

# **Laboratorio di Elettronica e Tecniche di Acquisizione Dati 2024-2025**

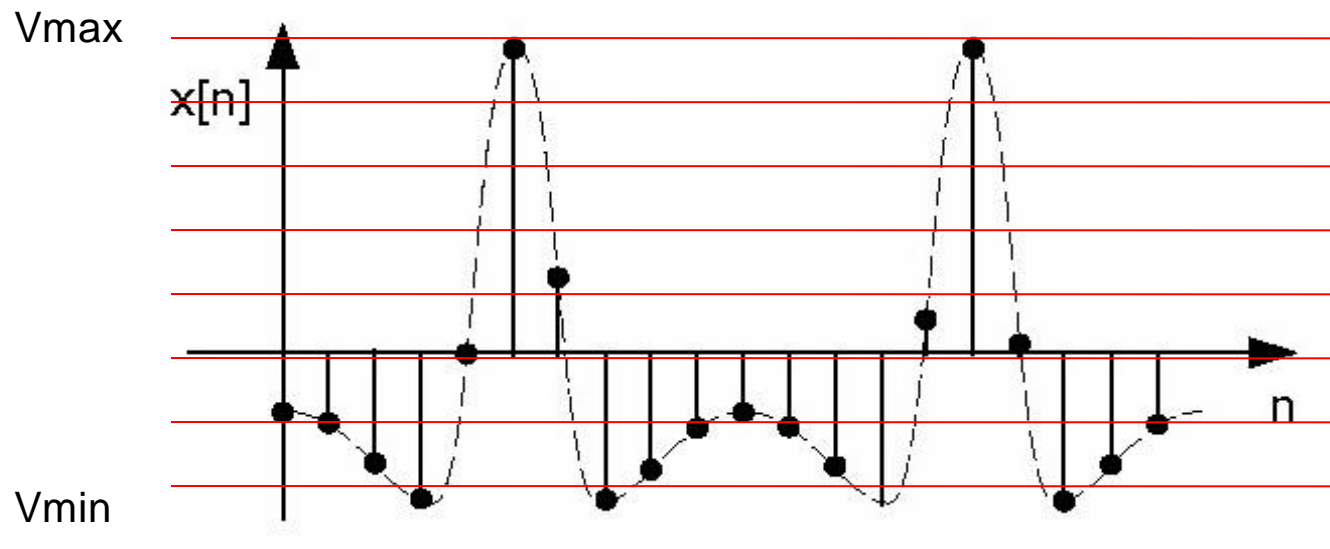
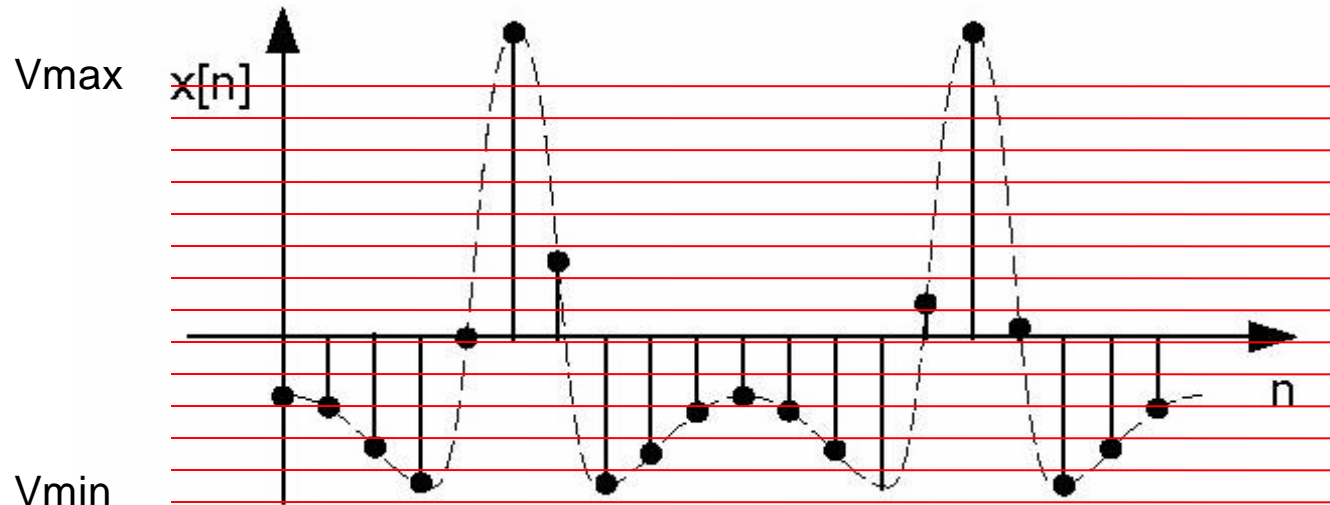
**Elettronica digitale (1<sup>a</sup> parte)**

**(cfr. <http://physics.ucsd.edu/~tmurphy/phys121/phys121.html>)**

# ADC (1)

- Dal punto di vista funzionale gli ADC sono dei *classificatori*:
  - l'intervallo di variabilità del segnale  $V_x$  viene diviso in  $n$  intervalli, detti *canali*, di ampiezza costante  $K$ . Definiamo quindi  $V_i = K i + V_o$
  - il segnale in ingresso  $V_x$  viene *classificato* nel canale  $i$ -esimo se è verificata la relazione
$$V_{i-1} < V_x < V_i$$
  - inevitabilmente si ha un errore di quantizzazione

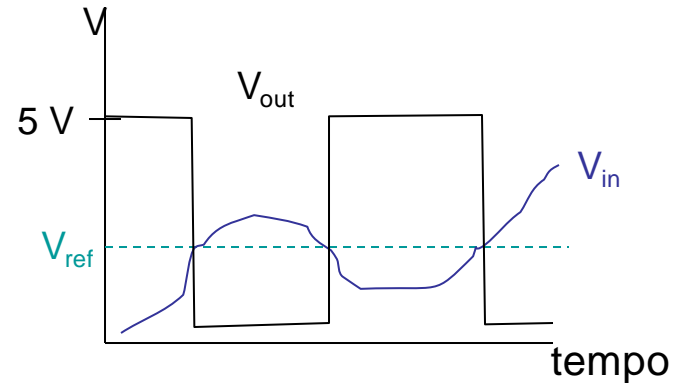
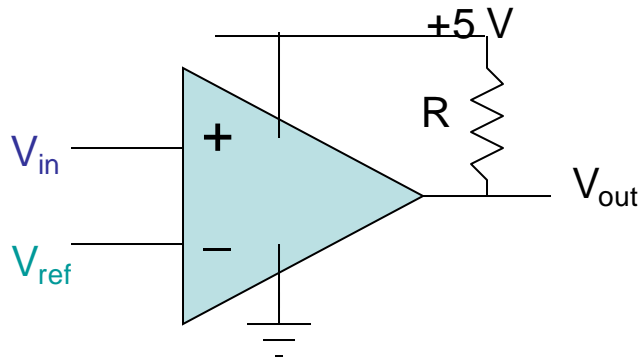
## ADC (2)



# Comparatori

- è spesso utile generare un segnale elettrico “forte” associato con un certo evento (cfr. *trigger*)
- possiamo utilizzare un comparatore per confrontare un segnale con una certa soglia
  - può essere una temperatura, una pressione, etc...: qualsiasi cosa che possa essere trasformata in un voltaggio
- possiamo utilizzare un operazionale senza feedback
  - input invertente alla soglia
  - input non-invertente collegato al segnale da testare
  - l'operazionale farà uscire un segnale (a fondo scala) negativo se il segnale è  $<$  della soglia, positivo se il segnale è  $>$  della soglia
- purtroppo l'operazionale è lento (basso “slew rate”)
  - $15 \text{ V}/\mu\text{s}$  significa  $2 \mu\text{s}$  per arrivare a fondo scala se alimentato  $\pm 15 \text{ V}$

