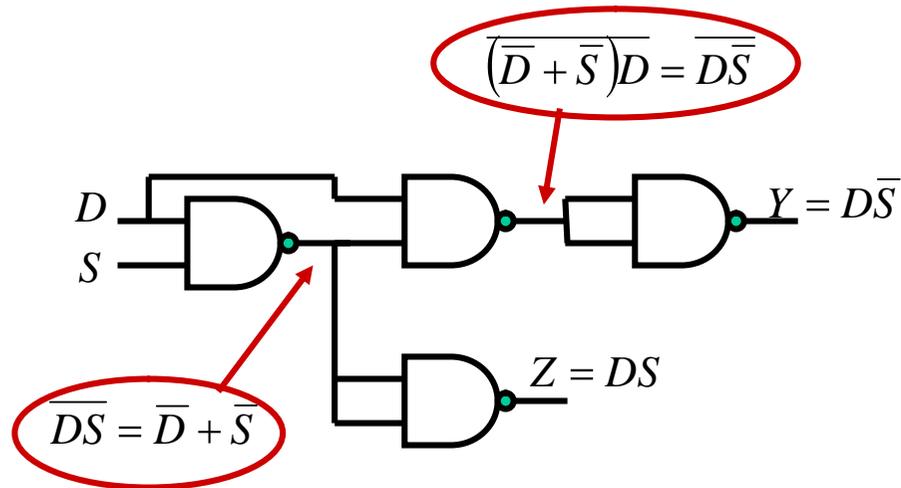
COME PREVEDIBILE ABBIAMO
DUE FUNZIONI LOGICHE:

$$Z = SD$$

$$Y = \bar{S}D$$

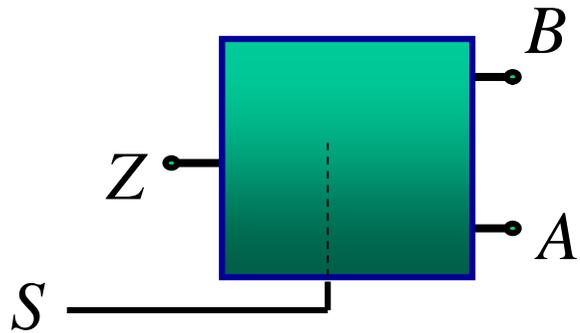


Con le porte universali:



Demultiplexer:

esempio: 1 su 2



S	D	y	Z
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

S	D	y	Z
0	X	D	0
1	X	0	D

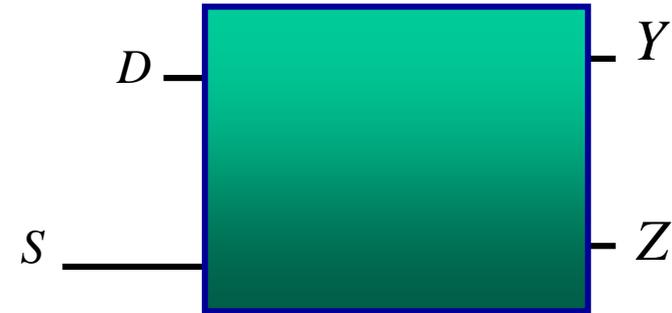
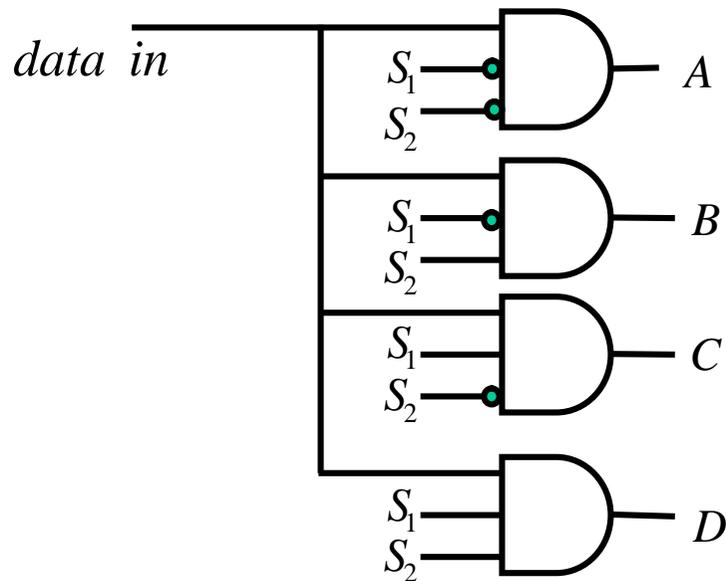


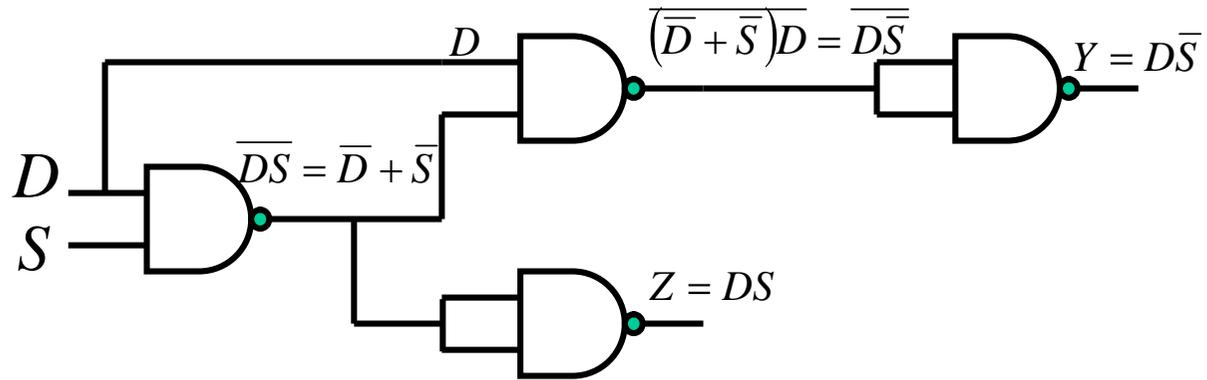
Fig. D-2-4

esempio: 1 su 4

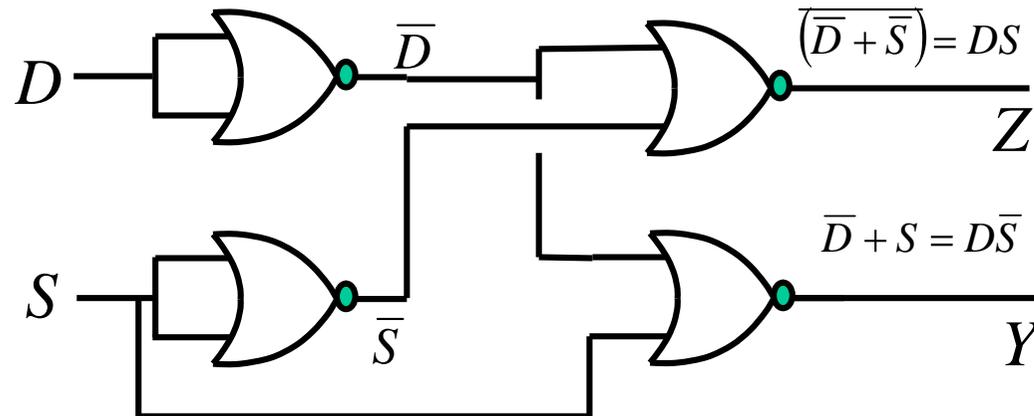


S1	S2	OUT
0	0	A
0	1	B
1	0	C
1	1	D

Demultiplexer con sole porte NAND:



Demultiplexer con sole porte NOR:



Condotta Pratica:

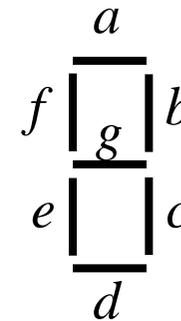
- 1. Assemblare il circuito MUX 2 su 1 (fig D-2-3)**
 - Realizzare il MUX con sole NAND**
- 2. Assemblare il circuito Demux 1 su 2**
 - Realizzarlo anche con sole NAND o NOR**
- 3. Provare ciascuno dei circuiti con gli interruttori manuali**
- 4. Provare ciascun circuito con generatore di Funzioni (F.G.) e oscilloscopio (C.R.O.)**
- 5. Provare anche 74153 (dual quad-in MUX)**
- 6. Provare anche 74154 (4in-16 out)**
- 7. Provare un 7485 (comparatore a 4 bit)**

Esempio pratico di progetto (parziale)

Convertitore BCD-7 segmenti

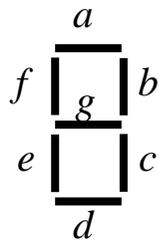
BCD: binary coded decimal

7 segmenti: è un tipo di display



Come si procede:

1. Dobbiamo rappresentare i numeri da 0 a 9:
di quanti bit abbiamo bisogno?
2. Quale (i) è (sono) la (le) funzione (i) logica?
3. Come si costruisce (ono)?



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

'46A, '47A, 'LS47 FUNCTION TABLE (T1)

DECIMAL OR FUNCTION	INPUTS						$\overline{\text{BI}}/\overline{\text{RBO}}^\dagger$	OUTPUTS							NOTE
	$\overline{\text{LT}}$	$\overline{\text{RBI}}$	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	ON	ON	ON	ON	ON	ON	OFF	1
1	H	X	L	L	L	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	H	X	L	L	H	L	H	ON	ON	OFF	ON	ON	OFF	ON	
3	H	X	L	L	H	H	H	ON	ON	ON	ON	OFF	OFF	ON	
4	H	X	L	H	L	L	H	OFF	ON	ON	OFF	OFF	ON	ON	
5	H	X	L	H	L	H	H	ON	OFF	ON	ON	OFF	ON	ON	
6	H	X	L	H	H	L	H	OFF	OFF	ON	ON	ON	ON	ON	
7	H	X	L	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF	
8	H	X	H	L	L	L	H	ON	ON	ON	ON	ON	ON	ON	
9	H	X	H	L	L	H	H	ON	ON	ON	OFF	OFF	ON	ON	
10	H	X	H	L	H	L	H	OFF	OFF	OFF	ON	ON	OFF	ON	
11	H	X	H	L	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON	
12	H	X	H	H	L	L	H	OFF	ON	OFF	OFF	OFF	ON	ON	
13	H	X	H	H	L	H	H	ON	OFF	OFF	ON	OFF	ON	ON	
14	H	X	H	H	H	L	H	OFF	OFF	OFF	ON	ON	ON	ON	
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
BI	X	X	X	X	X	X	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	H	L	L	L	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	L	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON	4

H = high level, L = low level, X = irrelevant

- NOTES:
1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
 2. When a low logic level is applied directly to the blanking input ($\overline{\text{BI}}$), all segment outputs are off regardless of the level of any other input.
 3. When ripple-blanking input ($\overline{\text{RBI}}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output ($\overline{\text{RBO}}$) goes to a low level (response condition).
 4. When the blanking input/ripple blanking output ($\overline{\text{BI}}/\overline{\text{RBO}}$) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

$\overline{\text{BI}}/\overline{\text{RBO}}$ is wire AND logic serving as blanking input ($\overline{\text{BI}}$) and/or ripple-blanking output ($\overline{\text{RBO}}$).

Consideriamo il segmento e

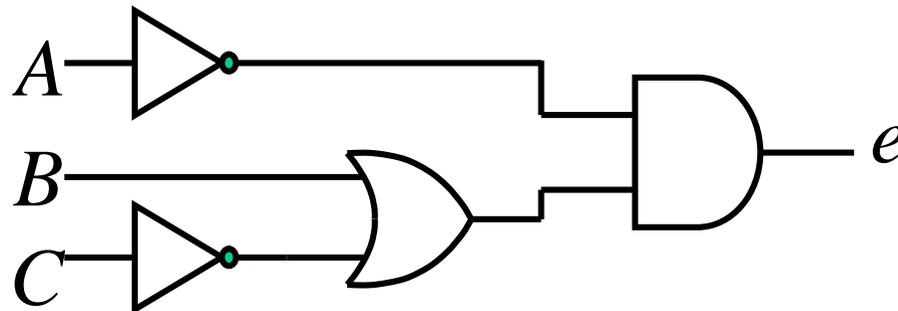
$$e = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D + \overline{A}BCD$$

....e semplifichiamo la funzione!

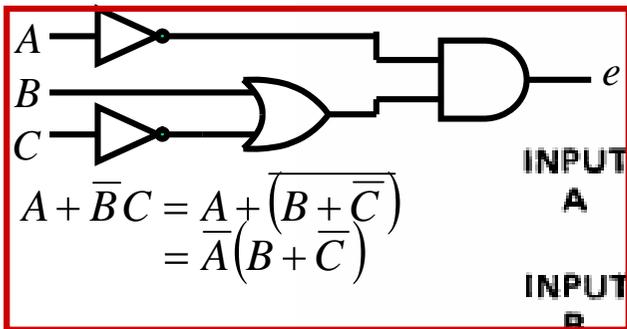
$$\begin{aligned} e &= \overline{A}\overline{C}\overline{D}(B + \overline{B}) + \overline{A}BC(D + \overline{D}) + \overline{A}\overline{C}D(B + \overline{B}) \\ &= \overline{A}\overline{C}\overline{D} + \overline{A}BC + \overline{A}\overline{C}D = \overline{A}\overline{C}(\overline{D} + D) + \overline{A}BC \end{aligned}$$

....poi si ricorre ad un trucco:

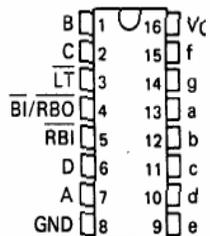
$$\begin{aligned} e &= \overline{A}\overline{C} + \overline{A}BC = \overline{A}\overline{C}(1 + B) + \overline{A}BC \\ &= \overline{A}\overline{C} + \overline{A}\overline{C}B + \overline{A}BC = \overline{A}\overline{C} + \overline{A}B = \overline{A}(\overline{C} + B) \end{aligned}$$



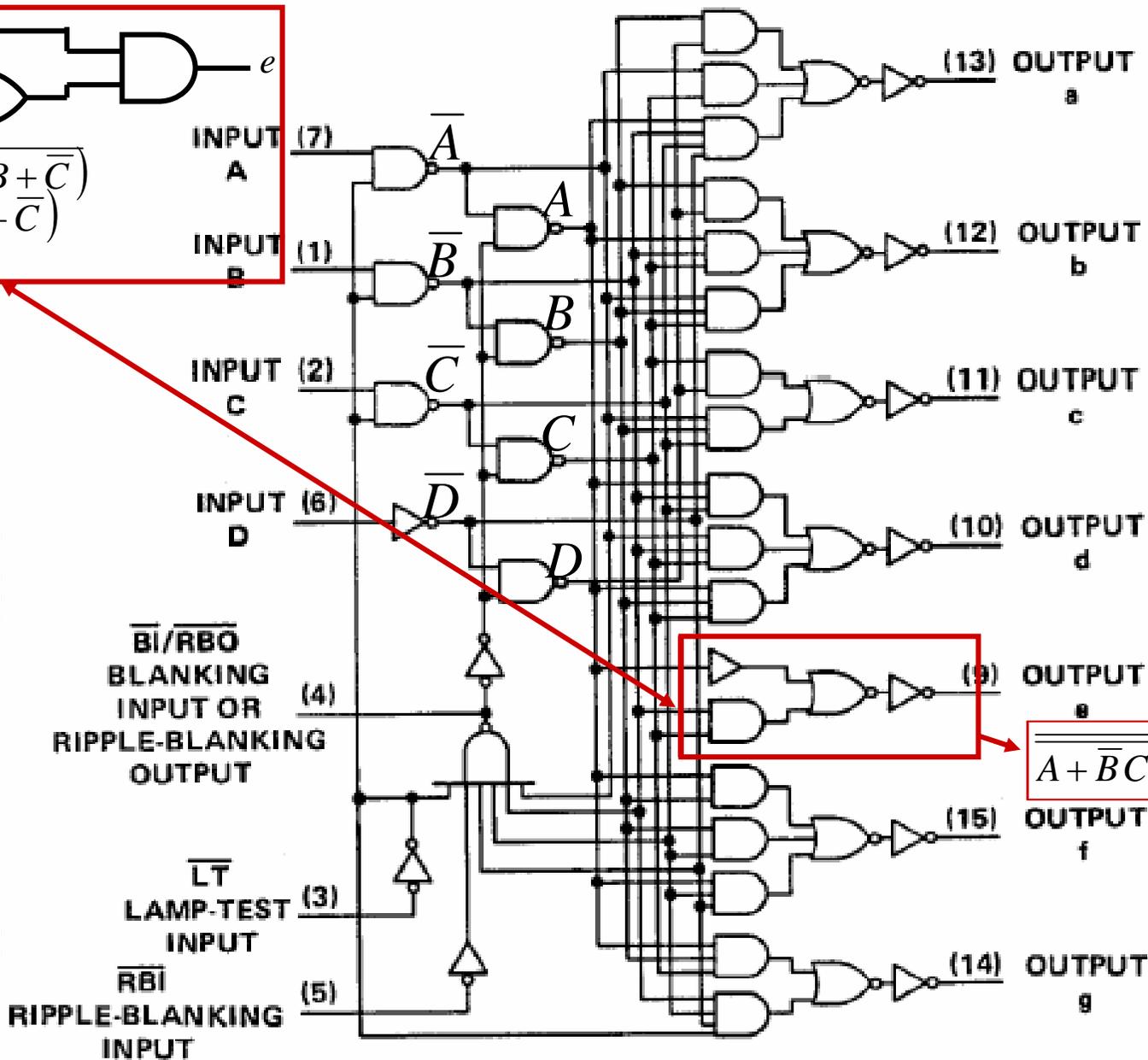
'46A, '47A, 'LS47



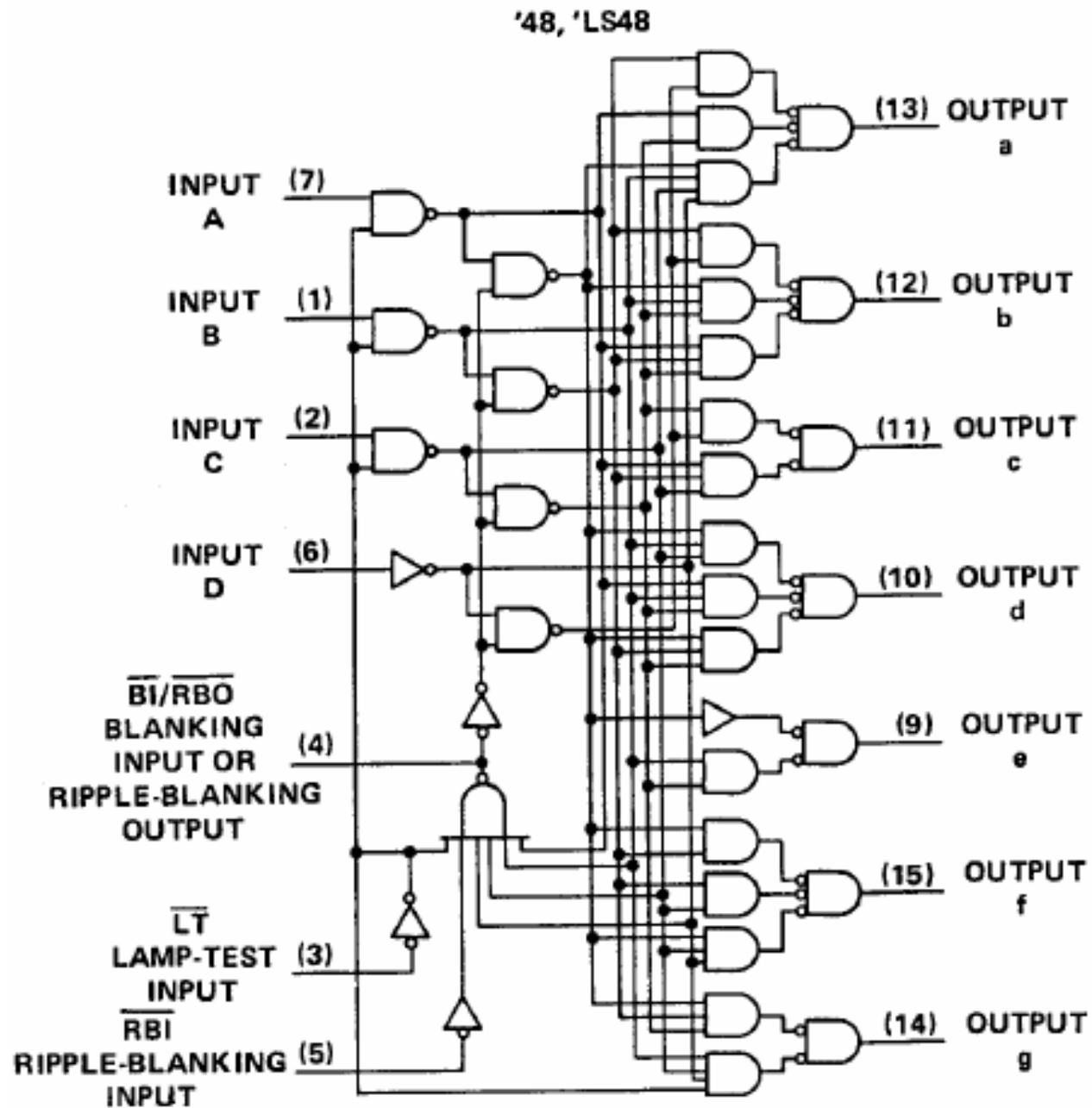
SN5446A, SN5447A, SN54LS4
 SN54LS48 ... J PACK
 SN7446A, SN7447A
 SN7448 ... N PACKA
 SN74LS47, SN74LS48 ... D OF
 (TOP VIEW)



SN54LS49 ... J OR W PA
 SN74LS49 ... D OR N PA
 (TOP VIEW)



$\overline{\overline{A + \overline{B}C}} = A + \overline{B}C$



Riepilogo delle esperienze da svolgere 27/04, 02/05 e 04/04

- Prova delle porte base AND e OR
- Verifica del Teorema di De Morgan
 - ✓ Realizzazione di AND con solo NOR e OR con solo NAND
- Realizzazione di XOR e XNOR con porte universali
 - ✓ uso di XOR come true/invert
 - ✓ uso di XNOR come comparatore
- MUX e DEMUX con diverse porte

lezione I di lab

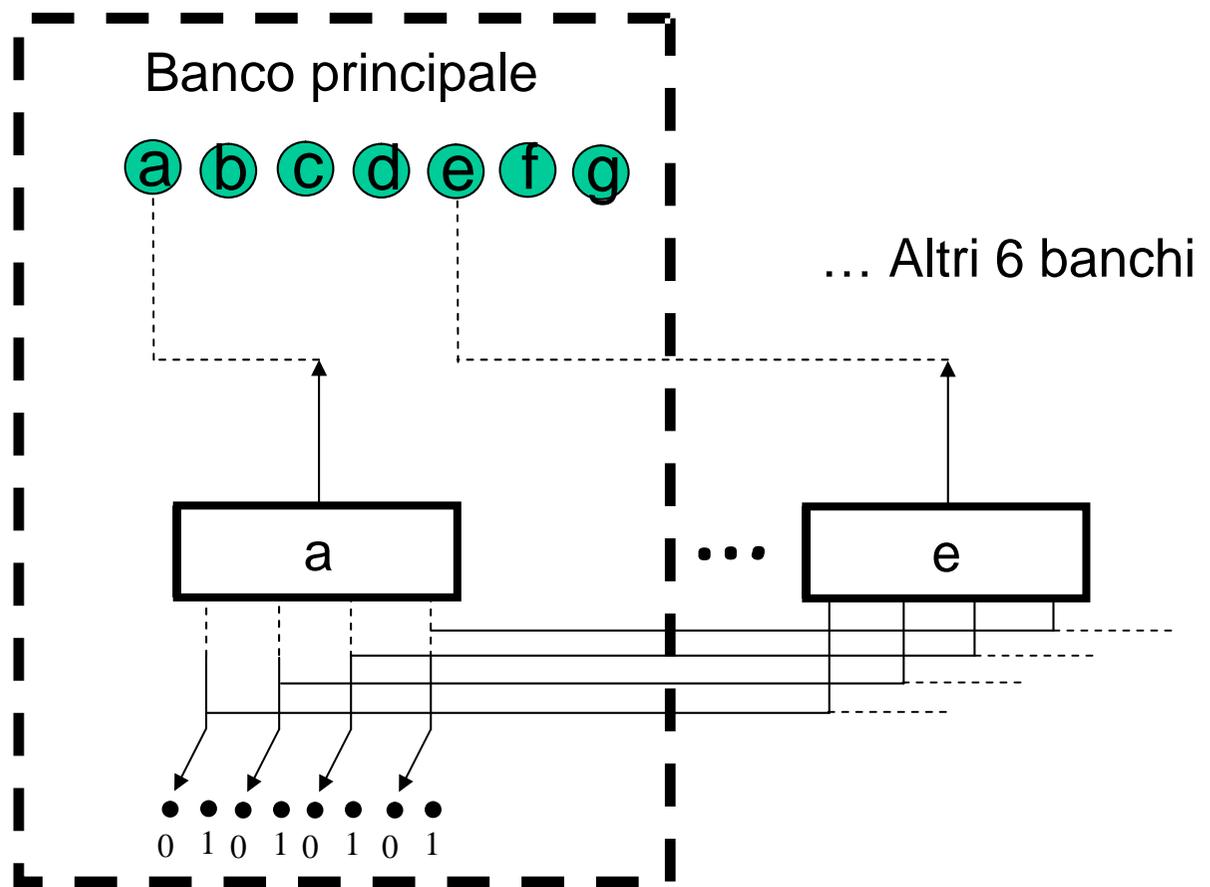
- BCD 7 segmenti
 - ✓ ogni gruppo realizzi il circuito corrispondente ad un singolo segmento:
 - studio della funzione algebrica
 - simulazione del circuito
 - realizzazione pratica

Si collega tutto insieme

lezione II di lab

BCD 7 segmenti

- ✓ ogni gruppo realizza il circuito corrispondente ad un singolo segmento:
 - ✓ studio della funzione algebrica: minterm, semplificazione algebrica con le mappe di Karnaugh, determinazione di eventuali operazioni in comune ai singoli segmenti
 - ✓ simulazione del circuito
 - ✓ realizzazione pratica



ELETTRONICA DEI SISTEMI DIGITALI

Parte II

Corso di Laurea in Informatica/TFI
Anno Accademico 2007-2008

Comparatori a più bit

- Abbiamo già visto il comparatore a un bit.
- Estendiamo il caso a 4 bit.
- Confrontare 4 bit significa:
 1. Confrontare i bit più significativi
 2. Decidere o...
 3. Confrontare i bit di ordine immediatamente successivo
 4. reiterare

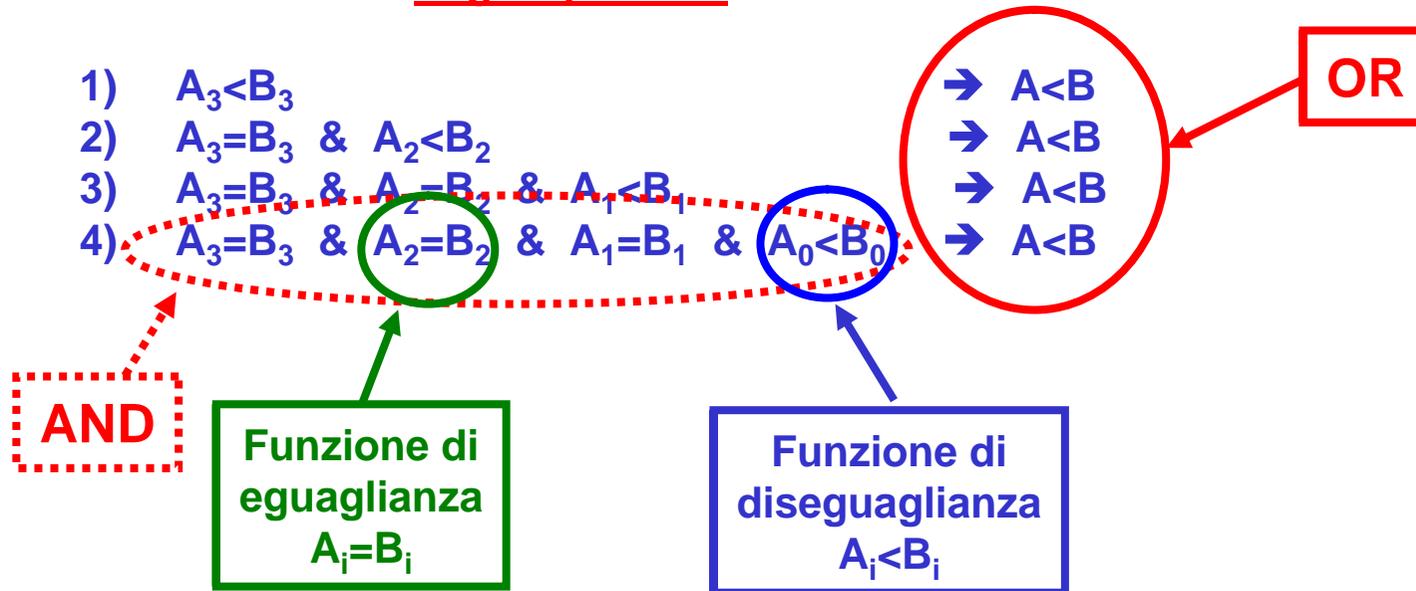
- Confrontiamo 2 numeri:

$$A_3A_2A_1A_0 \quad B_3B_2B_1B_0$$

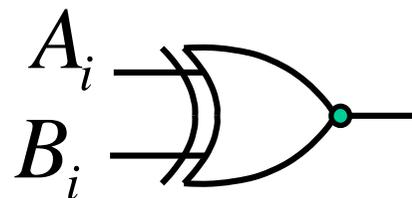
Logica per $A < B$

- | | | | |
|----|--|---------------------|-------------------------|
| 1) | $A_3 < B_3$ | $\rightarrow A < B$ | } OR delle 4 condizioni |
| 2) | $A_3 = B_3 \ \& \ A_2 < B_2$ | $\rightarrow A < B$ | |
| 3) | $A_3 = B_3 \ \& \ A_2 = B_2 \ \& \ A_1 < B_1$ | $\rightarrow A < B$ | |
| 4) | $A_3 = B_3 \ \& \ A_2 = B_2 \ \& \ A_1 = B_1 \ \& \ A_0 < B_0$ | $\rightarrow A < B$ | |

Logica per $A < B$

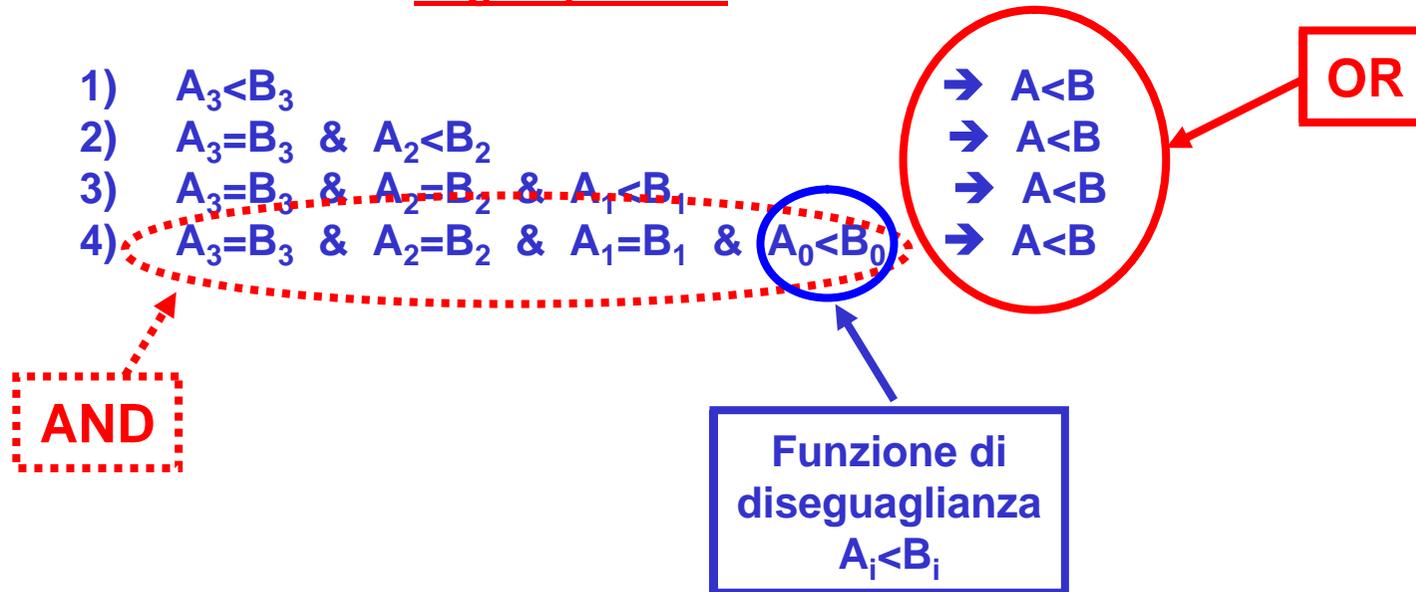


Funzione di eguaglianza (XNOR): E_3, E_2, E_1, E_0

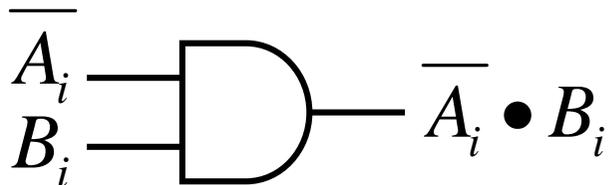

 $E_i = \overline{A_i \oplus B_i} \Leftrightarrow A_i = B_i$

A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Logica per A<B

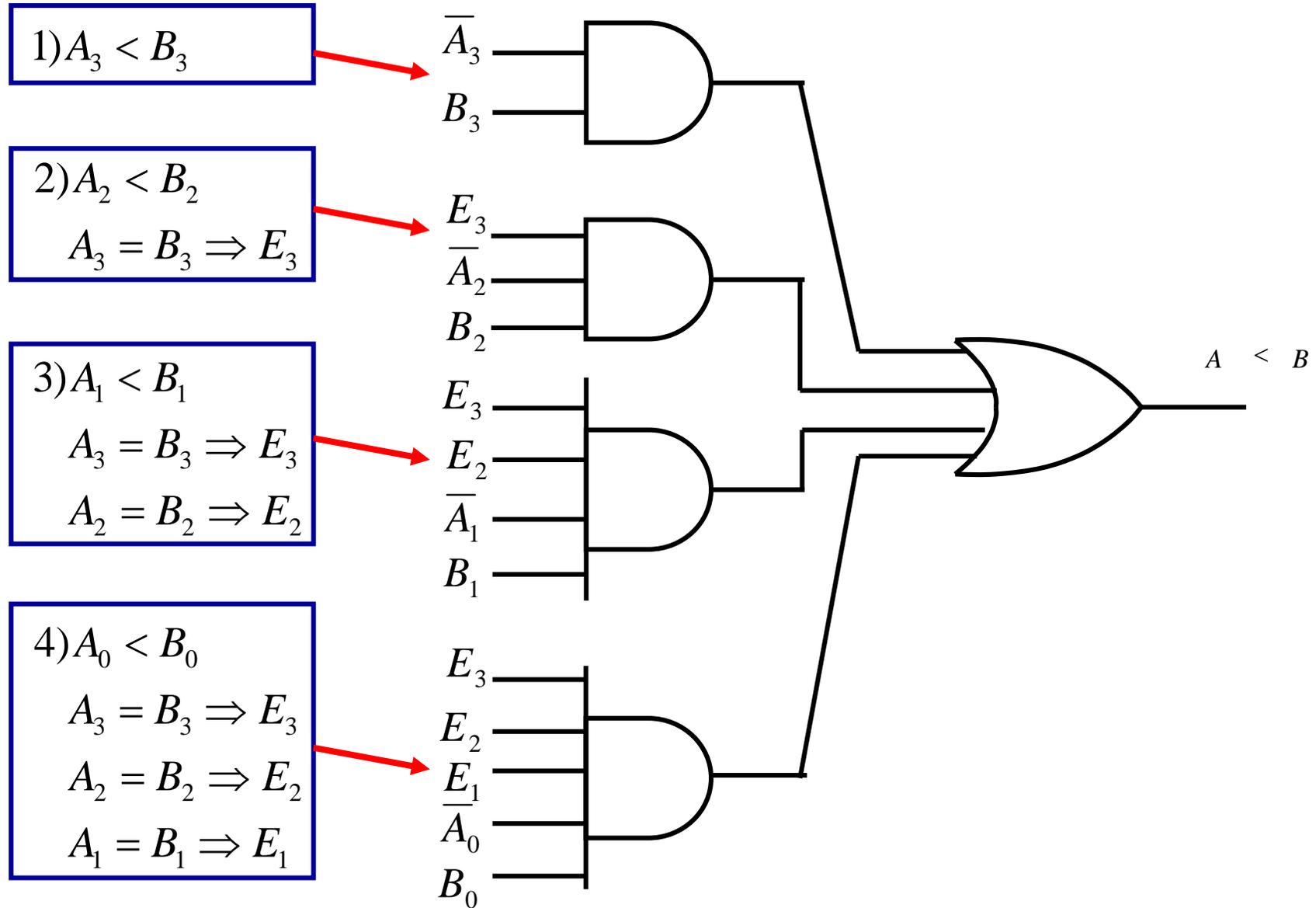


Funzione di diseuguaglianza: $A_i < B_i \Leftrightarrow \overline{A_i} \cdot B_i$

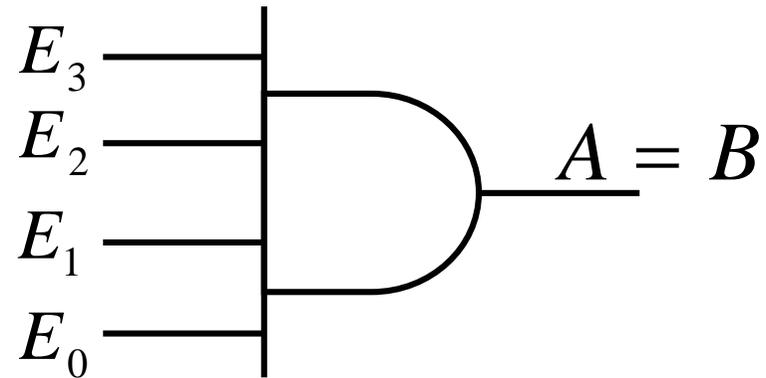


A	B	$\overline{A} \cdot B$
0	0	0
0	1	1
1	0	0
1	1	0

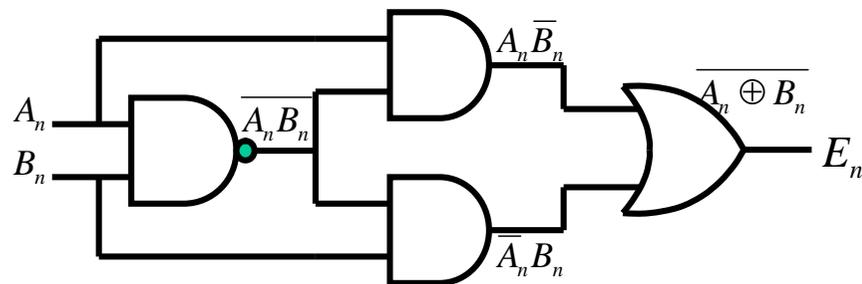
Logica per $A < B$ (per parole a 4 bit)



Logica per $A=B$ (per parole a 4 bit)



Le funzioni di eguaglianza:



Circuito XNOR per l'eguaglianza tra A_n e B_n

Logica per $A > B$ si può ricavare in diversi modi...

85

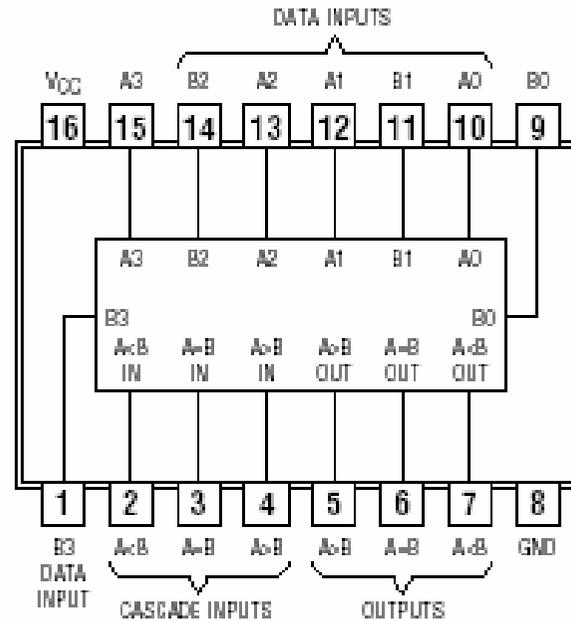
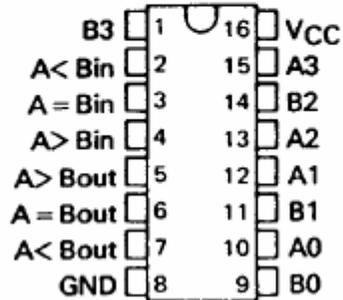
4-BIT MAGNITUDE COMPARATORS

SN5485, SN54LS85, SN54S85 . . . J OR W PACKAGE

SN7485 . . . N PACKAGE

SN74LS85, SN74S85 . . . D OR N PACKAGE

(TOP VIEW)

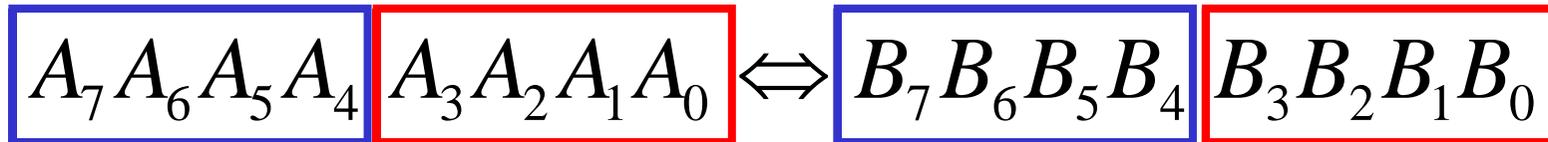


FUNCTION TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A2 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

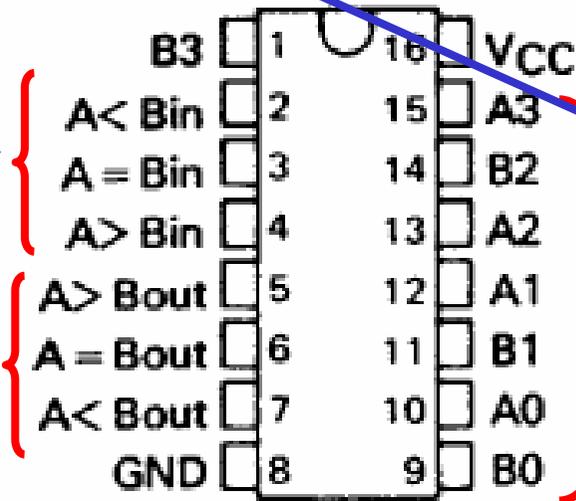
85

4-BIT MAGNITUDE COMPARATORS



3 bit di ingresso per eventuali confronti fra bit meno significativi

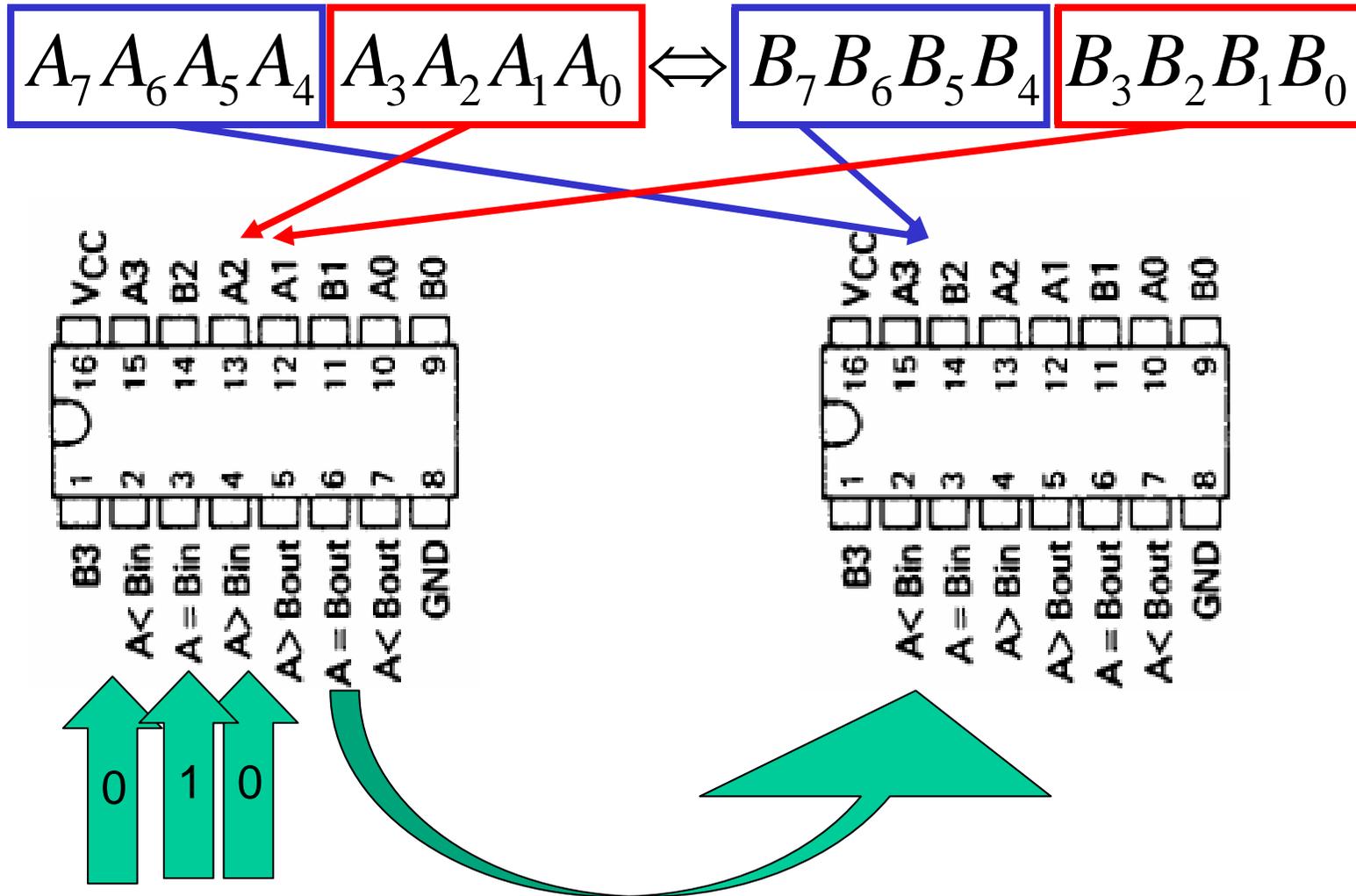
3 bit di uscita

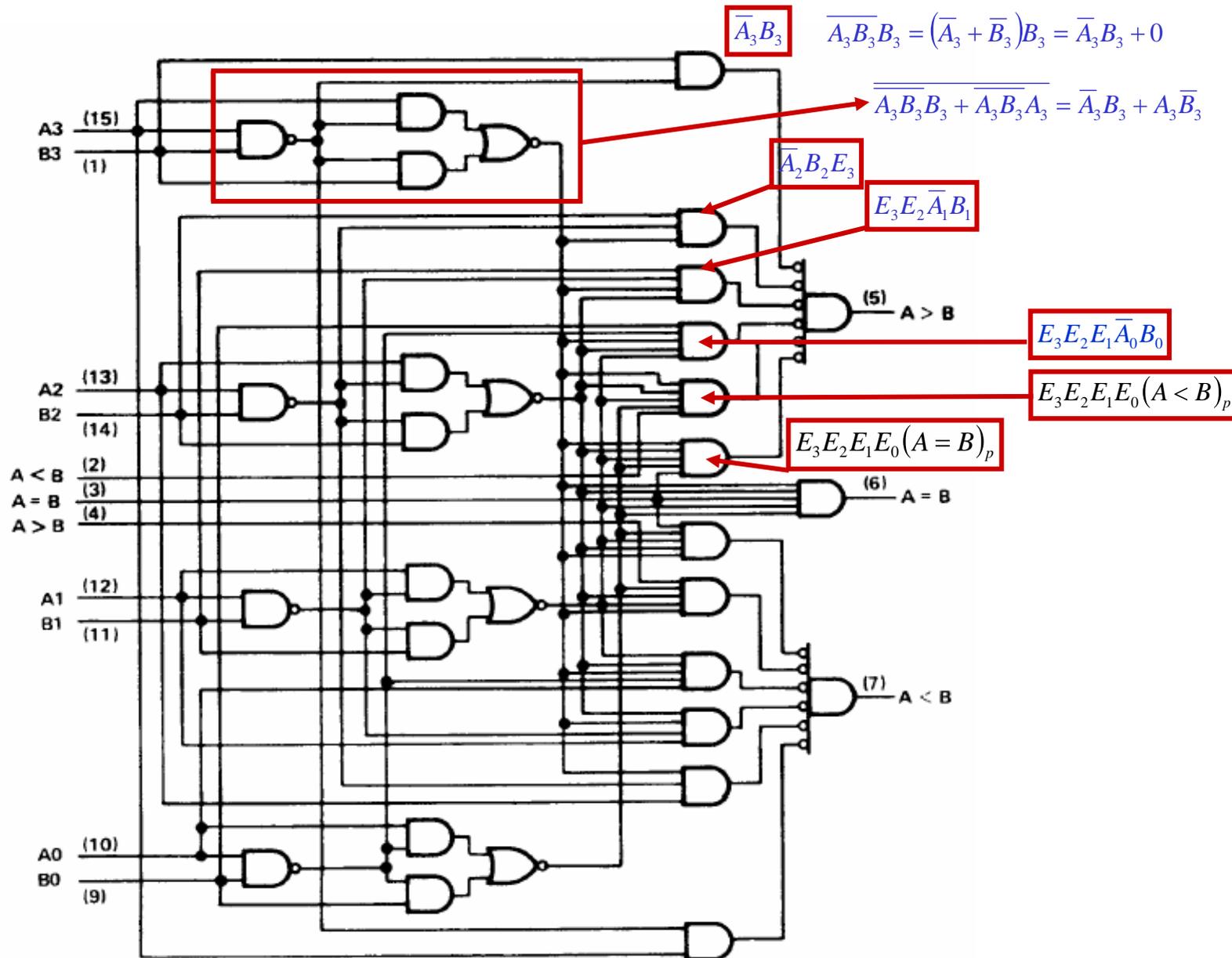


8 bit di ingresso

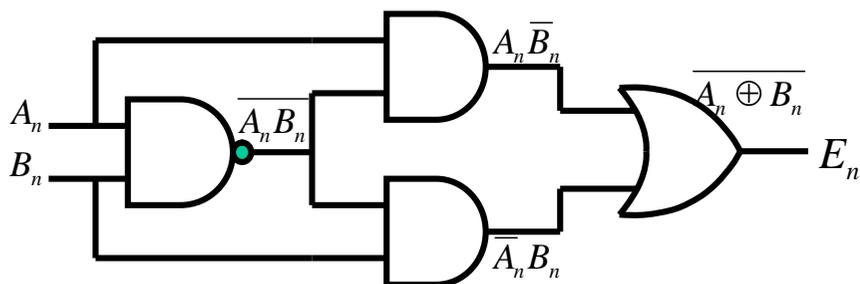
85

4-BIT MAGNITUDE COMPARATORS



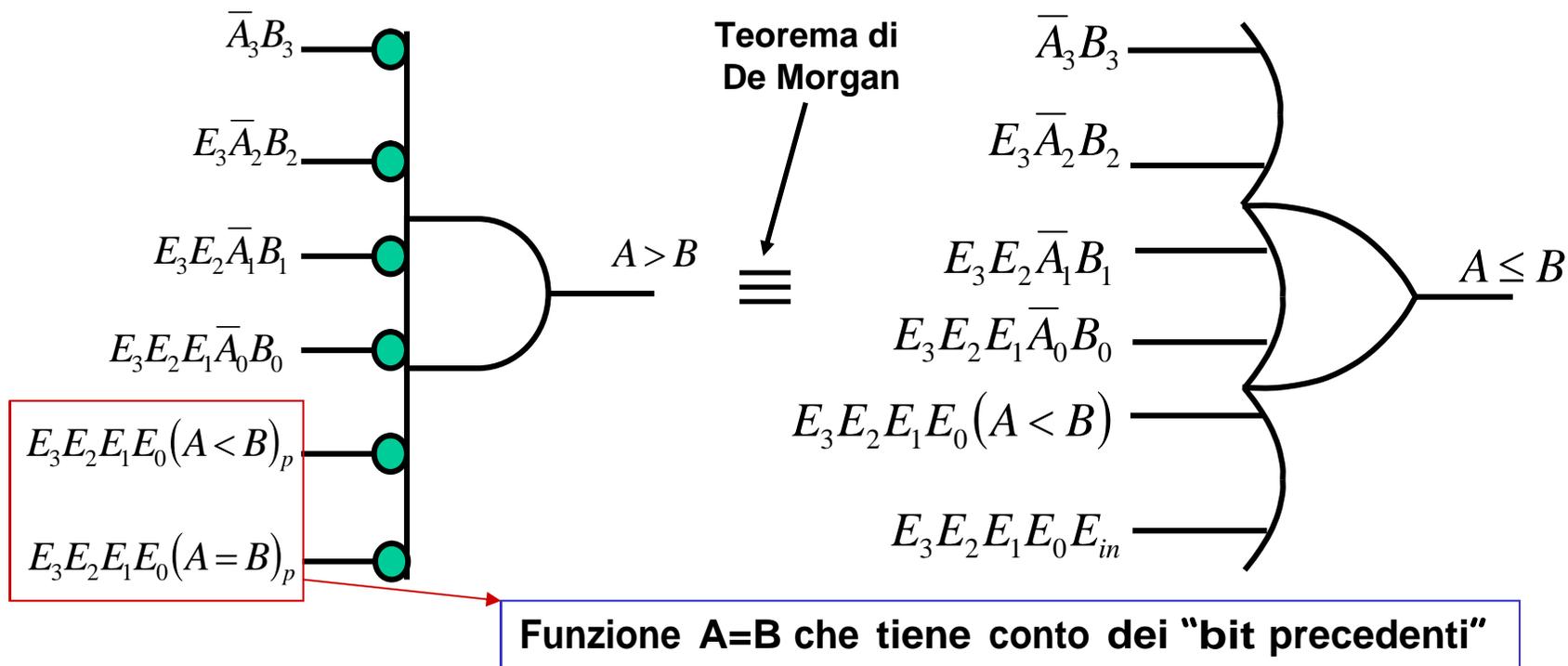


Le funzioni di eguaglianza nel 7485:

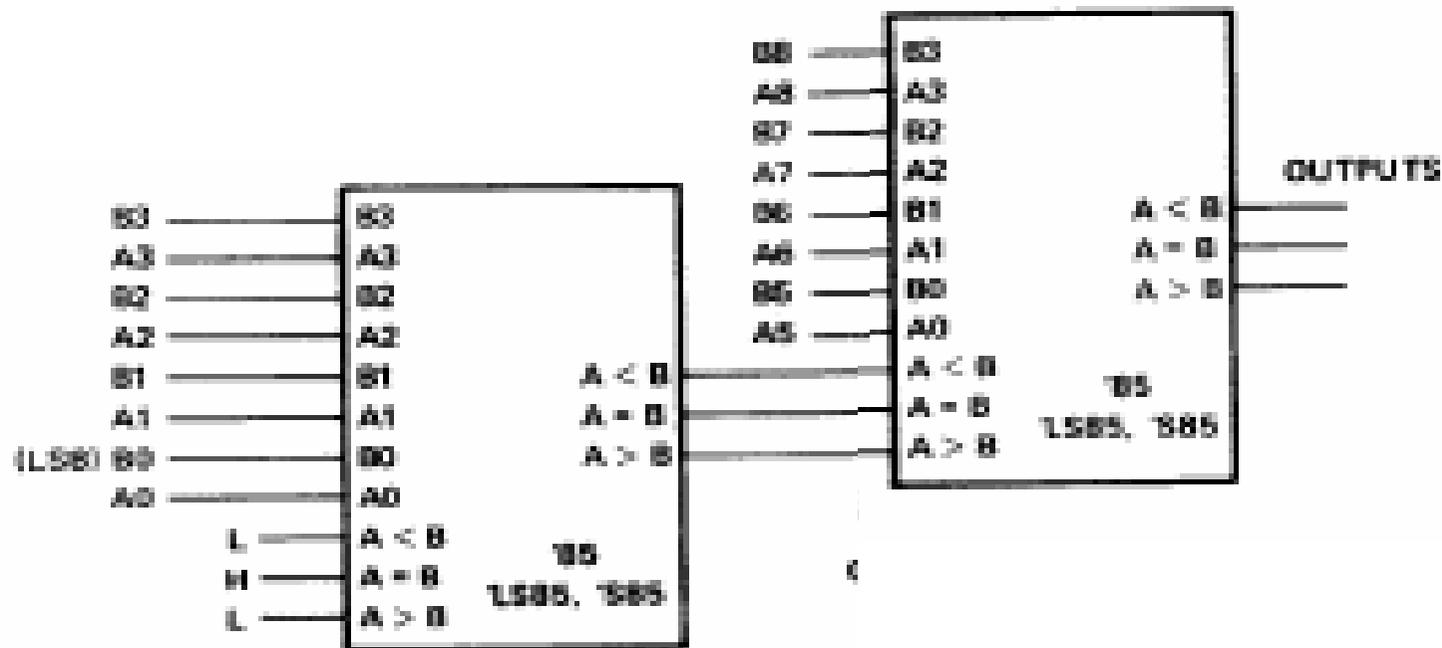


Circuito XNOR per l'eguaglianza tra A_n e B_n

Funzione $A > B$ nel 7485:

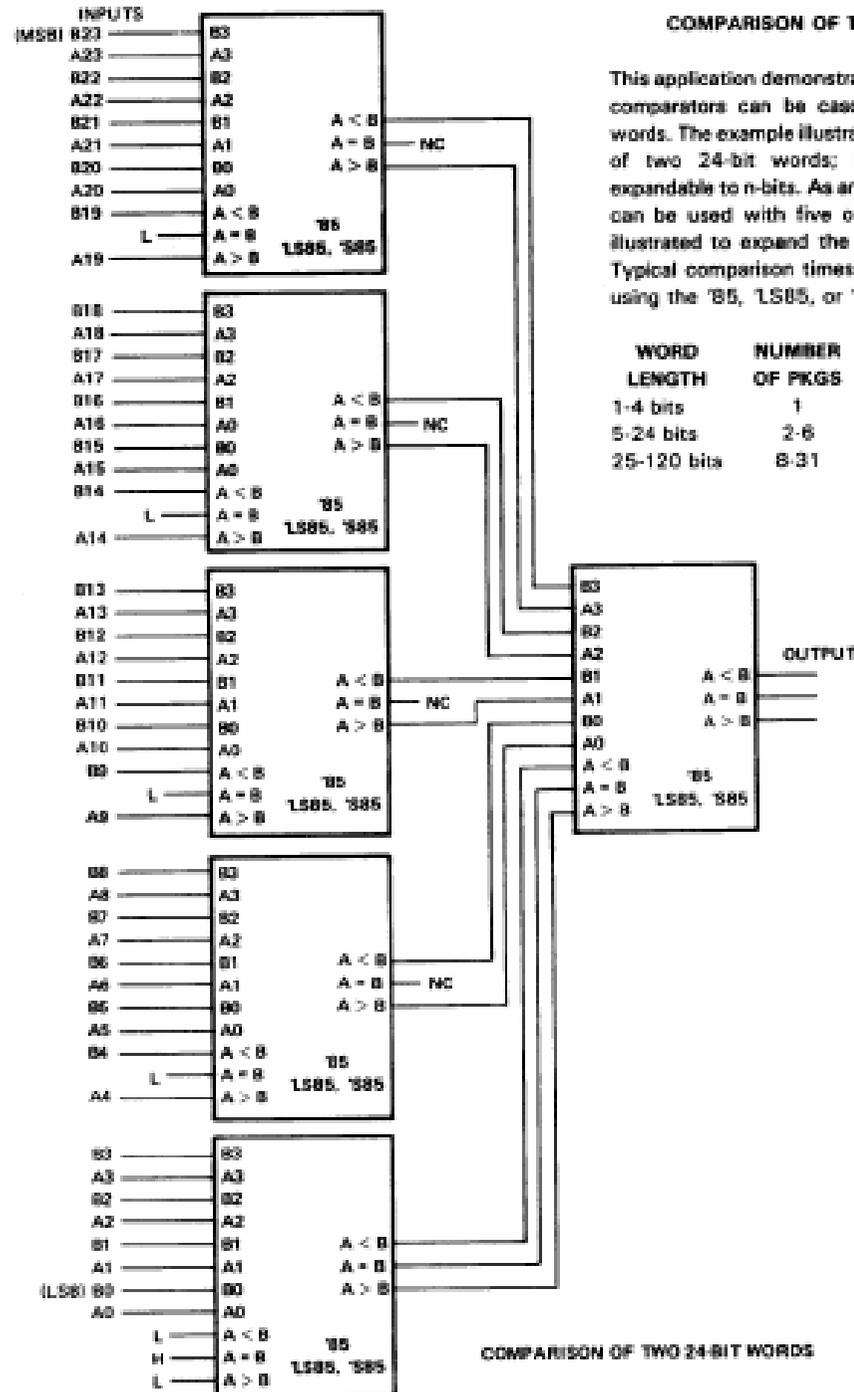


Applicazione di 2 '85 per confrontare 2 parole da 8 bit



Confronto di 2 parole da 24 bit

TYPICAL APPLICATION DATA



COMPARISON OF TWO N-BIT WORDS

This application demonstrates how these magnitude comparators can be cascaded to compare longer words. The example illustrated shows the comparison of two 24-bit words; however, the design is expandable to n-bits. As an example, one comparator can be used with five of the 24-bit comparators illustrated to expand the word length to 120-bits. Typical comparison times for various word lengths using the '85, 1585, or '885 are:

WORD LENGTH	NUMBER OF PKGS	'85	1585	'885
1-4 bits	1	23 ns	24 ns	11 ns
5-24 bits	2-6	48 ns	48 ns	22 ns
25-120 bits	8-31	68 ns	72 ns	33 ns

Perchè il comparatore?

- **Serve per confrontare due numeri (ovvio)**
- **Il risultato del confronto serve per le operazioni fra numeri binari da eseguire con circuiti logici...**
 - ➔ **realizziamo un circuito per la somma**
 - ➔ **ma con la somma possiamo eseguire anche sottrazioni**
 - ➔ **per decidere come *sottrarre sommando* dobbiamo confrontare A e B...**

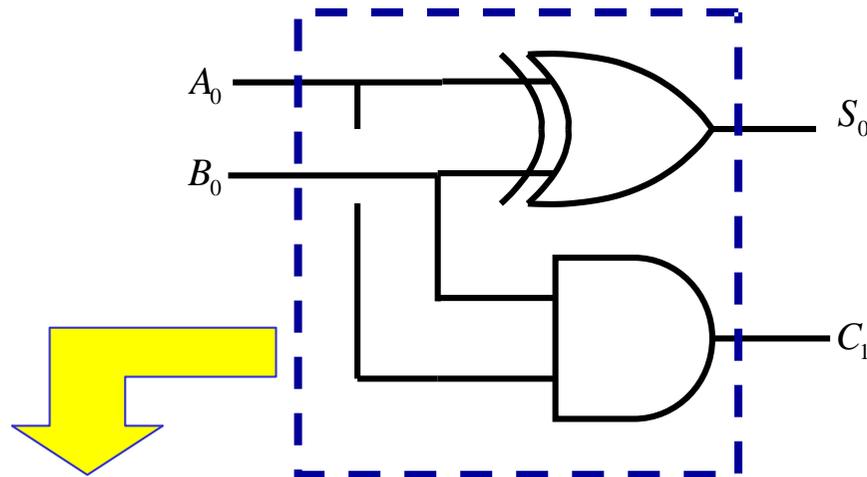
Sommatori e Sottrattori



Se dobbiamo produrre un circuito per la somma dobbiamo determinare la funzione logica tramite la relativa tavola della verità.

Nel caso di **2 bit**:

XOR - AND			
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Si chiama mezzo sommatore (Half Adder) perchè?

Non tiene conto dell'eventuale riporto in ingresso!

Sommatori e Sottrattoti



C_{n-1}	A_n	B_n	C_n	S_n
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

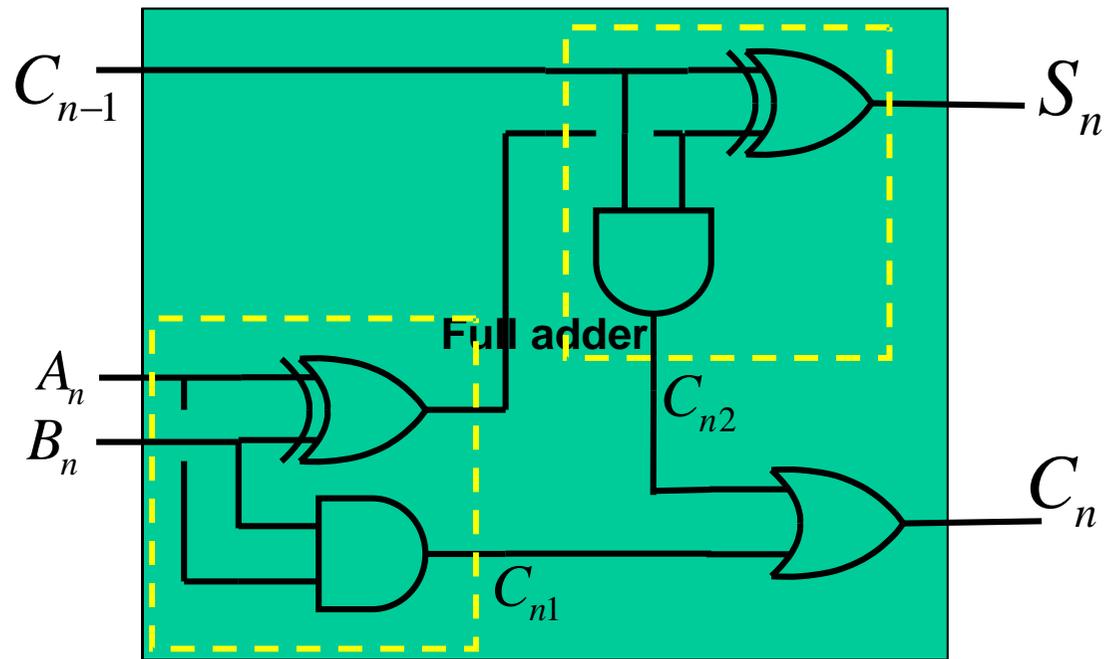
$$C_n = \overline{C_{n-1}} \underline{A_n} B_n + C_{n-1} \overline{A_n} B_n + C_{n-1} \underline{A_n} \overline{B_n} + C_{n-1} A_n B_n$$

$$S_n = C_{n-1} A_n B_n + C_{n-1} \overline{A_n} B_n + C_{n-1} \underline{A_n} \overline{B_n} + C_{n-1} \overline{A_n} \overline{B_n}$$

$$C_n = (\overline{C_{n-1}} + C_{n-1}) A_n B_n + C_{n-1} (\overline{A_n} B_n + A_n \overline{B_n}) = A_n B_n + C_{n-1} (A_n \oplus B_n)$$

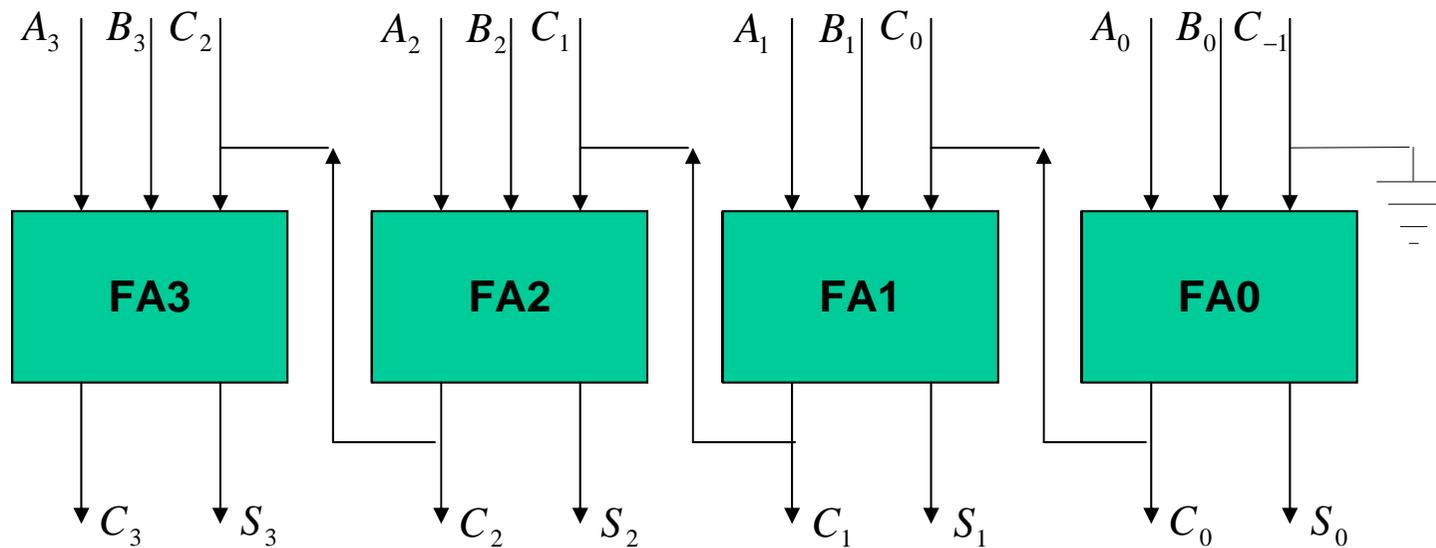
$$S_n = \overline{C_{n-1}} (\overline{A_n} B_n + A_n \overline{B_n}) + C_{n-1} (\overline{A_n} \overline{B_n} + A_n B_n) = C_{n-1} \oplus (A_n \oplus B_n)$$

Full adder



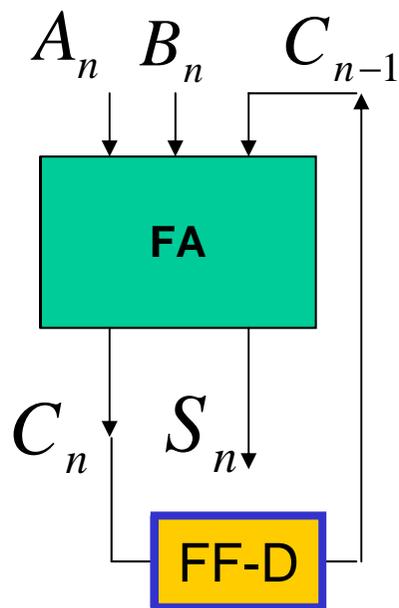
C_{n-1}	A_n	B_n	C_n	S_n	C_{n1}	C_{n2}
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	1	1	0	0	1
1	1	0	1	0	0	1
1	1	1	1	1	1	0

Sommatore binario parallelo a 4 bit realizzato con 4 sommatore completi in cascata:



✓ *ritardi diversi per le diverse uscite, il riporto C_3 arriva dopo tutto il resto*

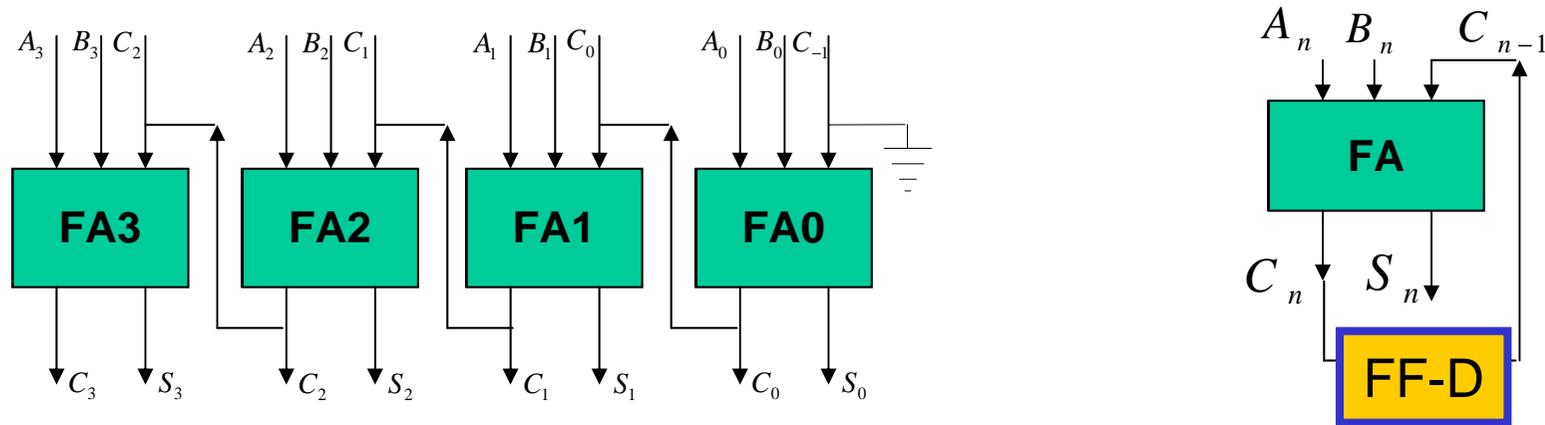
Sommatore binario seriale a n bit realizzato con 1 sommatore completo con retroazione del riporto:



- ✓ l'operazione di somma è eseguita in serie dai bit meno significativi ai bit più significativi
- ✓ il riporto precedente viene tenuto in memoria da un FF-D per essere sommato alla somma successiva

- ✓ l'inserimento dei bit e del riporto deve essere sincronizzato
→ il riporto deve essere opportunamente ritardato

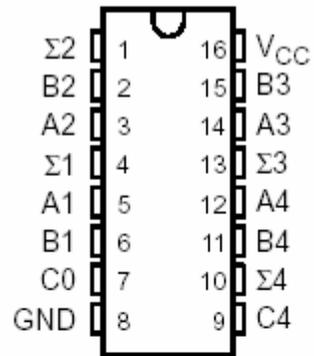
Confronto fra sommatore parallelo e seriale



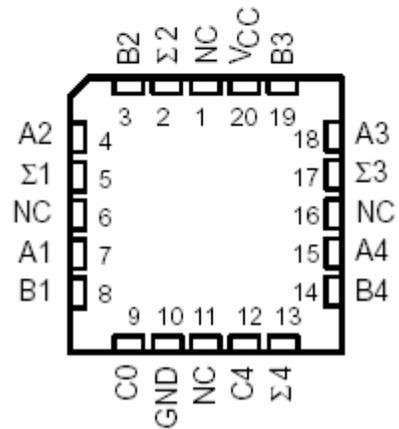
- ✓ il sommatore parallelo è più veloce del seriale, ma ha bisogno di un determinato numero di moduli
- ✓ il sommatore seriale è più lento del parallelo e deve essere sincronizzato, ma è necessario solo un modulo

**Full Adder with Fast Carry:
esegue i riporti in parallelo**

SN54F283 . . . J PACKAGE
 SN74F283 . . . D OR N PACKAGE
 (TOP VIEW)

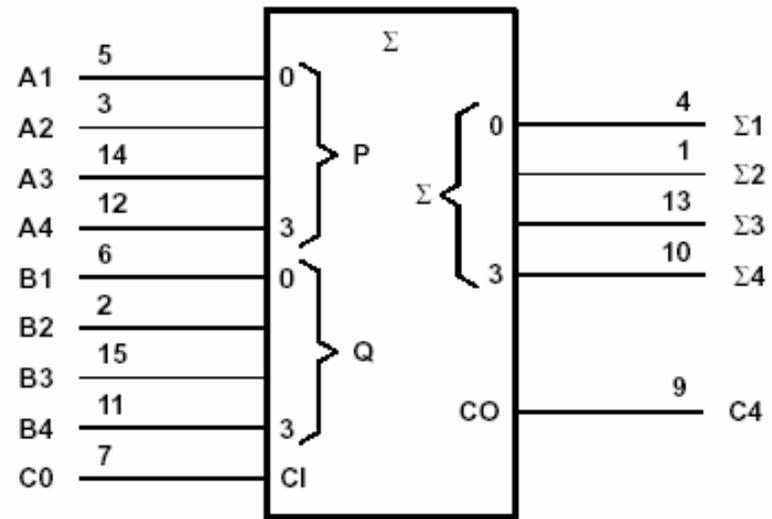


SN54F283 . . . FK PACKAGE
 (TOP VIEW)

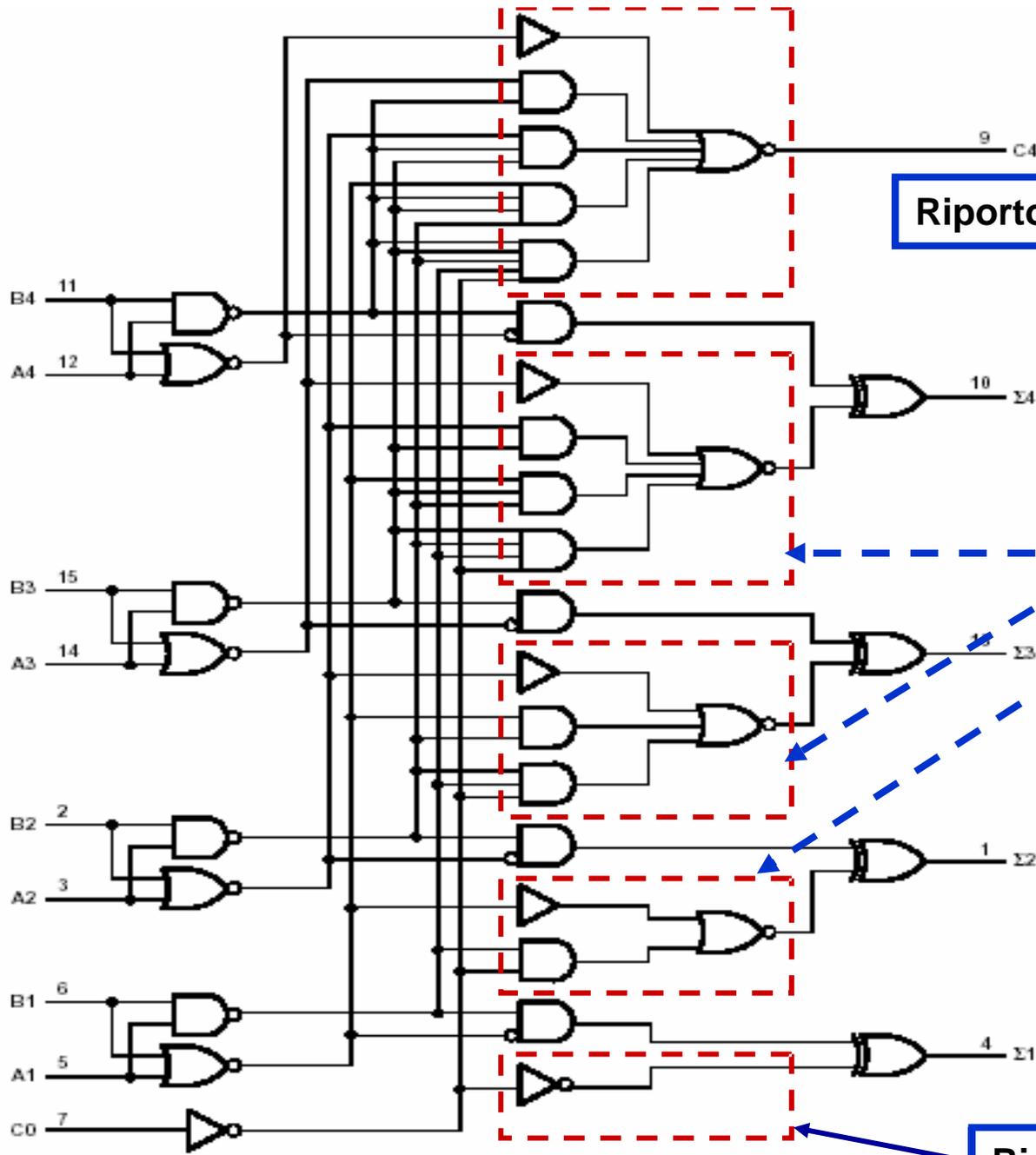


NC – No internal connection

SN54F283, SN74F283 4-BIT BINARY FULL ADDERS WITH FAST CARRY



logic diagram (positive logic)



Riporto esterno

SN54F283, SN74F283
4-BIT BINARY FULL ADDERS
WITH FAST CARRY

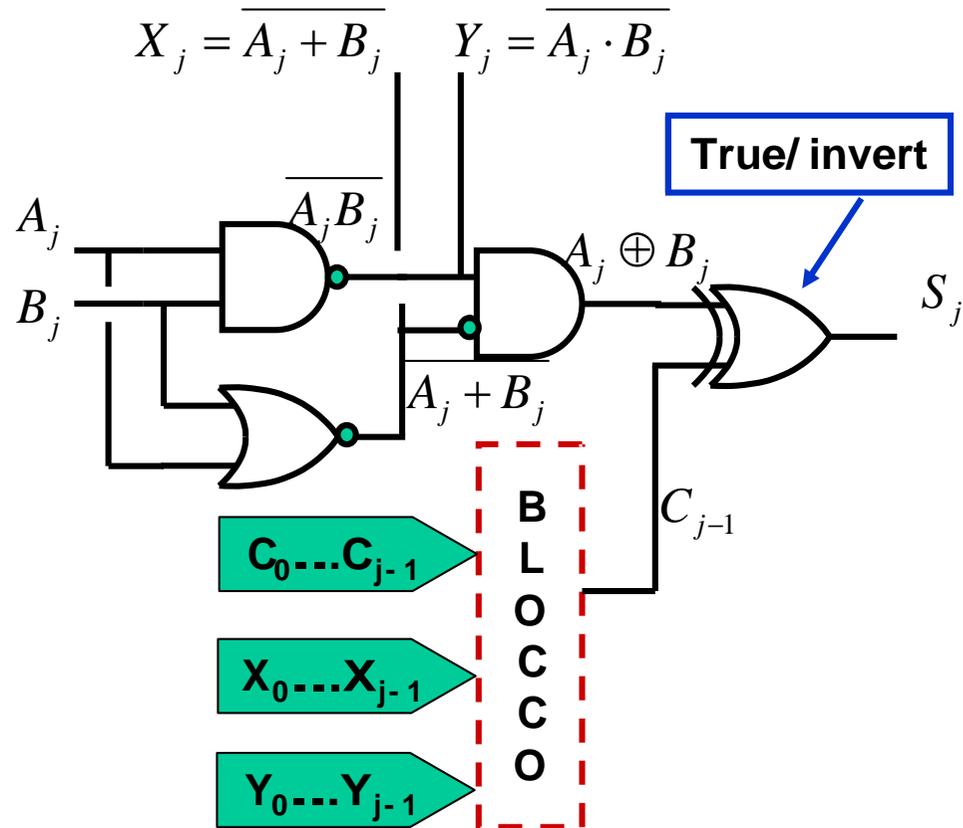
Riporti interni

I bit S_j della somma:

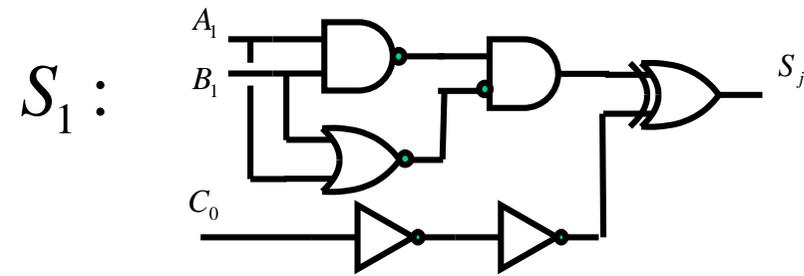
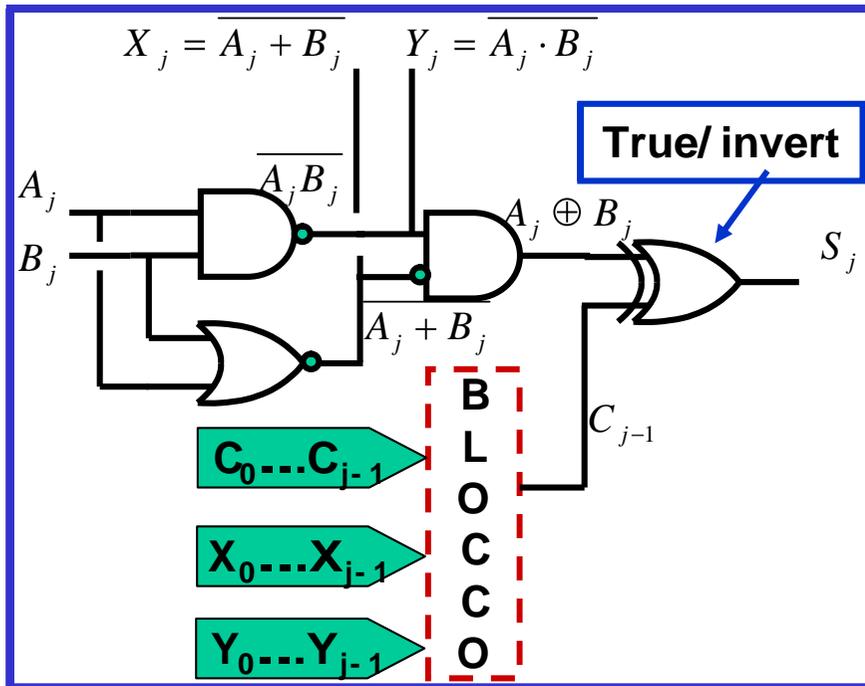
- circuiti con identica parte iniziale e finale
- passando da j a $j+1$ si ha una AND in più (+ un ingresso)
- il blocco fornisce il riporto

Riporto precedente

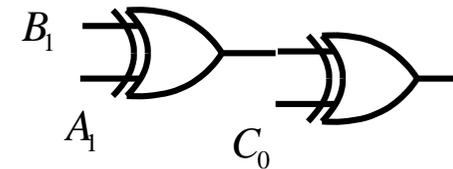
Il FULL ADDER con fast carry è formato da più moduli di questo tipo:



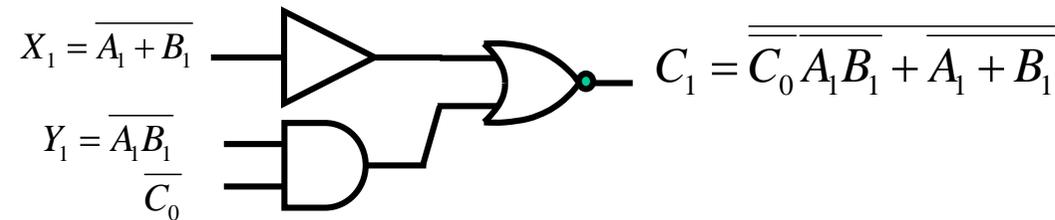
- ✓ stesso numero di porte per ogni singola funzione
- I bit S_j e C_n compaiono all'uscita senza ritardi



È equivalente a:



$S_2 :$ L' unica differenza è il blocco seguente:



Con C_1 che si aggiunge alla somma di A_2 B_2 ma:

$$\begin{aligned}
 C_1 &= \overline{\overline{C_0 A_1 B_1} (A_1 + B_1)} = (C_0 + A_1 B_1) \cdot (A_1 + B_1) \\
 &= C_0 A_1 + C_0 B_1 + A_1 B_1 = C_0 (A_1 + B_1) + A_1 B_1
 \end{aligned}$$

$$C_1 = \overline{\overline{C_0 A_1 B_1}} (A_1 + B_1) = (C_0 + A_1 B_1) \cdot (A_1 + B_1)$$

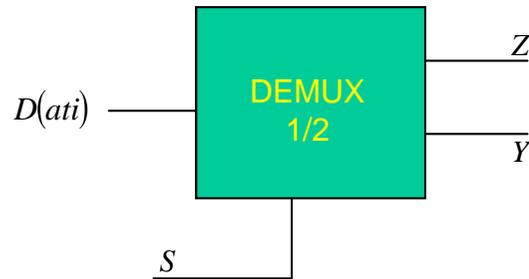
$$= C_0 A_1 + C_0 B_1 + A_1 B_1 = C_0 (A_1 + B_1) + A_1 B_1$$

C_0	A_1	B_1	C_1	$A_1 * B_1$	$A_1 + B_1$	$C_0(A_1 + B_1)$
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	0	1	0
0	1	1	1	1	0	0
1	1	0	1	0	1	1
1	0	1	1	0	1	1
1	0	0	0	0	0	0
1	1	1	1	1	1	1

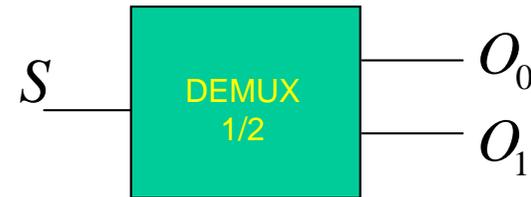
C_1 è vera se sono veri o l'uno ($A_1=B_1=1$) o l'altro: A_1 o $B_1=1$ e c'è un riporto precedente ($C_0=1$)

Decodificatori e Codificatori

Decodificatore (Demux)



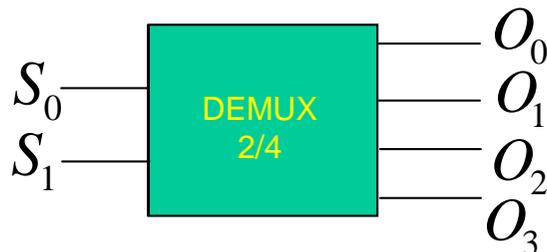
$$\begin{aligned} D &= 1 \\ Z &= O_0 \\ Y &= O_1 \end{aligned}$$



S = numero binario (0, 1)

O_0 uscita che corrisponde al numero decimale 0

O_1 uscita che corrisponde al numero decimale 1



S_1S_0 = numero binario (00, 01, 10, 11)

O_0 uscita che corrisponde al numero decimale 0

O_1 uscita che corrisponde al numero decimale 1

O_2 uscita che corrisponde al numero decimale 2

O_3 uscita che corrisponde al numero decimale 3

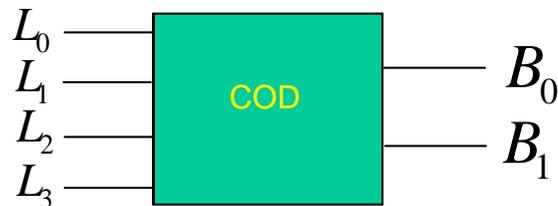
➤ **Demux: decodifica in decimale un numero binario**

✓ bit selezione costituiscono la parola binaria

✓ ogni linea di uscita corrisponde ad un numero decimale

Codificatore

- svolge la funzione inversa di un decodificatore
 - ✓ n linee di ingresso, ognuna corrispondente ad un numero decimale
 - ✓ N linee di uscite che costituiscono i bit della parola binaria corrispondente al numero decimale selezionato dalla linea di ingresso



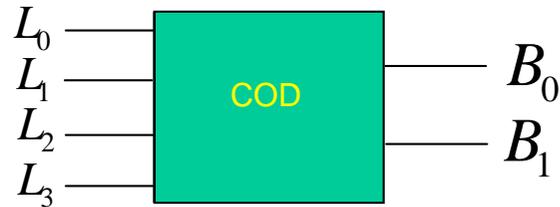
B_1B_0 = numero binario (00, 01, 10, 11)

L_0 linea d'ingresso che corrisponde al numero decimale 0
 L_1 linea d'ingresso che corrisponde al numero decimale 1
 L_2 linea d'ingresso che corrisponde al numero decimale 2
 L_3 linea d'ingresso che corrisponde al numero decimale 3

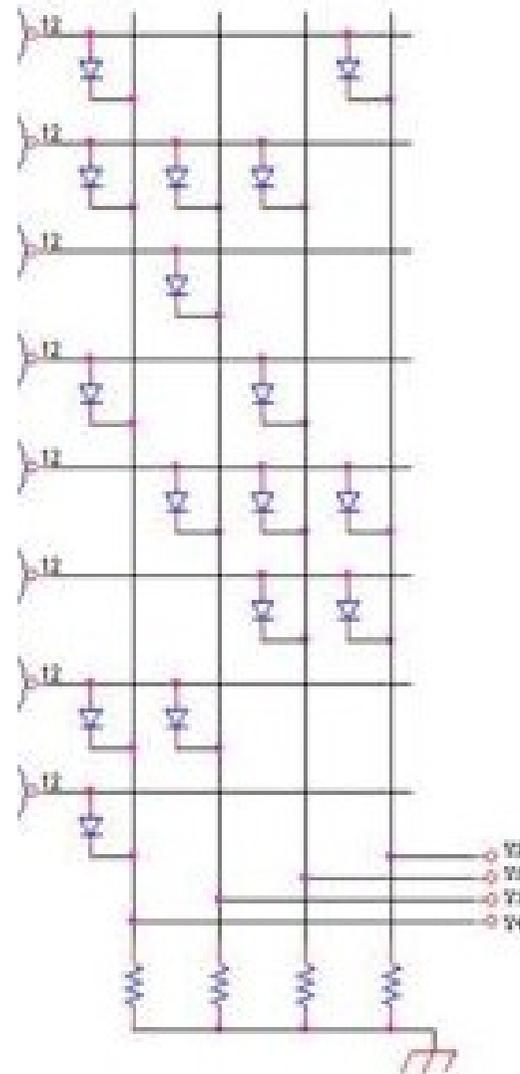
➤ non è un MUX

Codificatore

➤ In realtà viene realizzato con una matrice di diodi

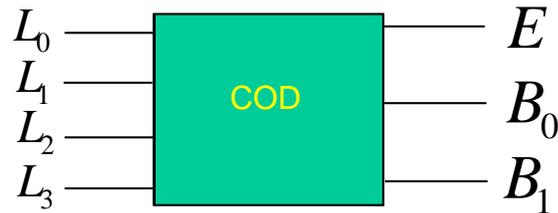


➤ Ma noi come esercitazione di laboratorio lo vogliamo realizzare con le porte logiche ... con una modifica



Codificatore

➤ esercitazione di laboratorio



$$L_0=1 (L_{1,2,3}=0) \rightarrow 0 \quad \rightarrow B_1B_0 = 00 \quad E=0$$

$$L_1=1 (L_{0,2,3}=0) \rightarrow 1 \quad \rightarrow B_1B_0 = 01 \quad E=0$$

$$L_2=1 (L_{0,1,3}=0) \rightarrow 2 \quad \rightarrow B_1B_0 = 10 \quad E=0$$

$$L_3=1 (L_{0,1,2}=0) \rightarrow 3 \quad \rightarrow B_1B_0 = 11 \quad E=0$$

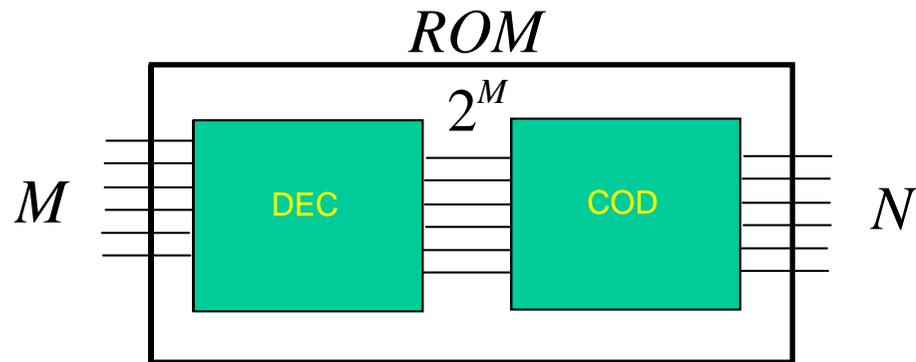
Ogni combinazione con più di un $L=1$ darà $B_1B_0=00$ e $E=1$

➤ Siccome noi pilotiamo L , potremmo anche accenderne più di una alla volta → nessun numero binario corrisponde a più linee accese → introduciamo l'indicatore di errore

➤ come facciamo?