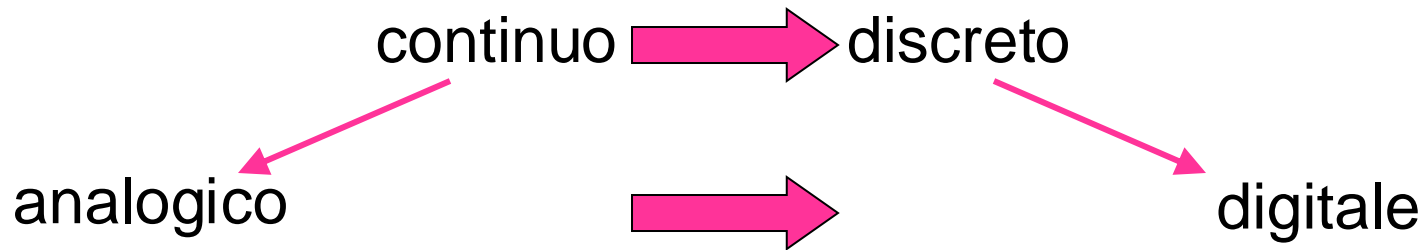
# Esempio (reale) di comparatore



- quando  $V_{in} < V_{ref}$ ,  $V_{out}$  va a  $V_{CC}^-$ , i.e. 0 (\*)
- quando  $V_{in} > V_{ref}$ ,  $V_{out}$  va a  $V_{CC}^+$ , i.e. 5V (\*)
  - nell'esempio è anche “pulled-up” (attraverso il resistore di “pull-up”, usualmente 1 k $\Omega$  o più): se la corrente richiesta dal carico è maggiore di quella possibile per l'op.amp (O(10 mA)), l'extra-corrente arriva dall'alimentazione esterna e non dall'op.amp
- l'uscita è una versione “digitale” del segnale
  - i valori “alto” e “basso” sono configurabili (ground e 5V, nell'esempio)
- possono essere utili anche per convertire un segnale “lento” in uno “veloce”
  - se è necessaria una maggiore precisione di “timing”

(\*) in realtà a meno di N potenziali di contatto...

# “digitale”



Stati logici solo due possibili stati  1, alto (H), vero (true)  
0, basso (L), falso (false)

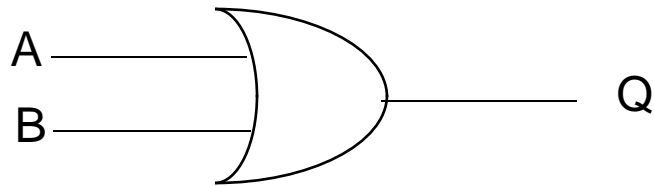
## Algebra booleana

sistema matematico per l'analisi di stati logici



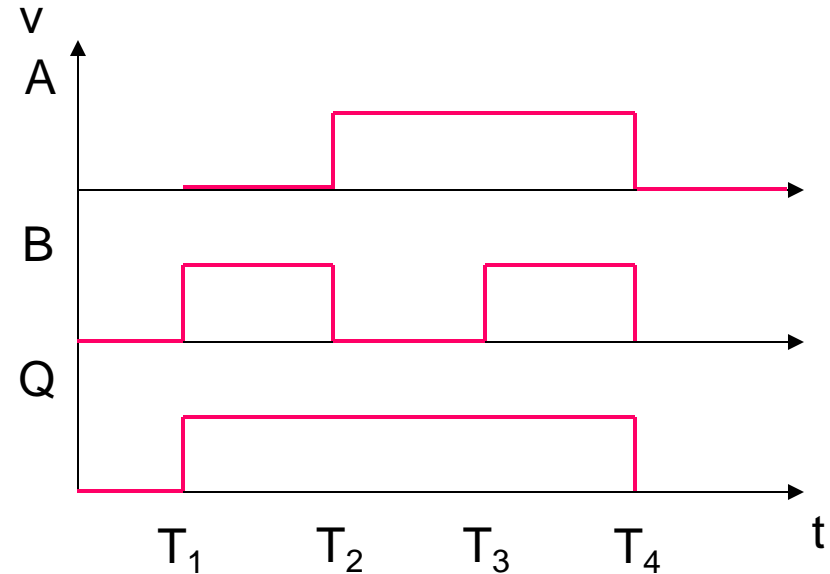
# Porte logiche di base - OR

**OR**



$$Q = A + B$$

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



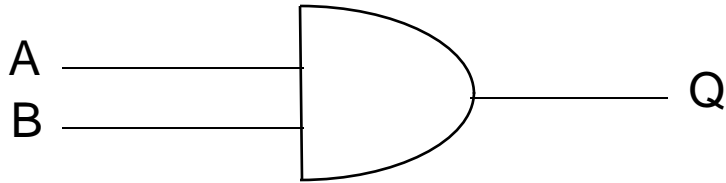
$$A + B + C = (A + B) + C = A + (B + C)$$

$$A + B = B + A$$

$$A + 1 = 1, A + A = A, A + 0 = A$$

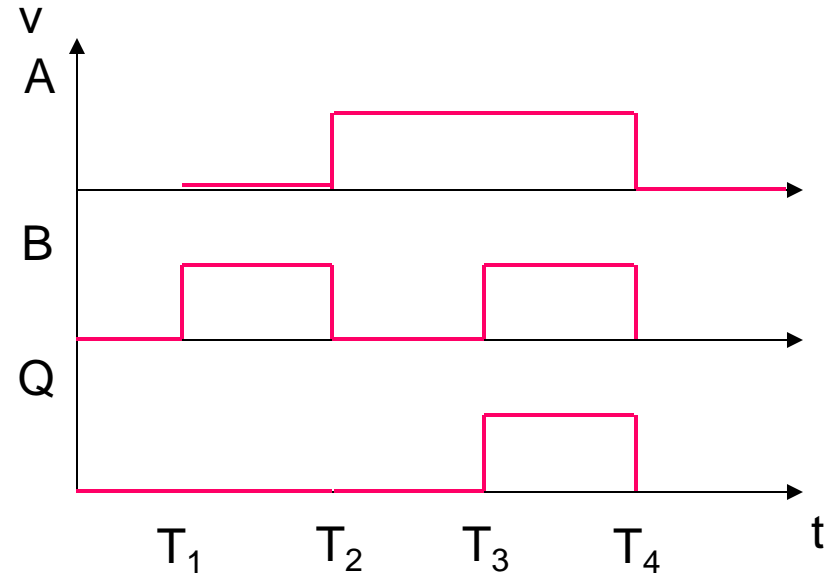
# Porte logiche di base - AND

**AND**



$$Q = A \cdot B$$

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



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

$$A \cdot B = B \cdot A$$

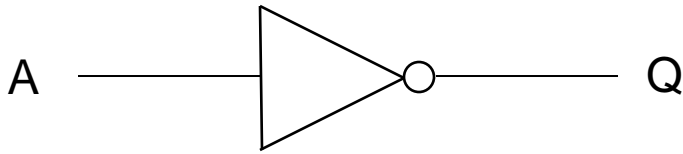
$$A \cdot 1 = A, A \cdot A = A, A \cdot 0 = 0$$

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



# Porte logiche di base - NOT

**NOT**



A	Q
0	1
1	0

$$\overline{\overline{A}} = A$$

$$\overline{\overline{A}} + A = 1$$

$$\overline{\overline{A}} \cdot A = 0$$

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

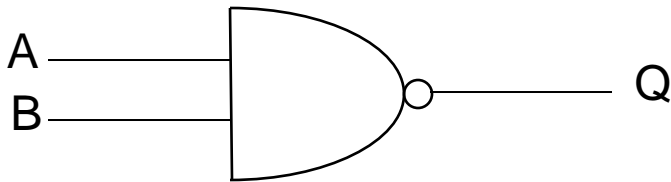
sapendo che

$$B + 1 = 1, A \cdot 1 = A, \overline{\overline{A}} + A = 1$$

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

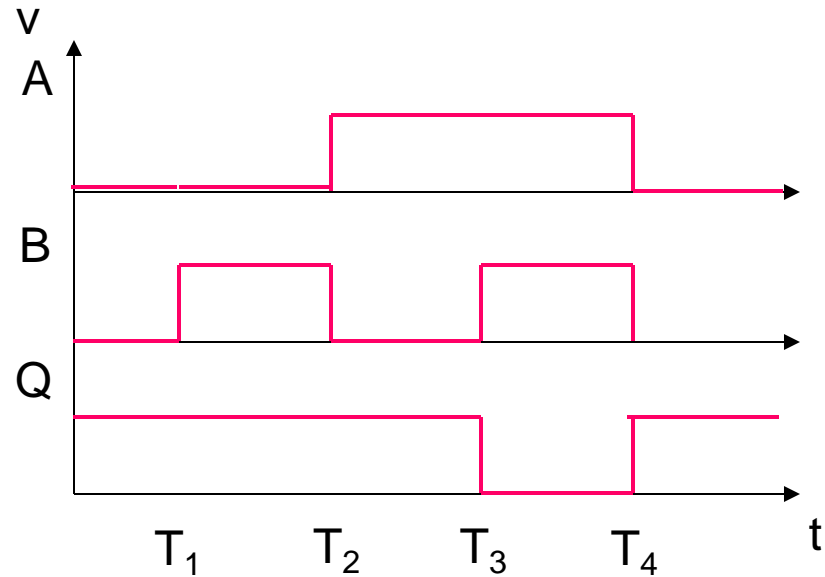
# Porte logiche di base – NAND

**NAND**



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

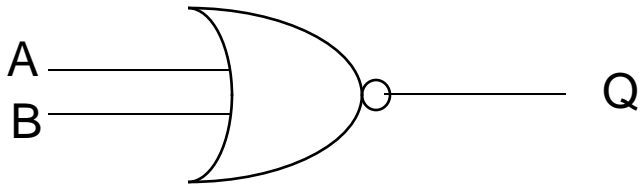
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