➤ Suggerimento: non è necessario scrivere tutta la tavola della verità

Decodificatore + Codificatore → ROM (Read Only Memory)



➤ Conversione di un codice a M -bit → codice a N -bit

ELETTRONICA DEI SISTEMI DIGITALI

Parte III

Corso di Laurea in Informatica e TFI
Anno Accademico 2007-2008

Logica Combinatoria → Logica Sequenziale

- Fino ad ora abbiamo considerato solo: LOGICA COMBINATORIA che non ha “**memoria**”:

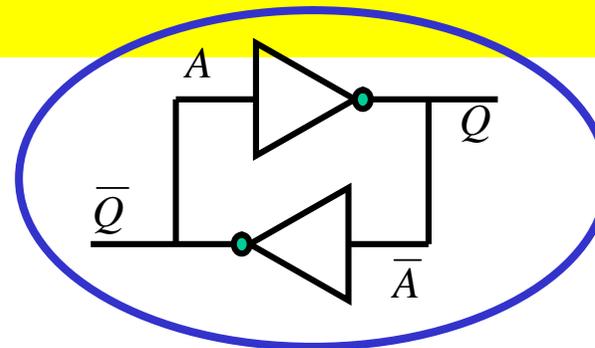
lo stato delle uscite al tempo T è univocamente determinato dallo stato degli ingressi allo stesso istante

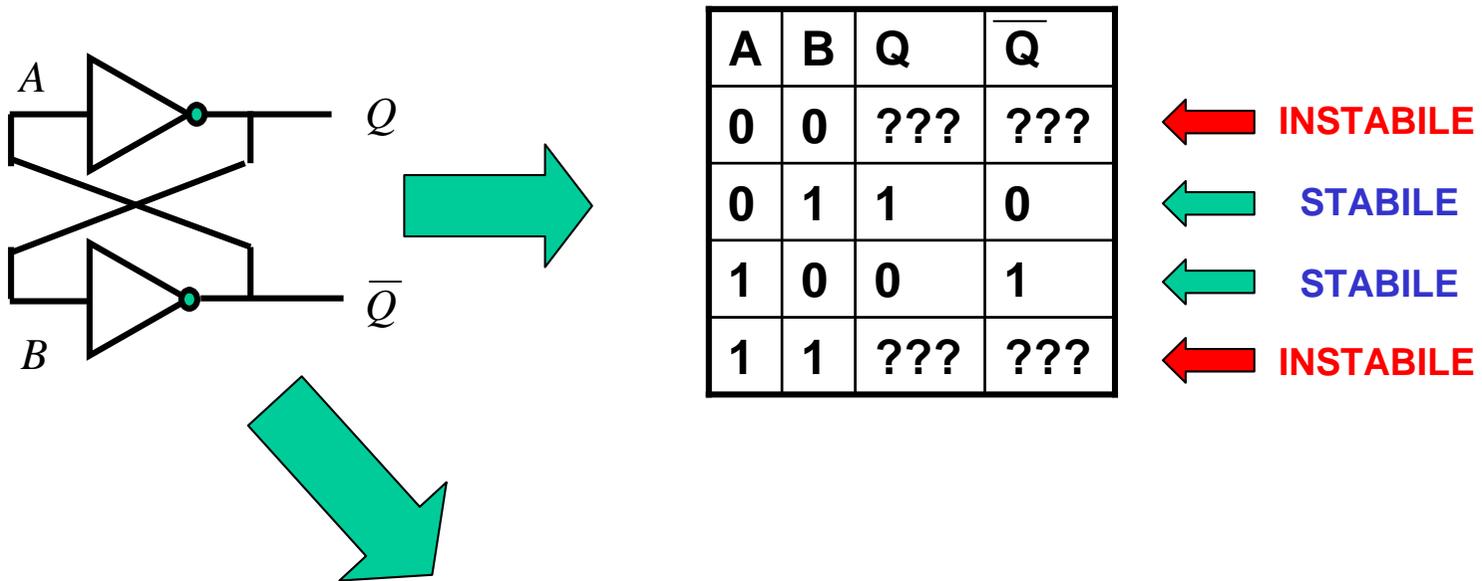
- Sappiamo che esiste anche una: LOGICA SEQUENZIALE che si “**ricorda**” della storia precedente:

lo stato delle uscite al tempo T dipende dagli stati che gli ingressi hanno assunto in tempi precedenti a T

La logica sequenziale si basa sul concetto di bistabilità (multiviratori):

1. Circuiti **astabili**
2. Circuiti **monostabili**
3. Circuiti **bistabili**



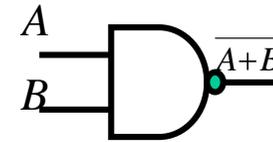
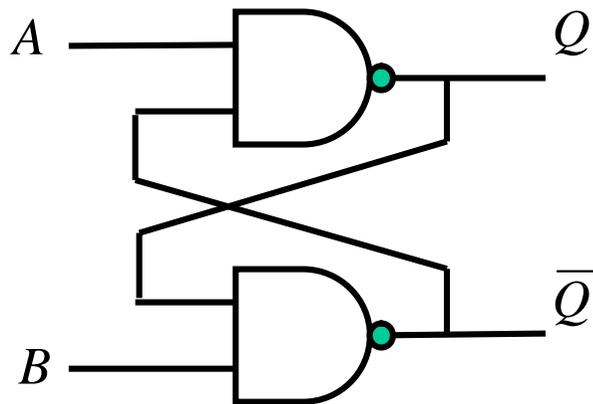


Ma quali problemi ha?

NON CI SONO INGRESSI!!!

Realizziamo un circuito simile con ingressi

Facciamo gli invertitori con NAND (o NOR)



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

$Q=0 \rightarrow A=1, B=0 \text{ o } 1 \quad (A=1, B=1)$

$Q=1 \rightarrow A=0, B=1 \quad (A=1, B=1)$

$A = 0, B = 0 \rightarrow Q = \overline{Q} \quad ???$

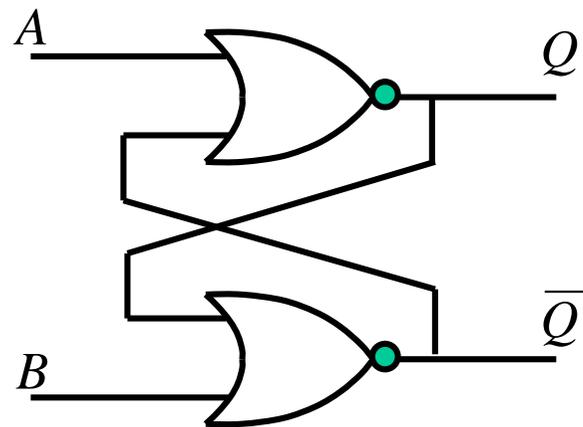


	A	B	Q_{n+1}
1	0	0	<i>n.p.</i>
2	0	1	1
3	1	0	0
4	1	1	Q_n

2(3) \rightarrow 4 Q invariato

2,3 \rightarrow 2,3 Q varia

Facciamo gli invertitori con NOR

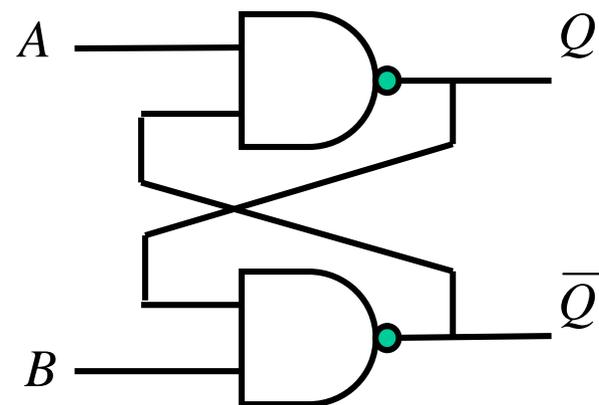


A	B	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	<i>n.p.</i>

I nomi assegnati ad A e B sono arbitrari, notazione standard è questa:

B=1 setta $Q=1 \rightarrow B=S$

A=1 resetta $Q=0 \rightarrow B=R$

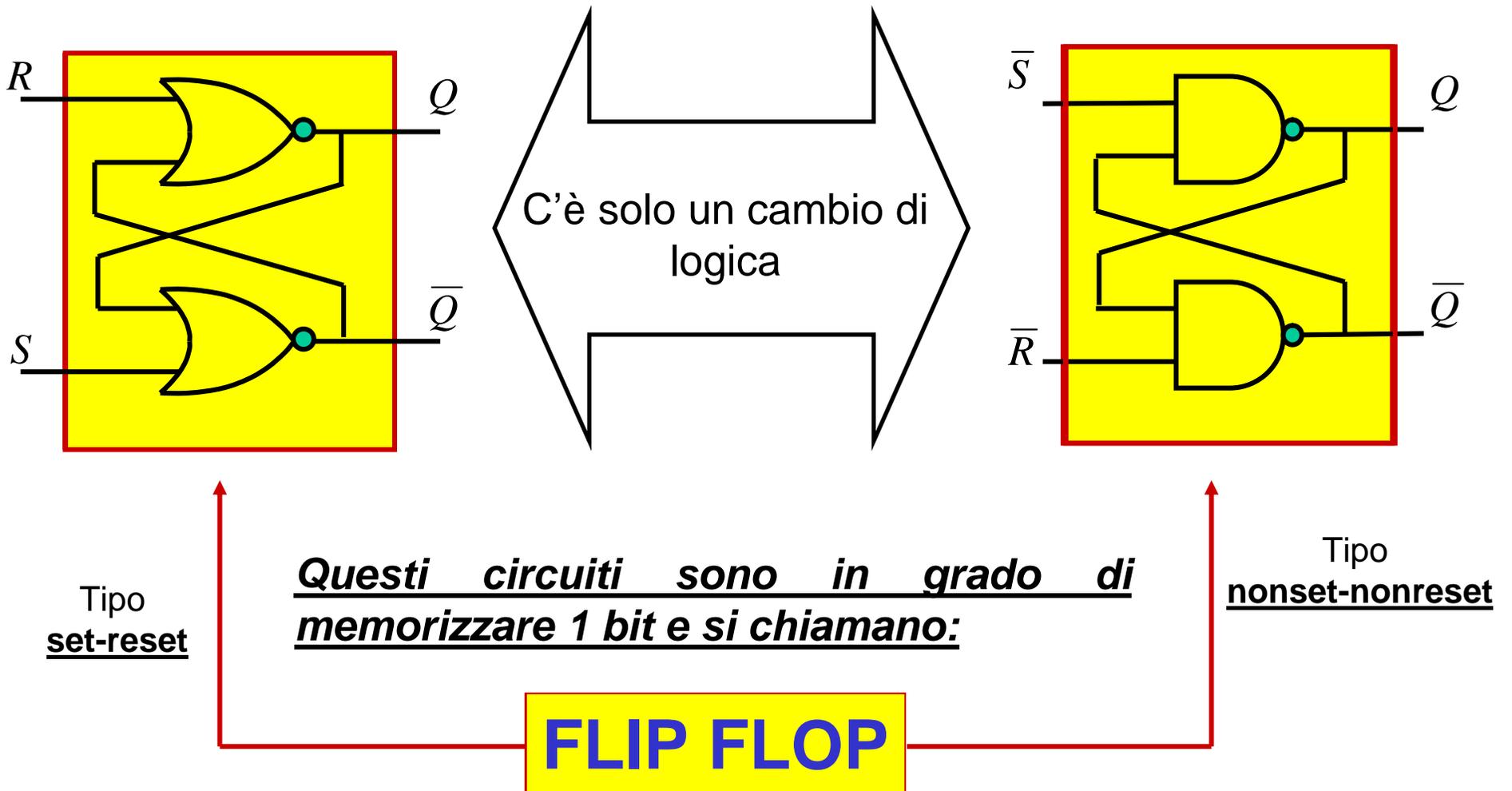


A	B	Q_{n+1}
0	0	<i>n.p.</i>
0	1	1
1	0	0
1	1	Q_n

A=0 setta $Q=1 \rightarrow A=\bar{S}$

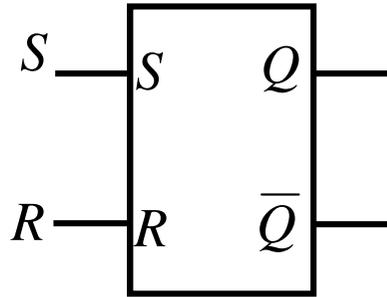
B=1 resetta $Q=0 \rightarrow B=\bar{R}$

Comunque sia si deve sempre far riferimento alla tavola della verità

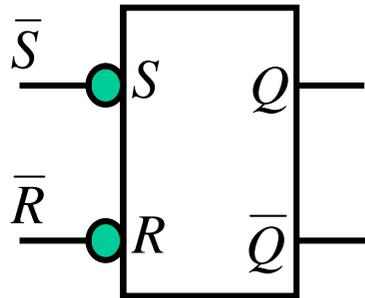
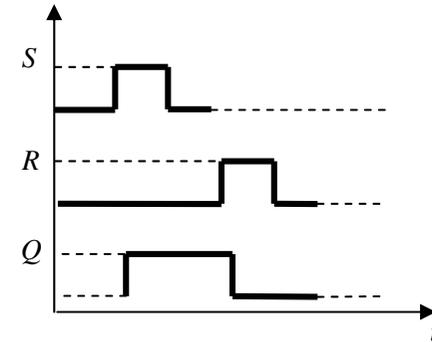


NB

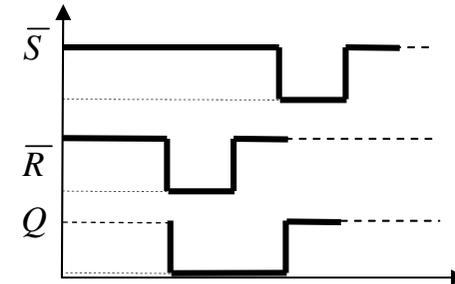
Hanno comportamento bistabile: sotto l'effetto di uno stimolo di comando in ingresso (S,R) generano uno stato stabile in uscita (Q, \bar{Q}) che permane anche una volta cessato il comando



S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	<i>n.p.</i>



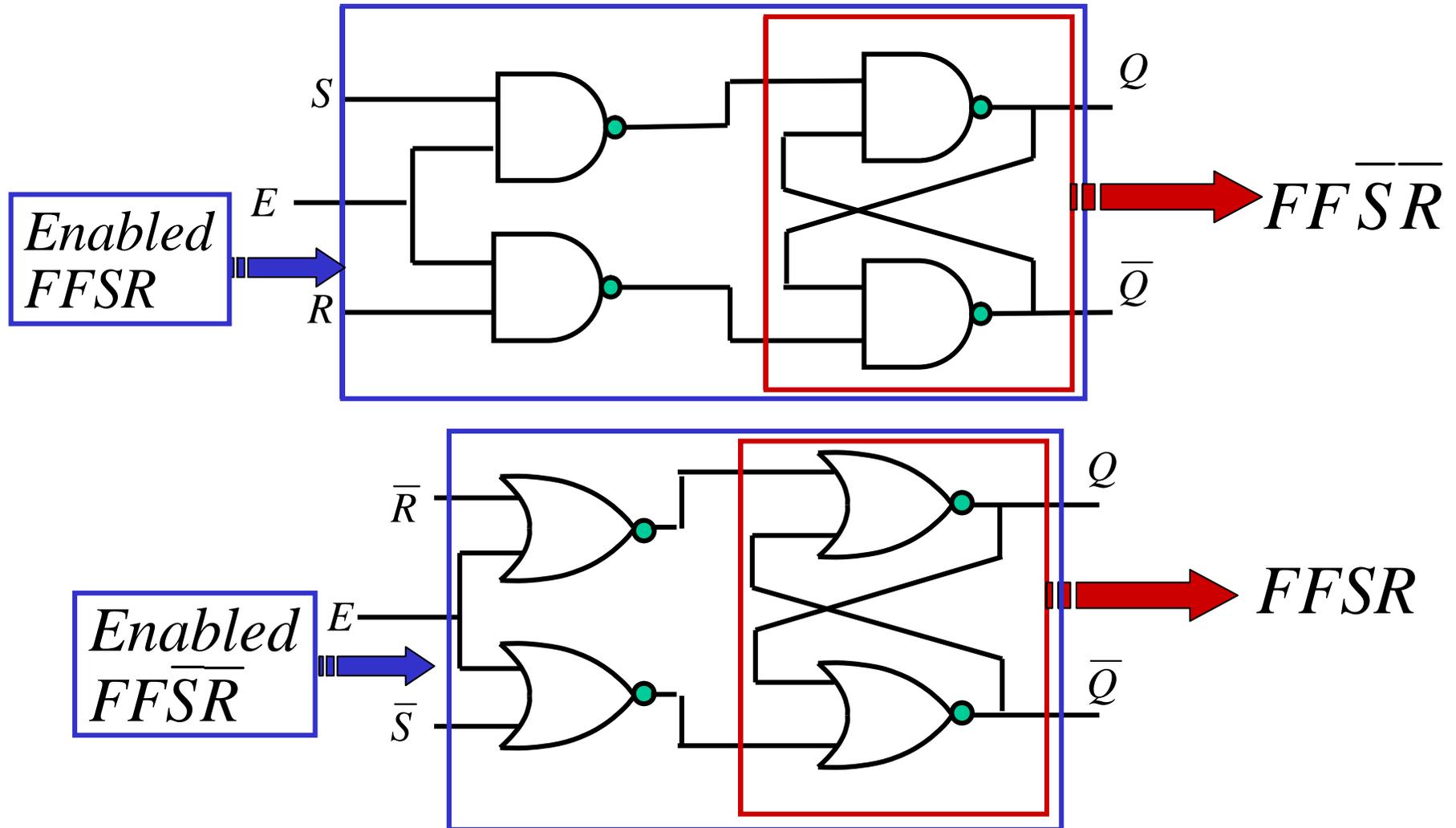
\bar{S}	\bar{R}	Q_{n+1}
1	1	Q_n
0	1	1
1	0	0
0	0	<i>n.p.</i>

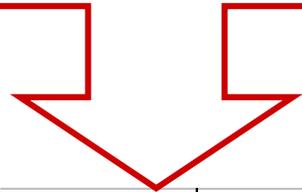
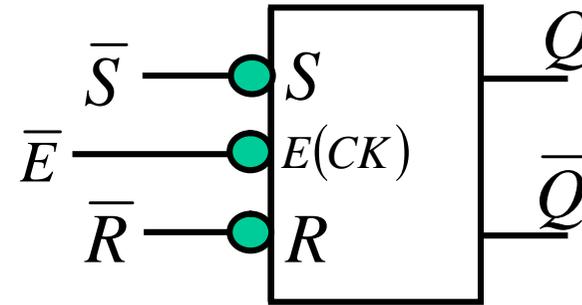
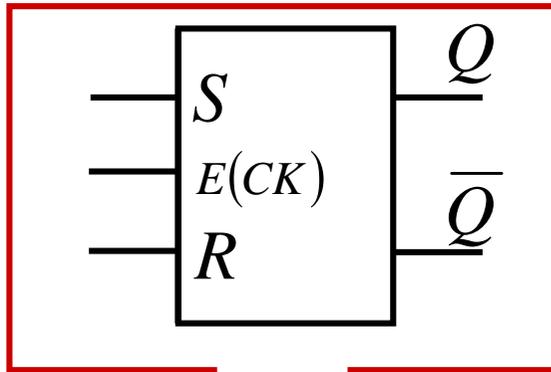


Attenzione:

1. Quali associamo ai circuiti realizzati con sole NOR e sole NAND?
2. si attiva un solo ingresso per volta
3. Il circuito è "sincronizzabile" (abilitazione)
4. In uno stesso FF si possono avere ingressi attivi alti e attivi bassi

FLIP FLOP..... con abilitazione





E	S	R	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	<i>n.p.</i>



Il FF è bloccato



Con E abilitato funziona come Un normale SR



Non permesso

Se usiamo l' ENABLE:

1. Per bloccare o meno il FF: **comportamento banale**
2. Per configurare i dati (S,R) per avere una certa uscita (Q, \bar{Q}) in un certo istante (E): **comportamento più furbo**

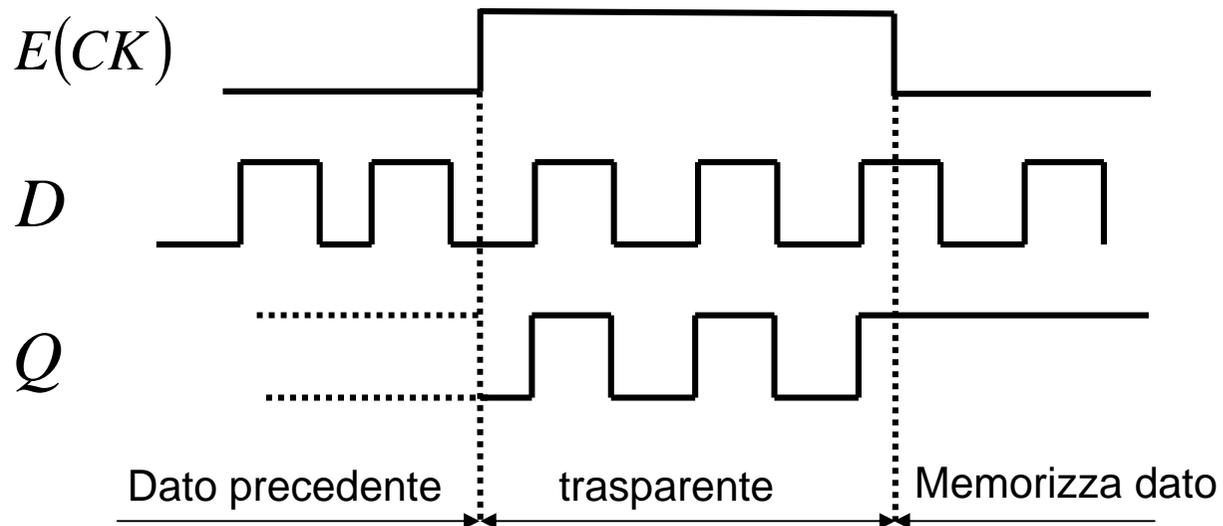
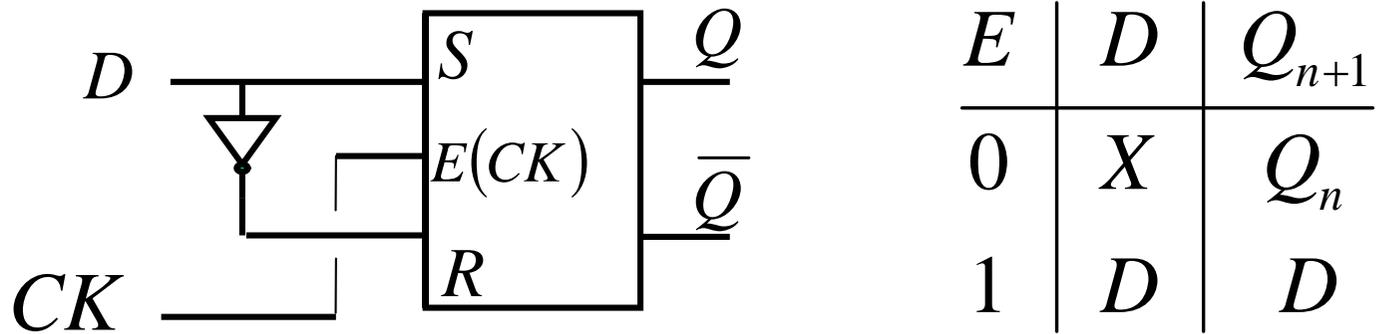
$$E=CK$$

In sostanza si hanno due modi:

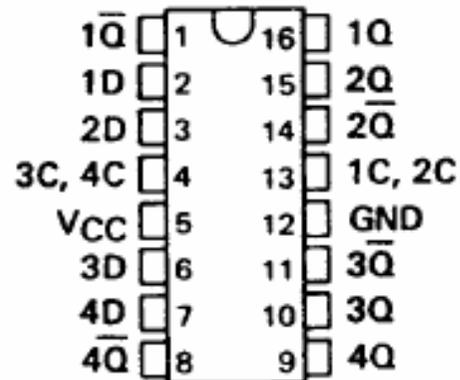
1. **Abilitare il FF: E fisso mentre variano S e R**
2. **Ritardare opportunamente il FF: R e S sono stabili prima e durante un impulso (ciclo ?) di clock (E)**

Il FF è una cella di memoria a un bit. Es: 7475 , 7477.....

LATCH "trasparente"



SN5475, SN54LS75 . . . J OR W PACKAGE
 SN7475 . . . N PACKAGE
 SN74LS75 . . . D OR N PACKAGE
 (TOP VIEW)



FUNCTION TABLE
 (each latch)

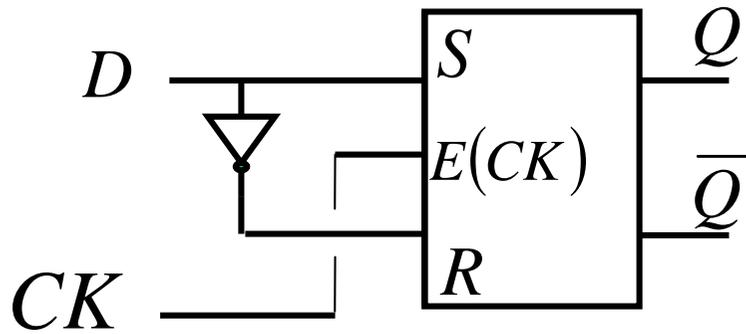
INPUTS		OUTPUTS	
D	C	Q	\bar{Q}
L	H	L	H
H	H	H	L
X	L	Q_0	\bar{Q}_0

H = high level, L = low level, X = irrelevant

Q_0 = the level of Q before the high-to-low transition of G

Esistono altri tipi di FF: **Toggle**: usato nei contatori, frequenzimetri, divisori ecc. ecc.

- **Cambia stato in uscita ad ogni impulso di clock;**
 - **Attenzione:** non deve essere trasparente!
- Proviamo infatti a modificare il nostro LATCH

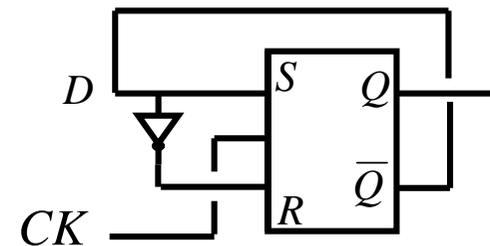
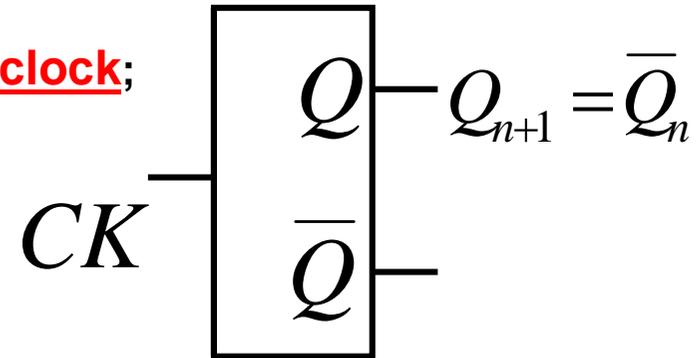


E	D	Q_{n+1}
0	X	Q_n
1	D	D

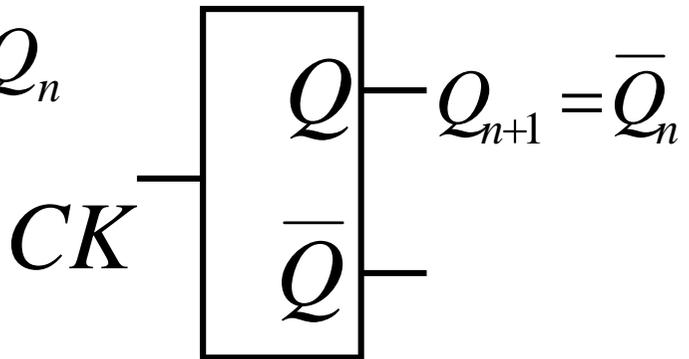
$$D = \overline{Q} \longrightarrow$$

E	D	Q_{n+1}
0	X	Q_n
1	$\overline{Q_n}$	$\overline{Q_n}$

$$\Rightarrow Q_{n+1} = \overline{Q_n} \Rightarrow Q_{n+2} = \overline{Q_{n+1}} = \overline{\overline{Q_n}} = Q_n$$



$$\Rightarrow Q_{n+1} = \overline{Q_n} \Rightarrow Q_{n+2} = \overline{Q_{n+1}} = \overline{\overline{Q_n}} = Q_n$$

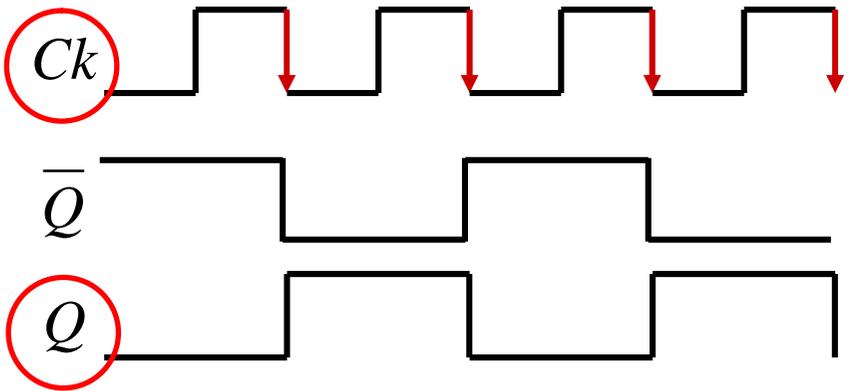


$$\Rightarrow \begin{cases} Q_{n+1} = \overline{Q_n} \\ Q_{n+2} = Q_n \end{cases}$$

- l'uscita Q oscilla
- è sincronizzata con il clock???

E	D	Q_{n+1}
0	X	Q_n
1	$\overline{Q_n}$	$\overline{Q_n}$

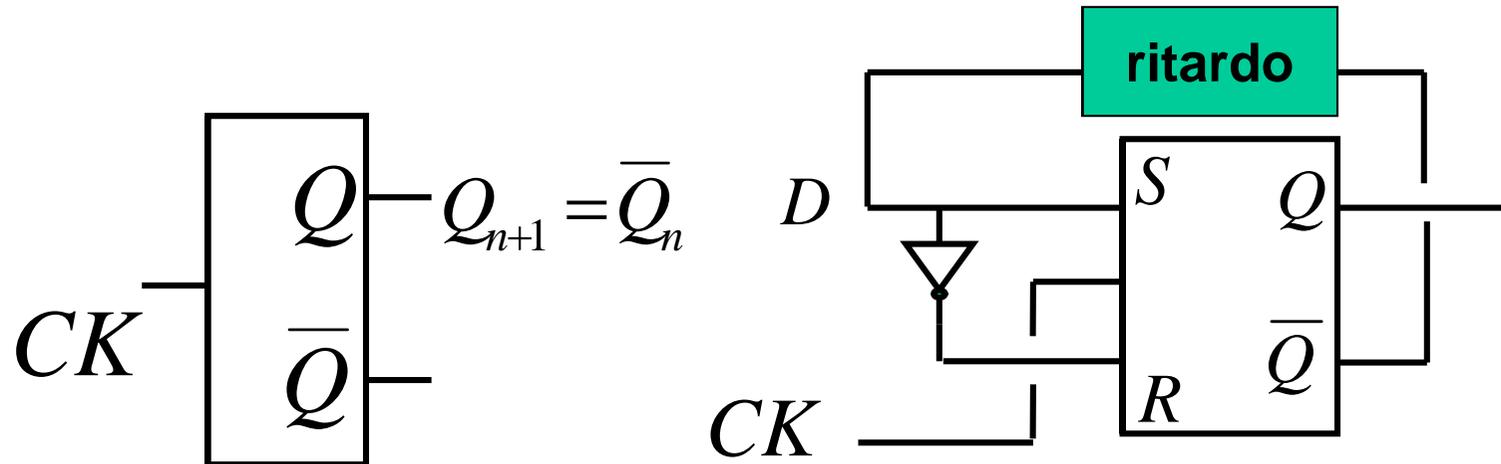
→ si verifica questo???



→ l'oscillazione si dovrebbe avere solo quando E=1 (in realtà non oscilla per niente)

Ma non funziona!!!
A causa della trasparenza

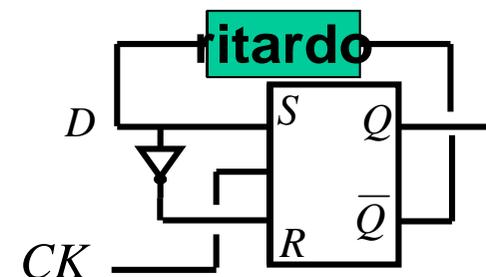
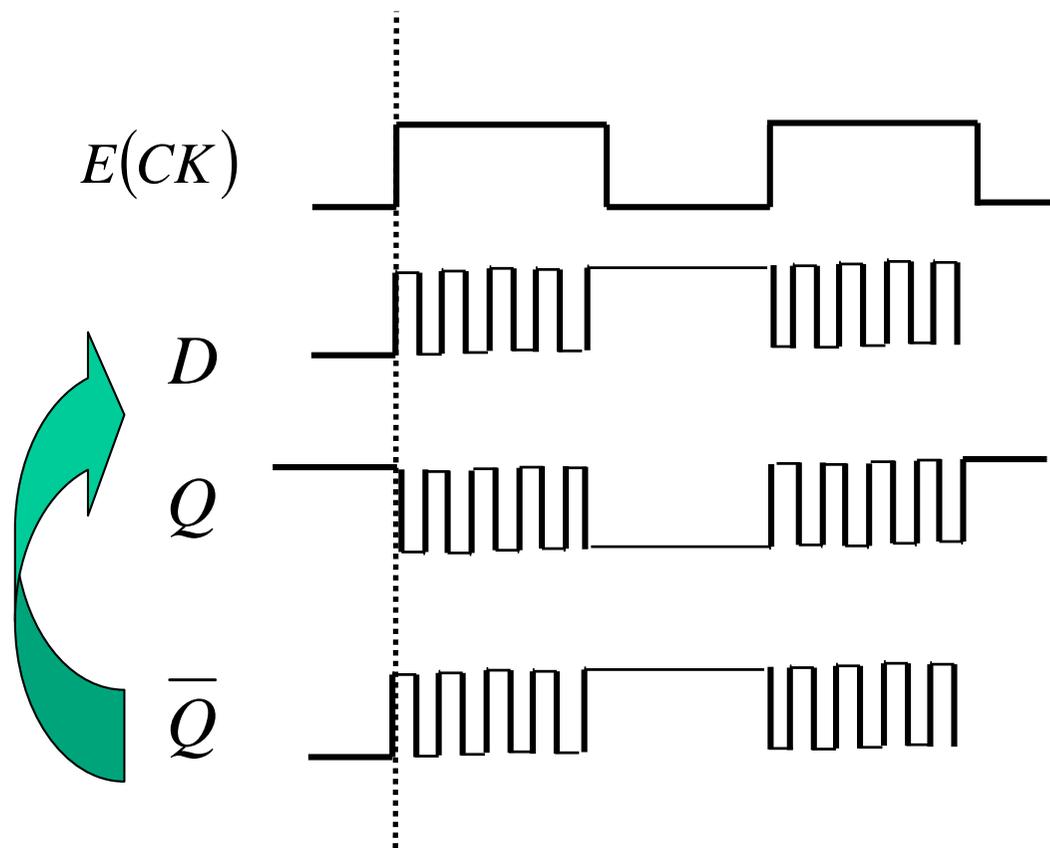
Proviamo infatti a modificare il nostro LATCH



Ma non funziona!!!
A causa della trasparenza

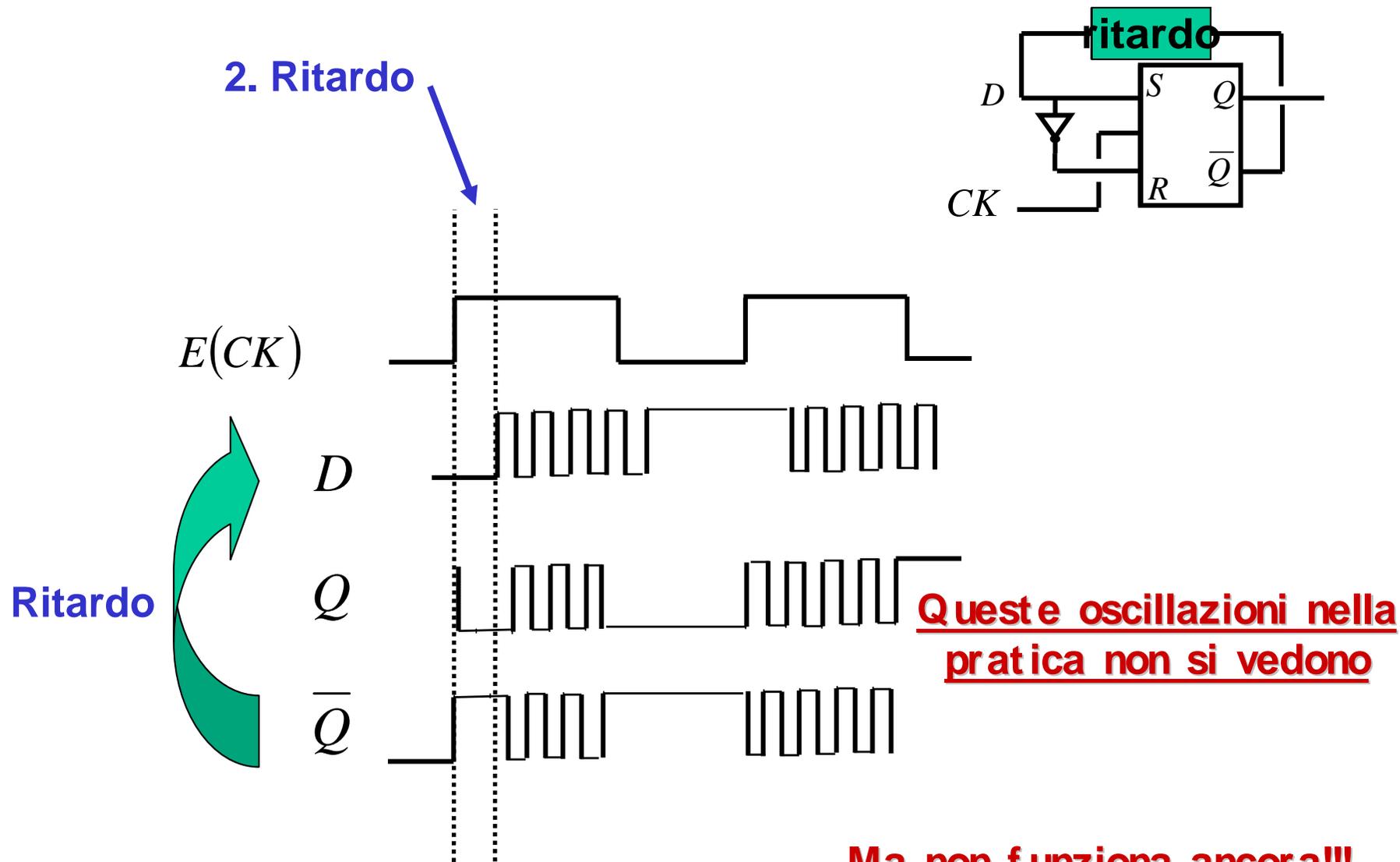


1. Ritardo nullo

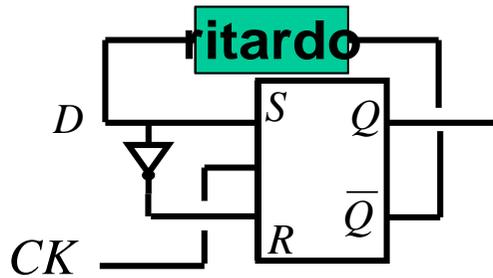


Queste oscillazioni nella pratica non si vedono

Ma non funziona!!!
A causa della trasparenza

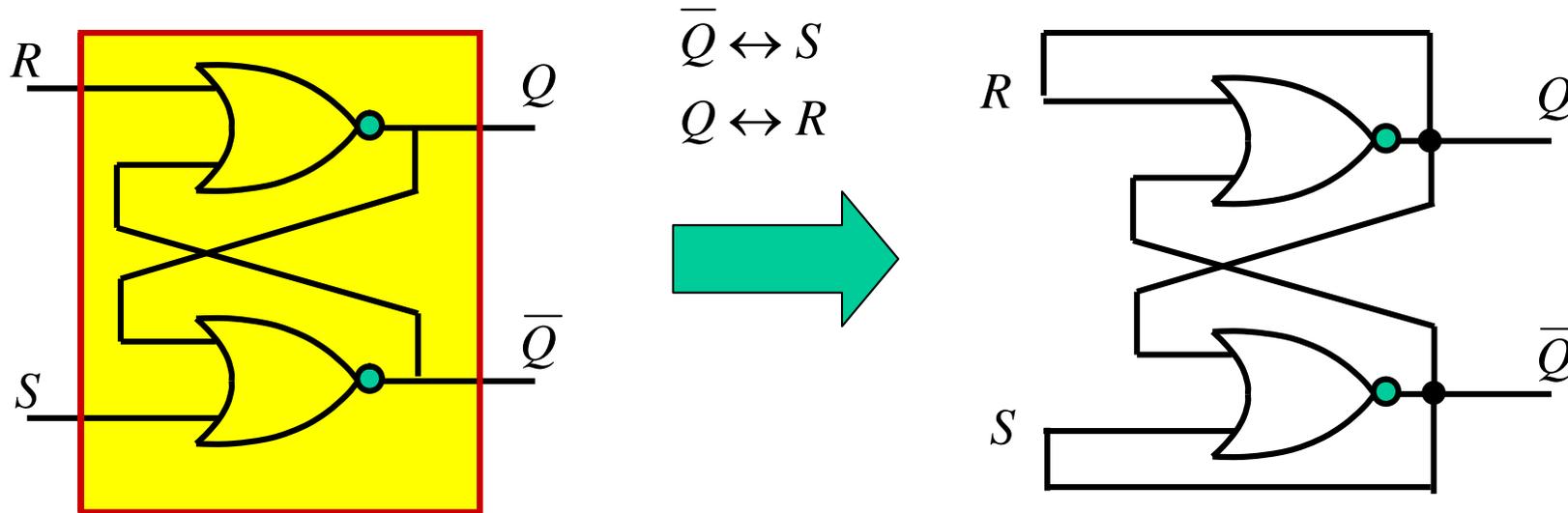


Per capire il fenomeno oscillazioni dobbiamo studiare i Tempi di propagazione dei segnali e di reazione delle porte

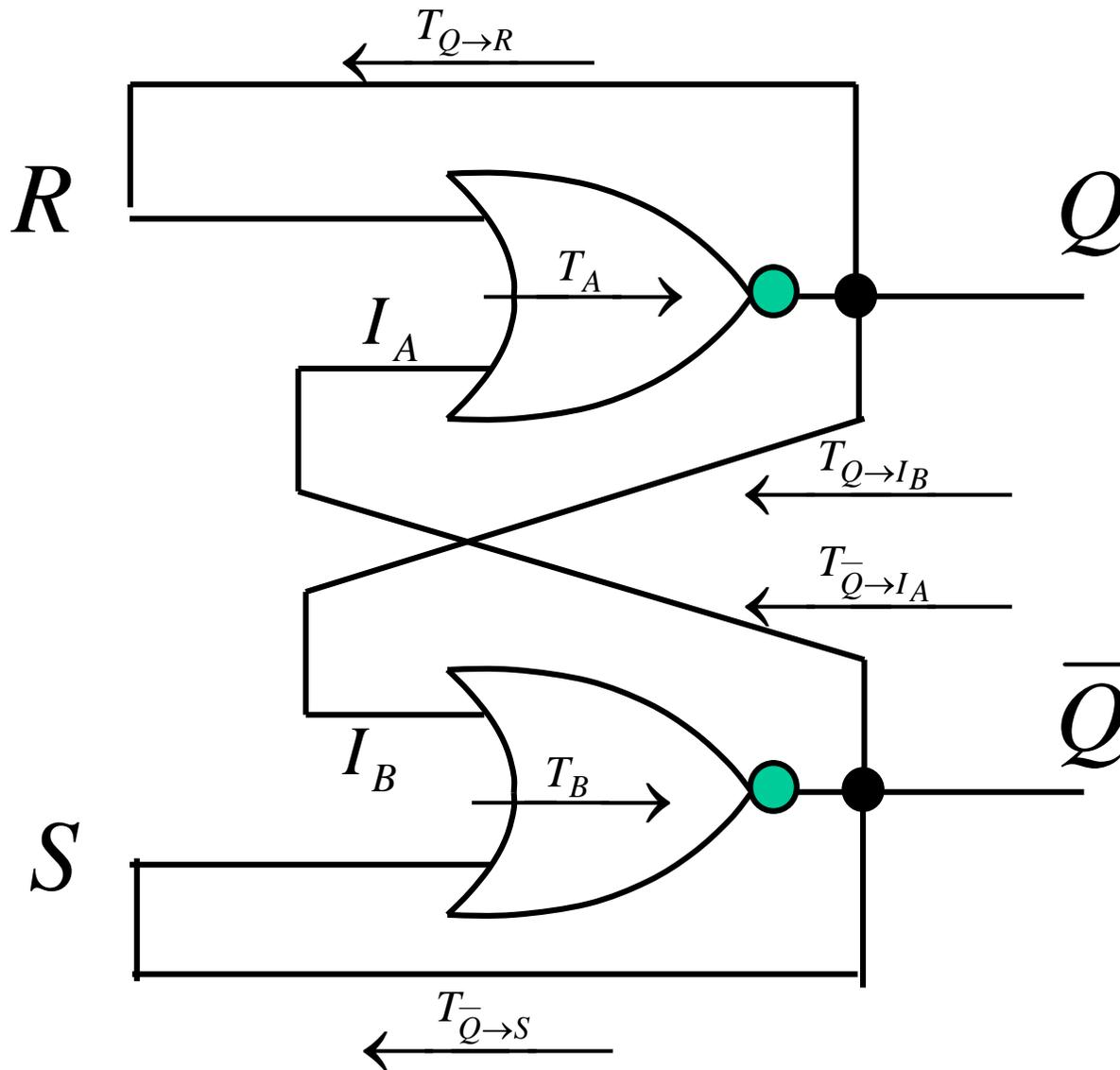


Siamo interessati al caso di CK=1
 → Usiamo un circuito più semplice

→ Prendiamo un FFSR senza CK e colleghiamo



Tempi di propagazione nelle connessioni e nelle porte

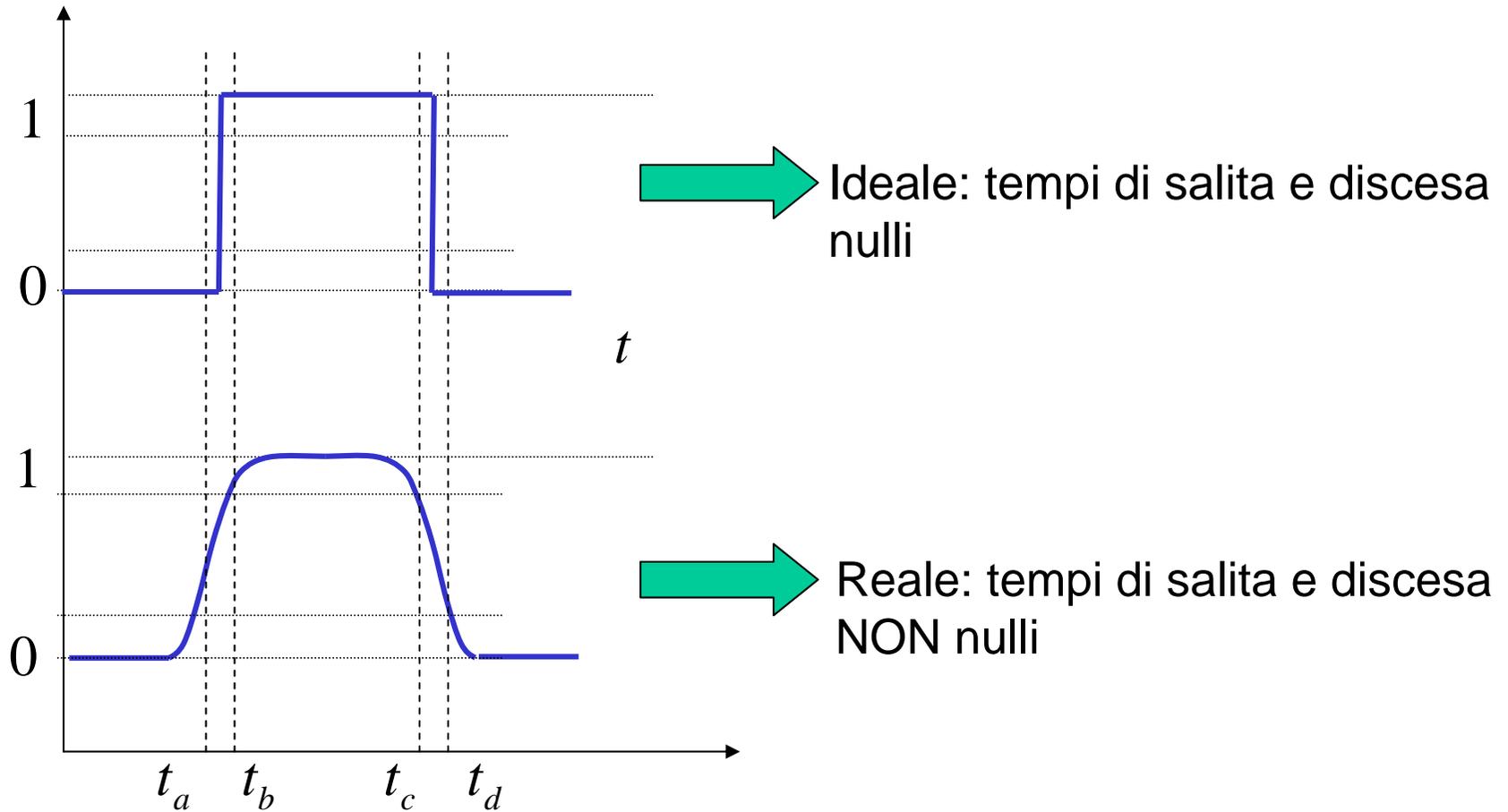


→ FF IDEALE
tempi uguali

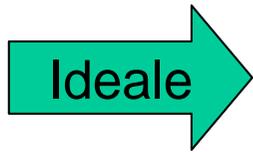
→ FF QUASI IDEALE
tempi simili

→ FF REALE
tempi diversi

Tempi di salita e di discesa dei segnali



Riepilogo sui tempi in gioco

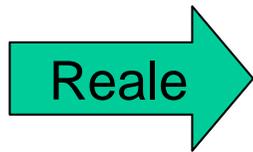
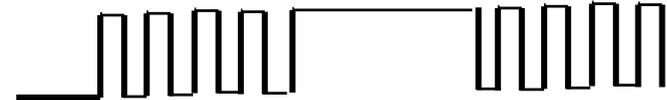


- Tempi di salita e discesa nulli
- FF IDEALE con tempi uguali

→ Si verifica il fenomeno di CORSA CRITICA

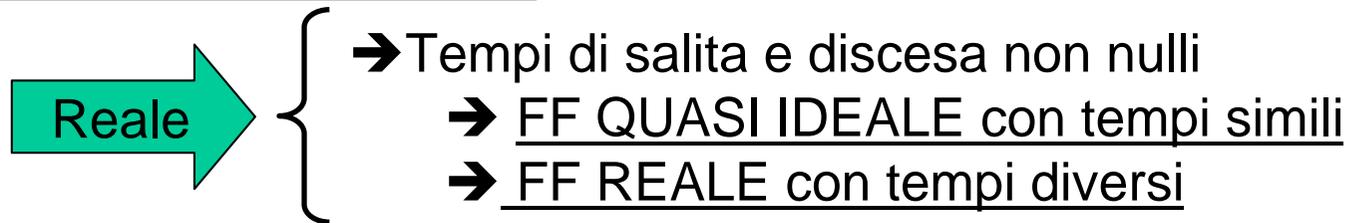
→ le uscite oscillano

→ la frequenza di oscillazione dipende dai ritardi interni *l'oscillazione è difficile da vedere*

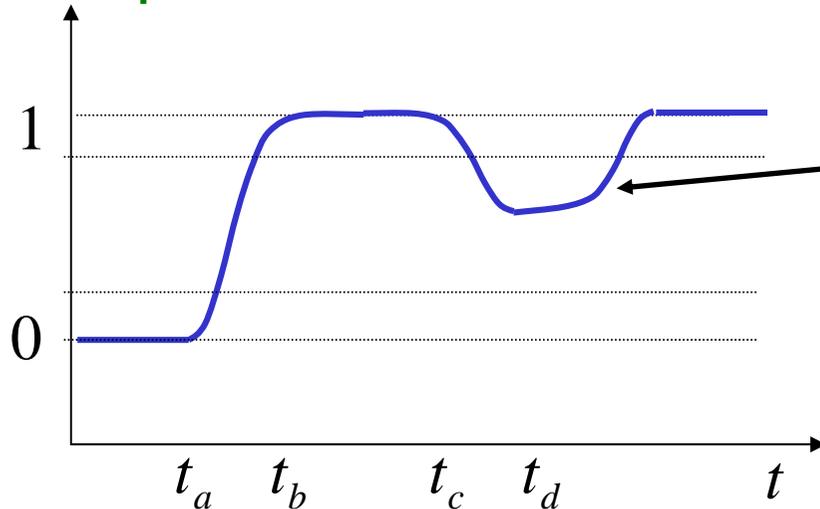


- Tempi di salita e discesa non nulli
- FF QUASI IDEALE con tempi simili
- FF REALE con tempi diversi

Riepilogo sui tempi in gioco



- Se FF è **QUASI** ideale (cioè tempi di propagazione molto simili):
lo stato delle uscite diventa **Metastabile** e dopo un certo tempo ritornerà random fra uno dei 2 possibili stati

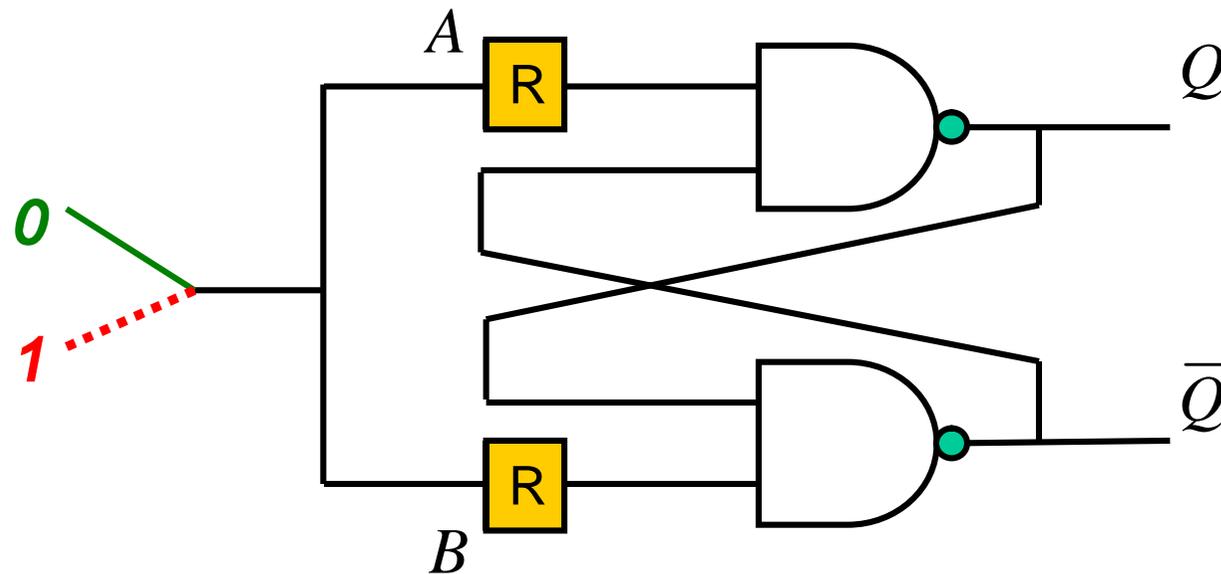


- ✓ **Uno stato metastabile e caratterizzato da un valore di tensione compreso fra il livello logico 0 e 1**
- ✓ **La probabilità che si verifichi uno stato metastabile decresce esponenzialmente con il tempo**

- Se FF è **REALE** (cioè tempi di propagazione diversi):
si ottiene sempre lo stesso stato delle uscite. Possiamo capire meglio questo con un altro esempio che introduciamo ora e studieremo in LAB.

Altro caso di comportamento anomalo

Simultanea attivazione e disattivazione degli ingressi



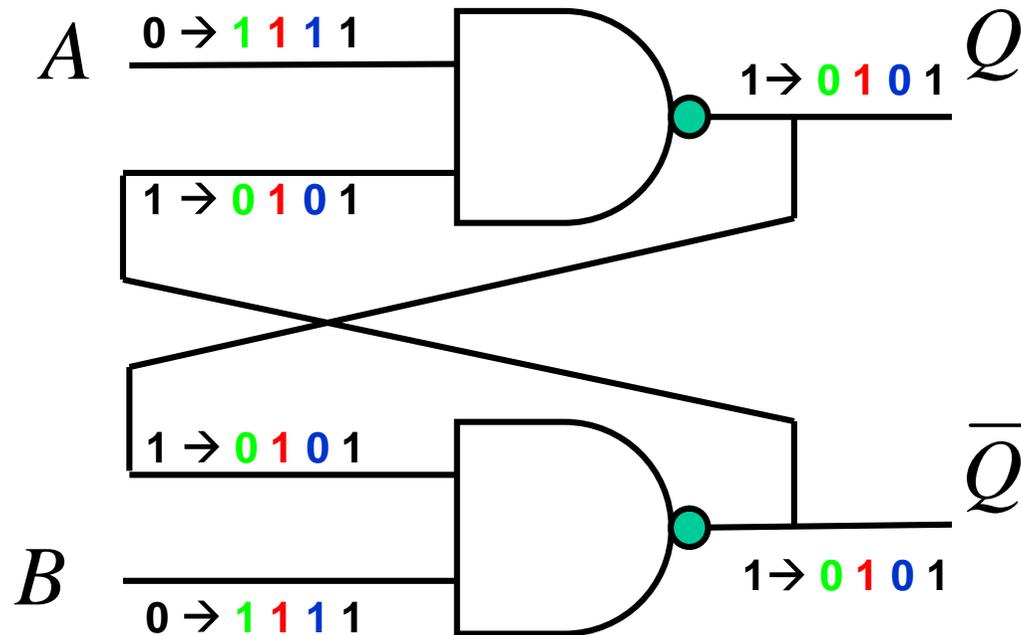
Comportamento 00→11 realizzato in lab:

→ dal lab vedremo risultati ambigui...

→ non appare chiaro cosa succede con 00→11

Simultanea 00→11: comportamento ideale *corsa critica*

FF ideale → ritardi delle 2 porte e delle retroazioni esattamente uguali



FF ideale → le uscite oscillano (00→11→00→ ...) (corsa critica)
→ la frequenza di oscillazione dipende dai ritardi interni
l'oscillazione è difficile da vedere

Simultanea 00→11: comportamento reale dal Lab *metastabilità*

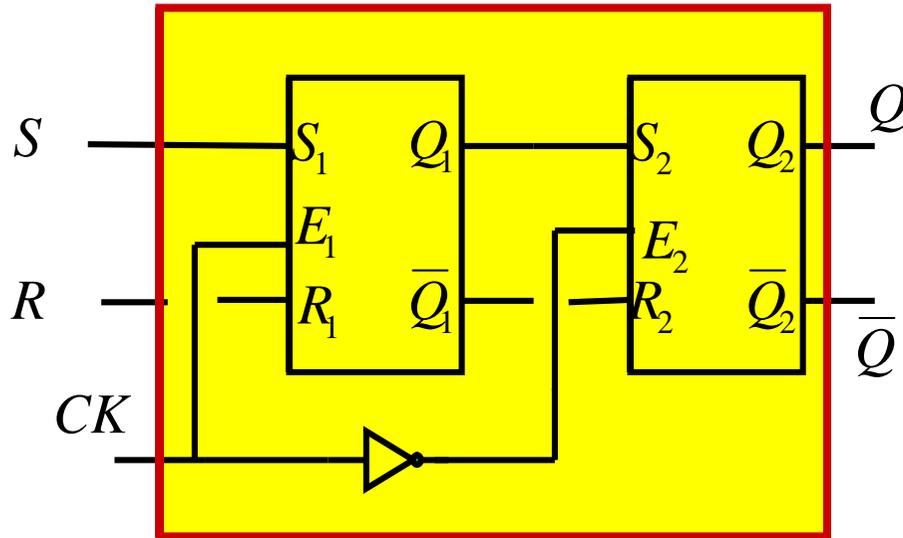
- Se FF è *QUASI* ideale (cioè tempi di propagazione molto simili):
lo stato delle uscite diventa **Metastabile** e dopo un certo tempo ritornerà random fra uno dei 2 possibili situazione che si verifica nei casi
- Se FF è *REALE* (cioè tempi di propagazione diversi):
si ottiene sempre lo stesso stato delle uscite

- ✓ *Uno stato metastabile è caratterizzato da un valore di tensione compreso fra il livello logico 0 e 1*
- ✓ *La probabilità che si verifichi uno stato metastabile decresce esponenzialmente con il tempo*

- ✓ *In laboratorio faremo alcune prove, poi interpreteremo i risultati ottenuti...*

Ritorniamo al FF-Toogle, dobbiamo risolvere il problema trasparenza

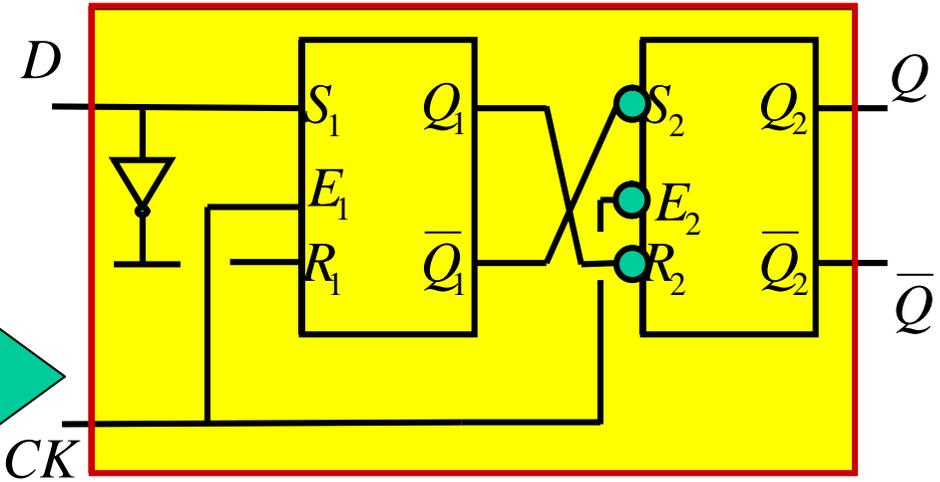
Tipica struttura non trasparente: **Master-Slave**



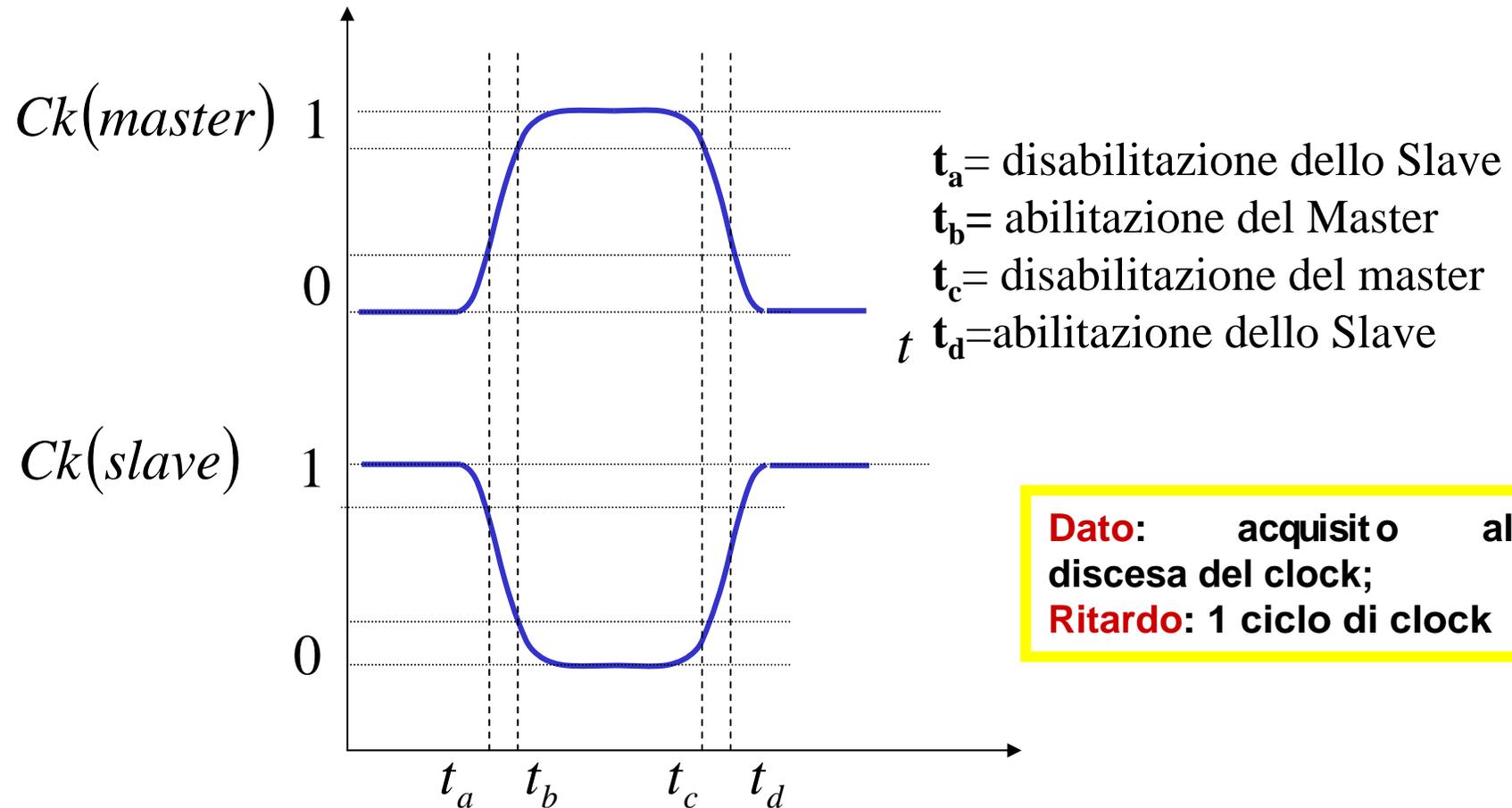
Struttura MS con un latch (SR) e un nonset- nonreset

struttura MS con due FF SR

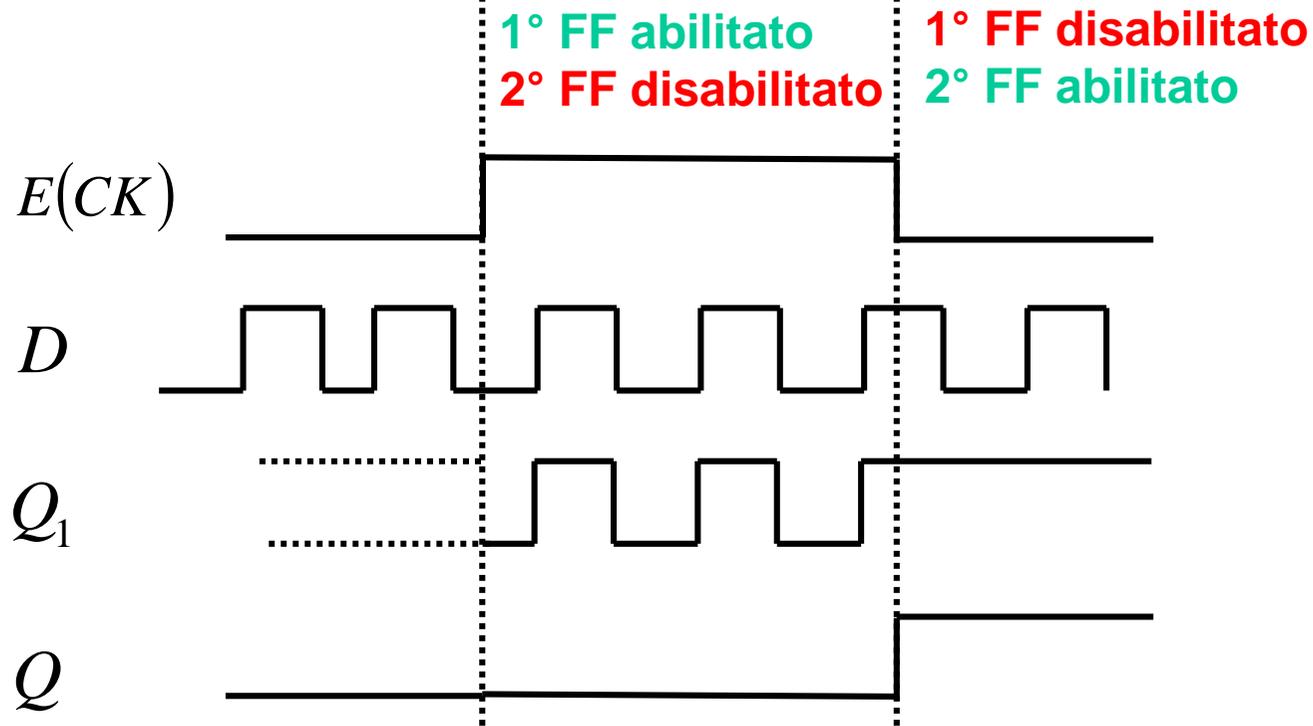
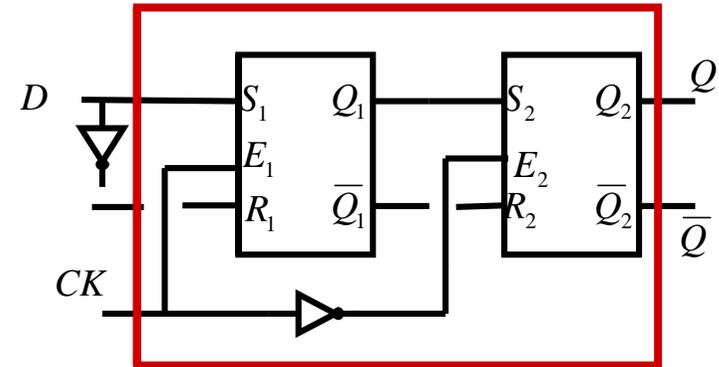
FF tipo D(elayed)



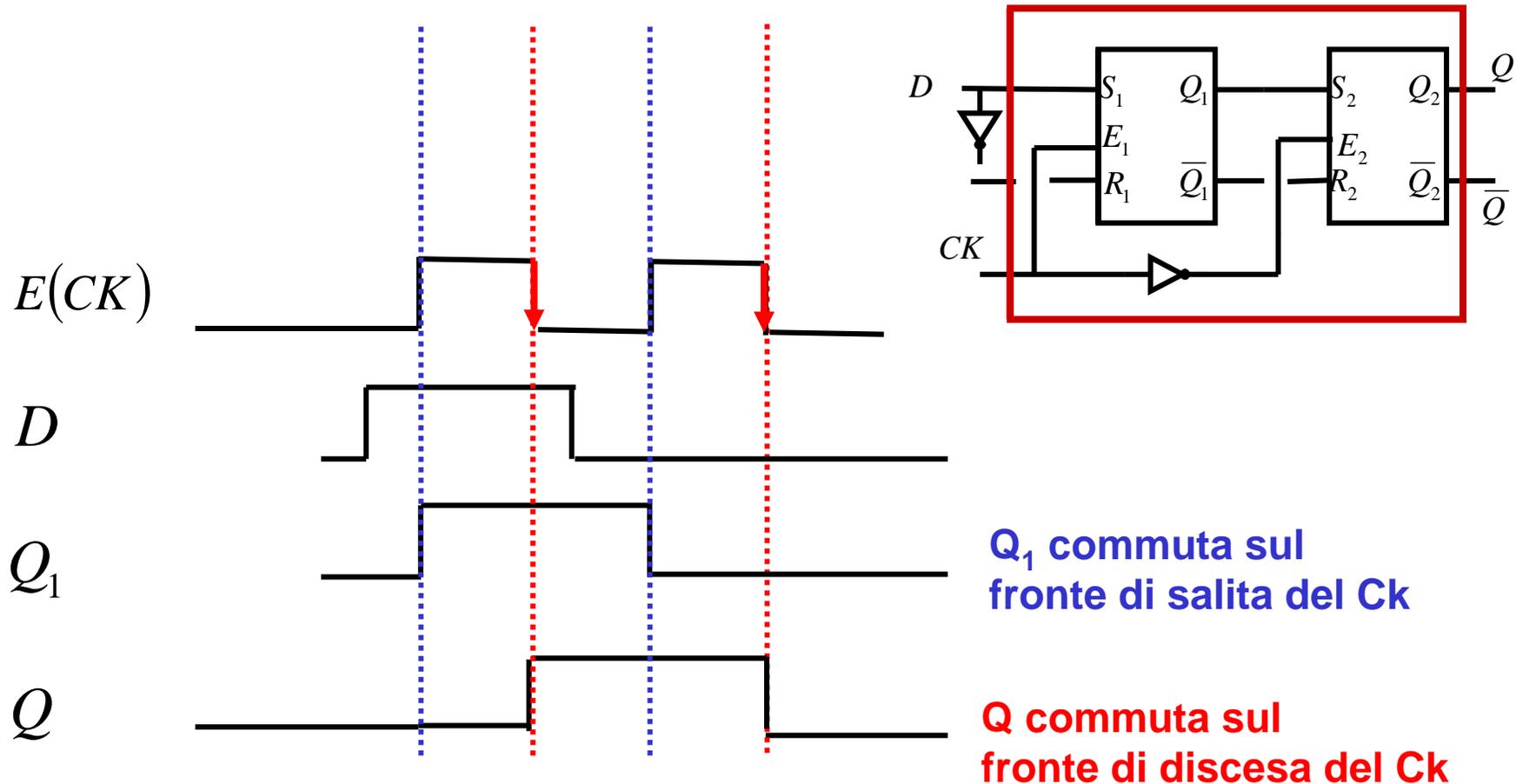
La struttura non trasparente risolve il problema del tempo di propagazione...



Verifichiamo la NON-Trasparenza del Master-Slave

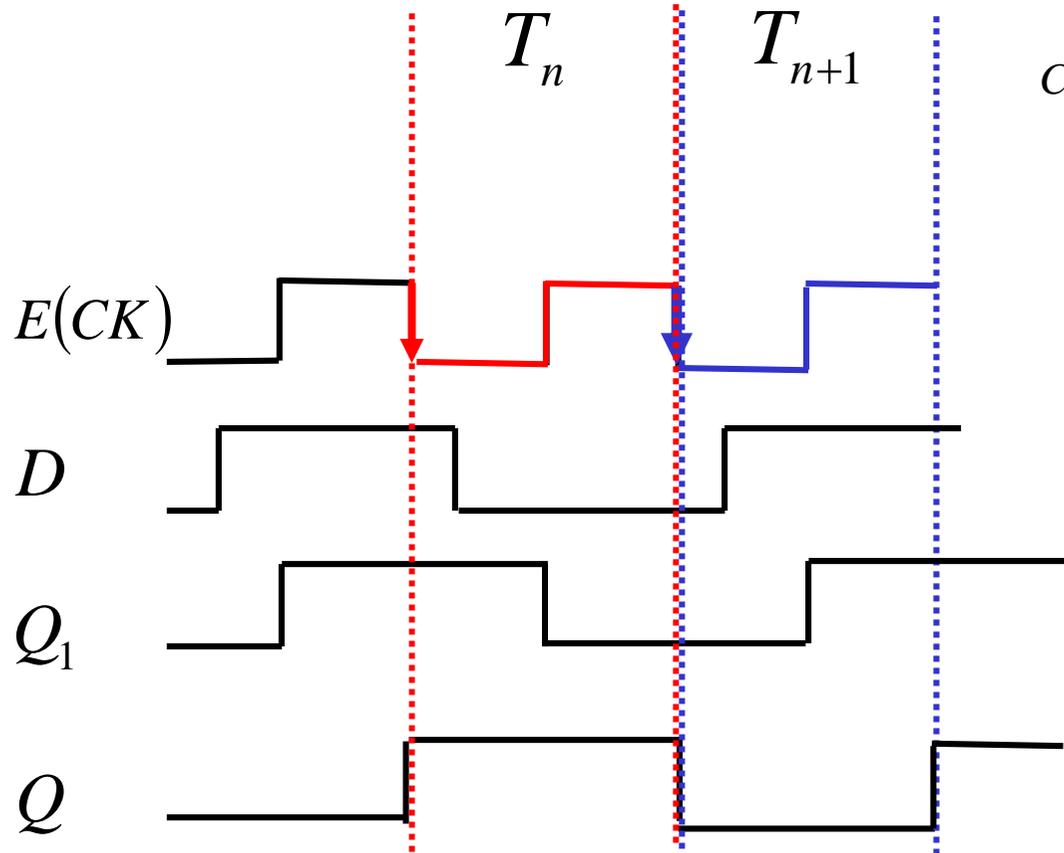
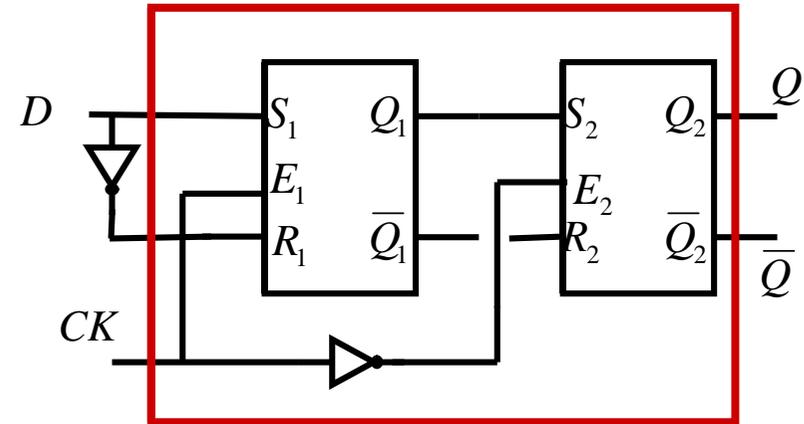


Master-Slave come memoria non trasparente



Usando un Ck negato si invertono le commutazioni sui fronti di salita e discesa

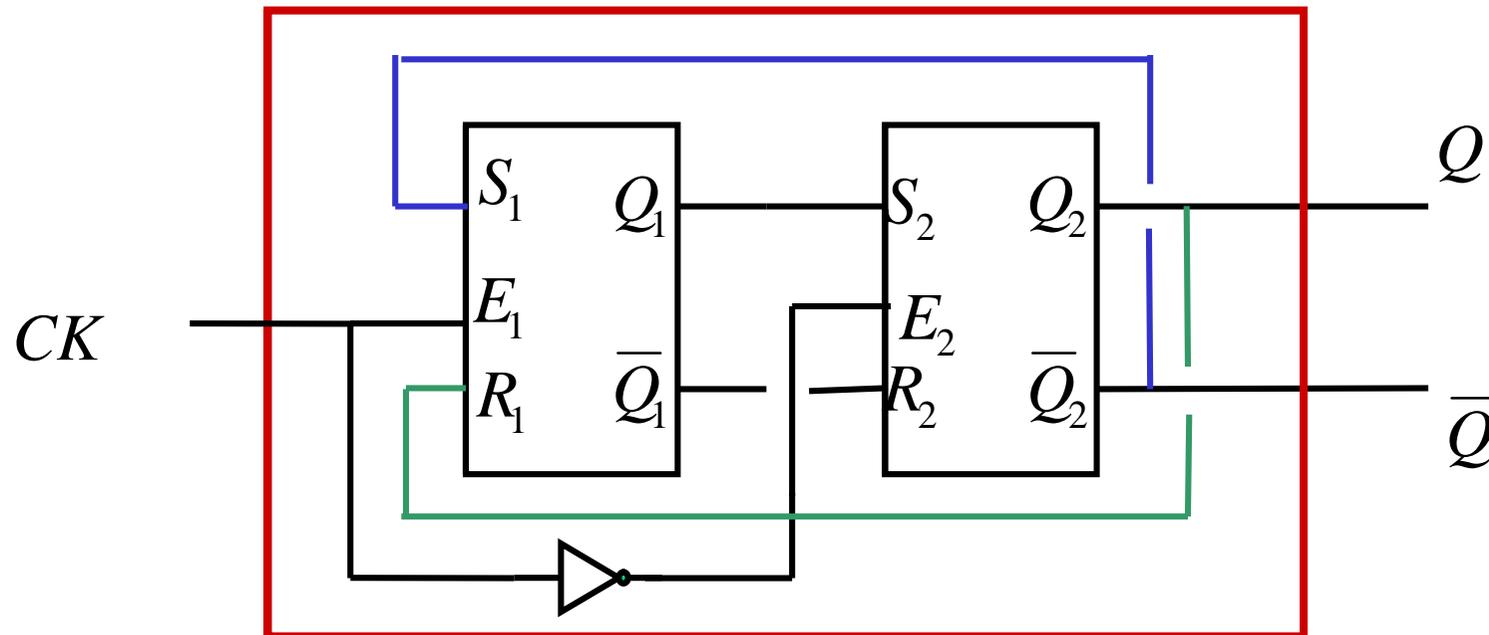
FF Master-Slave tipo-D (Delay)



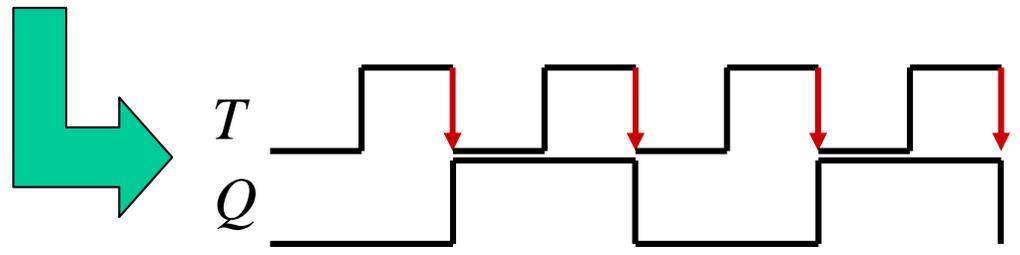
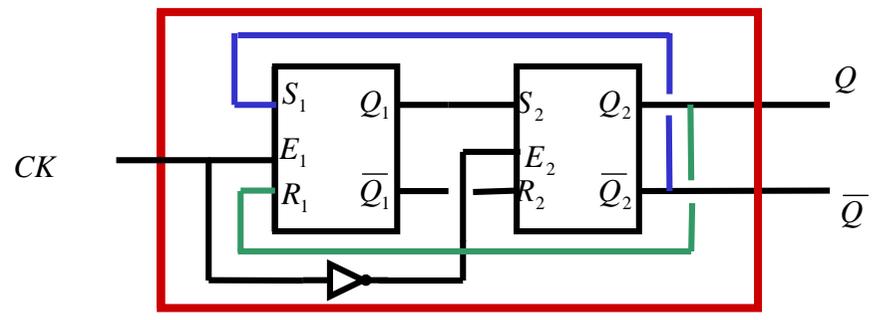
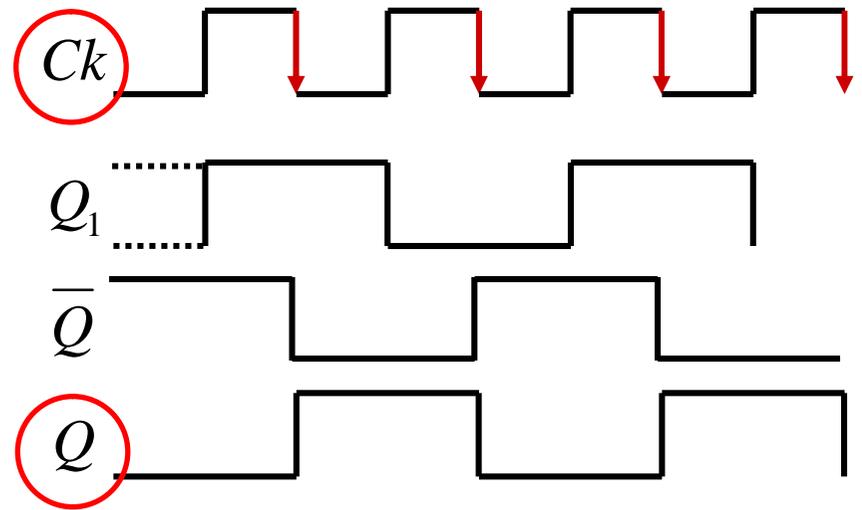
Al tempo T_{n+1} Q assume il valore assunto da D al tempo T_n

FF Master-Slave tipo-T (Toggle)

- Retroazione: $\bar{Q} \rightarrow S_1$
- Retroazione: $Q \rightarrow R_1$

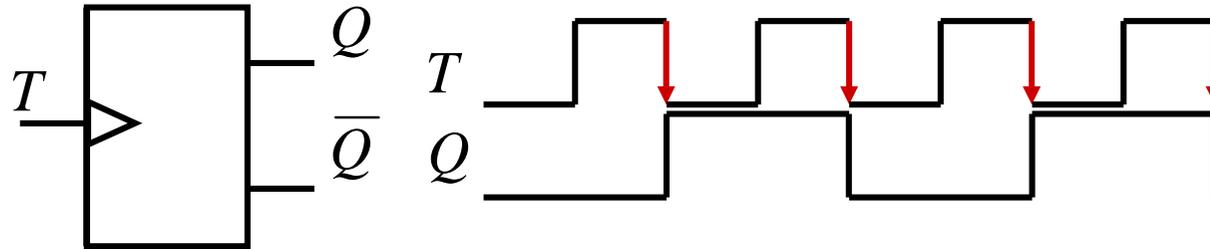


FF Master-Slave tipo-T (Toggle)



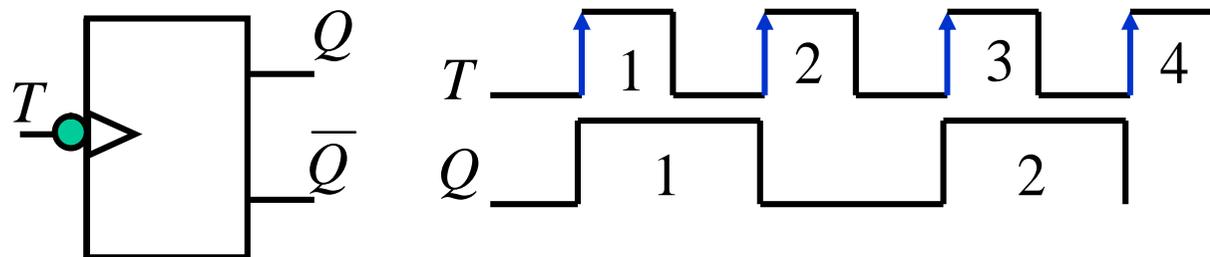
è un divisore per 2

Abbiamo realizzato un **FF Toggle** basandoci sulla retroazione ingresso- uscita



T	Q_n	Q_{n+1}
0	0	0
1	1	1
↓	0	1
↓	1	0

E' un divisore per due che commuta sul fronte:

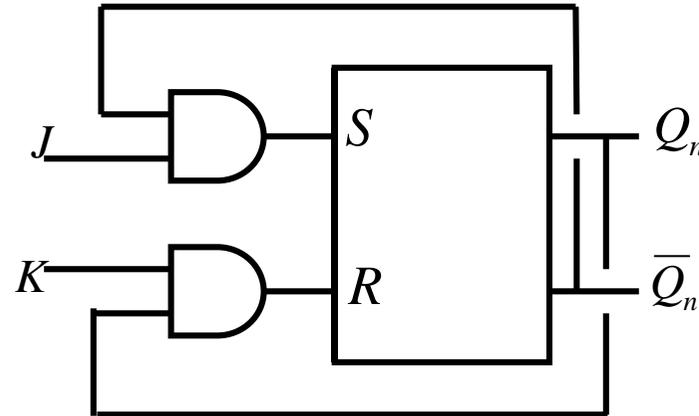
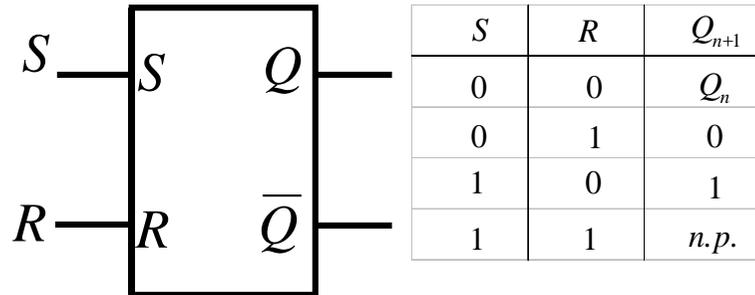