porta universale

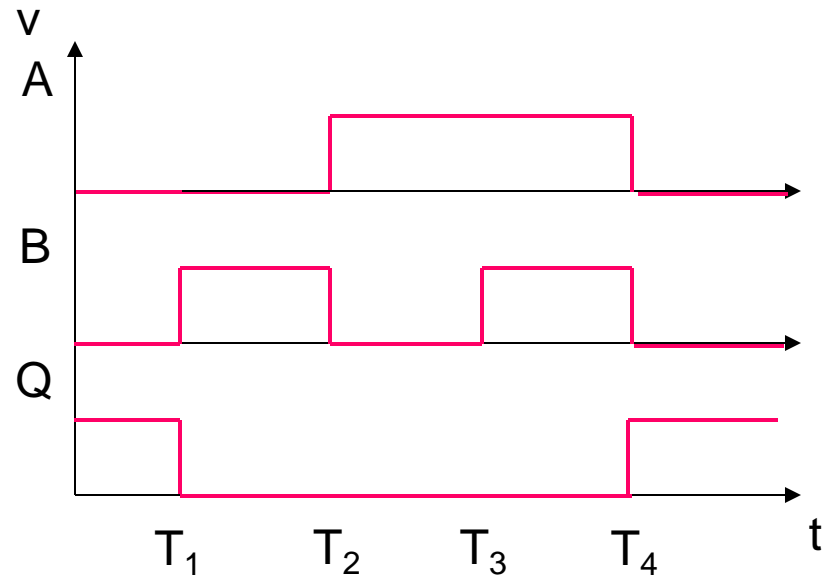
# Porte logiche di base – NOR

**NOR**



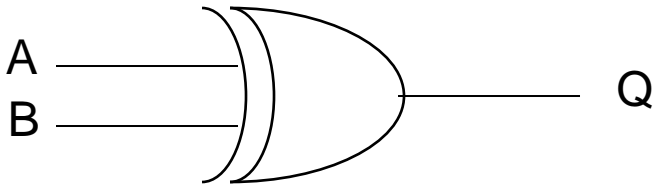
$$Q = \overline{A + B}$$

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



# Porte logiche di base – XOR

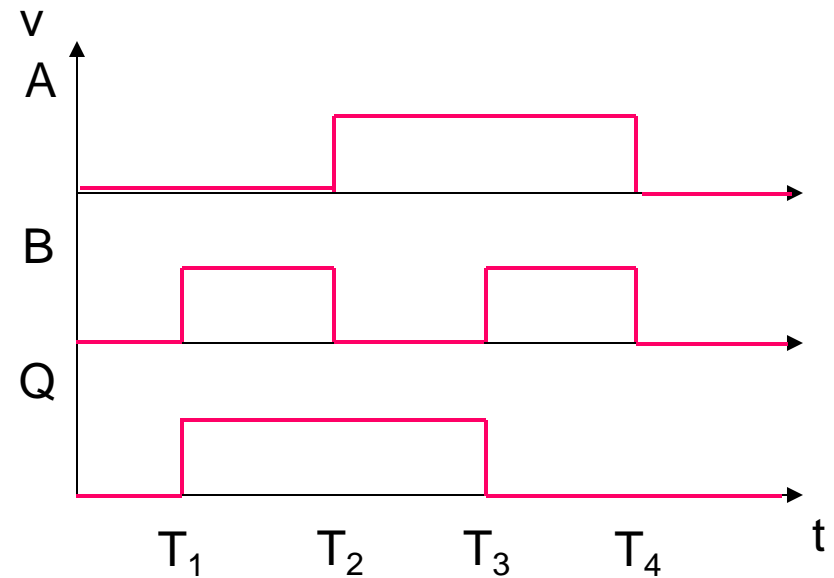
**XOR**



$$Q = A \oplus B$$

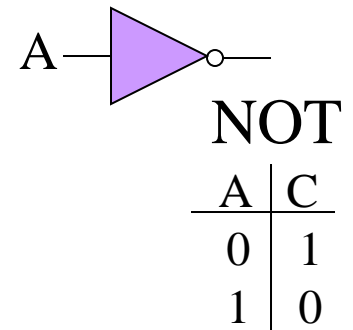
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

OR esclusivo



# Manipolazione dati

- tutta la “manipolazione” è basata sulla *logica*
- la logica segue regole ben precise, producendo uscite deterministiche, funzione solamente degli input



AND

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

OR

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

XOR

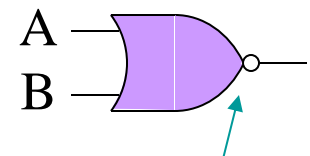
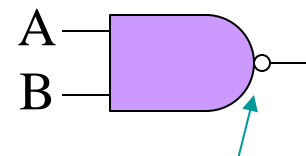
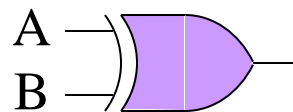
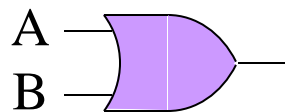
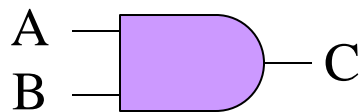
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

NAND

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

NOR

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



la “palletta” significa (e.g., NOT AND → NAND)

# Algebra Booleana

A	B	C	$f(A, B, C)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

## Prima forma canonica (esempio)

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

Ogni riga come prodotto (AND) dei termini naturali (se 1) o complementati (se 0)  
Somma (OR) delle righe con valore pari a 1.

## Seconda forma canonica (esempio)

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

Ogni riga come somma (OR) dei termini naturali (se 1) o complementati (se 0)  
Prodotto (AND) delle righe con valore pari a 0.

A	B	$f(A, B)$
0	0	0
0	1	1
1	0	0
1	1	0

# Algebra Booleana

A	B	C	$f(A, B, C)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Prima forma canonica (esempio)

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

Ogni riga come prodotto (AND) dei termini naturali (se 1) o complementati (se 0)

Somma (OR) delle righe con valore pari a 1.

Seconda forma canonica (esempio)

$$f(A, B) = (A + B)(\bar{A} + B)(\bar{A} + \bar{B})$$

Ogni riga come somma (OR) dei termini naturali (se 1) o complementati (se 0)

Prodotto (AND) delle righe con valore pari a 0.

A	B	$f(A, B)$
0	0	0
0	1	1
1	0	0
1	1	0

# Algebra Booleana

A	B	C	$f(A, B, C)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Prima forma canonica (esempio)

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

Ogni riga come prodotto (AND) dei termini naturali (se 1) o complementati (se 0).

Somma (OR) delle righe con valore pari a 1.

Seconda forma canonica (esempio)

$$f(A, B) = (A + B)(\bar{A} + B)(\bar{A} + \bar{B})$$

Ogni riga come somma (OR) dei termini naturali (se 1) o complementati (se 0)

Prodotto (AND) delle righe con valore pari a 0.

A	B	$f(A, B)$
0	0	0
0	1	1
1	0	0
1	1	0



# Algebra Booleana

A	B	C	$f(A, B, C)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

## Prima forma canonica (esempio)

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

Ogni riga come prodotto (AND) dei termini naturali (se 1) o complementati (se 0)  
Somma (OR) delle righe con valore pari a 1.