T	Q_n	Q_{n+1}
0	0	0
1	1	1
↑	0	1
↑	1	0

FF tipo J-K (*trasparente*) – Tavola della verità

Risolve il problema della configurazione non permessa del FF-SR

Aggiungiamo 2 porte AND ad un FF SR



J	K	$S = J\overline{Q_n}$	$R = KQ_n$	Q_{n+1}
0	0	0	0	Q_n
0	1	0	Q_n	0 _{nota1}
1	0	$\overline{Q_n}$	0	1 _{nota2}
1	1	Q_n	Q_n	$\overline{Q_n}$: <i>Toogle</i> _{nota3}

nota1: se $Q_n=0$ non c'è commutazione;
se $Q_n=1$ si ha un “**RESET**”

nota2: se $Q_n=1$ non c'è commutazione;
se è $Q_n=0$ si ha un “**SET**”

nota3: c'è il problema della trasparenza
che sappiamo risolvere

FF tipo J-K (trasparente) – analizziamo le note

J	K	$S = J\overline{Q_n}$	$R = KQ_n$	Q_{n+1}
0	0	0	0	Q_n
0	1	0	Q_n	0 _{nota1}
1	0	$\overline{Q_n}$	0	1 _{nota2}
1	1	$\overline{Q_n}$	Q_n	$\overline{Q_n} : Toogle$ _{nota3}

nota1: se $Q_n=0$ non c'è commutazione;
se $Q_n=1$ si ha un “**RESET**”

nota2: se $Q_n=1$ non c'è commutazione;
se è $Q_n=0$ si ha un “**SET**”

nota3: c'è il problema della trasparenza
che sappiamo risolvere

nota1: se $Q_n=0$ non c'è commutazione; se $Q_n=1$ si ha un “**RESET**”

S	R	Q_{n+1}	stato
0	$Q_n = 0$	$Q_n = 0$	memorizza 0
0	$Q_n = 1$	0	resetta a 0

 \Rightarrow

J	K	Q_{n+1}
0	1	0

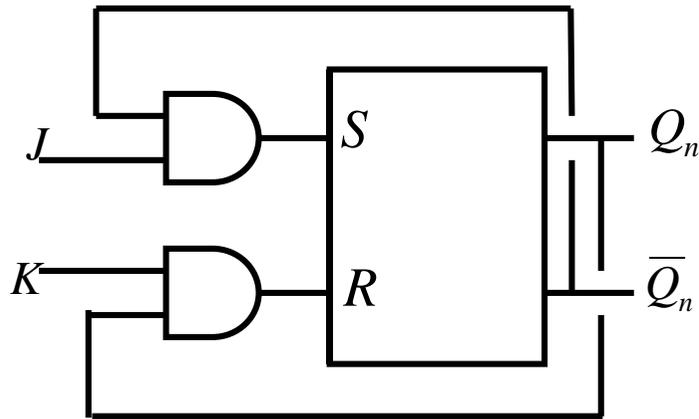
nota2: se $Q_n=1$ non c'è commutazione; se è $Q_n=0$ si ha un “**SET**”

S	R	Q_{n+1}	stato
$\overline{Q_n} = 0$	0	$Q_n = 1$	memorizza 1
$\overline{Q_n} = 1$	0	1	setta a 1

 \Rightarrow

J	K	Q_{n+1}
1	0	1

FF tipo J-K (*trasparente*) – riepilogo

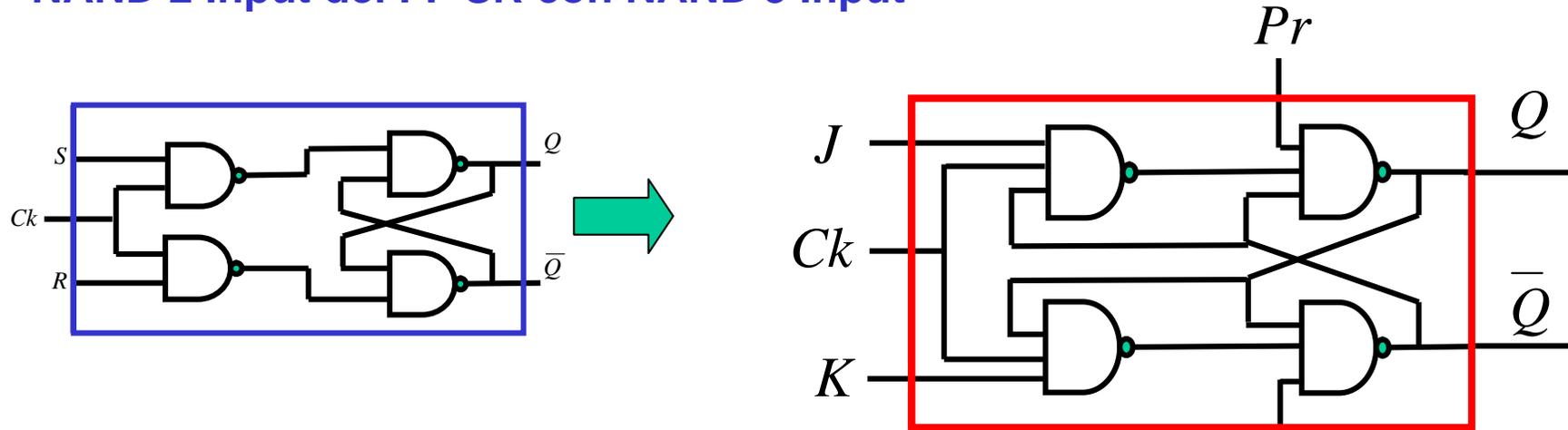


J	K	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	$\overline{Q_n}$

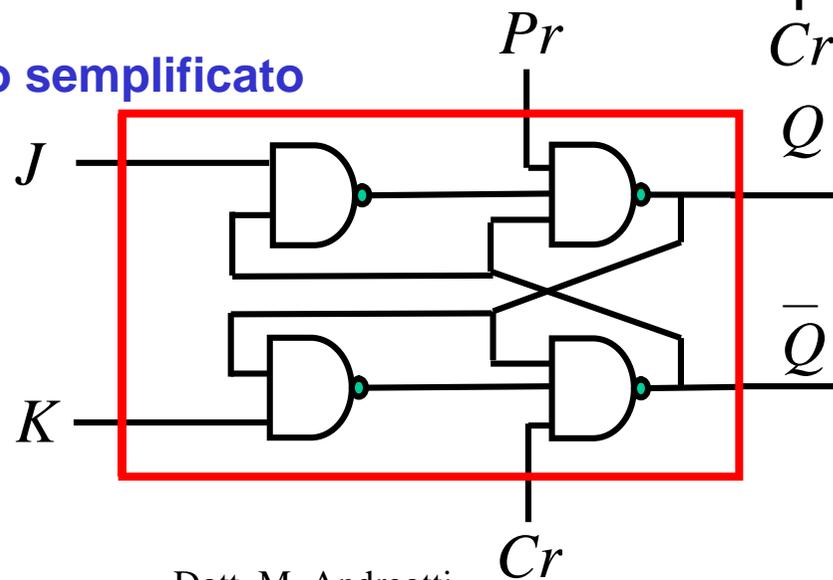
Il problema della trasparenza si risolve usando un FF-SR-MS

FF tipo J-K (*trasparente*) con *Preset* e *Clear*

Prendiamo un FF-SR con NAND (con clock) e sostituiamo NAND 2 Input del FF SR con NAND 3 Input

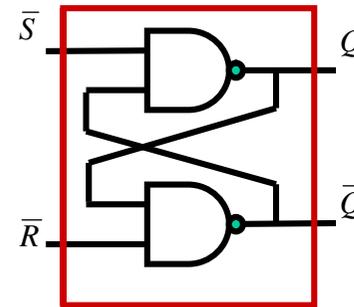
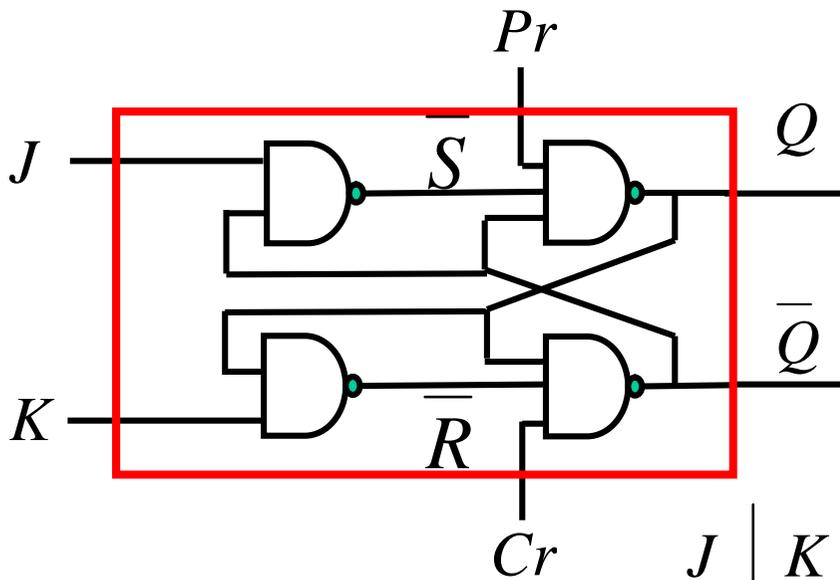


Consideriamo $Ck=1$,
quindi studiamo il circuito semplificato



FF-JK *Tavola della verità*

- FF JK ha due variabili di ingresso in più: *Pr* e *Cr*.
temporaneamente consideriamo Pr=Cr=1 in modo da abilitare le porte NAND

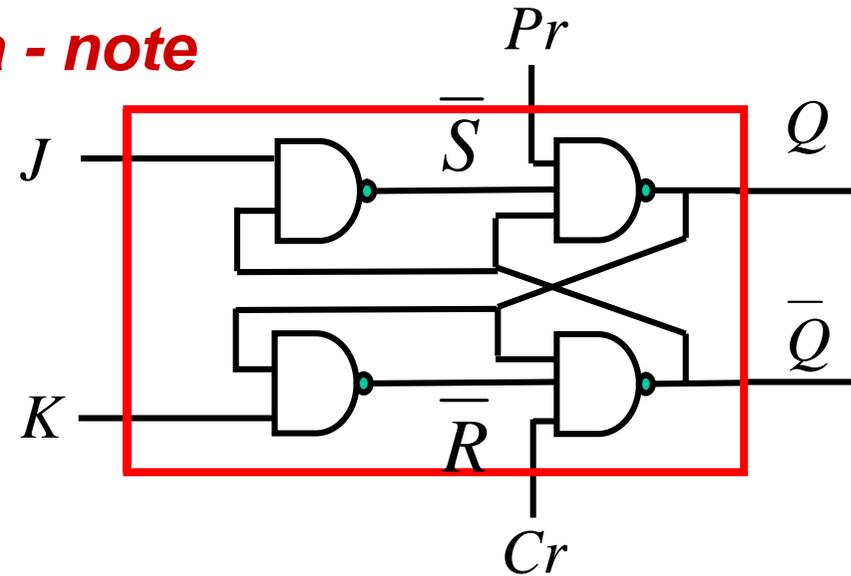


\bar{S}	\bar{R}	Q_{n+1}
1	1	Q_n
0	1	1
1	0	0
0	0	<i>n.p.</i>

J	K	$\bar{S} = \overline{JQ_n}$	$\bar{R} = \overline{KQ_n}$	Q_{n+1}
0	0	1	1	Q_n
0	1	1	$\overline{Q_n}$	0 _{nota1}
1	0	Q_n	1	1 _{nota2}
1	1	Q_n	$\overline{Q_n}$	$\overline{Q_n} : Toogle$ _{nota3}

FF-JK Tavola della verità - note

J	K	$\bar{S} = \overline{JQ_n}$	$\bar{R} = \overline{KQ_n}$	Q_{n+1}
0	0	1	1	Q_n
0	1	1	$\overline{Q_n}$	0 _{nota1}
1	0	Q_n	1	1 _{nota2}
1	1	Q_n	$\overline{Q_n}$	$\overline{Q_n}$: Toogle _{nota3}



nota1: se $Q_n = 0 \Rightarrow \bar{Q}_n = 1$ non c'è commutazione; se $Q_n = 1 \Rightarrow \bar{Q}_n = 0$ si ha un **“RESET”**

\bar{S}	\bar{R}	Q_{n+1}	stato
1	$\overline{Q_n} = 1$	$Q_n = 0$	memorizza 0 \Rightarrow
1	$\overline{Q_n} = 0$	0	resetta a 0

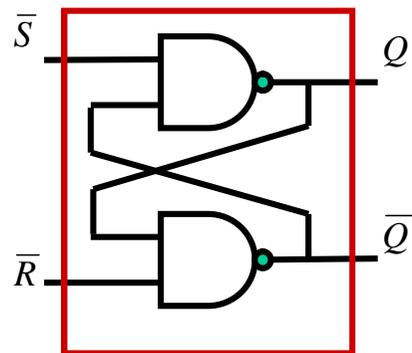
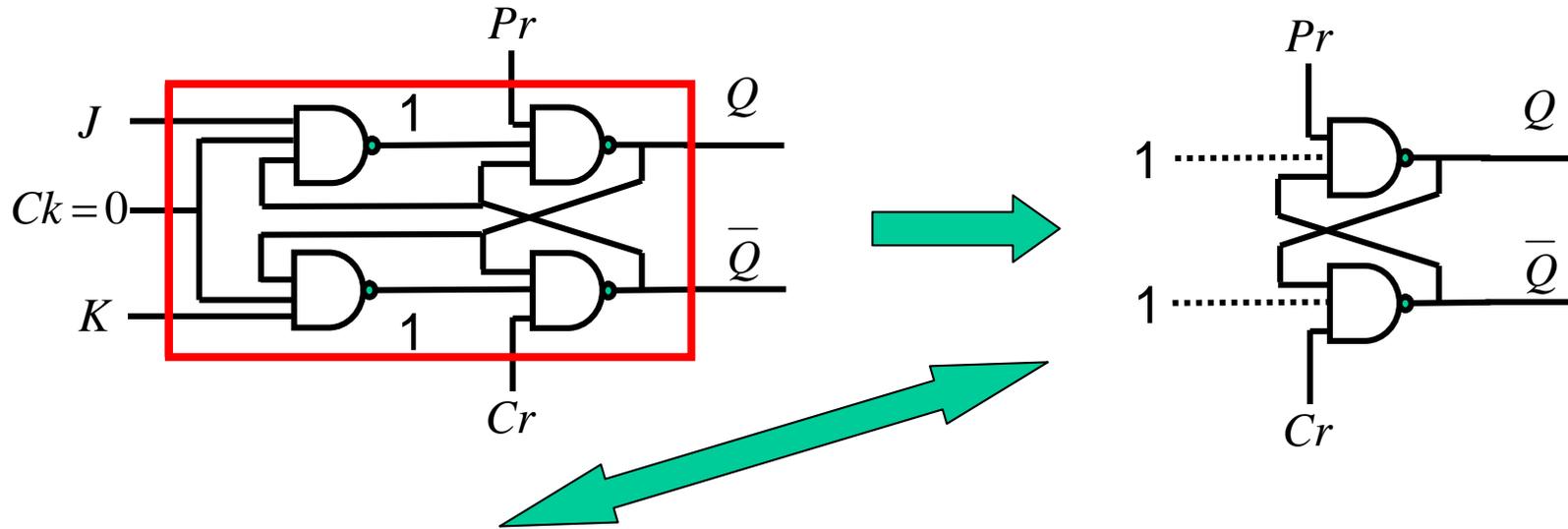
J	K	Q_{n+1}
0	1	0

nota2: se $Q_n = 1$ non c'è commutazione; se è $Q_n = 0$ si ha un **“SET”**

\bar{S}	\bar{R}	Q_{n+1}	stato
$Q_n = 1$	1	$Q_n = 1$	memorizza 1 \Rightarrow
$Q_n = 0$	1	1	setta a 1

J	K	Q_{n+1}
1	0	1

FF-JK Variabili Pr e Cr e Ck=0



\bar{S}	\bar{R}	Q_{n+1}	Pr	Cr	Q_{n+1}
1	1	Q_n	1	1	Q_n
0	1	1	0	1	1
1	0	0	1	0	0
0	0	<i>n.p.</i>	0	0	<i>n.p.</i>

$Pr = \bar{S}; Cr = \bar{R}$

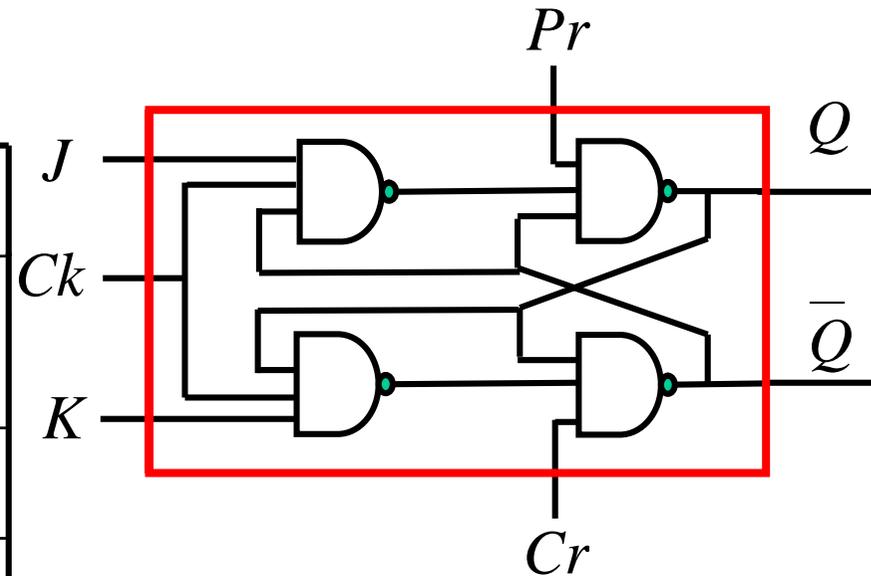
FF-JK Variabili sincrone e asincrone

- **J e K sono variabili sincrone:** modificano l'uscita Q solo quando il Ck abilita il FF
- **Pr e Cr sono variabili asincrone:** modificano l'uscita Q indipendentemente dallo stato del Ck
per Ck = 0 o 1 Pr e Cr agiscono in modi diversi

Pr = Preset (preassegnazione)

Cr = Clear (azzeramento)

Ck	Cr	Pr	Q	
1	1	1	tavola della verità	Abilitazione
0	0	1	0	Azzeramento
0	1	0	1	preassegnazione



~~Pr = Cr = 0~~

Ck=1 Pr,Cr variabili influenzano Q in modi diversi dalla Tab

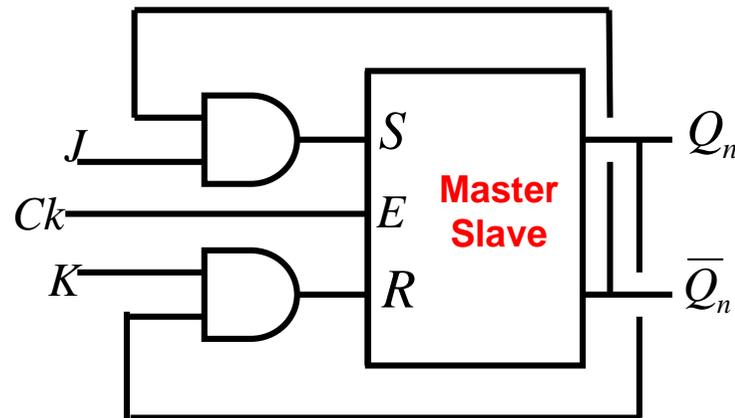
FF tipo J-K *Master-Slave*

- Il FF-JK descritto fin qui è trasparente:
presenta lo stesso problema incontrato prima



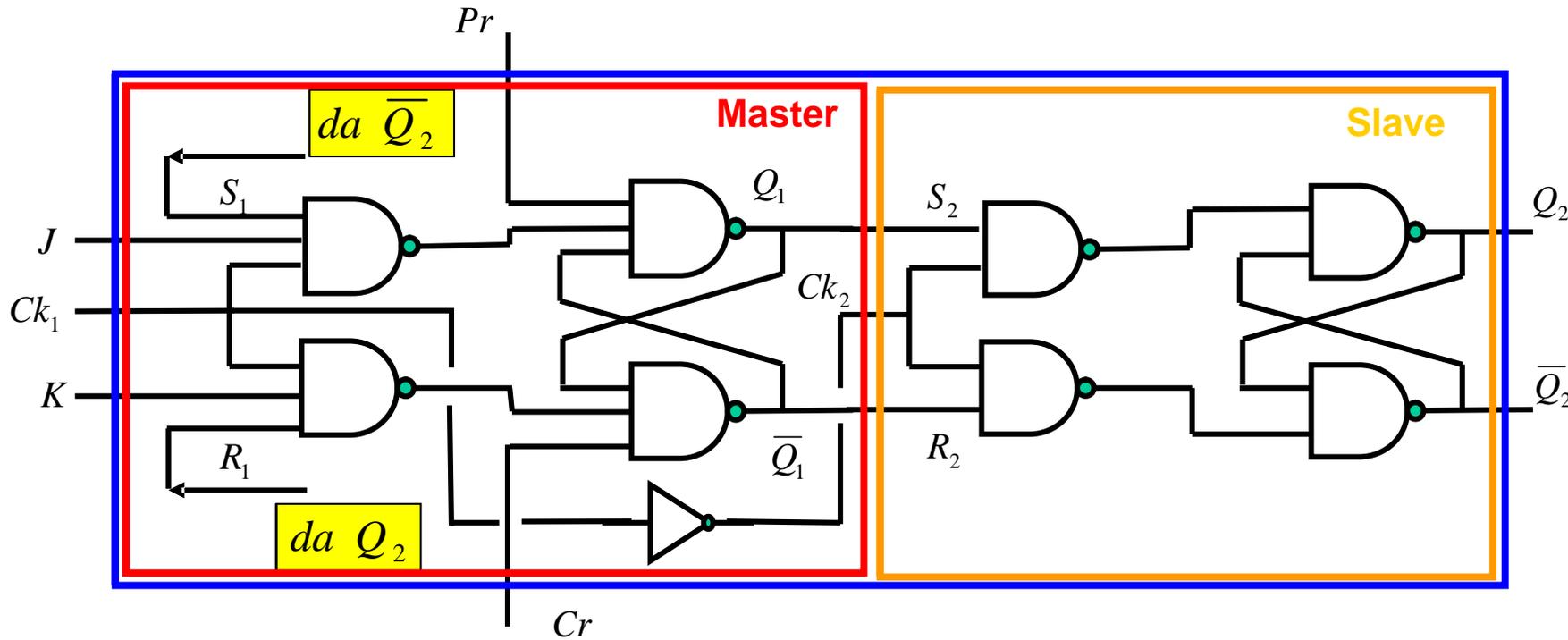
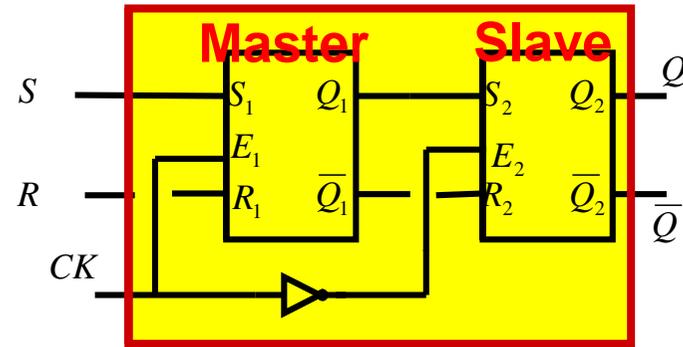
- Realizziamo il FF-JK con struttura MS

1° modo: analogamente a prima aggiungiamo 2 porte AND ad un FF SR con struttura MS



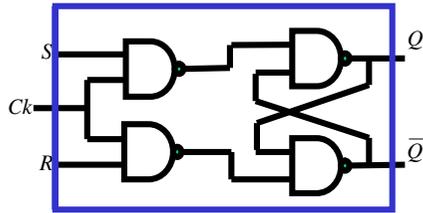
FF tipo J-K Master-Slave

2° modo: - analogamente a prima sostituiamo
 le NAND 2 Input del Master
 con NAND 3 Input
 - aggiungiamo le opportune
 retroazioni

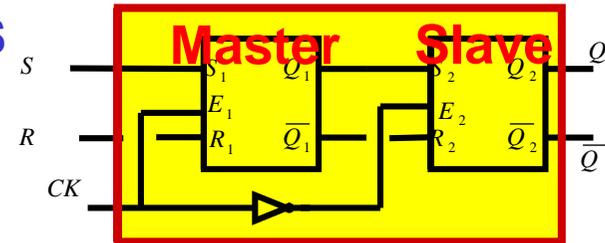


Riepilogo Flip-Flop

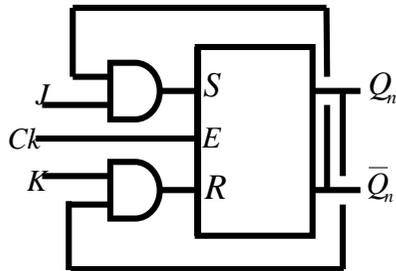
1) FF - SR



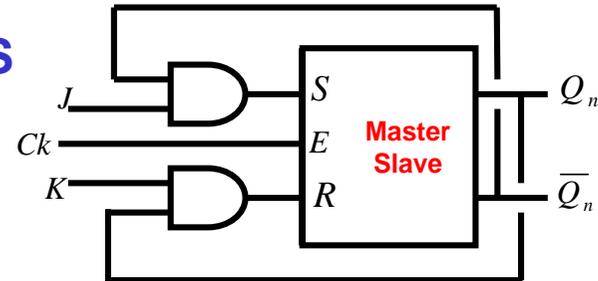
2) FF - SR - MS



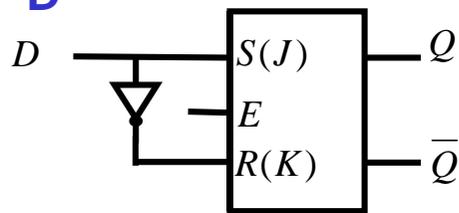
3) FF - JK



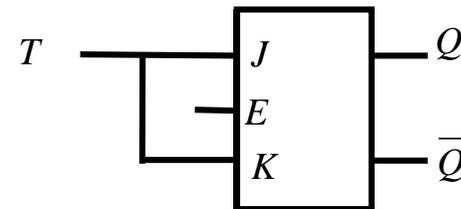
4) FF - JK - MS



5) FF - D

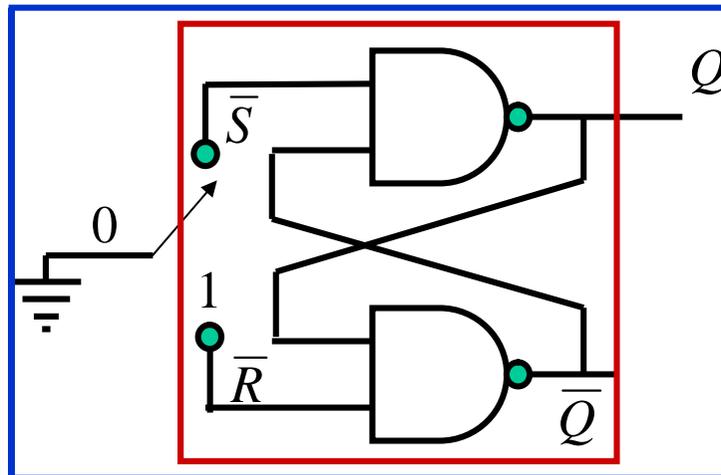


6) FF - T



Flip-Flop come interruttore antirimbalo

- quando si aziona un interruttore meccanico, prima che la variabile in uscita assuma il valore stabile definito deve subire un certo numero di oscillazioni
- il FF è un interruttore antirimbalo
 - infatti memorizza lo stato stabile che deve assumere



Commutazione dei Flip-Flop (*saremo vaghi... → LAB*)

- I FF commutano sul fronte di discesa (o salita) di un clock:
 - commutano seguendo le configurazioni degli ingressi e/o uscite (vedi collegamenti di retroazione)
 - durante le commutazioni gli ingressi devono essere stabili altrimenti si possono verificare delle irregolarità
- ad esempio l'utilizzo di uno switch normale anziché di uno antirimbalzo può portare a irregolarità

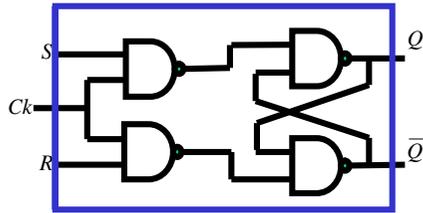
..... ma per saperne di più aspettiamo il laboratorio

Applicazione dei Flip-Flop

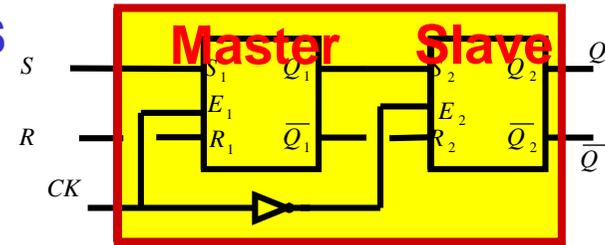
- Contatori
- Registri a scorrimento

Riepilogo Flip-Flop

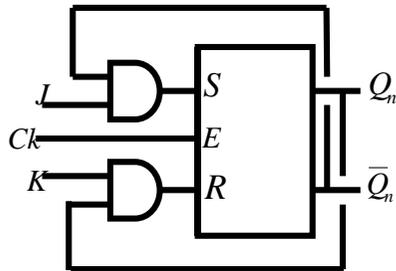
1) FF - SR



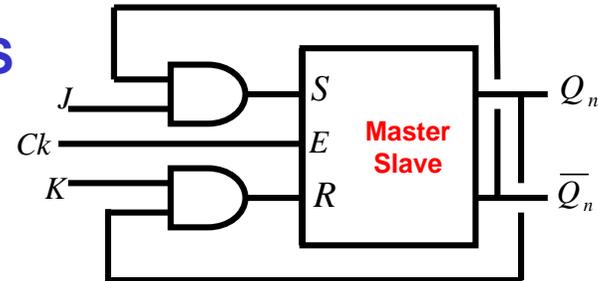
2) FF - SR - MS



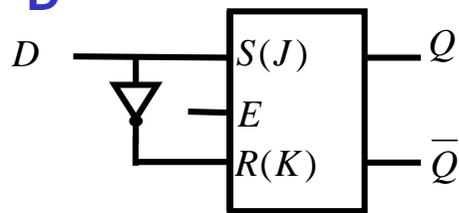
3) FF - JK



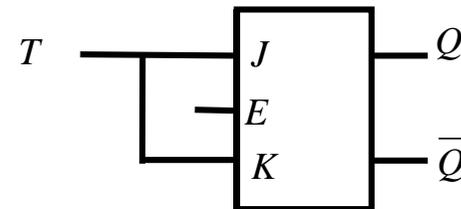
4) FF - JK - MS



5) FF - D

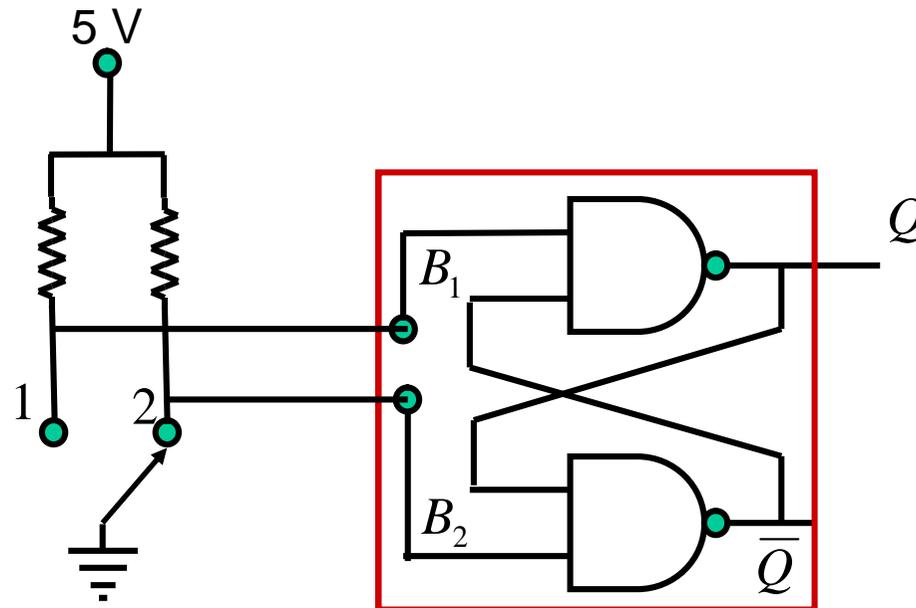


6) FF - T



Flip-Flop come interruttore antirimbalo

- quando si aziona un interruttore meccanico, prima che la variabile in uscita assuma il valore stabile definito deve subire un certo numero di oscillazioni
- il FF è un interruttore antirimbalo
 - infatti memorizza lo stato stabile che deve assumere



Commutazione dei Flip-Flop (*saremo vaghi... → LAB*)

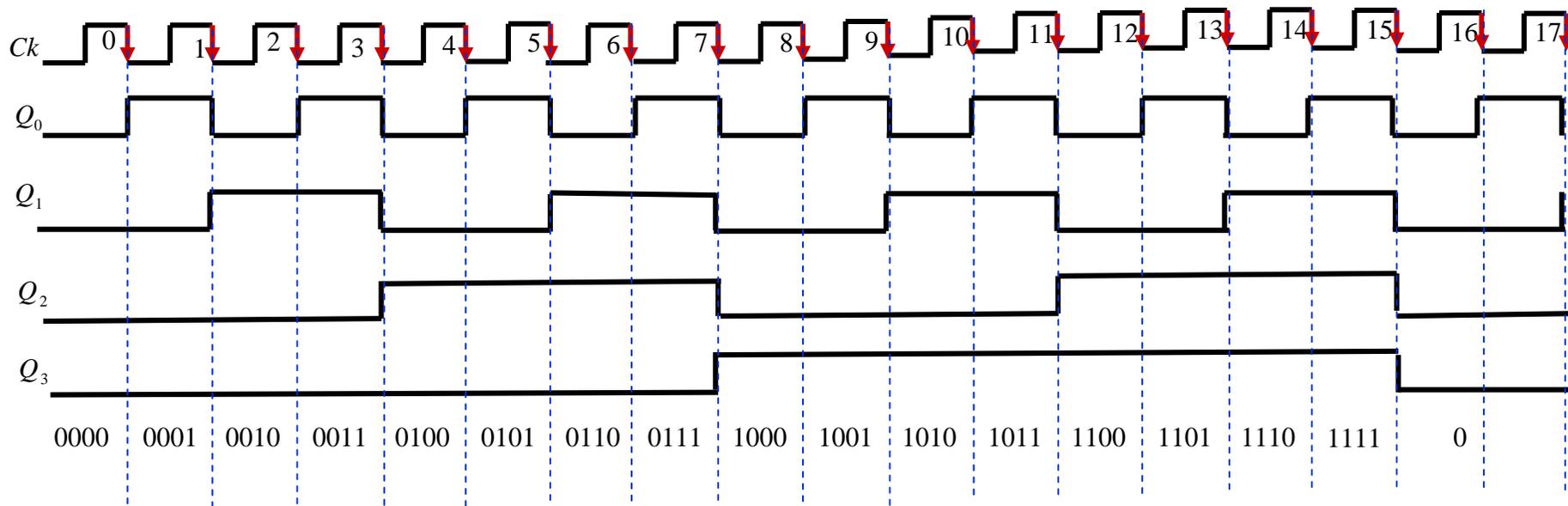
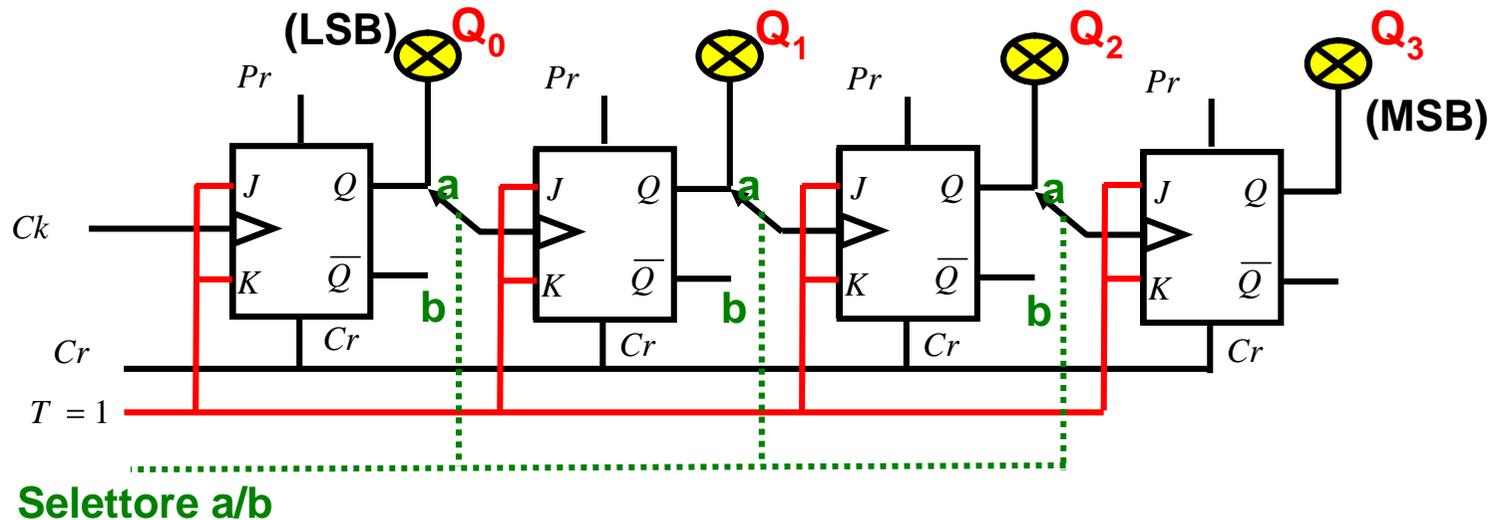
- I FF commutano sul fronte di discesa (o salita) di un clock:
 - commutano seguendo le configurazioni degli ingressi e/o uscite (vedi collegamenti di retroazione)
 - durante le commutazioni gli ingressi devono essere stabili altrimenti si possono verificare delle irregolarità
- ad esempio l'utilizzo di uno switch normale anziché di uno antirimbalzo può portare a irregolarità

..... ma per saperne di più aspettiamo il laboratorio

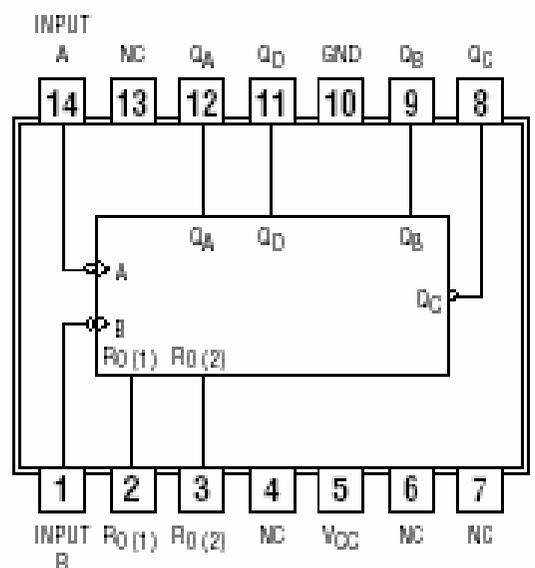
Applicazione dei Flip-Flop

- Contatori
- Registri a scorrimento

Contatore Asincrono modulo 16



Contatore Asincrono modulo 16



NC – No internal connection

TRUTH TABLE

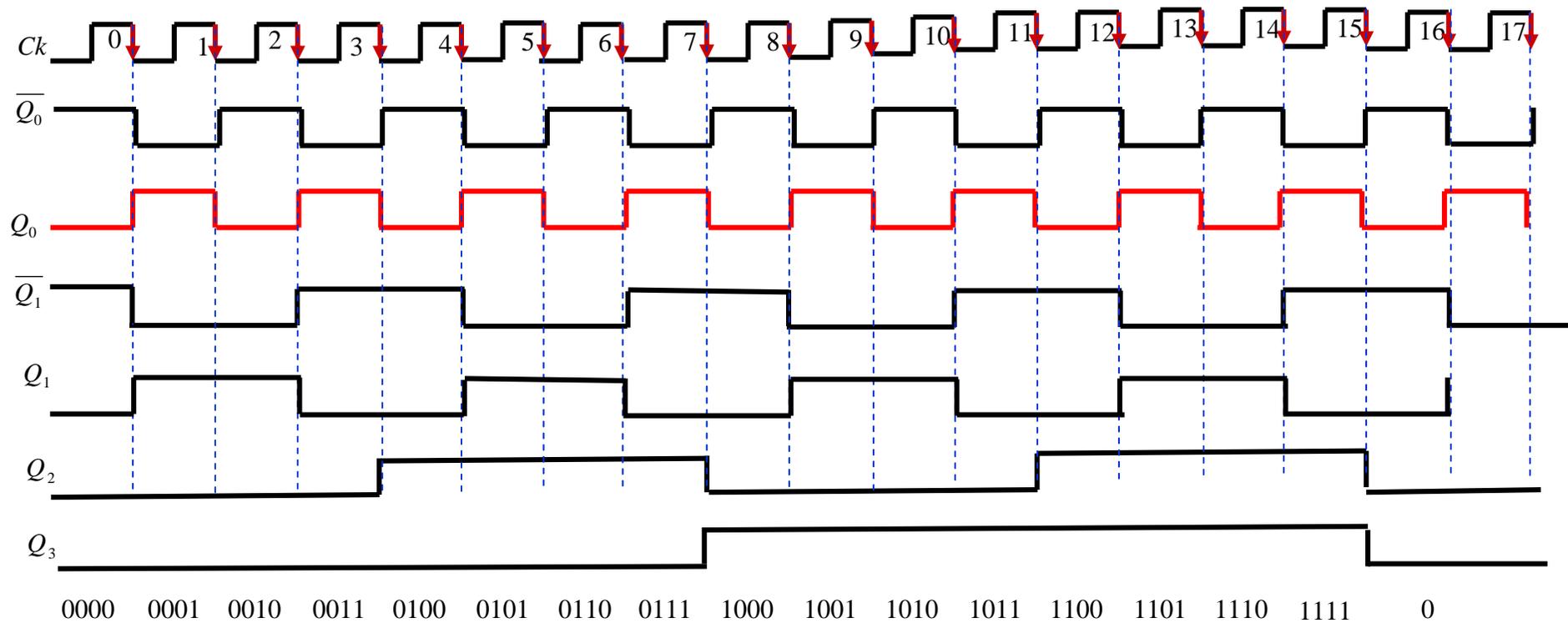
COUNT	OUTPUTS			
	Q ₀	Q ₁	Q ₂	Q ₃
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

H = High Voltage Level, L = Low Voltage Level

Contatore Asincrono (o Ripple Counter) modulo 16

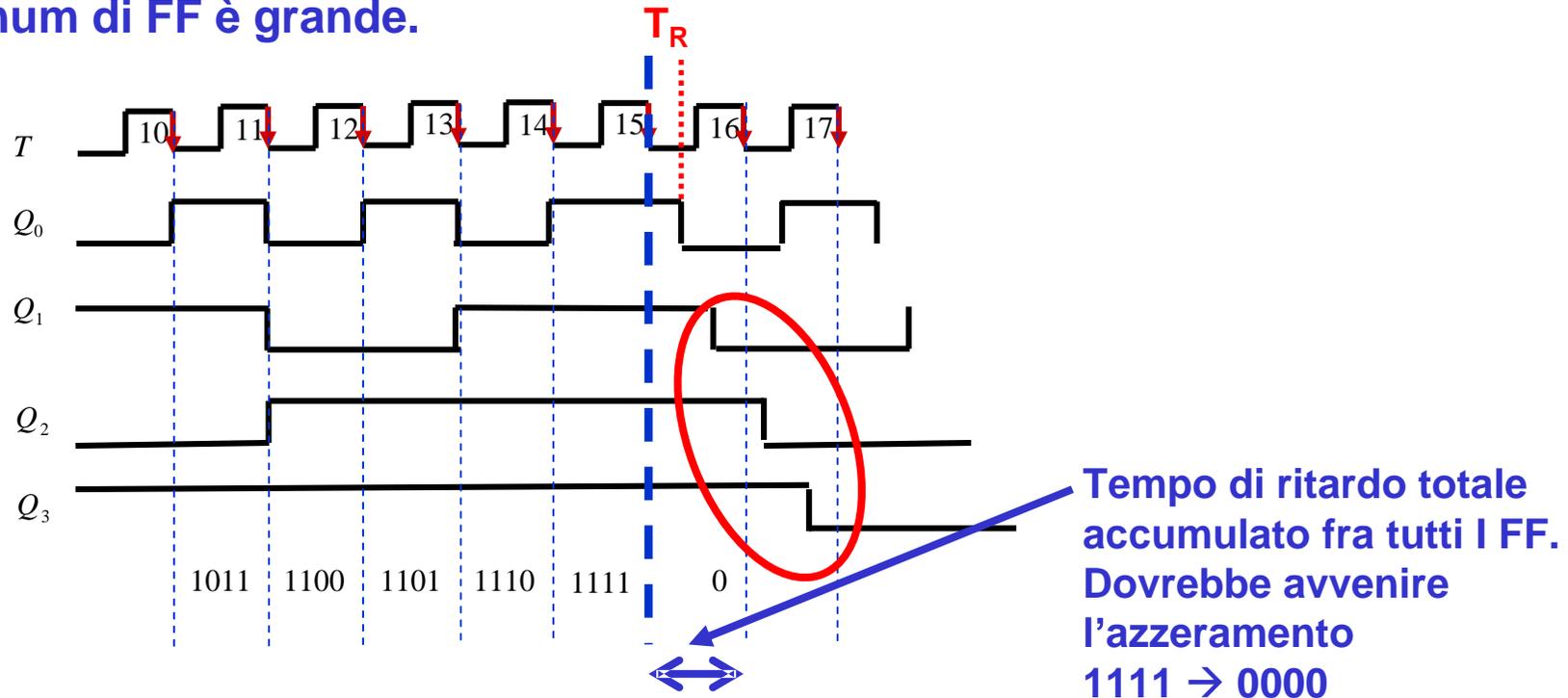
- Abbiamo realizzato un contatore con le seguenti caratteristiche:
 - **asincrono**: Ck ad ogni FF arriva in tempi diversi
 - **modulo 16**: può contare da 0 a 15 (4 FF, ogni FF è una cifra del num bin)
 - **avanti(a) / indietro(b)**: può contare in avanti (0→15) se Sel = a
può contare indietro (15→ 0) se Sel = b
 - Q₀ è il bit meno significativo (LSB)
 - Q₃ è il bit più significativo (MSB)
 - sia in avanti che indietro i bit sono sempre le uscite Q dei FF
 - il selettore non è altro che un MUX

Contatore Asincrono modulo 16 Indietro



Problemi del contatore asincrono

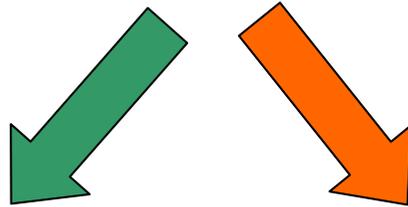
- Il tempo di propagazione del Ck (propagazione del riporto della somma) tra FF successivi può essere un problema, specialmente quando il num di FF è grande.



- Se $T_{Rtot} > T_{Ck}$
 - non è possibile leggere il contenuto del contatore fra 2 impulsi di Ck
 - introduciamo i contatori sincroni

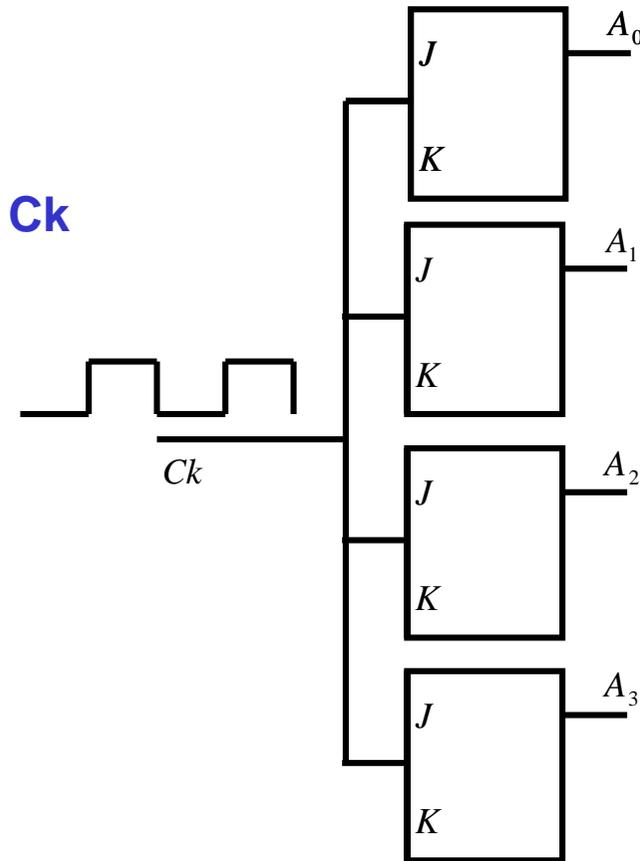
Contatore Sincrono modulo 16

- Il contatore sincrono si basa sul funzionamento in parallelo dei FF
- Ogni FF è comandato dallo stesso Ck

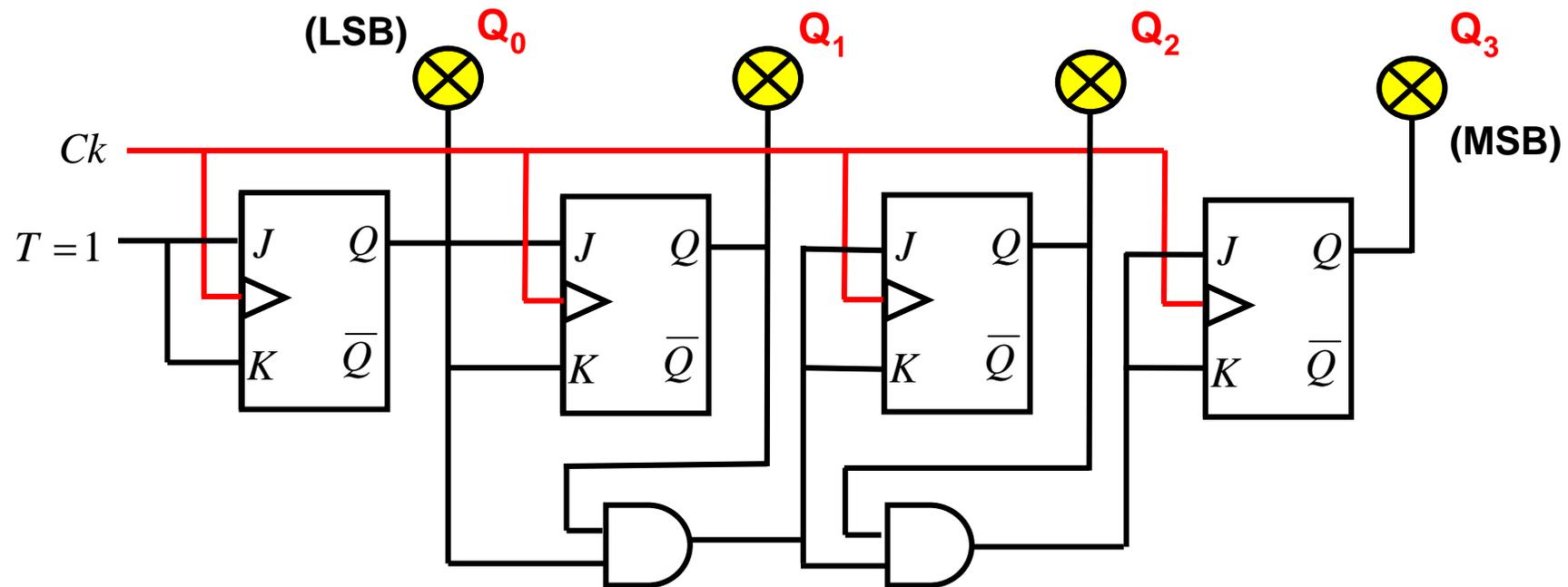


**Contatore con
propagazione del
riporto in serie**

**Contatore con
propagazione del
riporto in parallelo**



Contatore Sincrono con riporto in serie

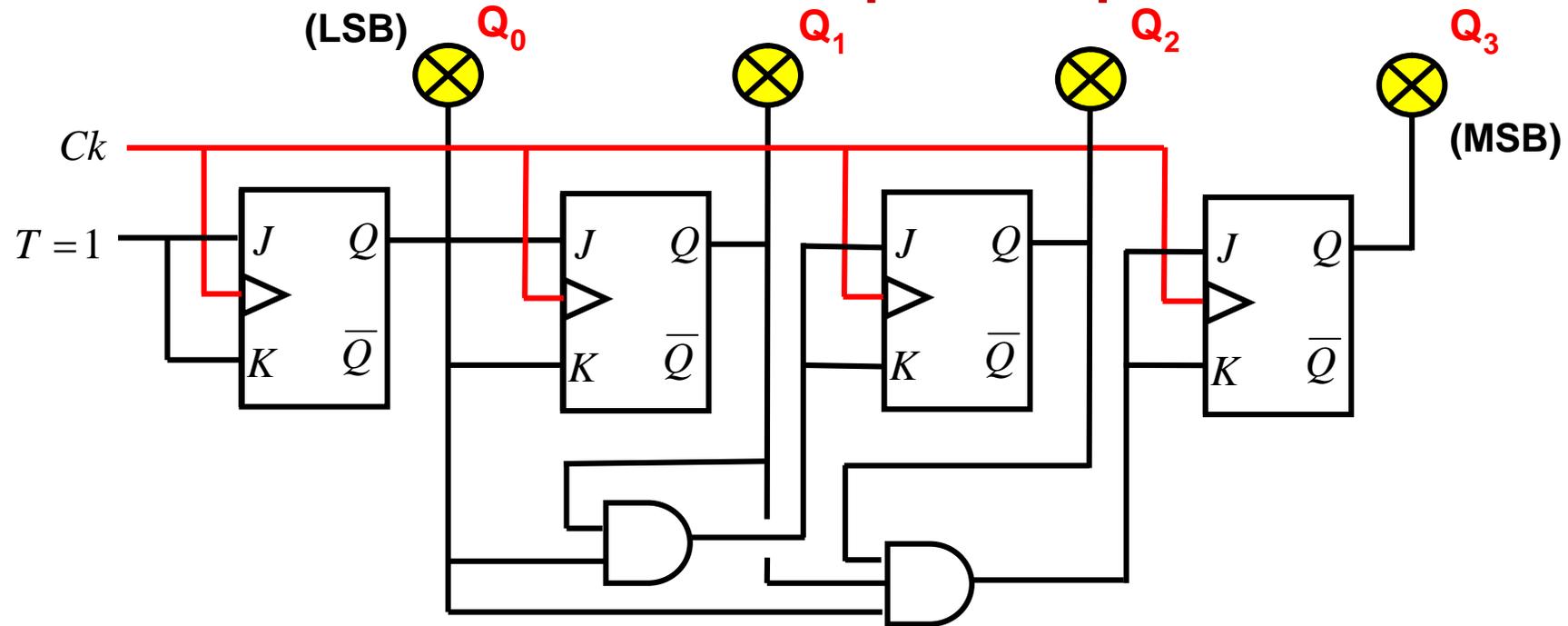


- Il Ck arriva simultaneamente a tutti i FF
- Il riporto attraversa in serie tutte le porte di controllo

→ anche in questo caso si dovrà rispettare un T_{\min} fra due impulsi di Ck:

$$T_{\min} = T_{FF} + (n - 2) \cdot T_{AND}$$

Contatore Sincrono con riporto in parallelo



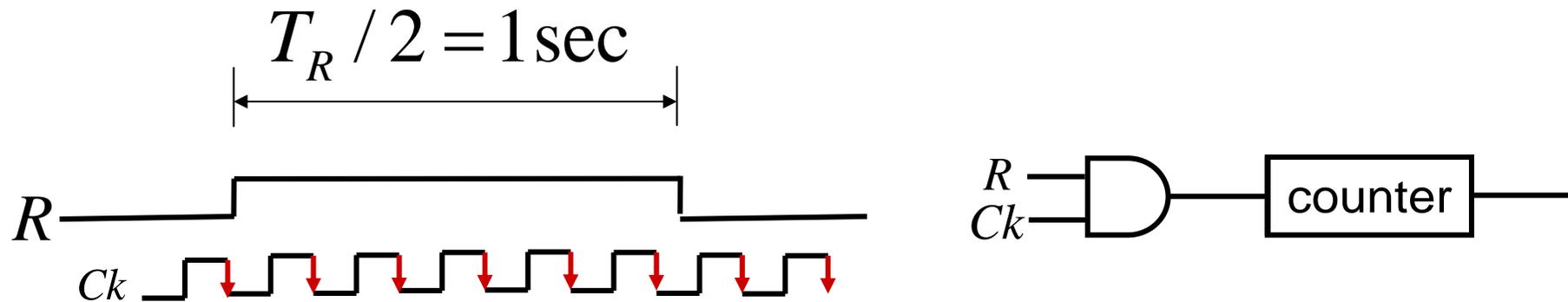
- Il Ck arriva simultaneamente a tutti i FF
- Il riporto attraversa simultaneamente le porte AND in parallelo
→ in questo caso si riduce ulteriormente T_{\min} fra due impulsi di Ck:

$$T_{\min} = T_{FF} + T_{AND}$$

- **Inconveniente: necessità di utilizzare porte AND a più ingressi**

Contatore per misurare la frequenza

- Supponiamo di avere un segnale con frequenza sconosciuta (Ck)
- Usiamo un'onda con frequenza nota (es. 0.5Hz)

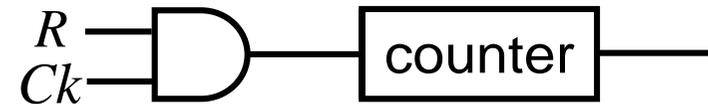
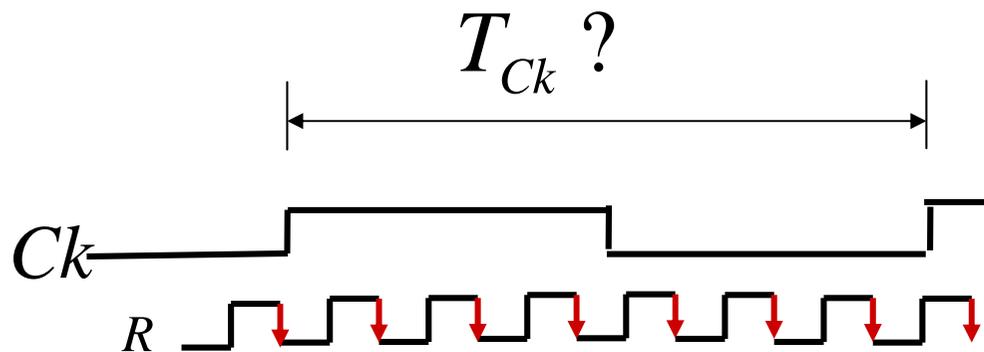


$$T_{Ck} = \frac{T_R / 2}{n} \Rightarrow f_{Ck} = \frac{2 \cdot n}{T_R}$$

incertezza su n $\sigma_n = \pm 1$ (ciclo)

Contatore per misurare il tempo

- Supponiamo di avere un segnale con frequenza sconosciuta (Ck)
- Usiamo un'onda con frequenza nota



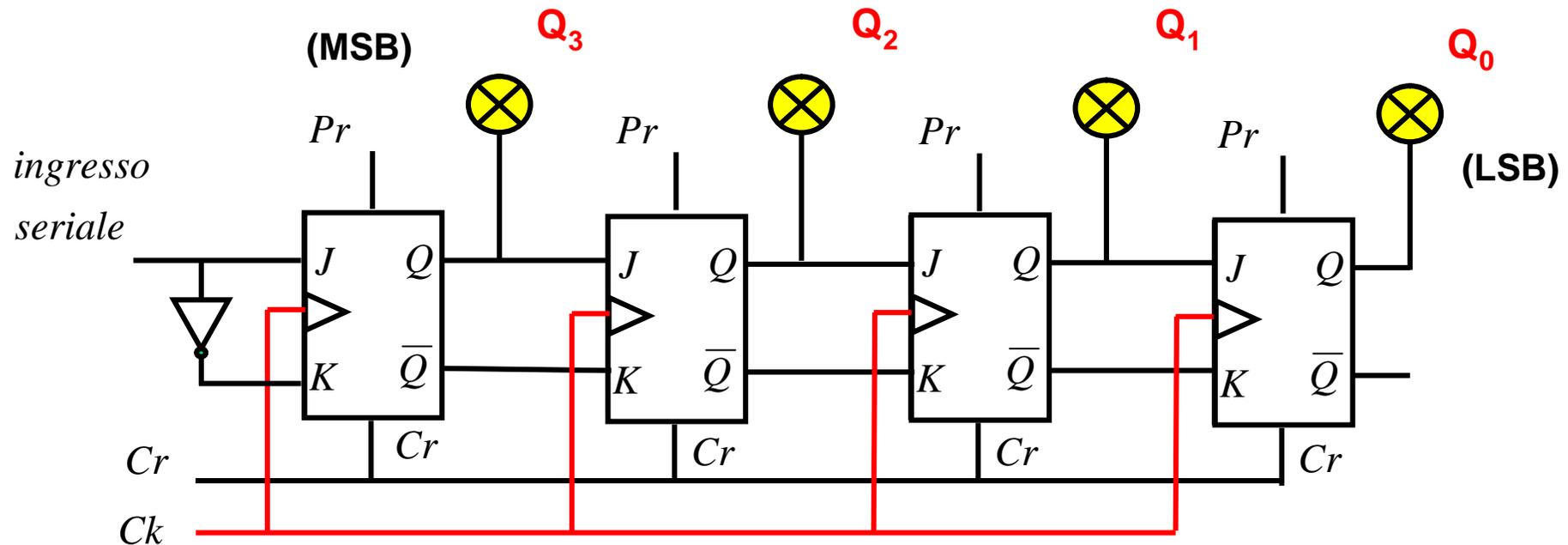
incertezza su n $\sigma_n = \pm 1$ (ciclo)

$$T_{Ck} = 2 \cdot n \cdot T_R \Rightarrow T_{Ck} = \frac{2 \cdot n}{f_R}$$

- Frequenze elevate del rif → conveniente misurare tempi
- Frequenze basse del rif → conveniente misurare frequenze

Registro a scorrimento

- Un FF memorizza una parola di 1 bit
 - con più FF possiamo memorizzare parole a più bit
 - ogni bit viene inserito ad ogni ciclo di Ck
 - si inseriscono i bit dal meno significativo a più significativo



- **Ad ogni ciclo di Ck il bit si sposta da Q_n a Q_{n-1}**
 - **il bit scorre verso destra → registro a scorrimento**

Registro a scorrimento: un esempio

➤ Supponiamo di voler memorizzare il numero 1101:

→ inseriamo la cifra meno significativa (1°)

D=1, si compie un ciclo di Ck (0→1→0)

→ $Q_3=1$

→ inseriamo la 2° cifra

D=0, ciclo di Ck

→ $Q_3=0$ $Q_2=1$

→ inseriamo la 3° cifra

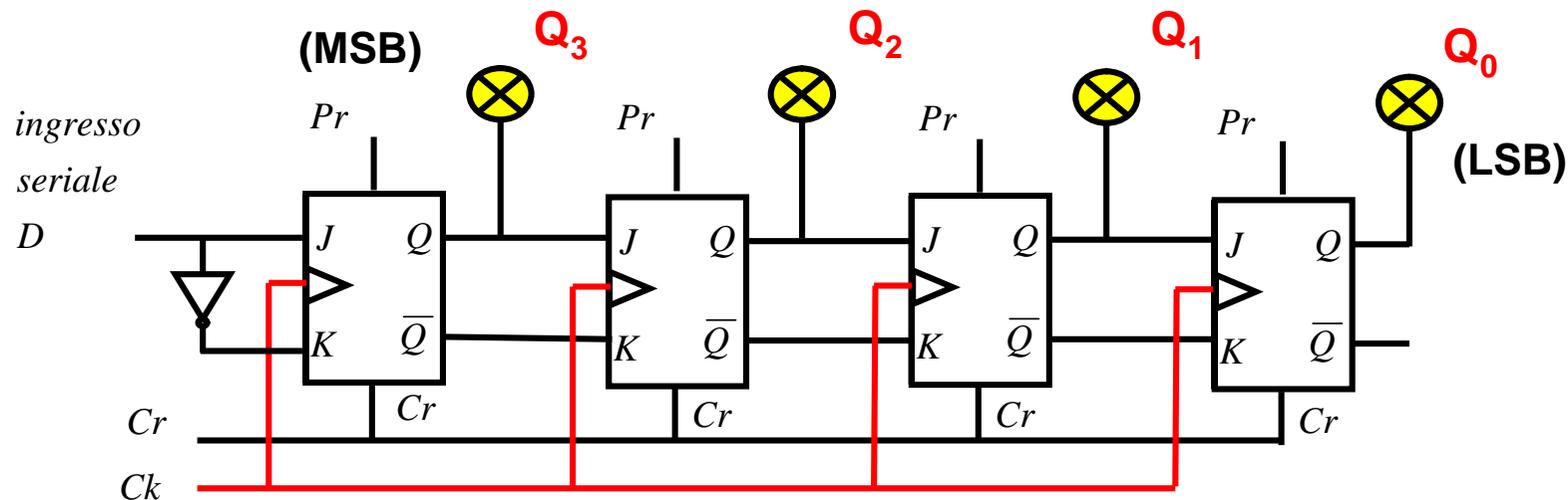
D=1, ciclo di Ck

→ $Q_3=1$ $Q_2=0$ $Q_1=1$

→ inseriamo la 4° cifra

D=1, ciclo di Ck

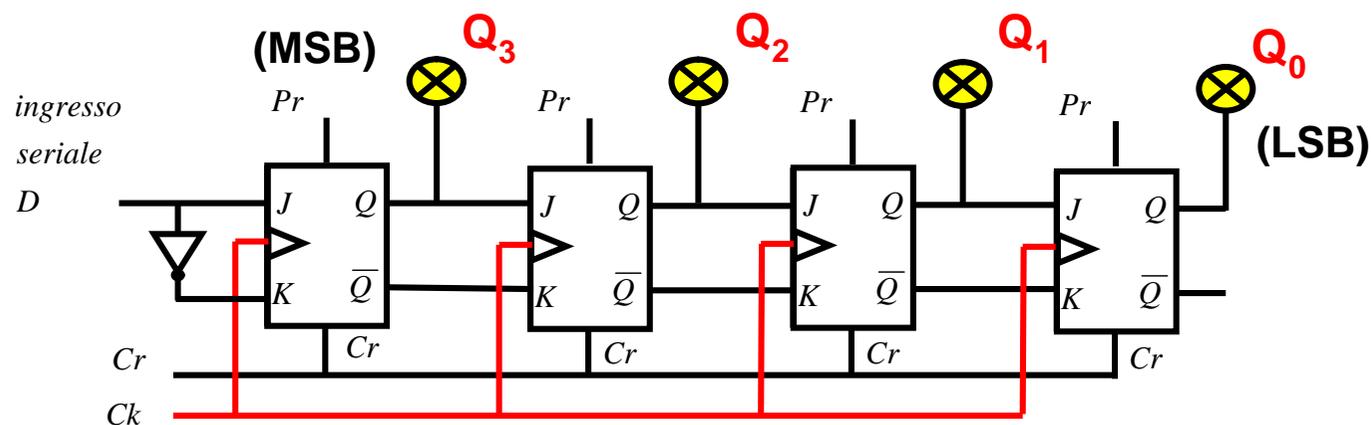
→ $Q_3=1$ $Q_2=1$ $Q_1=0$ $Q_0=1$



D deve rimanere stabile per tutto il ciclo di Ck di memorizzazione

Tipi di Registri a scorrimento: SIPO, SISO, PISO, PIPO

- Il registro appena studiato è di tipo SIPO:
→ **S**erial **I**nput **P**arallel **O**utput
- Gli ingressi vengono inseriti in serie
- Le uscite si attivano in parallelo

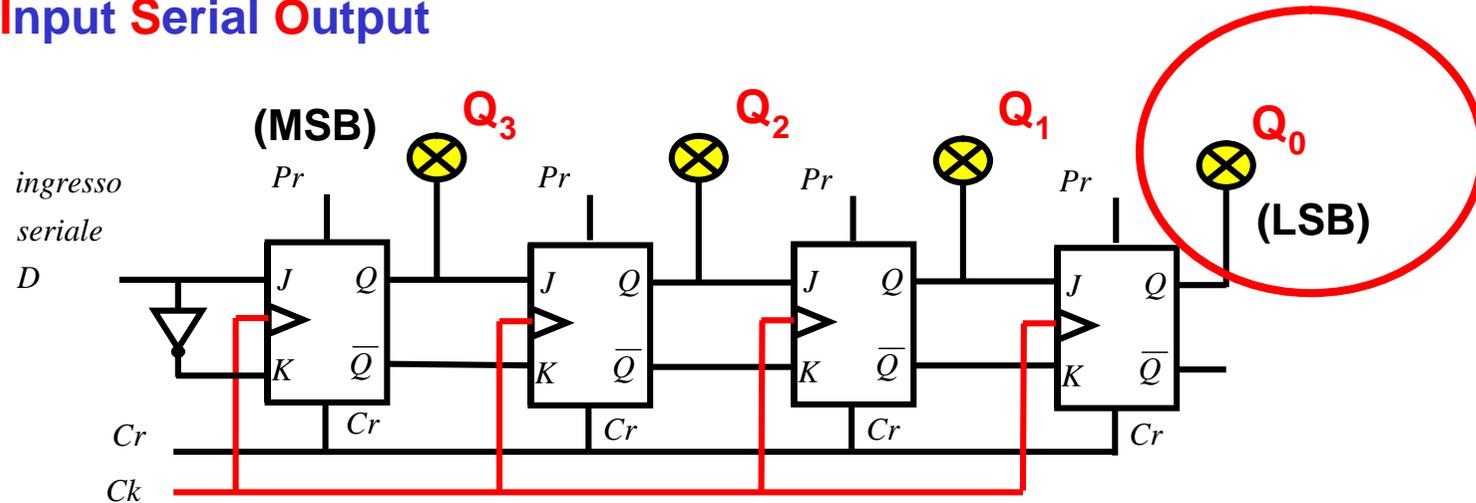


Serial/**P**arallel **I**nput **S**erial/**P**arallel **O**utput

S/P I S/P O

Registro a scorrimento SISO

→ Serial Input Serial Output

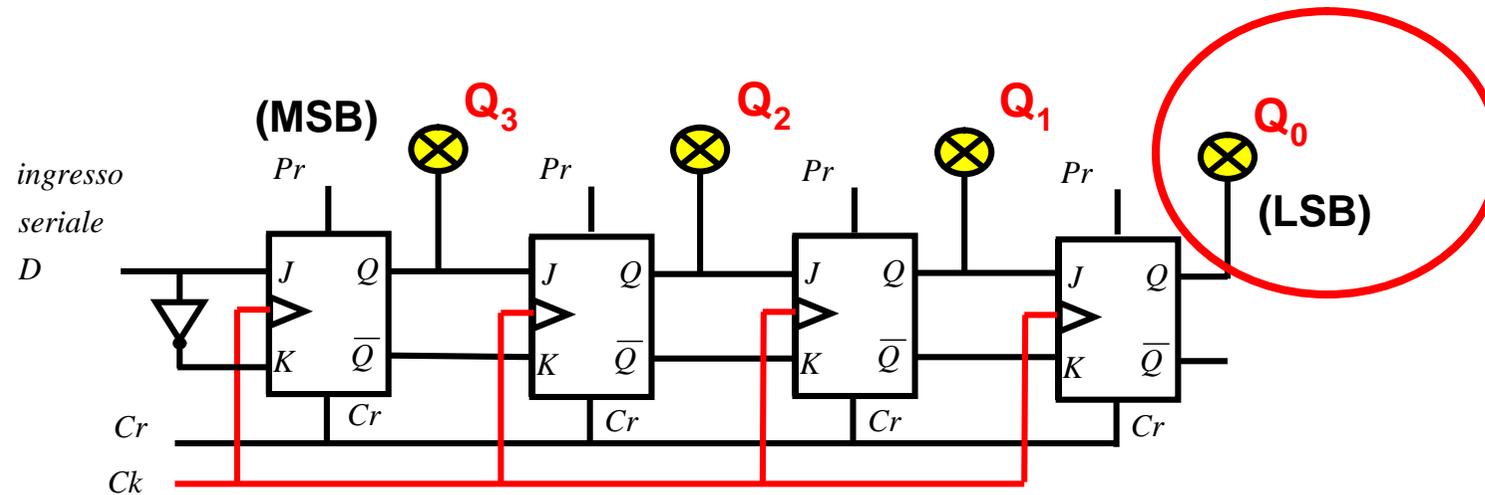


- Si inserisce il numero in seriale (come prima)
- Si pone $D=0$
- Il numero si legge usando solo l'uscita Q_0
- Si inviano $n(3)$ cicli di Ck in modo da far assumere all'uscita Q_0 in successione ad ogni ciclo il valore di ogni cifra del numero

dalla meno significativa (dopo 0 cicli) → più significativa (dopo 3 cicli)

Registro a scorrimento PISO

→ Parallel Input Serial Output

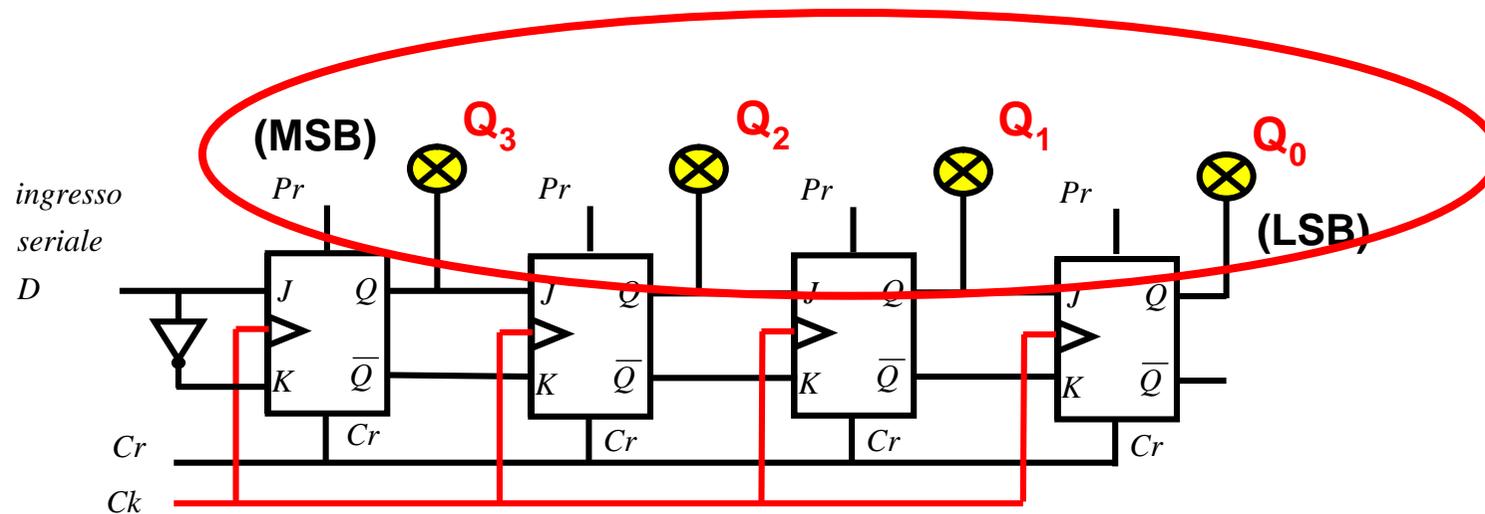


- L'inserimento avviene in parallelo con *Pr* e *Cr*
- La lettura avviene in serie come in SISO

Registro a “scorrimento” PIPO

→ **Parallel Input Parallel Output**

In questo caso non è più un registro a scorrimento perché la parola non scorre all'interno del registro.



- L'inserimento avviene in parallelo con Pr e Cr
- La lettura avviene in parallelo (banale)

Registro a scorrimento come divisore per 2

- Memorizziamo una certa parola → 1011 → 11₁₀
- Portiamo l'Ingresso seriale a 0
- Dopo un ciclo di Ck la parola si sposta tutta a destra perdendo l'ultimo bit → 0101 → 5₁₀
- La parola risultante risulta la metà (approssimata per difetto) della parola iniziale → 11/2=5.5 → 5₁₀

Prendiamo in generale un numero binario $A_3A_2A_1A_0$

Consideriamo quello shiftato $A_3A_2A_1$

Facciamo la divisione

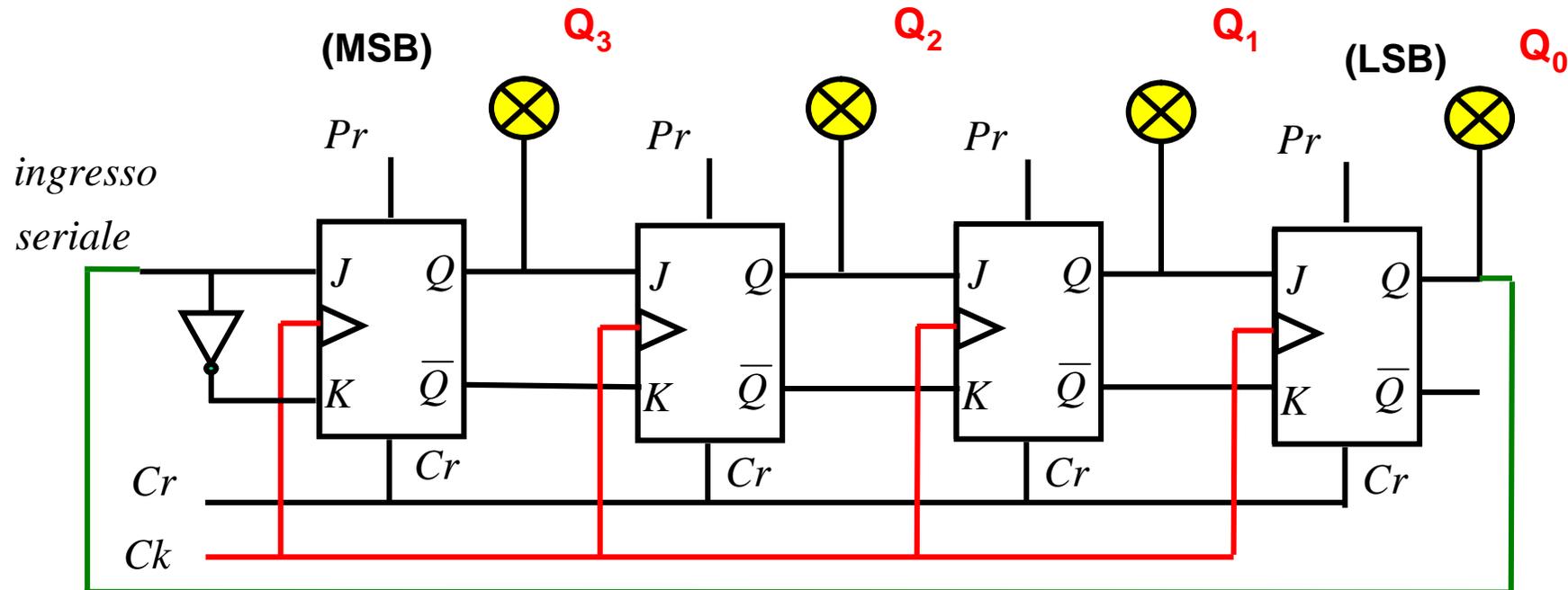
$$\begin{array}{r|l}
 \overbrace{A_3A_2A_1} \overline{A_0} & A_3A_2A_1 \\
 \hline
 \text{— } A_0 & \\
 \text{resto } A_0 & (10)_2 = (2)_{10}
 \end{array}$$

✓ se il resto $A_0=0$ → il risultato è esatto

✓ se il resto $A_0=1$ → il risultato è approssimato per difetto

Memoria Dinamica o Circolare

- Prendiamo un registro a scorrimento di tipo PISO
- Facciamo un collegamento di retroazione $Q_0 \rightarrow$ ingresso seriale

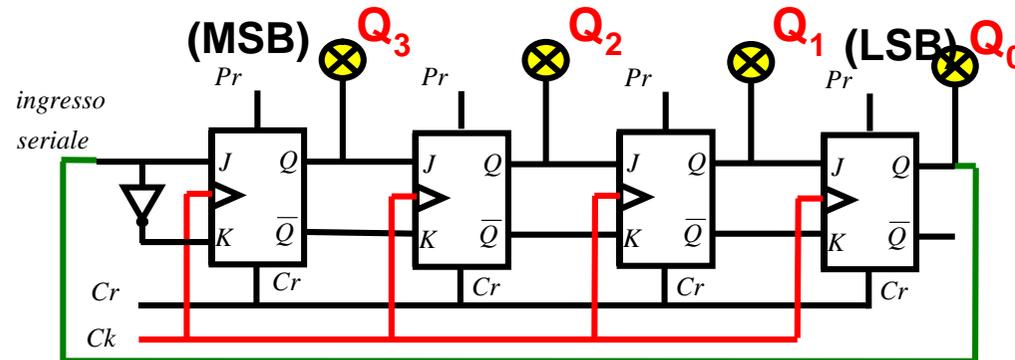


- Impostiamo un numero in modo parallelo (Pr , Cr)
- Applicando continuamente un Ck , all'interno del registro scorre sempre la sequenza impostata all'inizio
→ memoria dinamica o circolare

Contatore ad Anello

- Realizziamo una memoria circolare con N FF
- Preassegnamo $Q_0=1$ e $Q_1=Q_{N-1}=0$
- Dopo n ($n < N$) impulsi di $Ck \rightarrow Q_{N-n}=1$ e tutti $Q_i=0$

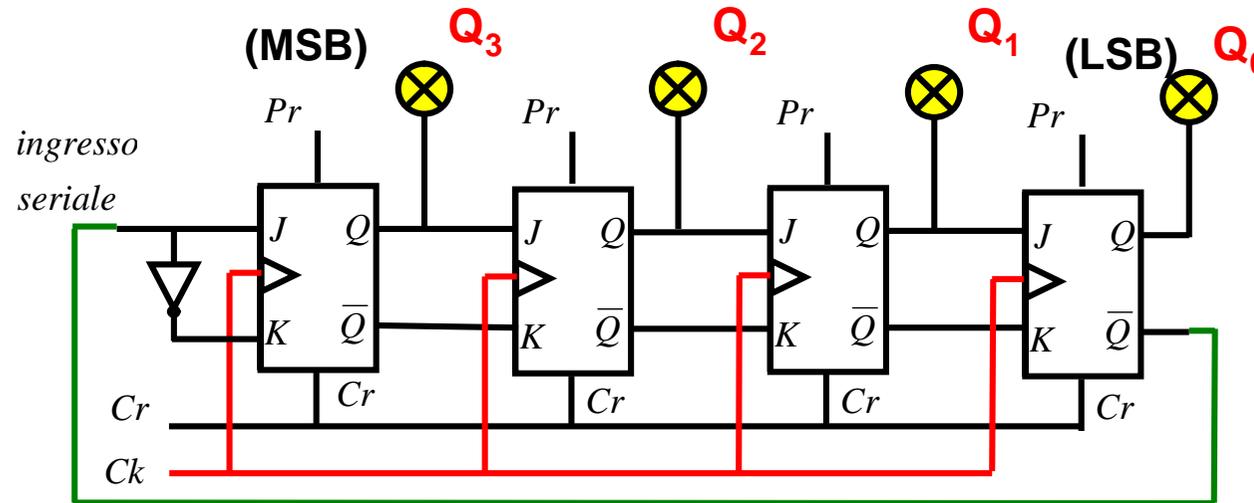
→ dalla posizione in cui si trova 1 riusciamo a determinare quanti impulsi di Ck sono stati eseguiti



- Contatore
- Abilitatore in tempi differenti
- Divisore per N, in quanto si ha un impulso all'uscita Q_0 ogni N impulsi di Ck

Contatore ad Anello Incrociatoo

- Prendiamo un registro a scorrimento di tipo PISO
- Facciamo un collegamento di retroazione $\overline{Q}_0 \rightarrow$ ingresso seriale

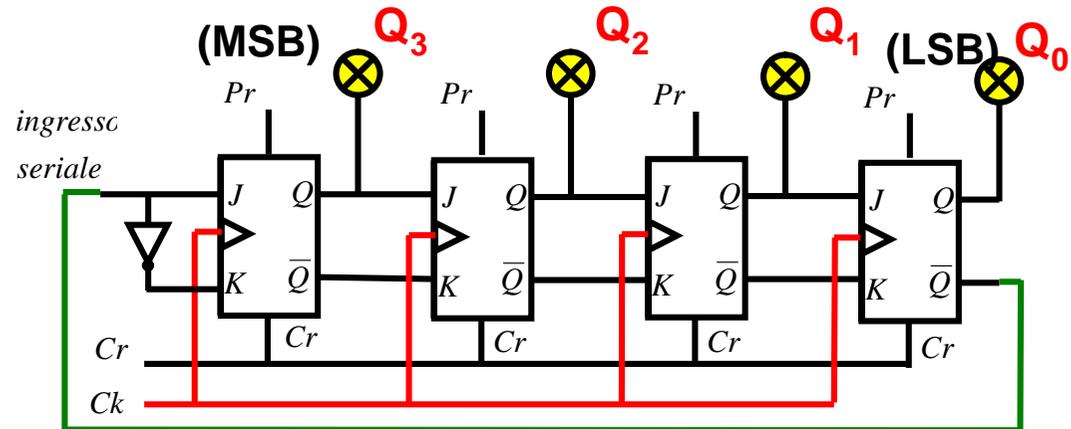


- Preassegnamo tutte le uscite a 0 ($\rightarrow \overline{Q}_0=1$)
- Dopo N impulsi di Ck le uscite sono tutte a 1
- Dopo altri N impulsi saranno di nuovo tutte a 0
- Si ritorna alla condizione iniziale dopo 2N impulsi di Ck
- **Divisore per 2N, contatore per 2N**

Contatore (ad Anello Incrociato) per 2N

➤ Chiamato anche contatore Moebius o contatore Johnson

N° cicli	Q ₃	Q ₂	Q ₁	Q ₀
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0



➤ Con un opportuno decodificatore si può contare



Attività di Laboratorio
ELETTRONICA DEI
SISTEMI DIGITALI

Parte III

Corso di Laurea in Informatica e TFI
Anno Accademico 2007-2008

Esperienza Contatori/Registri

Comprende le prove:

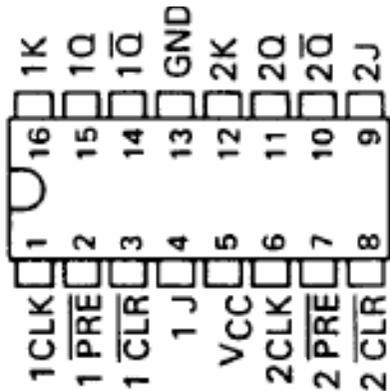
- **Uso di un FF-JK con Preset e Clear**
- **Realizzazione di un contatore binario asincrono avanti/indietro a 4 bit con 4 FF JK**
- **Realizzazione di un registro a scorrimento a 4 bit con 4 FF JK:**
 - **Funzionamento SISO,SIPO,PIPO,PISO.**
- **Realizzazione di un contatore ad anello.**
- **Realizzazione di un contatore ad anello incrociato.**

IC da utilizzare sono

- **'76 : contiene 2 FF-JK con Pr a Cr**
- **'107 : contiene 2 FF-JK con solo Cr**
- **'157 : contiene 4 MUX a 2 ingressi**
- **(oppure) '86 : contiene 3 XOR**

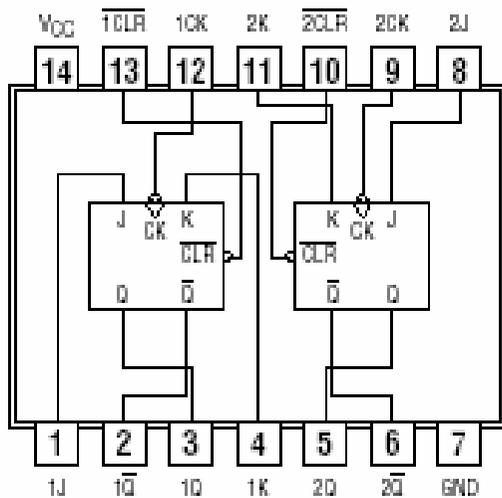


SN5476, SN54LS76A . . . J PACKAGE
 SN7476 . . . N PACKAGE
 SN74LS76A . . . D OR N PACKAGE
 (TOP VIEW)



107

DUAL J-K FLIP-FLOPS WITH CLEAR



'76 FUNCTION TABLE

INPUTS					OUTPUTS	
PRE	CLR	CLK	J	K	Q	Q̄
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H↑	H↑
H	H	⌈	L	L	Q ₀	Q̄ ₀
H	H	⌈	H	L	H	L
H	H	⌈	L	H	L	H
H	H	⌈	H	H	TOGGLE	