## Seconda forma canonica (esempio)

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

Ogni riga come somma (OR) dei termini naturali (se 1) o complementati (se 0)  
Prodotto (AND) delle righe con valore pari a 0.

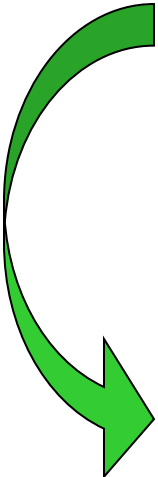
A	B	$f(A, B)$
0	0	0
0	1	1
1	0	0
1	1	0

# Algebra Booleana

Algebra booleana

trasformare una funzione logica in un' altra  
di più facile implementazione hardware

Teoremi di De Morgan


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

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

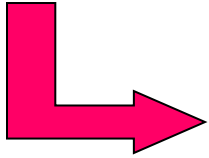
Il complemento dell' AND di più variabili  
logiche è dato dall' OR dei complementi

Il complemento dell' OR di più variabili  
logiche è dato dall' AND dei complementi



# Algebra Booleana

Un circuito AND per logica positiva funziona come un OR per logica negativa

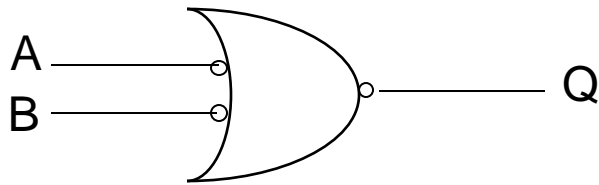


non è necessario usare i tre circuiti di base

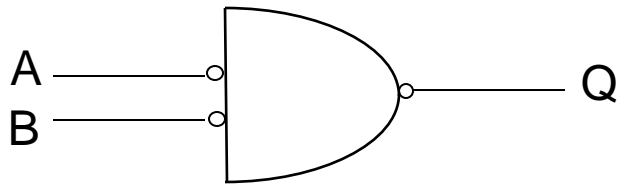
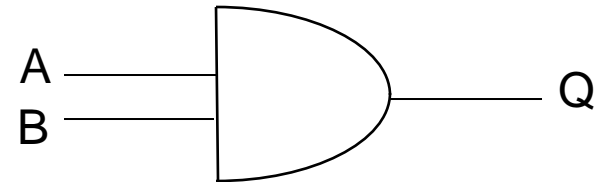
bastano due



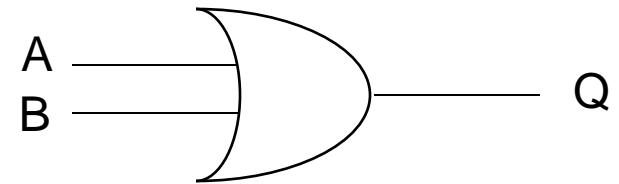
OR e NOT oppure AND e NOT



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



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



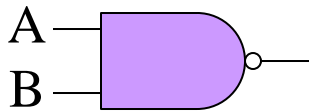
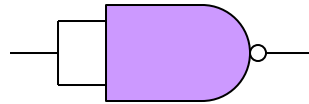
# Tutta la logica con la sola NAND

NAND

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

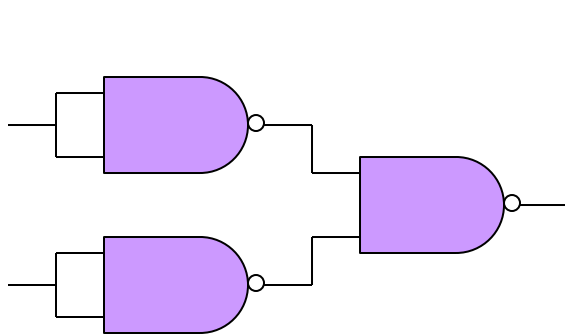
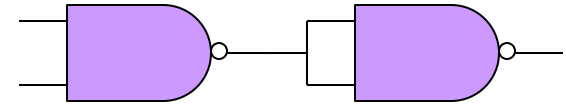
NOT

A	C
0	1
1	0



AND

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

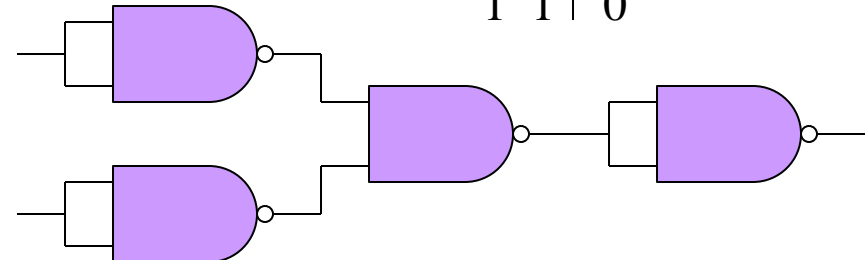


OR

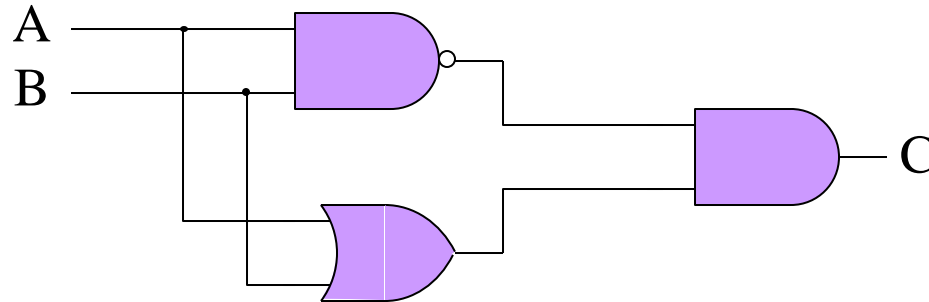
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

NOR

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



# Tutta la logica con la sola NAND



$$\text{XOR} = (A \text{ NAND } B) \text{ AND } (A \text{ OR } B)$$

- la OR già sappiamo come farla di sole porte NAND
- 6 NAND in totale: 3 per la OR, 2 per la AND e 1 per la NAND
- questa è una XNOR, che utilizzando un'altra NAND viene negata, cioè diventa la XOR desiderata

# Aritmetica

- sommiamo due numeri binari:

$$00101110 = 46$$

$$+ 01001101 = 77$$

$$01111011 = 123$$

- come lo abbiamo fatto? Definiamo le nostre “regole”:

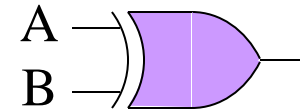
$$0 + 0 = 0;$$

$$0 + 1 = 1 + 0 = 1;$$

$$1 + 1 = 10 \text{ (2): (0, riporto 1);}$$

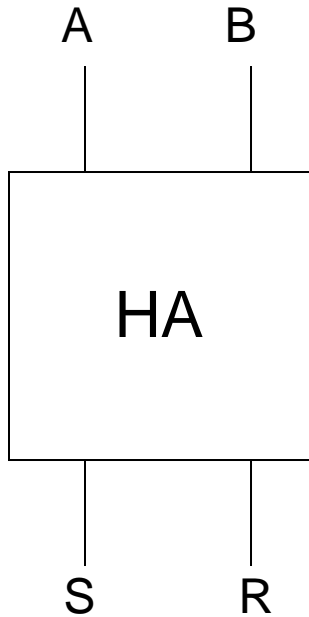
$$1 + 1 + (1 \text{ di riporto}) = 11 \text{ (3): (1, riporto 1)}$$

- proviamo ad associare una porta logica a queste “regole”
  - essendo una somma pensiamo subito alla OR
  - il caso “ $1+1=0$  (riporto1)” si adatta meglio alla XOR
  - ancora manca la gestione del riporto



XOR		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

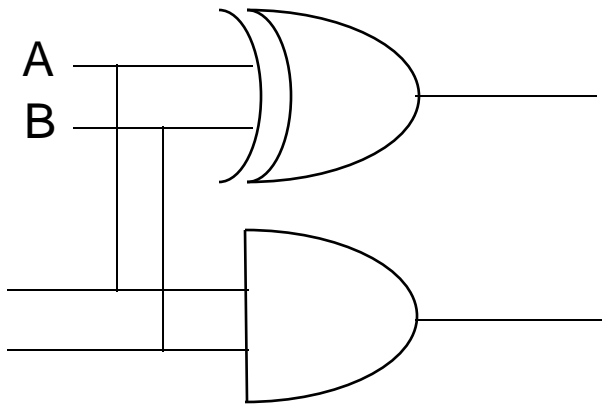
# Half Adder



A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

XOR

AND



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



$$R = A \cdot B$$

# Half Adder

Somma binaria è analoga alla somma decimale:

- 1) sommare i due bit corrispondenti al digit  $2^n$
- 2) sommare il risultato al riporto dal digit  $2^{n-1}$

Il circuito sommatore a due ingressi è detto Half Adder  
(ne occorrono due per fare una somma completa: riporto precedente!)

due input  i bit da sommare  
due output  la somma e il riporto

può essere costruito con i circuiti di base



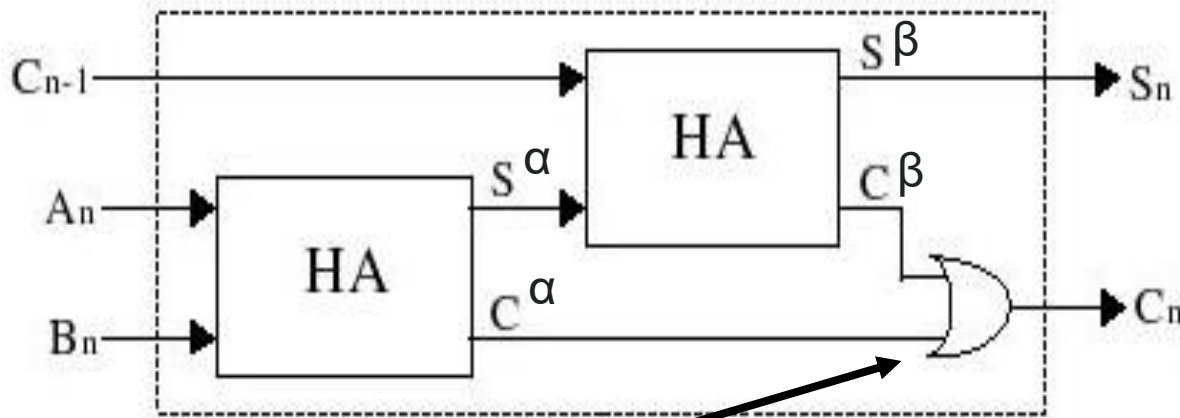
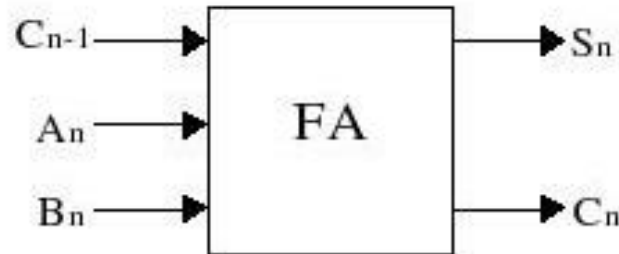
# Full Adder

Tabella di verità della somma di 3 bit

$A_n$	$B_n$	$R_{n-1}$	$S_n$	$R_n$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

# Full Adder

L'insieme di due half-adder e una porta logica OR, opportunamente collegati, restituisce un [full-adder](#)



Il caso in cui entrambi gli "C" ( $C^\alpha$  e  $C^\beta$ ) sono 1 sarebbe un problema (come facciamo col resto?), ma non esiste (quindi OR o XOR sono equivalenti):

- $C^\alpha = 1$  solo se  $A_n$  e  $B_n$  sono entrambi 1  $\rightarrow S^\alpha = 0$
- ma se  $S^\alpha = 0 \rightarrow C^\beta = 0$ , indipendentemente da  $C_{n-1}$

Half Adder:

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

# Full Adder

Tabella di verità della somma di 3 bit

$A_n$	$B_n$	$R_{n-1}$	$S_n$	$R_n$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$A_n$	$B_n$	$R_{n-1}$	$S_n$	$R_n$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

# Full Adder

Espressione booleana corrispondente alla tabella di verità

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

$$R_n = \overline{A_n} B_n R_{n-1} + A_n \overline{B_n} R_{n-1} + A_n B_n \overline{R_{n-1}} + A_n B_n R_{n-1}$$

possiamo riscrivere  $R_n$ , sapendo che  $Q+Q+Q = Q$

$$R_n = (\overline{A_n} B_n R_{n-1} + A_n B_n R_{n-1}) + (A_n \overline{B_n} R_{n-1} + A_n B_n R_{n-1}) + (A_n B_n \overline{R_{n-1}} + A_n B_n R_{n-1})$$

$$R_n = (\overline{A_n} + A_n) B_n R_{n-1} + (\overline{B_n} + B_n) A_n R_{n-1} + (\overline{R_{n-1}} + R_{n-1}) A_n B_n$$

$$R_n = B_n R_{n-1} + A_n R_{n-1} + A_n B_n = \boxed{A_n B_n + (A_n + B_n) R_{n-1}}$$

# Full Adder

$$S_n = \overline{A_n} \overline{B_n} R_{n-1} + \overline{A_n} B_n \overline{R_{n-1}} + A_n \overline{B_n} \overline{R_{n-1}} + A_n B_n R_{n-1}$$
$$R_n = \overline{A_n} B_n R_{n-1} + A_n \overline{B_n} R_{n-1} + A_n B_n \overline{R_{n-1}} + A_n \overline{B_n} \overline{R_{n-1}}$$

possiamo riscrivere la somma  $S_n$

$$S_n = R_{n-1} \left( A_n B_n + \overline{A_n} \overline{B_n} \right) + \overline{R_{n-1}} \left( \overline{A_n} B_n + A_n \overline{B_n} \right)$$

ma  $A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$  (la tab. di verità della XOR è simmetrica)

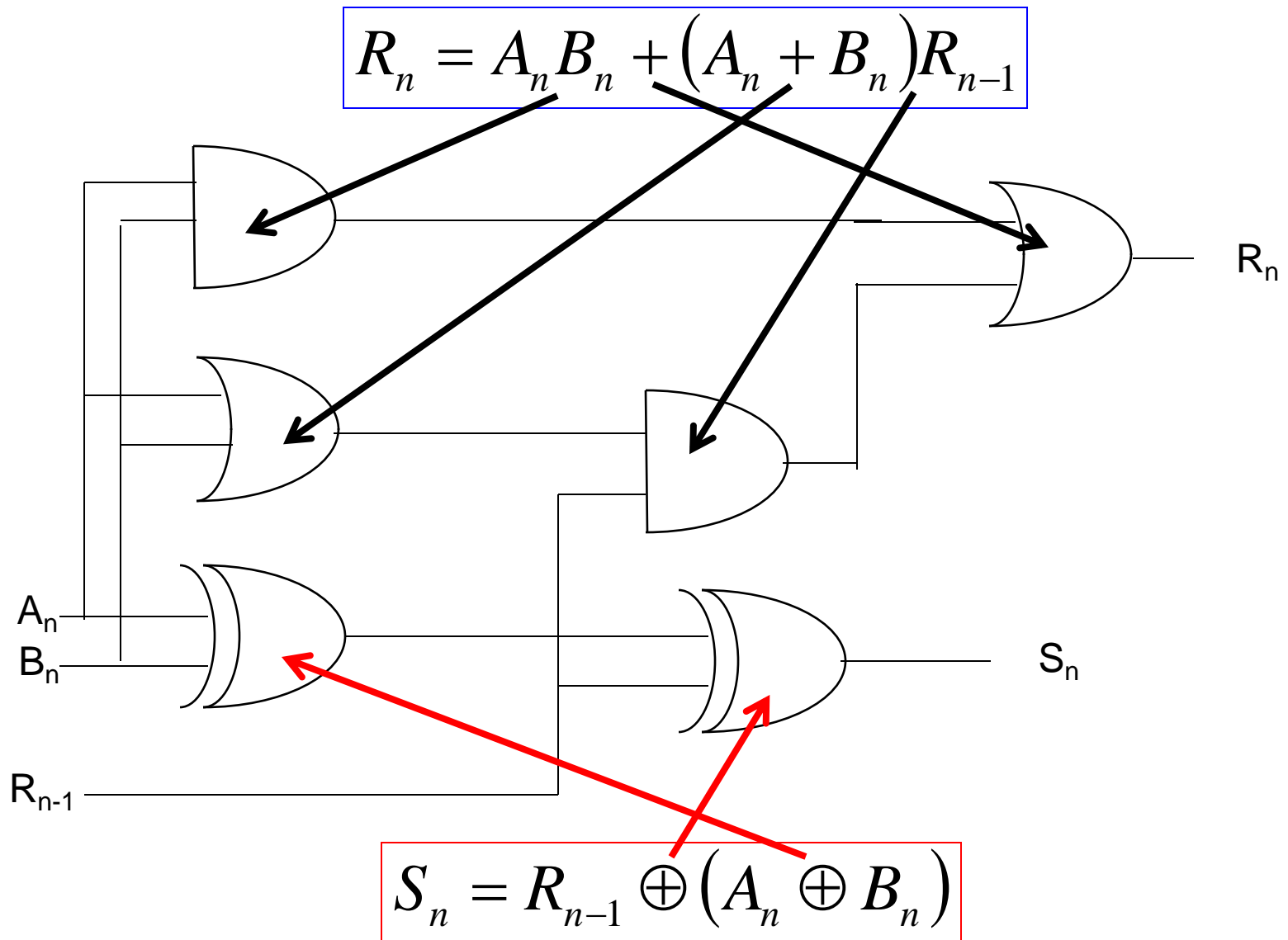
$$\left( A_n B_n + \overline{A_n} \overline{B_n} \right) = \overline{A_n \oplus B_n}$$
$$\left( \overline{A_n} B_n + A_n \overline{B_n} \right) = A_n \oplus B_n$$