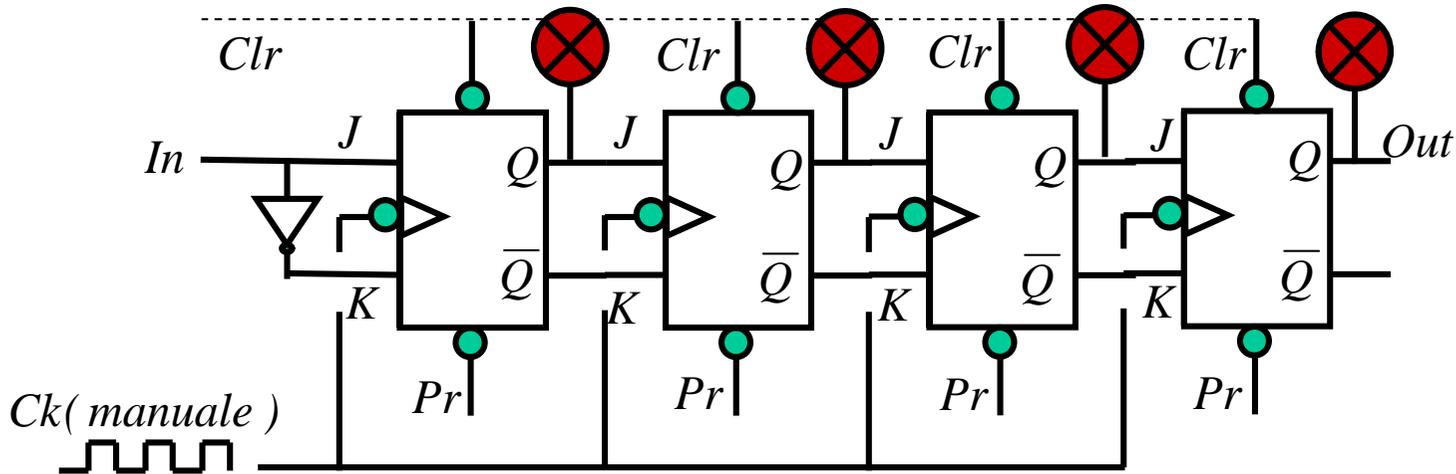
FUNCTION TABLES '107

INPUTS				OUTPUTS	
CLR	CLOCK	J	K	Q	Q̄
L	X	X	X	L	H
H	⌈	L	L	Q ₀	Q̄ ₀
H	⌈	H	L	H	L
H	⌈	L	H	L	H
H	⌈	H	H	TOGGLE	

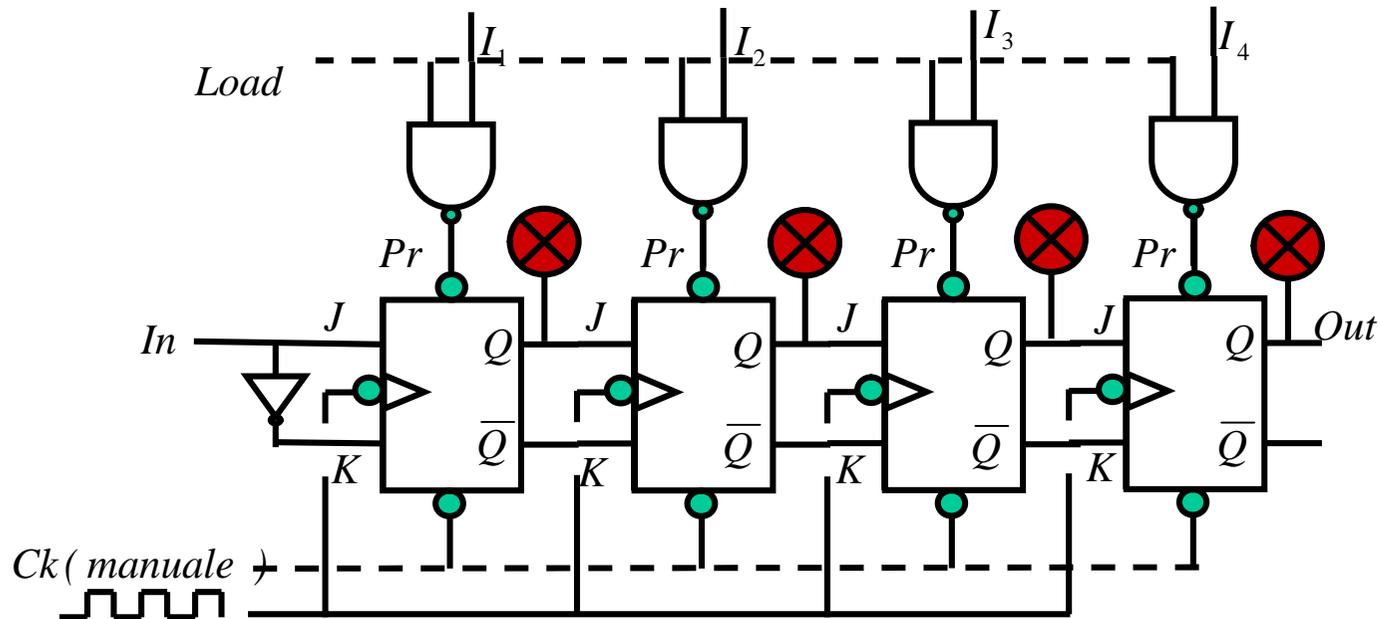
'LS107A, 'HC107

INPUTS				OUTPUTS	
CLR	CLOCK	J	K	Q	Q̄
L	X	X	X	L	H
H	↓	L	L	Q ₀	Q̄ ₀
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q ₀	Q̄ ₀

Registro a scorrimento con 4 FF JK (7476): SI SO,SI PO



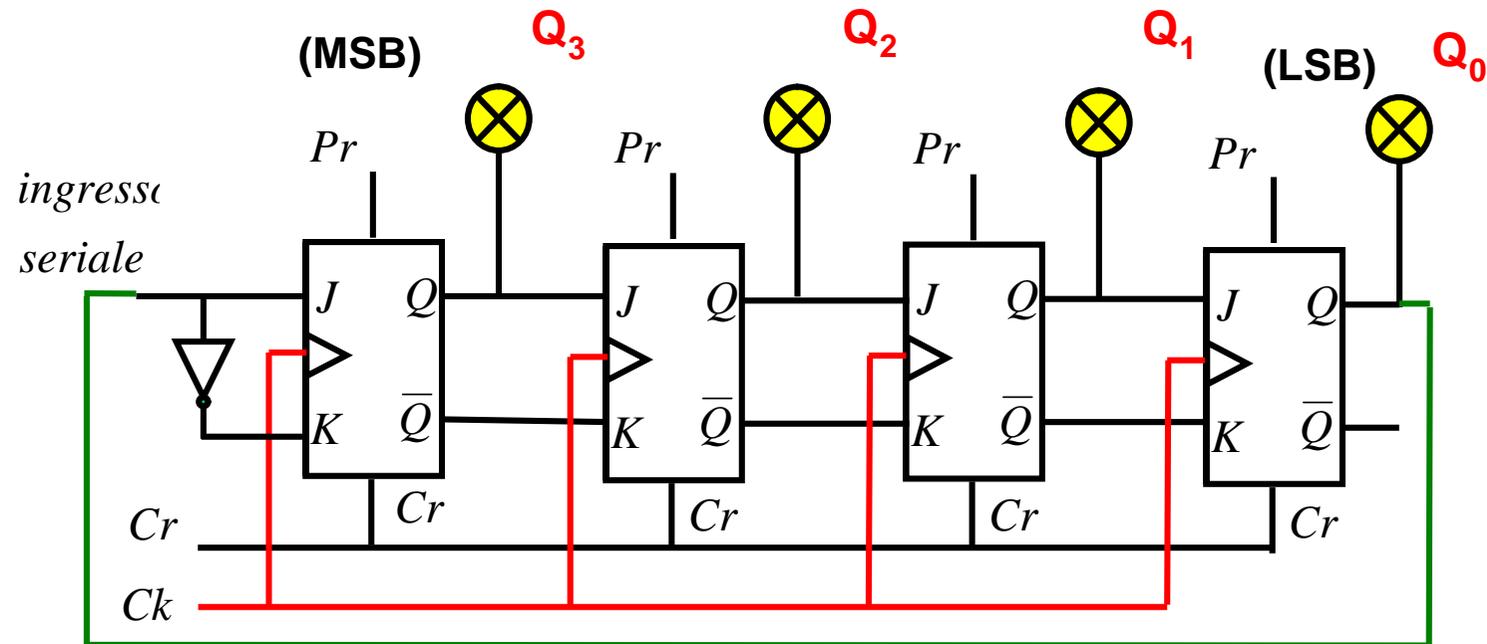
Registro a scorrimento con 4 FF JK (7476): PI SO,PI PO



Attenzione:
 Non consente
 l'introduzione di zeri ma
 si può usare il clear.
 E' anche un:
Generatore di sequenza
Contatore ad anello

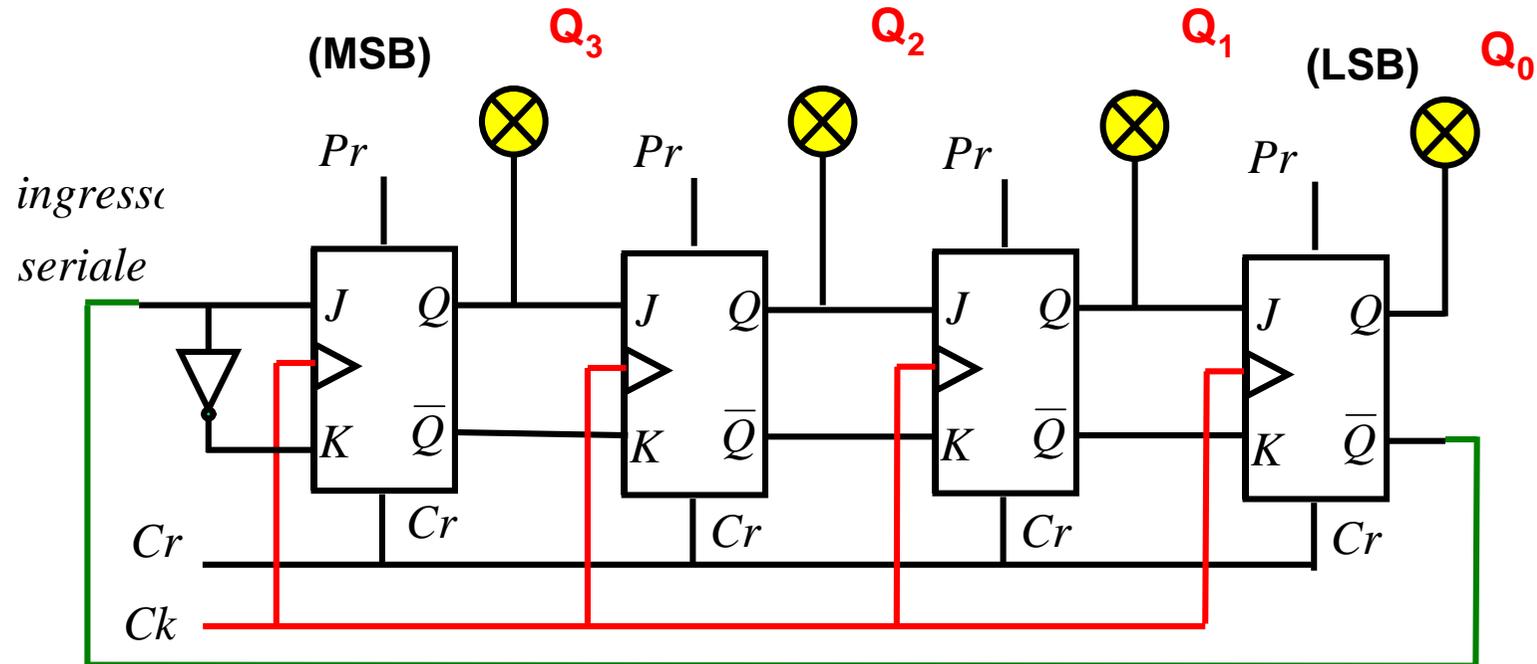
Contatore ad Anello

→ Con la retroazione $Q \rightarrow$ *ingresso seriale* realizziamo un contatore ad anello



Contatore ad Anello Incrociatoo

- Prendiamo il registro a scorrimento di tipo PISO
- Facciamo un collegamento di retroazione $\overline{Q}_0 \rightarrow$ *ingresso seriale*



➔ Realizziamo un contatore ad anello incrociatoo

ELETTRONICA DEI SISTEMI DIGITALI

Parte IV

Corso di Laurea in Informatica TFI
Anno Accademico 2007-2008

Porte Logiche con uscite: Totem Pole, Open Collector e Three-State

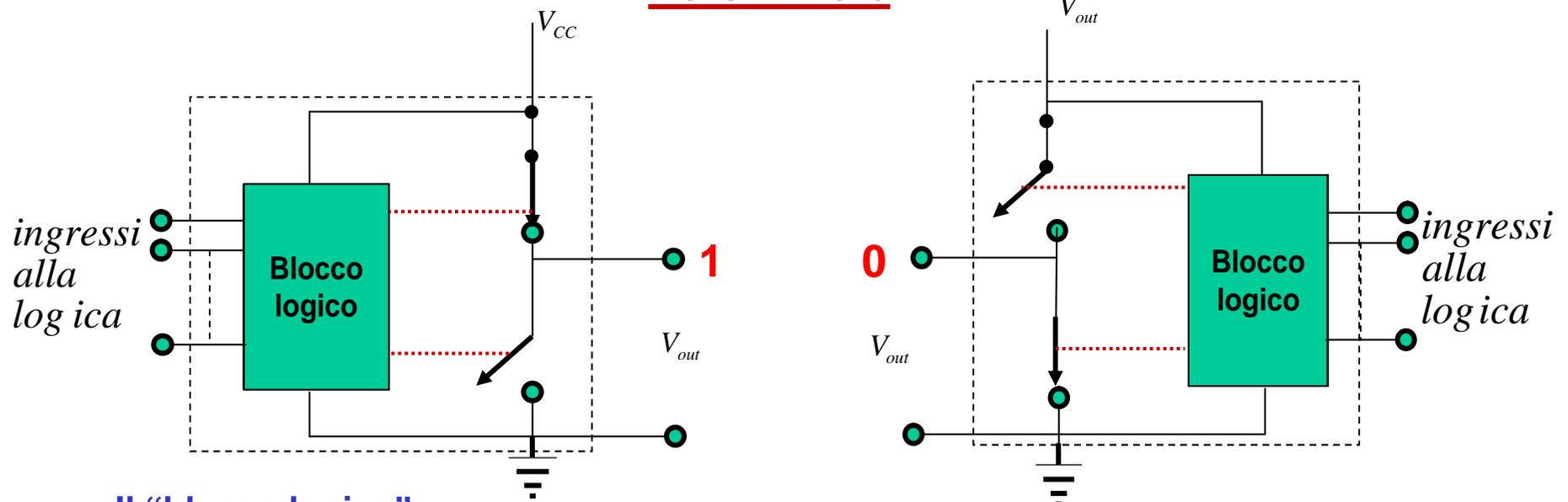
Totem Pole

- Tutte le porte utilizzate finora hanno uscite Totem Pole: **2 possibili stati**

 - Ogni IC ha un'alimentazione che ha i ruoli:
 - Fornisce I livelli “alti” di uscita
 - Fornisce la potenza per pilotare I circuiti di “carico” (fan-out)
 - Non costituisce un segnale

 - I livelli sono ottenuti mediante dispositivi (transistor) che funzionano come “interruttori” comandati. In questi gli stati:
 - di conduzione (interruttore chiuso)
 - di non conduzione (interruttore aperto)
- sono determinati dallo stato di una variabile di comando elettrica.**

Totem Pole



Il “blocco logico”:

- decide, in base alla logica (AND,OR,...) se l' uscita deve essere alta o bassa

Le linee tratteggiate:

- rappresentano una “immaginaria connessione meccanica” che predispone lo stato degli interruttori

Totem pole

- gli interruttori sono sempre in stati opposti
- uscita collegata a V_{CC} -> stato alto
- uscita collegata a Gnd -> stato basso

Totem Pole

I Transistor sono “interruttori imperfetti”:

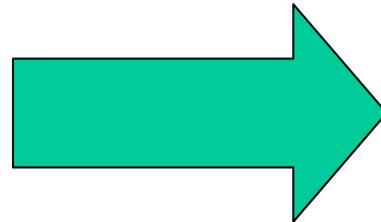
- lo stato basso non è zero
- lo stato alto non è 5V

lo stato con entrambi gli interruttori chiusi non è ammissibile!

L' integrato è predisposto per evitare quella configurazione ma bisogna evitare di:

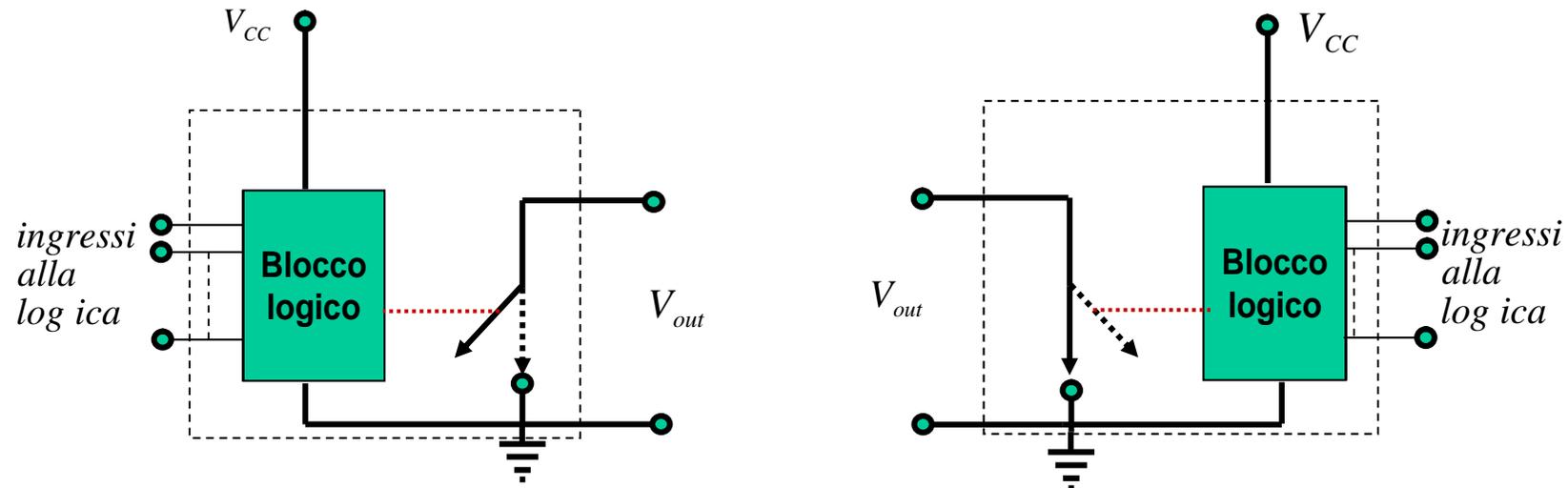
- collegare più uscite insieme
- collegare uscite con switch logici

Totem Pole



**2 possibili stati di uscita:
Alto
Basso**

Introduciamo un 3° stato d'uscita: Open Collector (Collettore Aperto)



➤ Altri 2 stati dell'uscita sono:

→ **A conduzione verso massa:**

- Tensione di uscita "prossima a zero" → come prima livello Basso

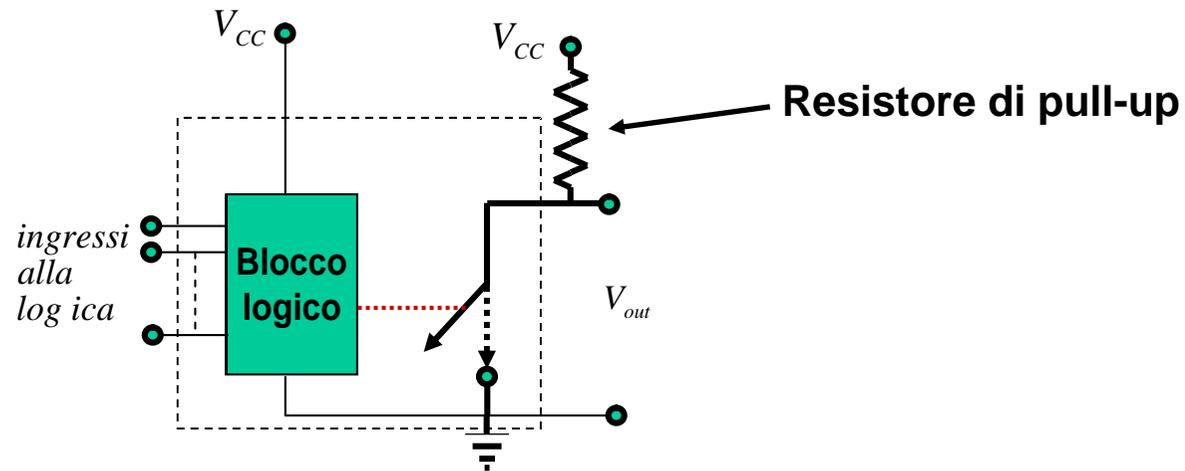
→ **Open Collector:**

- Tensione di uscita come un "filo sconnesso"

→ questa stessa uscita può essere collegata ad altre uscite OC

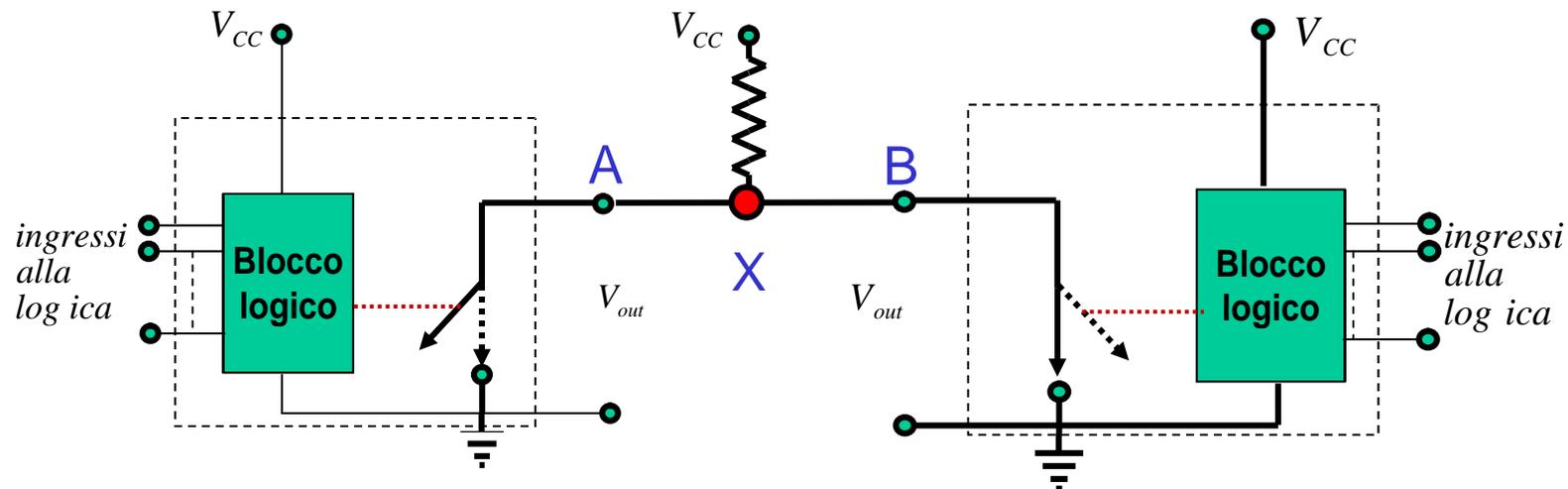
Open Collector + pull up

- Resistore di **pull up** e **wired AND** (AND cablato)



- In questo caso l' alimentazione ha un ruolo anche nella logica:
→ gli stati possibili in questo caso sono Alto/Basso
Vedere gli integrati '01 e '03

Open Collector + pull up come AND cablato (wired AND)



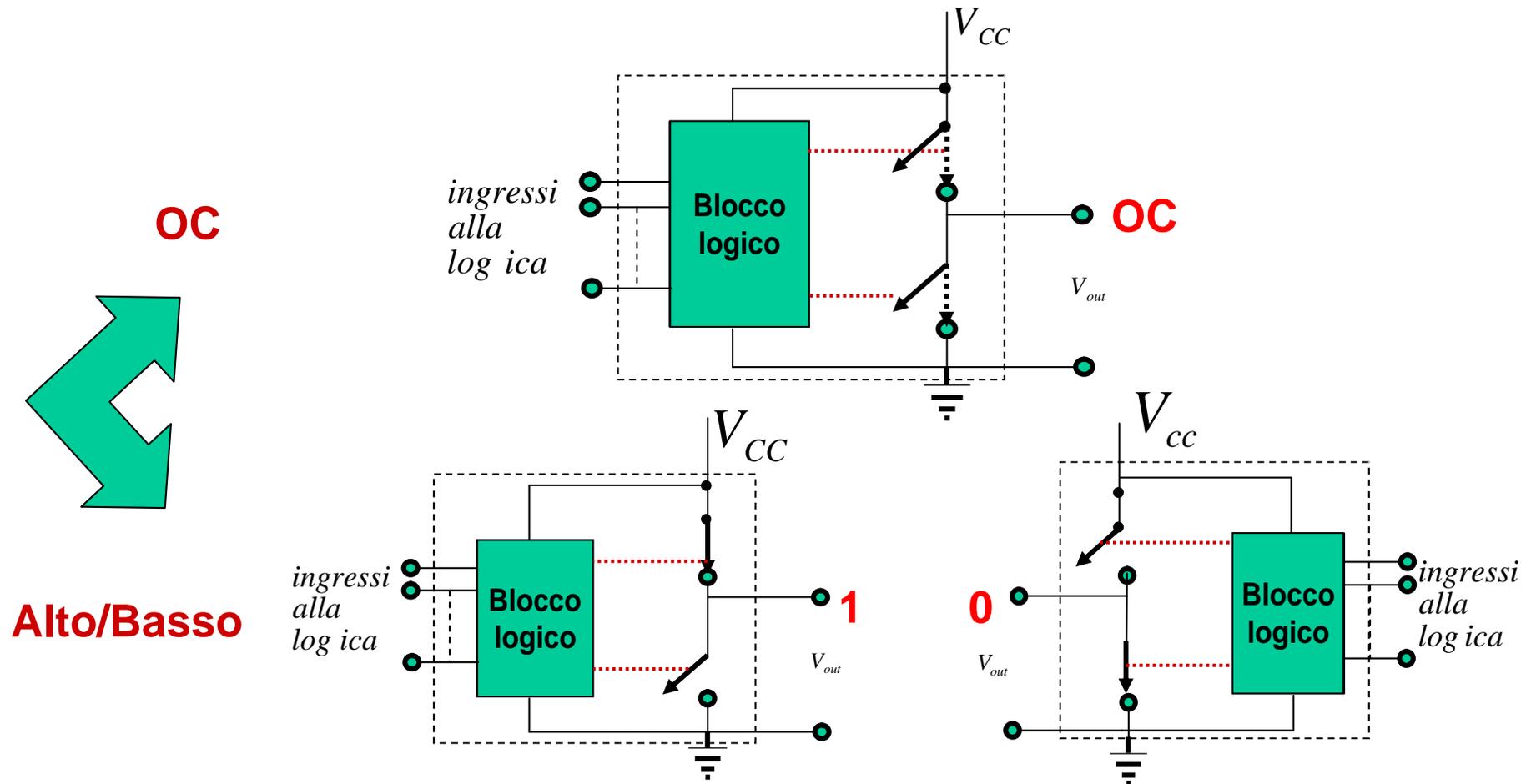
- $X=AB \rightarrow$ l'uscita X è l'AND fra A e B :
 - quando una delle uscite è a 0 $X=0$
 - quando entrambe le uscite sono OC $X=1$

OC \rightarrow AND cablato: non è possibile realizzarlo con porte TP

TP \rightarrow uscite collegate insieme con livelli diversi \rightarrow corto circuito

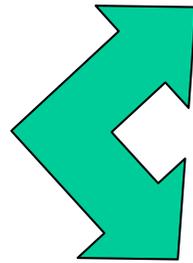
Three-State: Alto/Basso/Open Collector

- Necessità di far condividere a più uscite una stessa linea → serve lo stato OC
- E' necessario un circuito che realizzi 3 stati d'uscita: Alto/Basso/OC



Three-State: Alto/Basso/Open Collector

➤ **Bit di selezione**

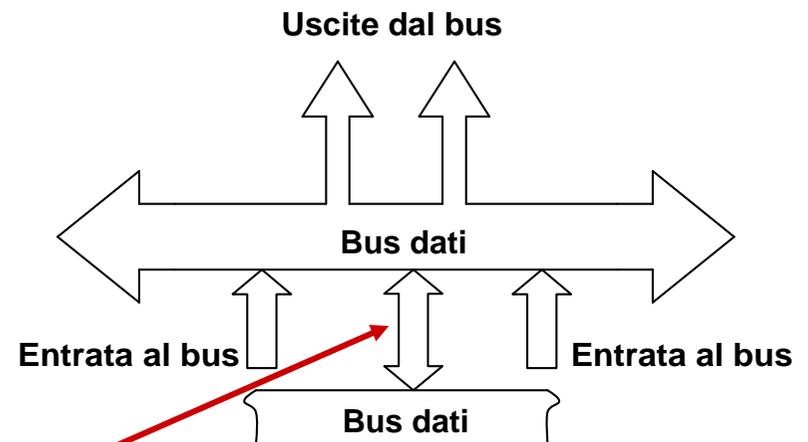


Alto/Basso

OC (alta impedenza): assumerà il valore deciso da altre uscite collegate

Utilizzati nei sistemi complessi:

- trasferimento di dati a molti bit
- trasferimento di dati tra più blocchi
- improponibile un sistema 1bit=1filo!!
- necessità di condividere le linee
- interfacciamento bus di dati/ utenze

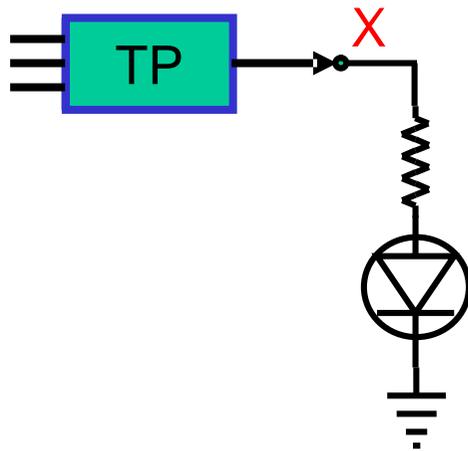


Collegamento bi- direz. al bus

Visualizzare gli stati di un'uscita

Totem Pole (TP)

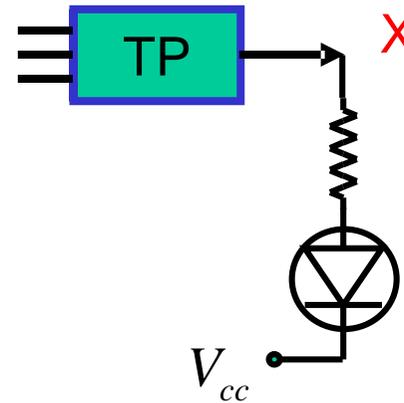
➤ LED attivi alti o attivi bassi



LED AA

$X=0 \rightarrow \text{LED} = 0$

$X=1 \rightarrow \text{LED} = 1$



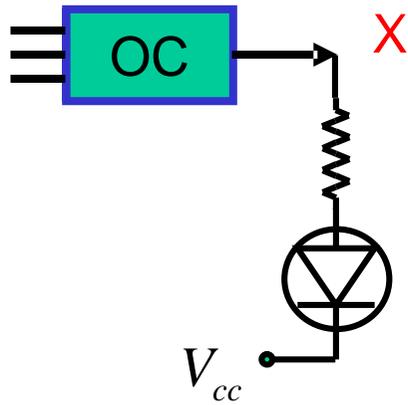
LED AB

$X=0 \rightarrow \text{LED} = 1$

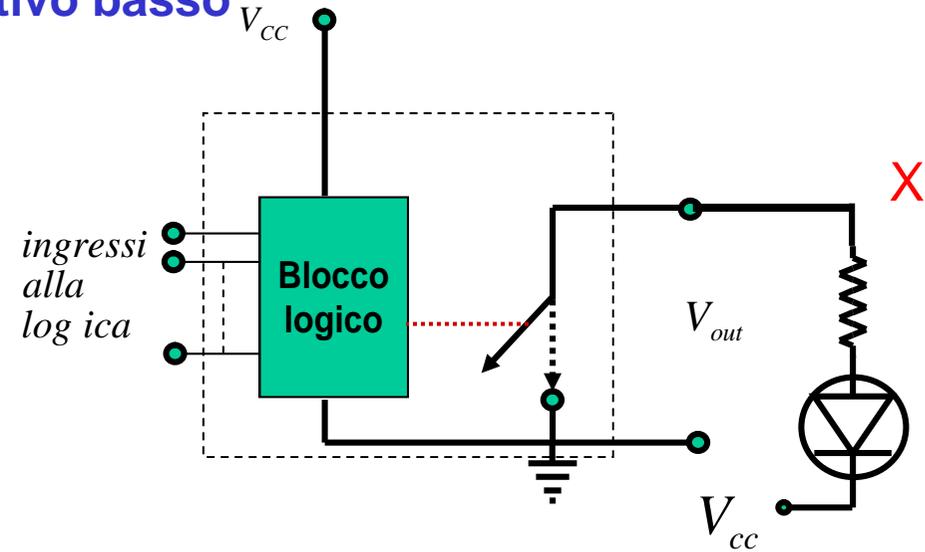
$X=1 \rightarrow \text{LED} = 0$

Visualizzare gli stati di un'uscita

Open Collector (OC)



➤ LED attivo basso



LED AB

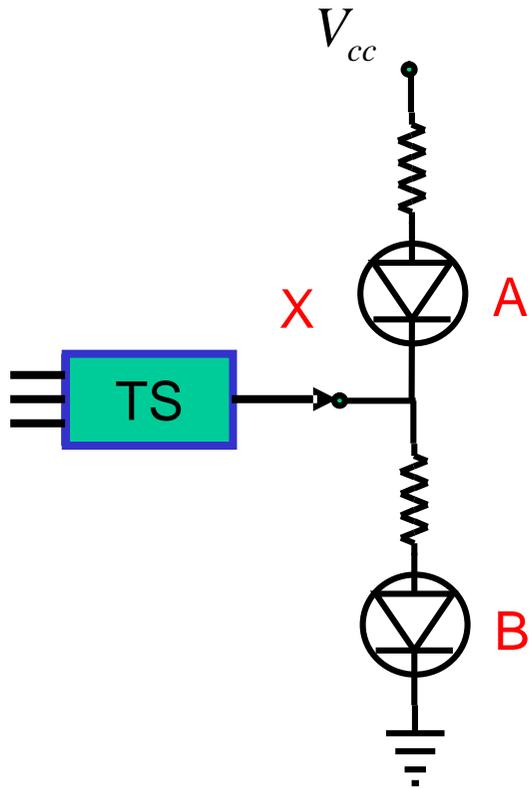
$X=0 \rightarrow \text{LED} = 1$

$X=OC \rightarrow \text{LED} = 0$

Visualizzare gli stati di un'uscita

Tri-State (TS)

- Con un solo LED attivo basso o attivo alto non si possono visualizzare i 3 stati
→ usiamo 1 LED AB (A) e 1 LED AA (B)



1 LED AA e 1 LED AB

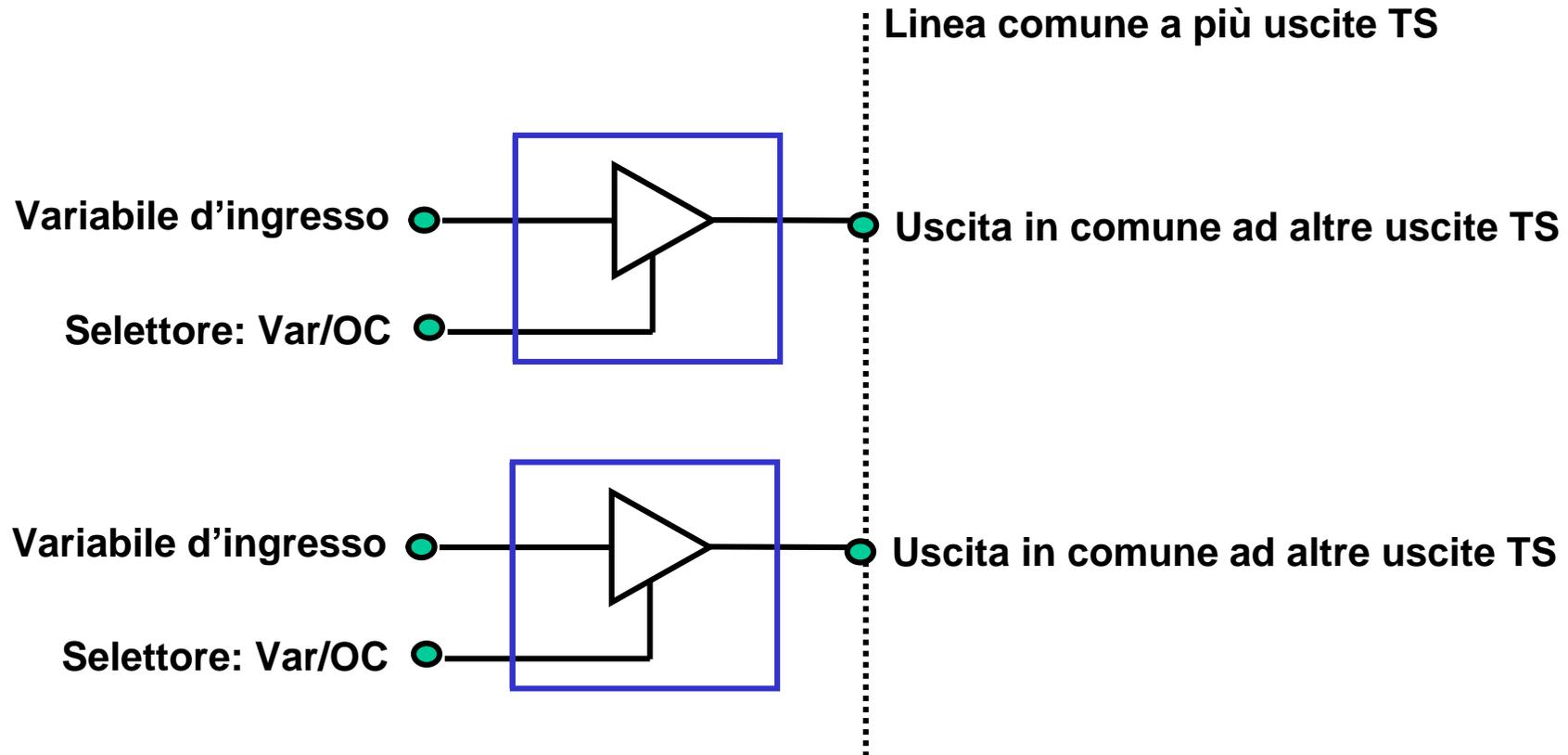
$X=0 \rightarrow \text{LED-A} = 1 \quad \text{LED-B} = 0$

$X=1 \rightarrow \text{LED-A} = 0 \quad \text{LED-B} = 1$

$X=OC \rightarrow \text{LED-A} = \frac{1}{2} \quad \text{LED-B} = \frac{1}{2}$

*circa metà intensità luminosa
ad ogni LED è applicata $V_{cc}/2$*

Tipico IC: buffer Tri-State



- ***Le uscite assumeranno il valore OC (Alta Impedenza) quando il selettore disabilita il buffer***
- ***Le uscite dovranno essere abilitate una per volta !!!***

Attività di Laboratorio
ELETTRONICA DEI
SISTEMI DIGITALI

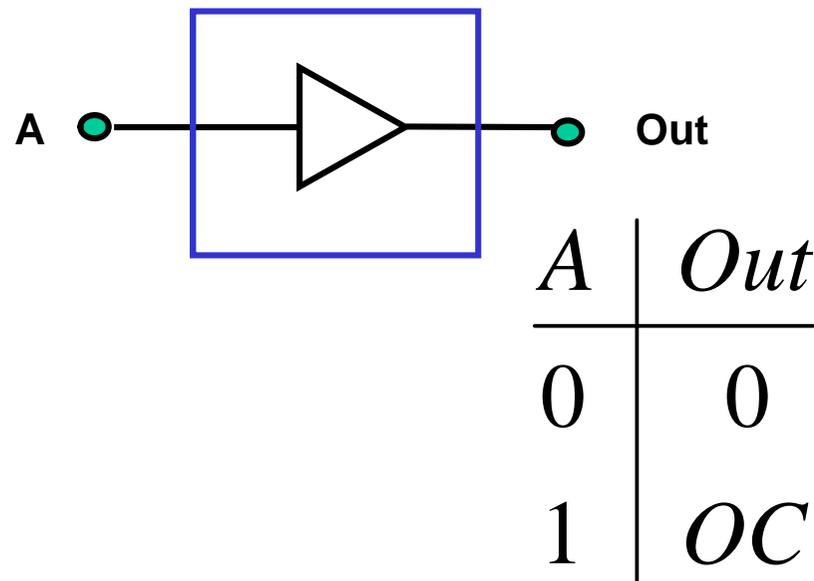
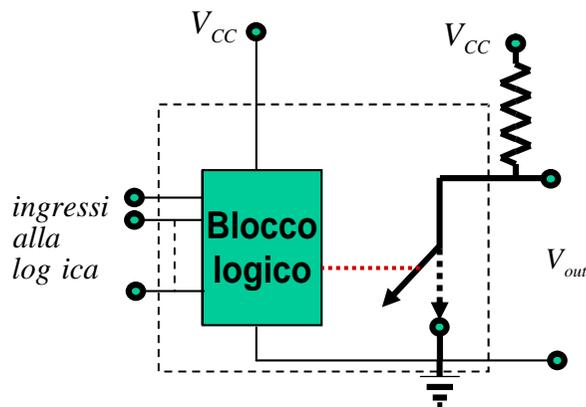
Parte IV

Corso di Laurea in Informatica e TFI
Anno Accademico 2007-2008

Esperienza OC

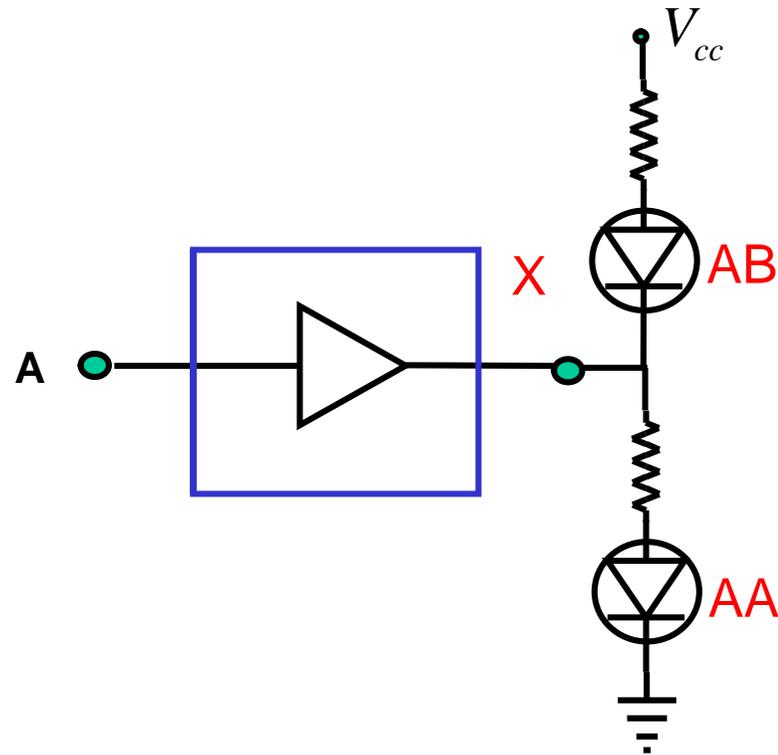
Comprende le prove:

1. Uso di un IC 7407 e uso di dispositivi LED attivi bassi e attivi alti e loro combinazione per esaminare lo stato delle uscite TP e OC.
2. Uso del DMM per la misura della tensione all'uscita dell'IC in configurazione TP e OC.
3. Realizzazione di AND cablato.



Esperienza OC

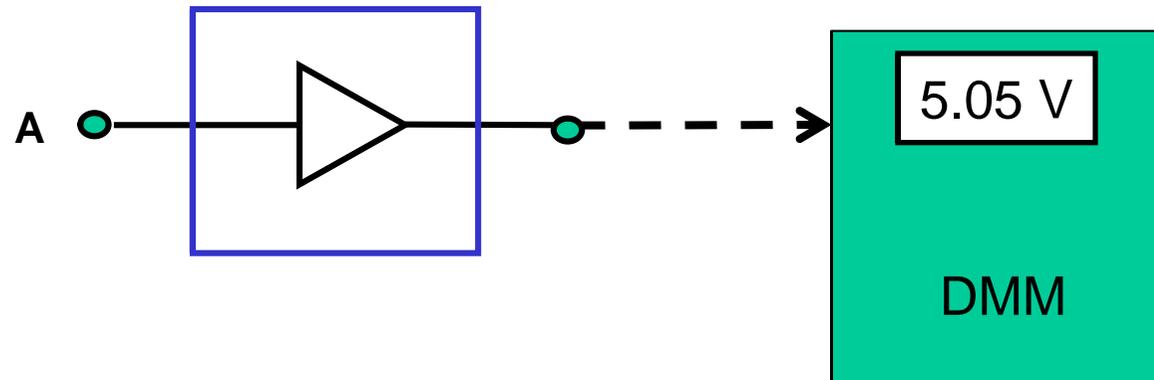
1. **Uso di un IC 7407 e uso di dispositivi LED attivi bassi e attivi alti e loro combinazione per esaminare lo stato delle uscite TP e OC.**



Scrivere la tavola della verità con le indicazioni della luminosità dei led AB e AA

Esperienza OC

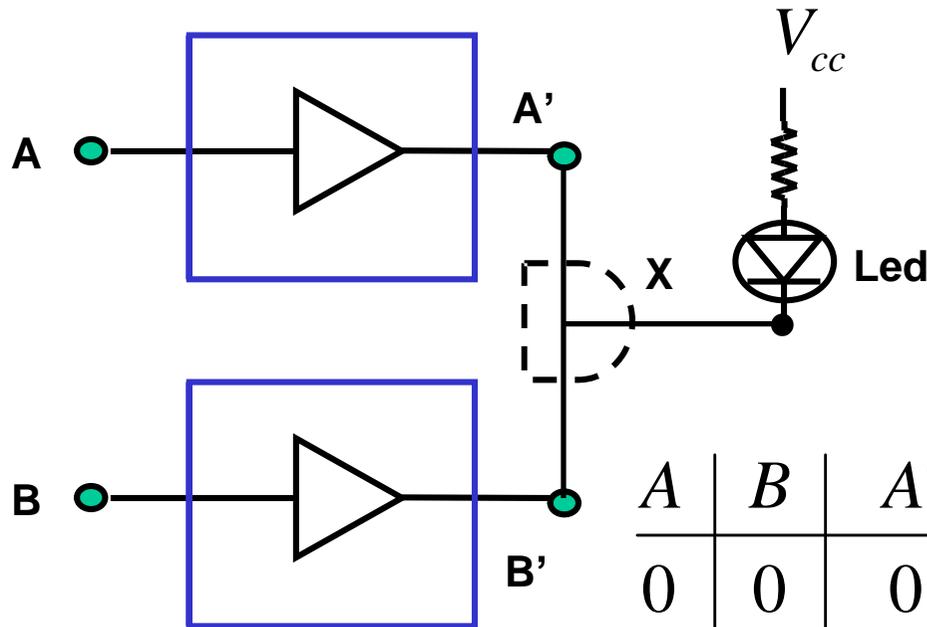
2. Uso del DMM per la misura della tensione all'uscita dell'IC in configurazione TP e OC.



Scrivere una tabella riportando le misure del DMM in corrispondenza dei valori di A

Esperienza OC

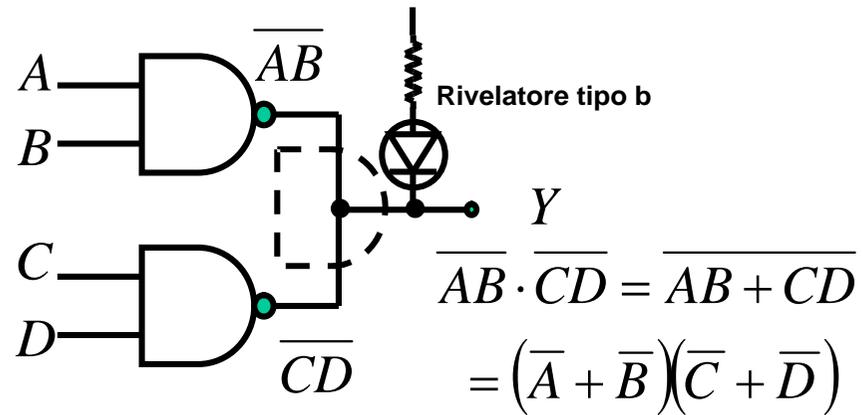
3. Realizzazione di AND cablato.



<i>A</i>	<i>B</i>	<i>A'</i>	<i>B'</i>	<i>X</i>	<i>Led</i>	<i>A · B</i>
0	0	0	0	0	<i>On</i>	0
0	1	0	<i>OC</i>	0	<i>On</i>	0
1	0	<i>OC</i>	0	0	<i>On</i>	0
1	1	<i>OC</i>	<i>OC</i>	1	<i>Off</i>	1

Scrivere la tabella della verità facendo le possibili combinazioni fra A e B

In alternativa si può usare una 7401: quadrupla NAND OC:



A	B	C	D	Y
0	x	0	x	1
0	x	x	0	1
x	0	x	0	1
x	0	0	x	1
1	1	x	x	0
x	x	1	1	0

Attività di Laboratorio
ELETTRONICA DEI
SISTEMI DIGITALI

Progetti di fine corso

Corso di Laurea in Informatica e TFI
Anno Accademico 2007-2008

1. Esperienza Sommatore/Sottrattore

Comprende le prove:

- **Uso di un sommatore a 4 bit in unione ad una memoria per:**
 - **Addizionare numeri binari**
 - **Sottrarre numeri binari**
 - **Realizzare un collegamento E.A.C. (End Around Carry)**
- **Abbiamo 4 switch per impostare 2 numeri**
 - **usiamo una memoria (IC '95 o '75) per memorizzare un numero**
 - **usiamo i 4 switch per impostare il secondo numero**
- **Per sottrarre dobbiamo complementare a 1 (vedi slide 3):**
 - **XOR per invertire 1 \leftrightarrow 0 (2 IC '86 con 4 XOR)**
- **Sommatore**
- **Una logica di controllo per selezionare Somma/Sottrazione**

Sottrazione alternativa = Somma + Complemento a 1

- Eseguire il complemento a 1 del sottraendo
- Sommare il risultato di a. al minuendo

c. Il minuendo è maggiore del sottraendo?

SI:

d. Sommare il riporto a LSB (E.A.C.)

$$\begin{array}{r}
 1110 - \\
 0110 = \\
 \hline
 \end{array}
 \rightarrow \text{compl.}(1) \rightarrow
 \begin{array}{r}
 1110 + \\
 1001 = \\
 \hline
 10111 \\
 \boxed{1} \rightarrow \\
 \hline
 1000
 \end{array}$$

Lo riportiamo (EAC):

1000

No

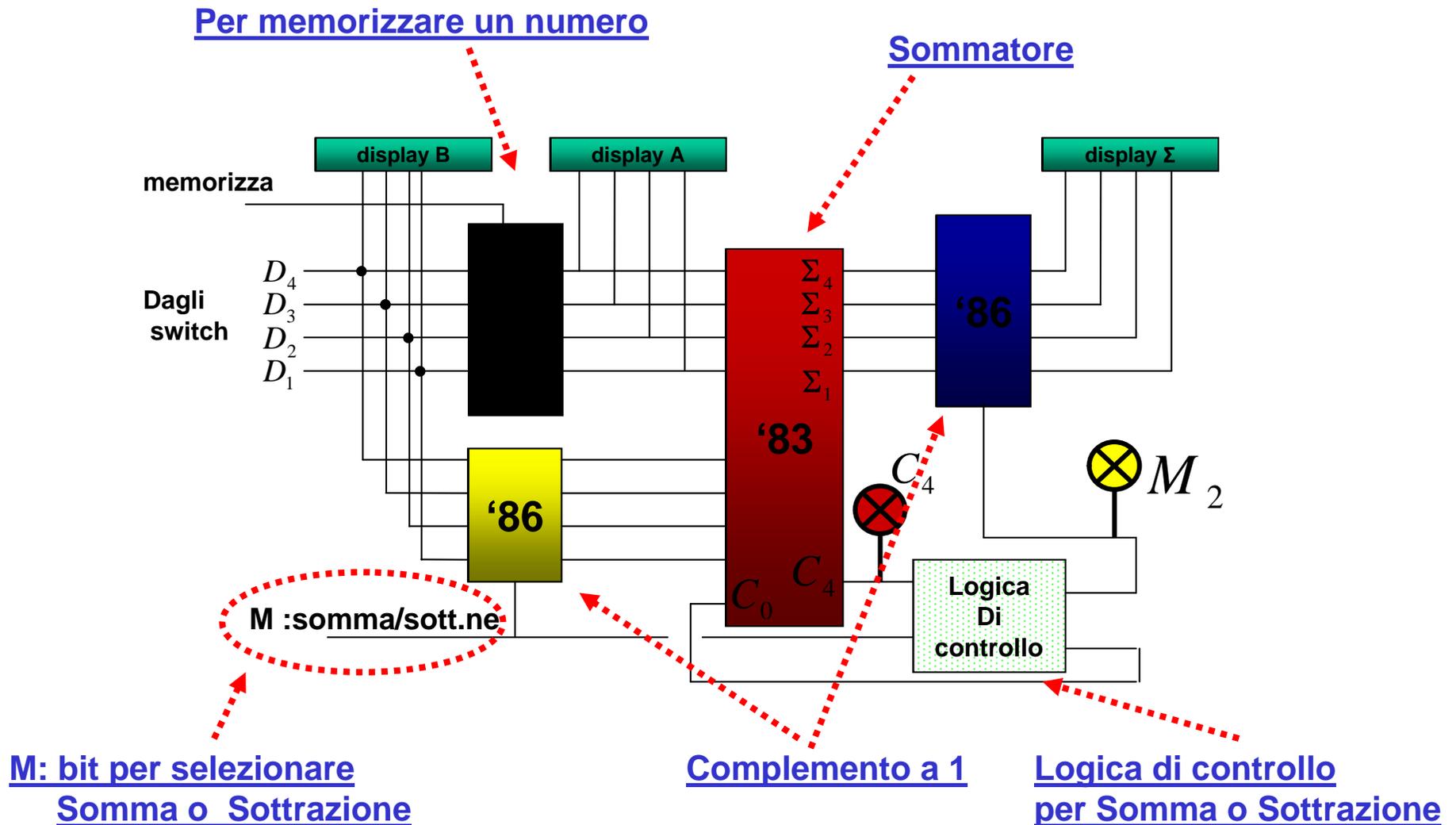
d. Complementare il risultato e CHS

$$\begin{array}{r}
 0110 - \\
 1110 = \\
 \hline
 \end{array}
 \rightarrow \text{compl.}(1) \rightarrow
 \begin{array}{r}
 0110 + \\
 0001 = \\
 \hline
 0111 \\
 \leftarrow
 \end{array}$$

compl.(1)
e CHS

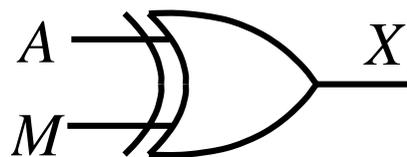
-1000

Schema



Complemento a 1

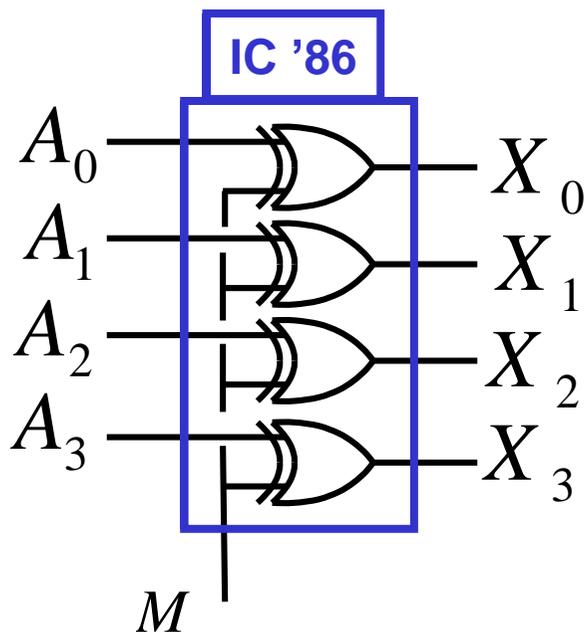
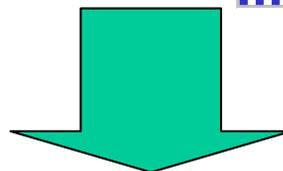
➤ IC '86: 4 XOR



A	M	X
0	0	0
0	1	1
1	0	1
1	1	0

Se $M=0 \rightarrow X=A$

Se $M=1 \rightarrow X=\overline{A}$
 $\rightarrow X=1-A$

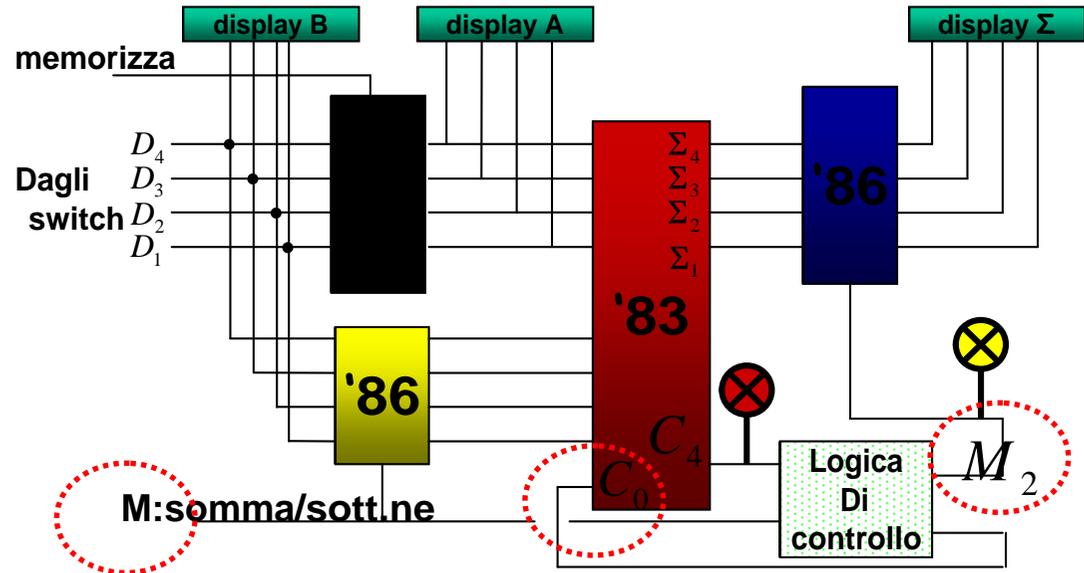


$$M = 0 \Rightarrow X_i = A_i$$

$$M = 1 \Rightarrow X_i = \overline{A_i} = 1 - A_i$$

$$\Rightarrow X_3 X_2 X_1 X_0 = 1111 - A_3 A_2 A_1 A_0$$

Logica di controllo e complemento a 1



minuendo > sottraendo

$$\begin{array}{r}
 1110 - \\
 0110 = \\
 \hline
 \end{array}
 \rightarrow \text{compl.}(1) \rightarrow
 \begin{array}{r}
 1110 + \\
 1001 = \\
 \hline
 10111 \\
 \rightarrow 1 \\
 \hline
 1000
 \end{array}$$

Lo riportiamo (EAC):

- complemento a 1 sottraendo → M = 1
- Alla somma sommiamo il riporto 1 → C₀ = 1
- Nessun complemento del risultato → M₂ = 0

sottraendo > minuendo

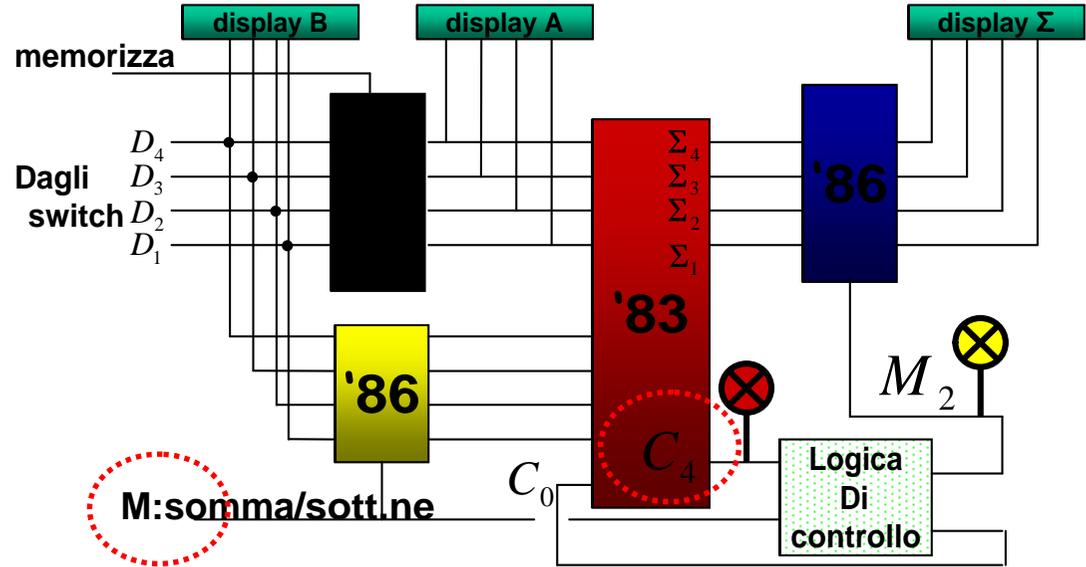
$$\begin{array}{r}
 0110 - \\
 1110 = \\
 \hline
 \end{array}
 \rightarrow \text{compl.}(1) \rightarrow
 \begin{array}{r}
 0110 + \\
 0001 = \\
 \hline
 0111 \\
 \leftarrow \text{compl.}(1) \text{ e CHS} \\
 \hline
 -1000
 \end{array}$$

- complemento a 1 sottraendo → M = 1
- nessun riporto da sommare → C₀ = 0
- complemento della somma (e CHS indicato dal led M₂) → M₂ = 1

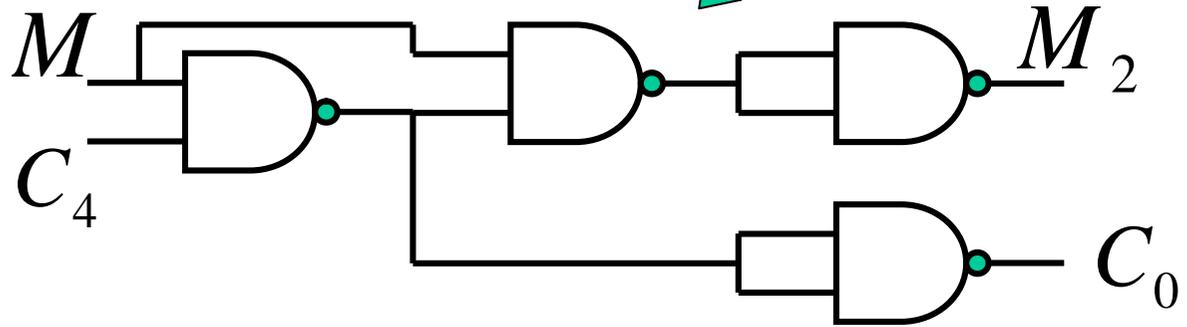
Logica di controllo

M=0 → somma
 → C_4 è un vero riporto
M=1 → sottrazione:
 • se $C_4=1 \rightarrow C_0=1$
 no complemento
 • se $C_4=0 \rightarrow C_0=0$
 e si complementa

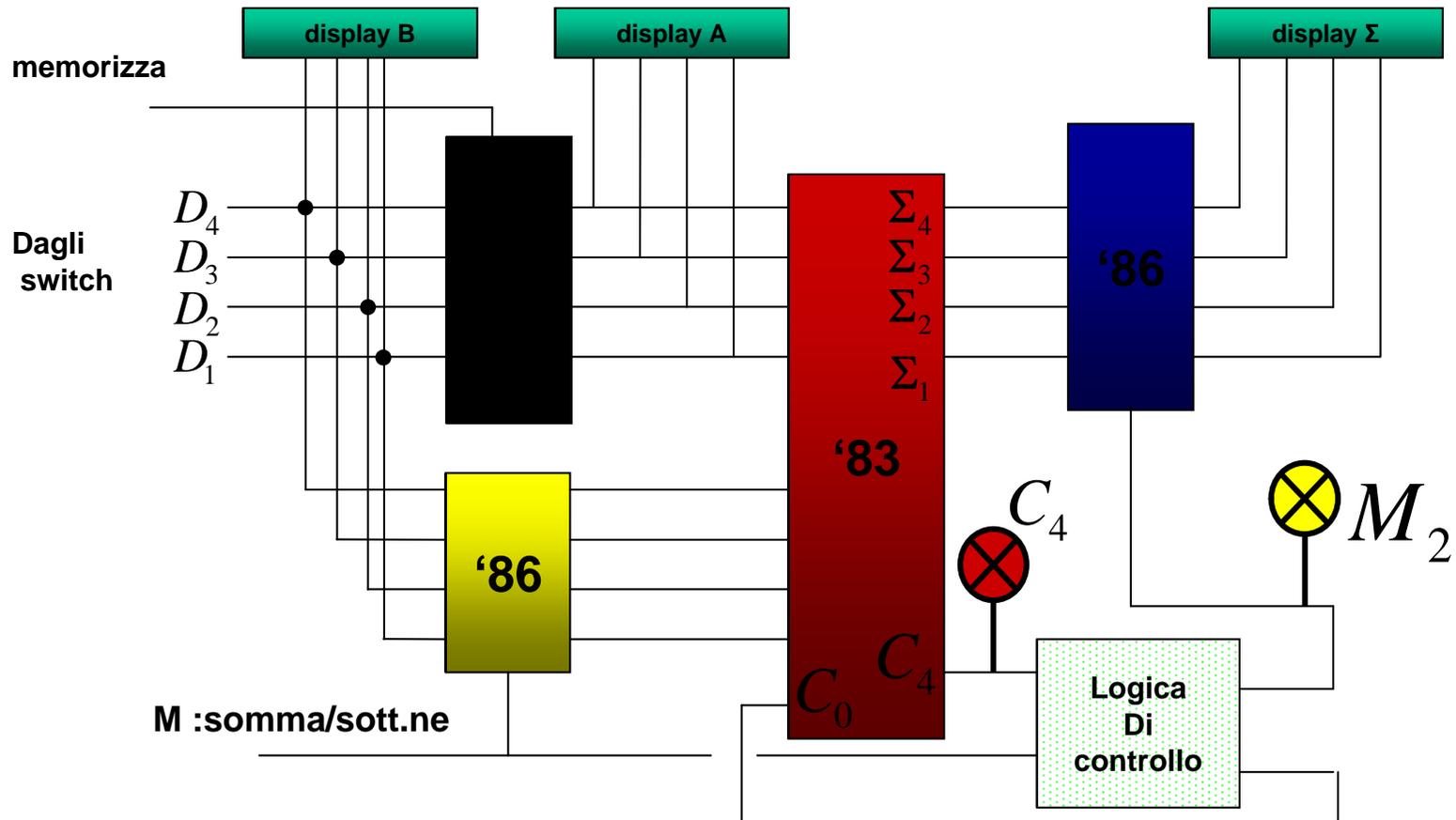
M	C_4	C_0	M_2
0	X	0	0
1	0	0	1
1	1	1	0



$$\begin{aligned}
 C_0 &= M C_4 = \overline{\overline{M C_4}} \\
 M_2 &= M (\overline{M} + \overline{C_4}) = M \cdot \overline{M C_4}
 \end{aligned}$$



Schema



Provare il sommatore

- **Somma (M=0) senza riporto:**

$$0101 + 0011 = 1000$$

- **Somma (M=0) con riporto:**

$$1011 + 0111 = 10010 \rightarrow 0010 \text{ con riporto di } 1$$

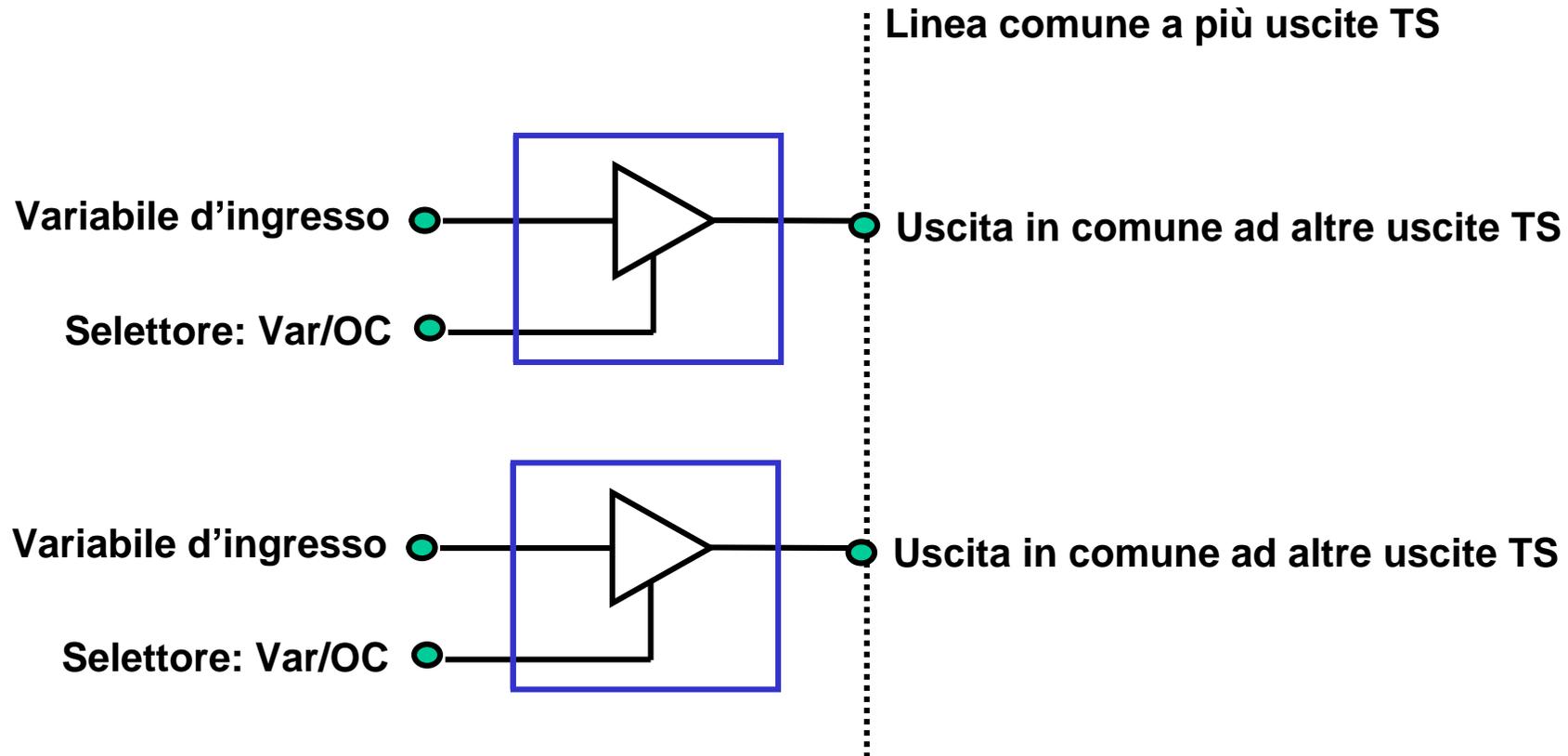
- **Sottrazione (M=1) con minuendo > sottraendo:**

$$1110 - 0110 = 1000$$

- **Sottrazione (M=1) con sottraendo > minuendo:**

$$0110 - 1110 = -1000 \rightarrow 1000 \text{ con } M_2=1$$

2. Comunicazione dati tramite Bus in comune

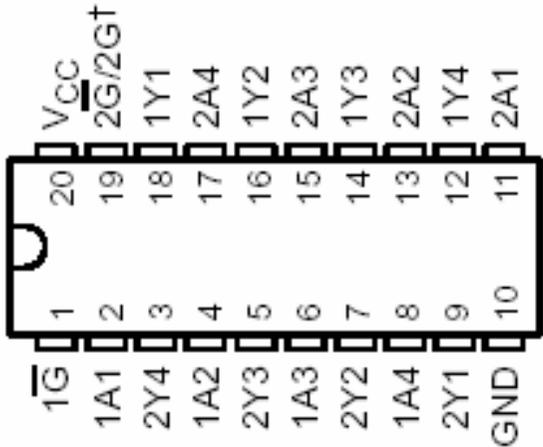


- ***Le uscite assumeranno il valore OC (Alta Impedenza) quando il selettore disabilita il buffer***
- ***Le uscite dovranno essere abilitate una per volta !!!***

3. Collaudare un buffer 74240 usando il DMM e tutti i rivelatori visti:

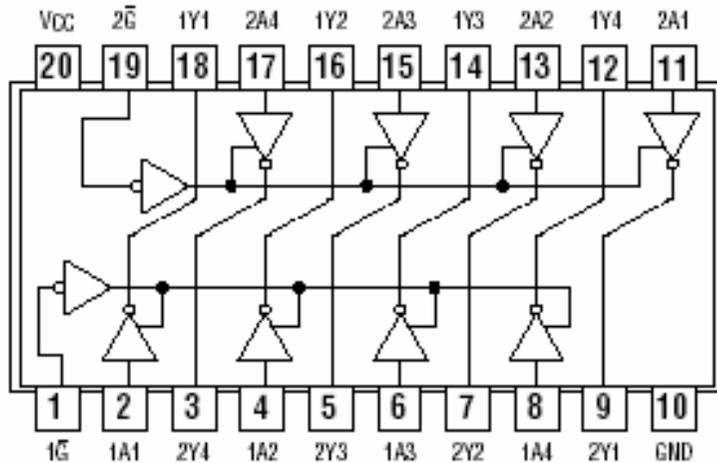
- Pilotare con gli switch i dati in ingresso e l'abilitazione G

SN54LS', SN54S' ... J OR W PACKAGE
 SN74LS240, SN74LS244 ... DB, DW, N, OR NS PACKAGE
 SN74LS241 ... DW, N, OR NS PACKAGE
 SN74S' ... DW OR N PACKAGE
 (TOP VIEW)

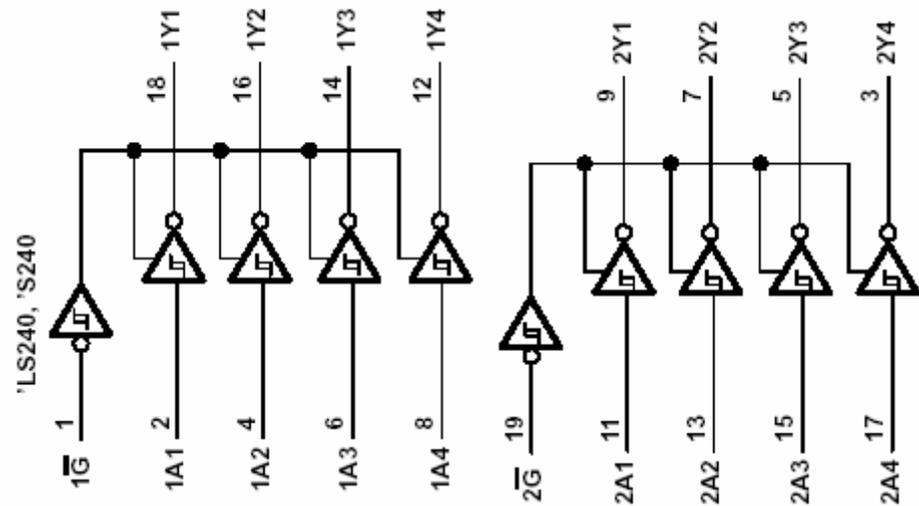


240

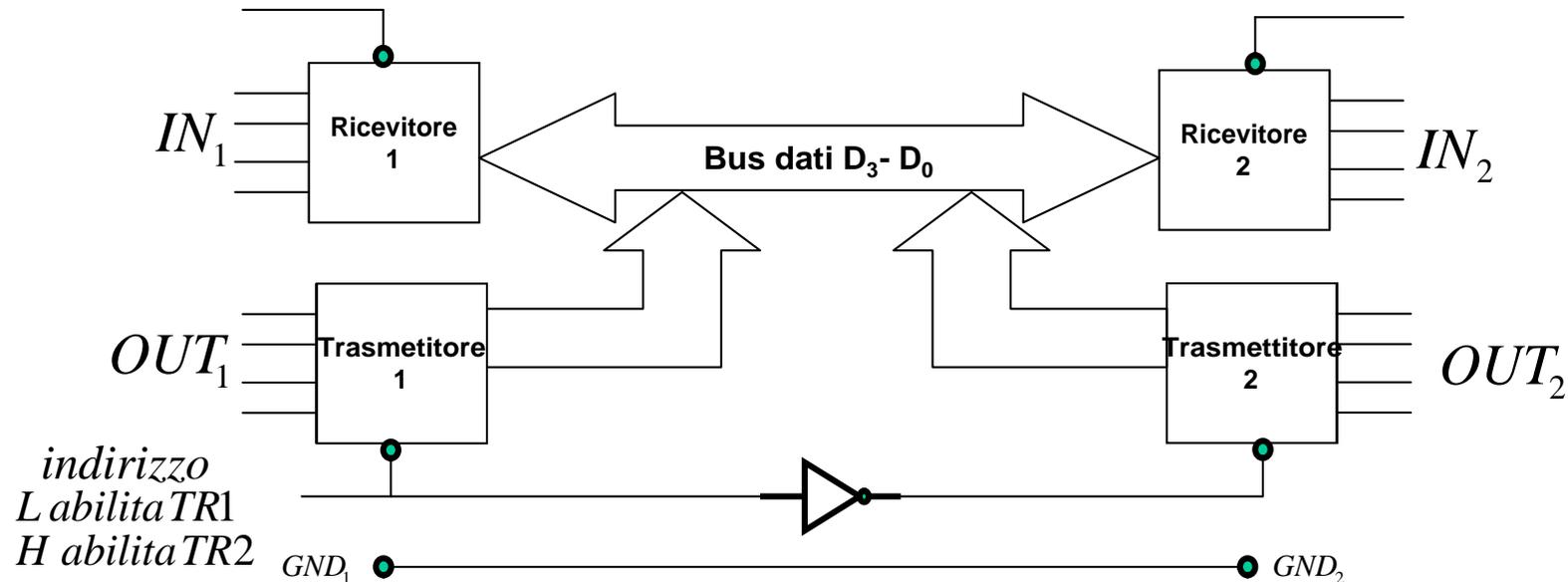
OCTAL BUFFERS/LINE DRIVERS/LINE RECEIVERS



logic diagram



4. **Uso del buffer 3-state per la trasmissione bidirezionale di dati a 4 bit tramite bus di comunicazione e linea di indirizzo che abilita una stazione alla volta.**



Due stazioni ricetrasmittenti (due gruppi frontali) comunicano attraverso un BUS di dati. Ciascuna è identificata da un codice di indirizzo (0=master;1=Slave) ed è abilitata a trasmettere solo quando il bit di indirizzo corrisponde al suo codice.

La linea GND1-GND2 **è fondamentale!!!**

3. Inserimento dati da tastiera

Comprende le prove:

- **Realizzazione di trasmissione dati da tastiera tramite l'uso di:**
 - ✓ **Semplice tastierino numerico con pulsanti da 0 a 9 con contatti normalmente aperti**
 - ✓ **Un MUX 16-to-1 74150**
 - ✓ **2 contatori sincroni 74161 con preset**

Il circuito deve:

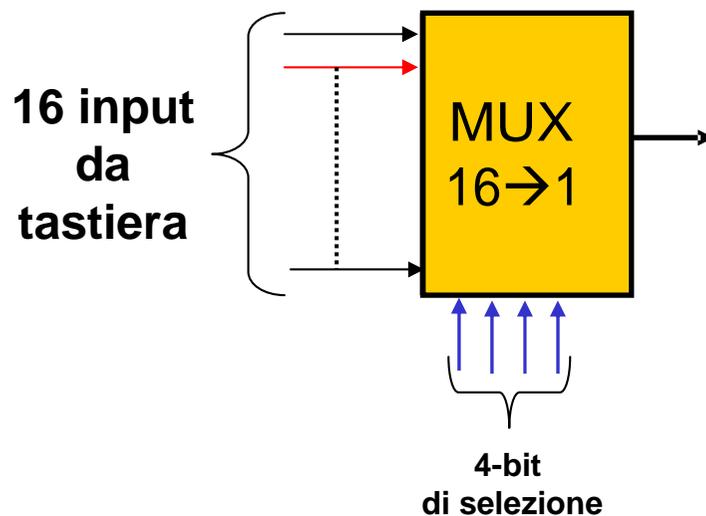
- **Segnalare se è stato premuto un tasto sul tastierino**
- **A tasto premuto generare il codice binario corrispondente**

Esistono diversi modi per realizzare tale circuito:

- **Utilizzo di un codificatore (circuito combinatorio) troppo semplice**
- **Metodo complesso come l'interfaccia PS/2**
- **Tecnica dello scanning**

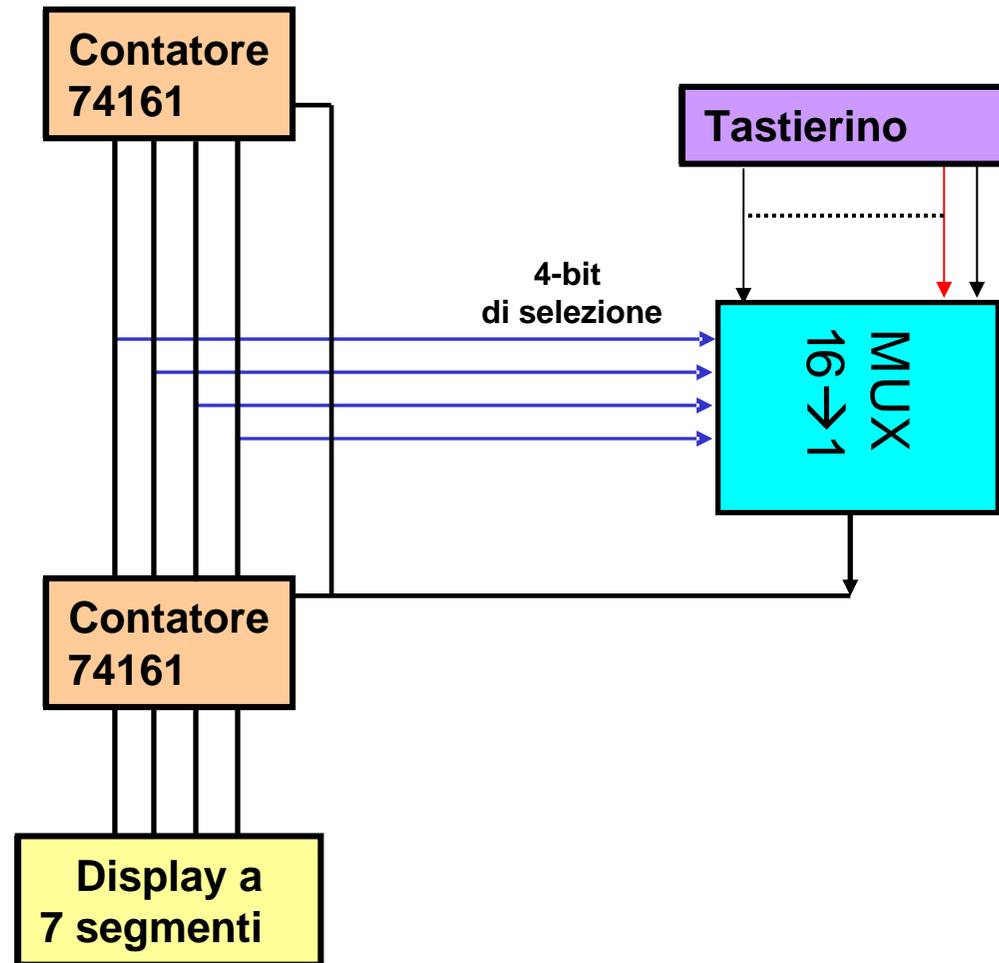
Tecnica dello scanning

- **Consiste nella scansione ciclica di tutti i pulsanti della tastiera:**
 - ✓ **Se non si trova nessun pulsante premuto si procede con lo scanning**
 - ✓ **Se si trova un pulsante premuto si procede alla decodifica come risultato dello scanning**



- **4-bit: ciclicamente assumono tutte le combinazioni da 0000 → 1111**
- **Ciclicamente Output del MUX sono gli Input da 0 → 15**
 - **Se nessun pulsante è premuto → in uscita al MUX c'è sempre 0 (o 1)**
 - **Se un pulsante è premuto, quando i bit di selezione smistano il corrispondente ingresso in uscita**
- **In uscita al MUX c'è un 1 (o 0)**
- **I bit di selezione sono proprio la combinazione binaria del numero decimale premuto sulla tastiera**

Realizzazione del circuito per la tecnica dello scanning



Esecuzione dell'esperienza

- **Provare il funzionamento dei contatori**
- **Provare il funzionamento del MUX**
- **Provare la catena Tastira-MUX inserendo i 4-bit con switch**
- **Realizzare il circuito completo collegando un display 7-seg anche ai 4-bit**
- **Collaudare il circuito con un clock lento (1-10 Hz)**
- **Osservare cosa accade premendo più di un tasto contemporaneamente**
- **Collaudare il circuito a frequenze più elevate (100 - 1000 Hz).**

4. Utilizzo di una Unità Aritmetico-Logica

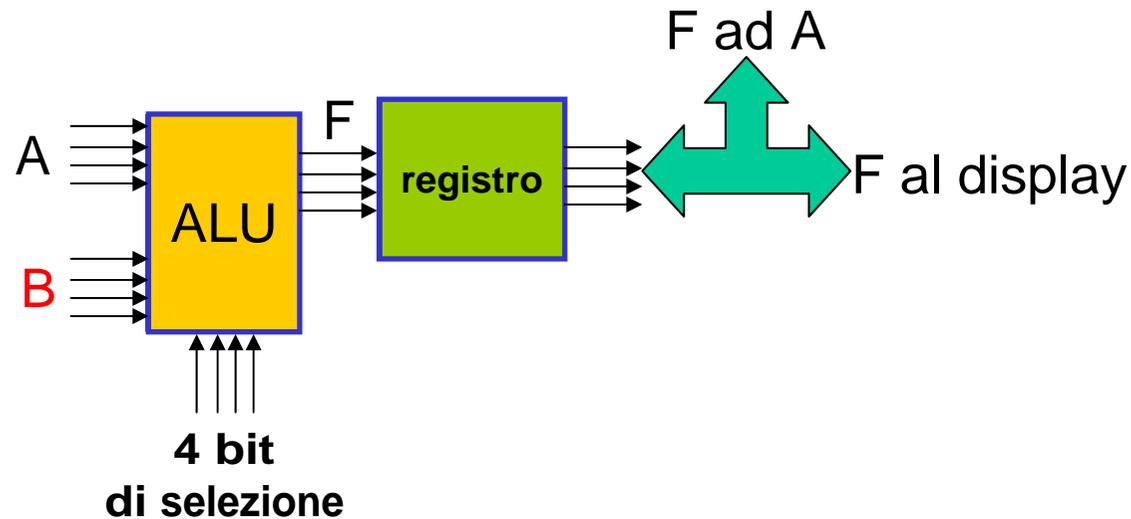
Comprende le prove:

- **Utilizzo di una ALU (Unità Aritmetico-Logico):**
 - ✓ Operazioni logiche fra 2 parole a 4 bit
 - ✓ Operazioni aritmetiche fra 2 o più parole a 4 bit
- ☐ **La ALU si utilizza in associazione con:**
 - ✓ Registro per la memorizzazione di una parola
 - ✓ Contraves per l'inserimento della seconda parola

La ALU è provvista di 4 bit di selezione per:

- Effettuare operazioni logiche
- Effettuare operazioni aritmetiche

ALU



- ✓ **A: è inserito mediante la memorizzazione di F**
- ✓ **B: è inserito con un contraves = 4 interruttori configurati da un comando rotativo**
- ✓ **4-bit di selezione: usiamo i soliti switch**

Esperienza con la ALU

- ✓ Operazioni logica fra A e B
- ✓ Operazioni aritmetiche fra A e B

- **Come si procede:**
 - ✓ per inserire A, impostare B, selezionare la funzione della ALU $F=B \rightarrow A=F$ in quanto abbiamo la retroazione
 - ✓ Impostare B
 - ✓ Impostare l'operazione desiderata con i 4-bit
 - ✓ Verificare il risultato

5. Misura di tempi e frequenze

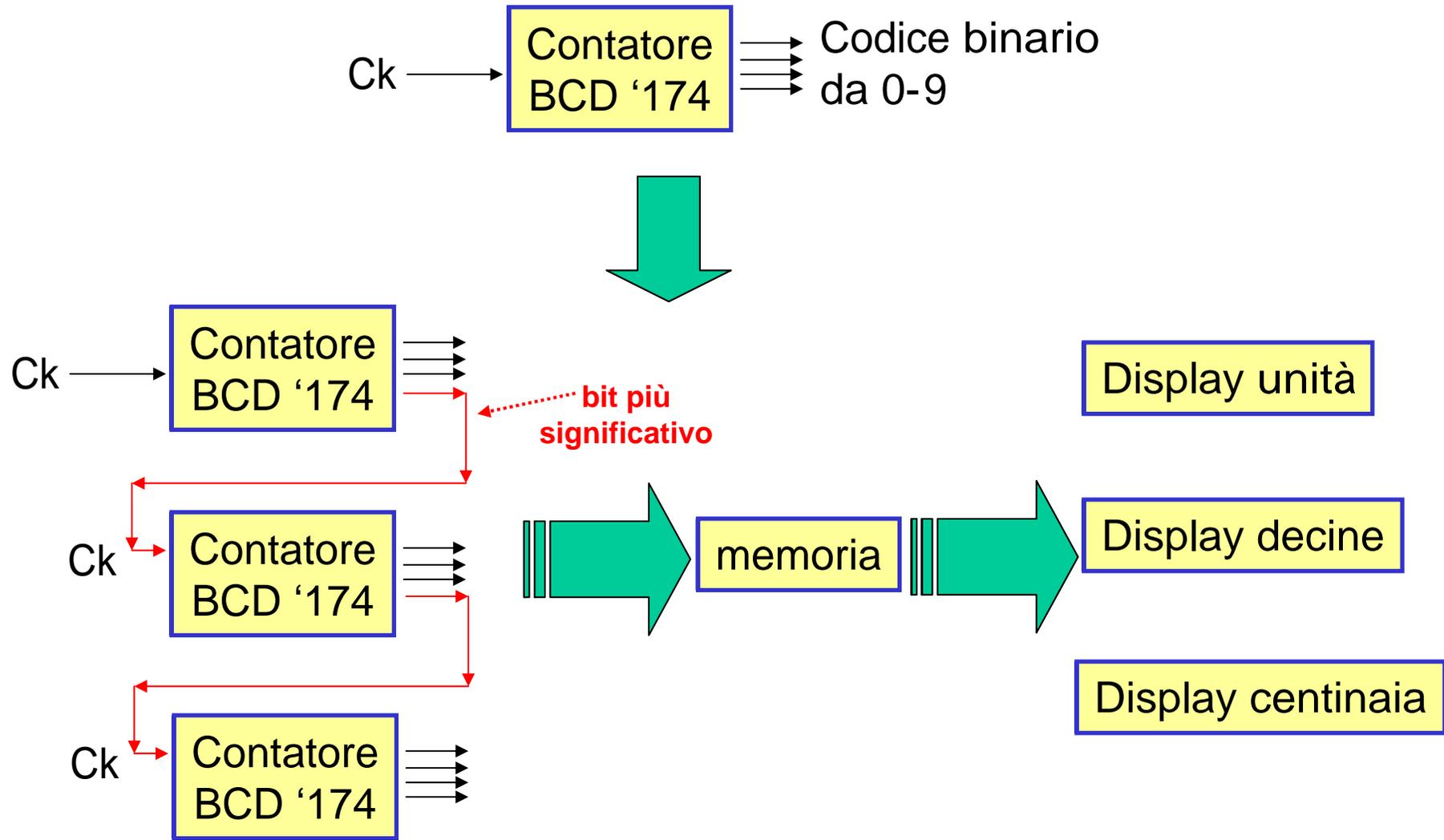
Comprende le prove:

- **Realizzazione di un contatore di impulsi TTL a 3 cifre decimali per:**
 - ✓ **Misura di frequenze**
 - ✓ **Misura di periodi di tempo**

IC utilizzato:

- **3 contatori BCD (decimale codificato in binario) → 12 bit**
- **2 memorie di 6 bit**
- **3 display 7-segmenti**

contatore



Misure di tempi o frequenze



- ✓ Realizzare il circuito indicato nelle dispense Lab
- ✓ Utilizzarlo per la misura di frequenza e di periodo di tempo

➤ Segnale incognito x con f_x

➤ Segnale di riferimento r con f_r (~ 2 Hz)

✓ $f_x > f_r \rightarrow$ x-Ck r-G \rightarrow misura di frequenza

✓ $f_x < f_r \rightarrow$ x-G r-Ck \rightarrow misura di periodo

$$f_x = \frac{2N}{T_r} \Rightarrow \frac{\Delta f_x}{f_x} = \frac{\Delta N}{N}$$

$$T_x = \frac{2N}{f_r} \Rightarrow \frac{\Delta T_x}{T_x} = \frac{\Delta N}{N}$$

➤ Scegliamo r secondo i seguenti criteri:

- ✓ effettuare la miglior misura sapendo che $\Delta N = \pm 1$
- ✓ effettuare la misura in tempi brevi

Altri progetti

- Realizzazione di un circuito per codifica Grey
- Realizzazione di circuiti per la codifica/decodifica Manchester e comunicazione dati fra 2 postazioni
- Realizzazione di un orologio digitale per contare secondi e minuti
- Realizzazione di un circuito per il controllo di un semaforo

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.