quindi

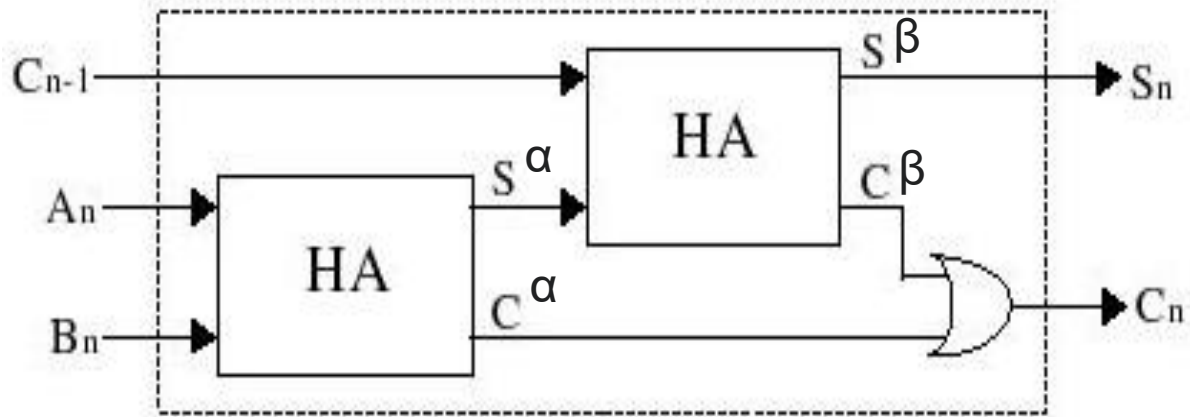
$$S_n = R_{n-1} \cdot \overline{A_n \oplus B_n} + \overline{R_{n-1}} \cdot A_n \oplus B_n$$

$$S_n = R_{n-1} \oplus (A_n \oplus B_n)$$

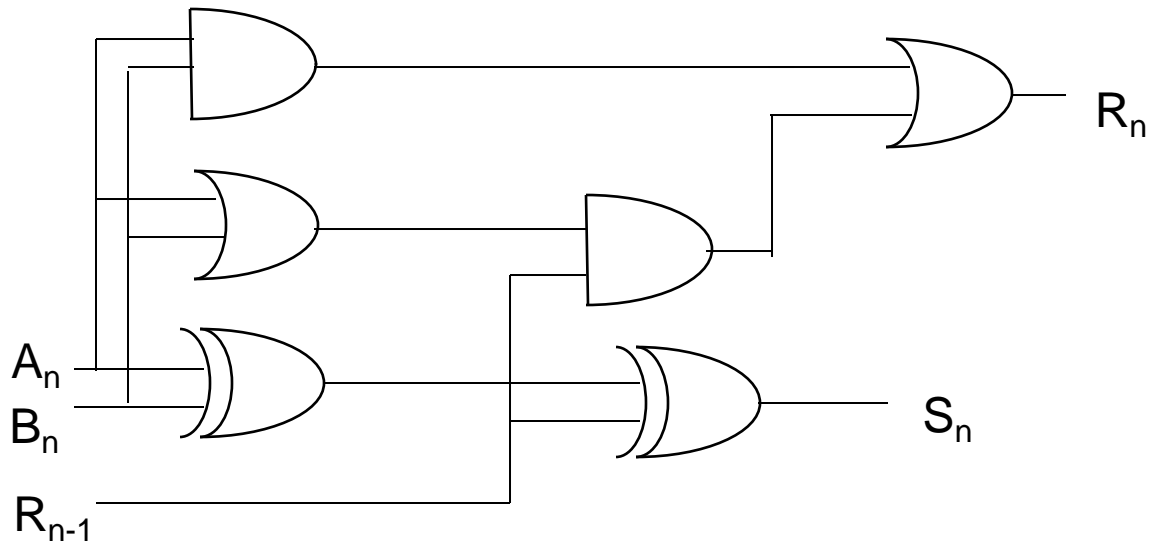
# Full Adder - circuito



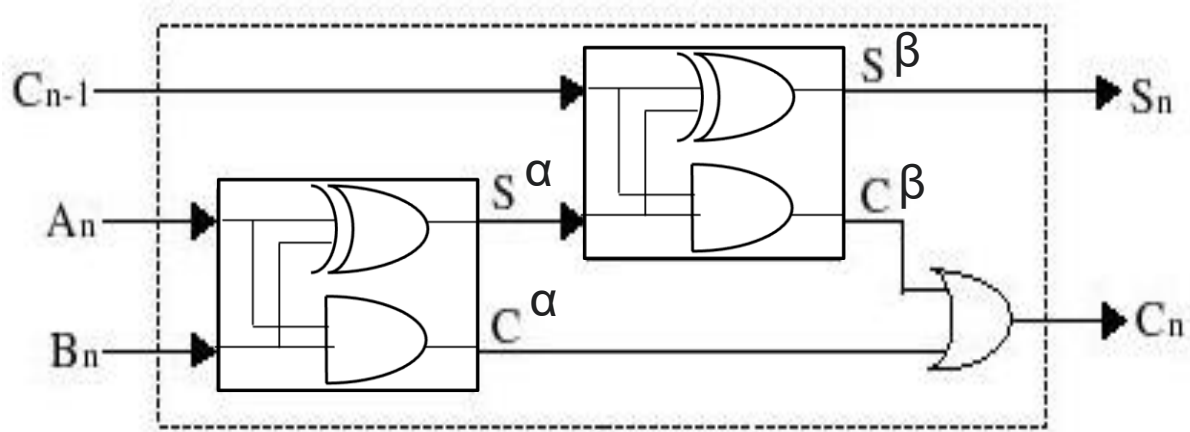
# Full Adder



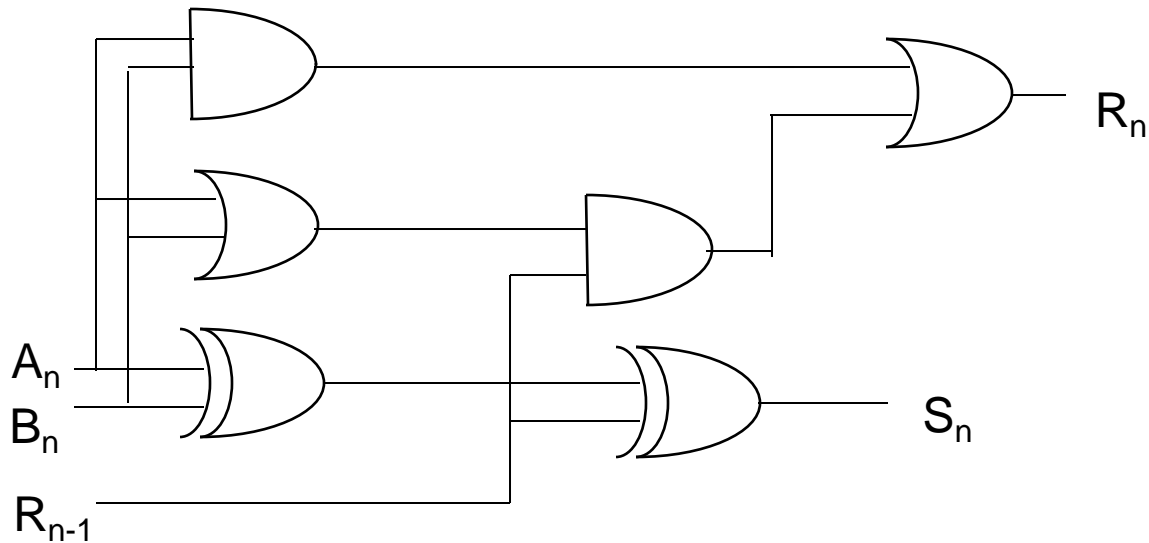
VS.



# Full Adder

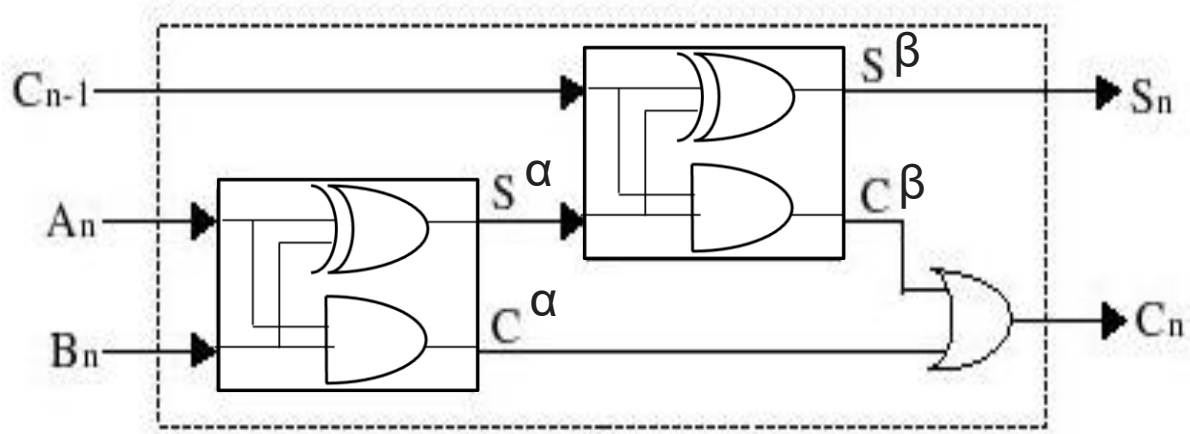


VS.

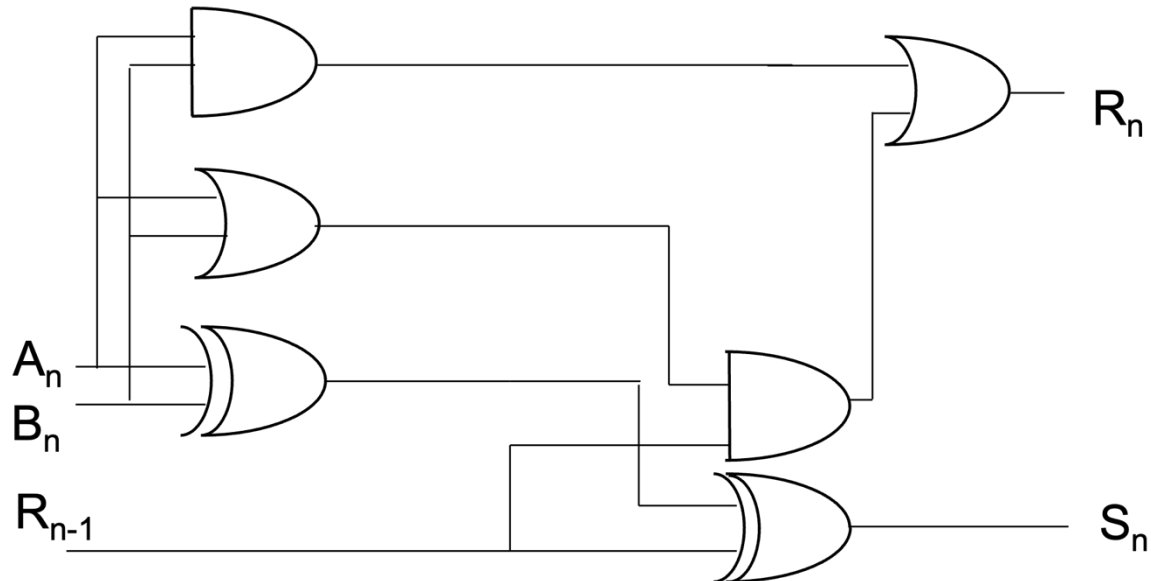




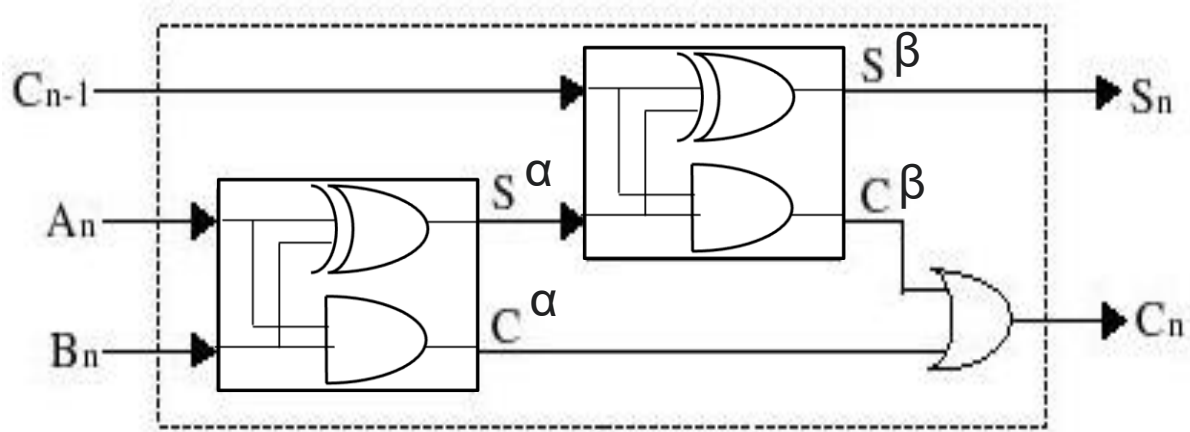
# Full Adder



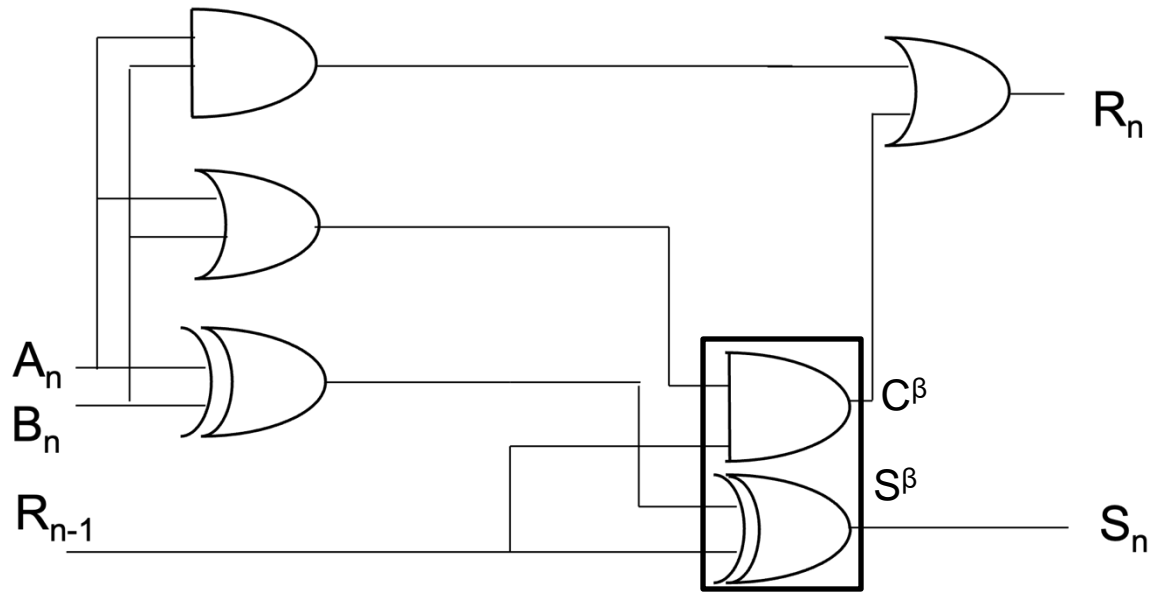
VS.



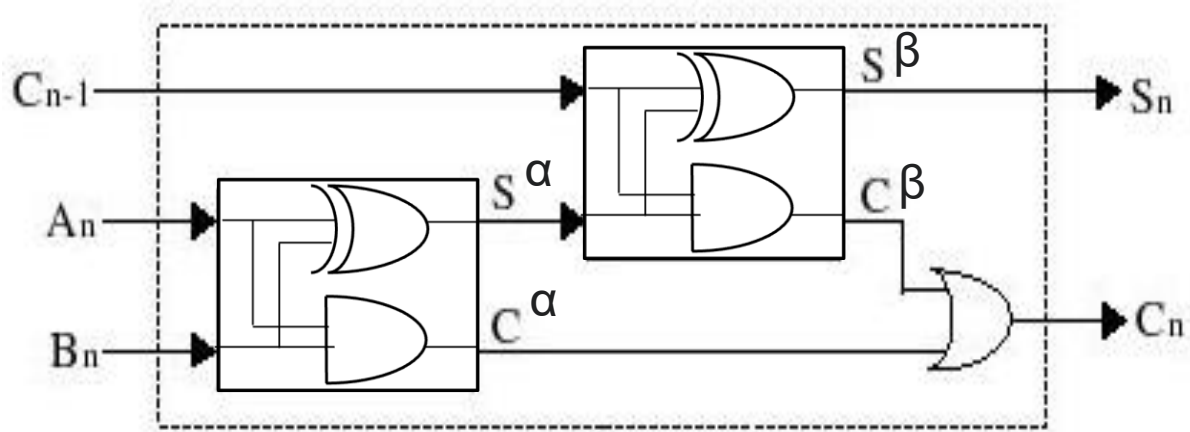
# Full Adder



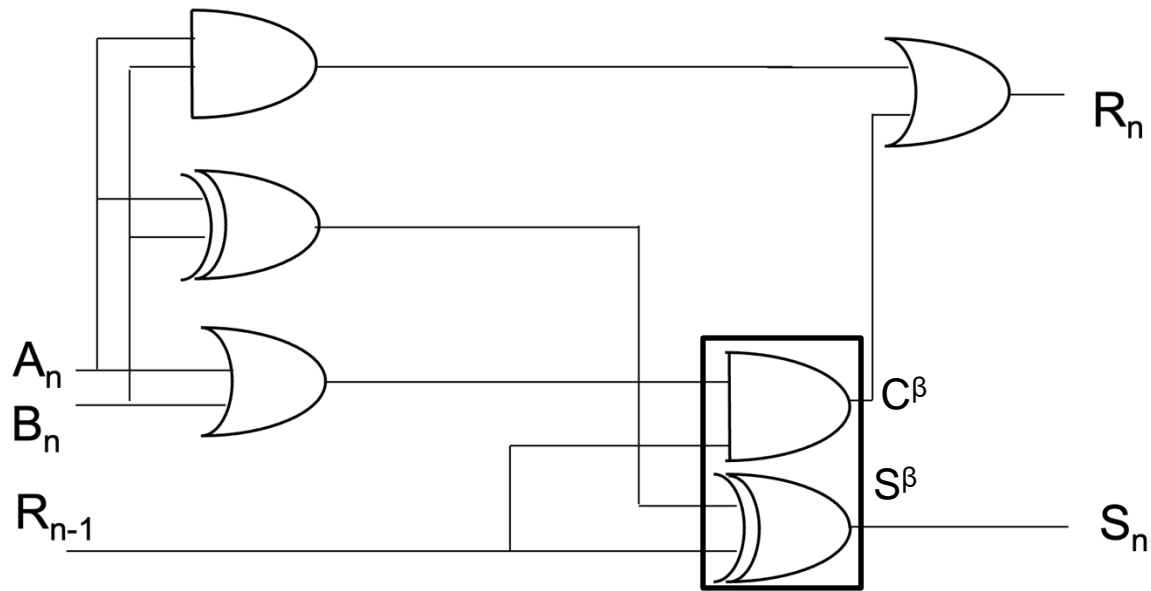
VS.



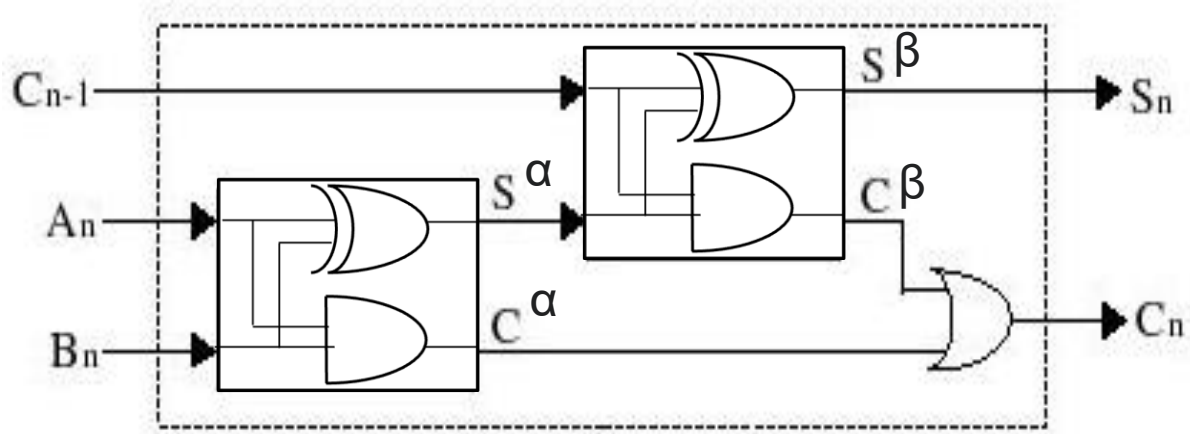
# Full Adder



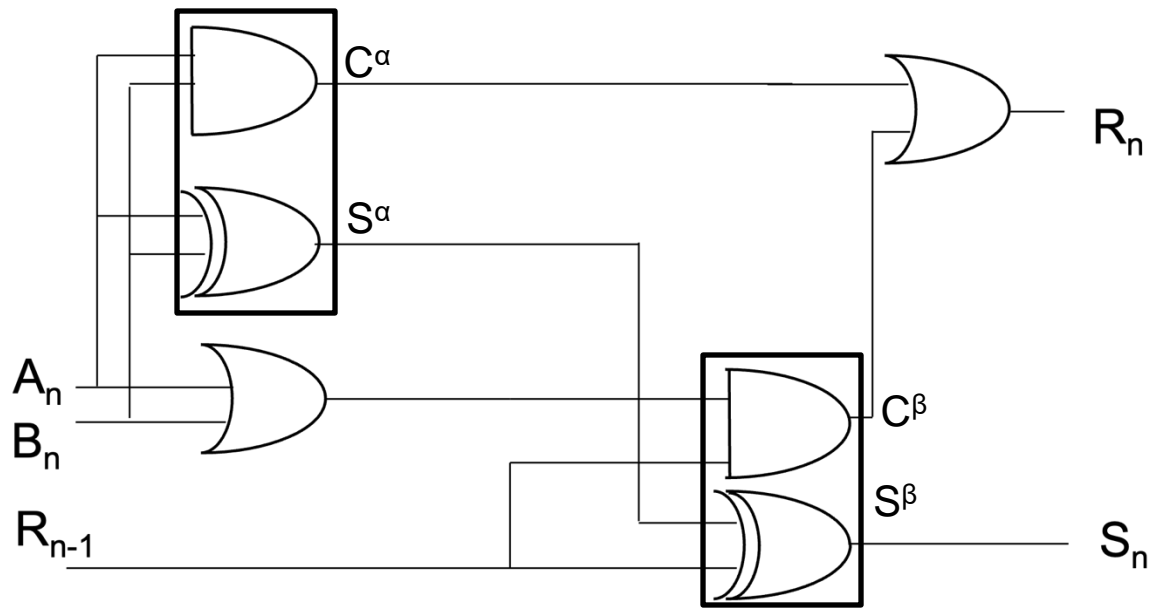
VS.



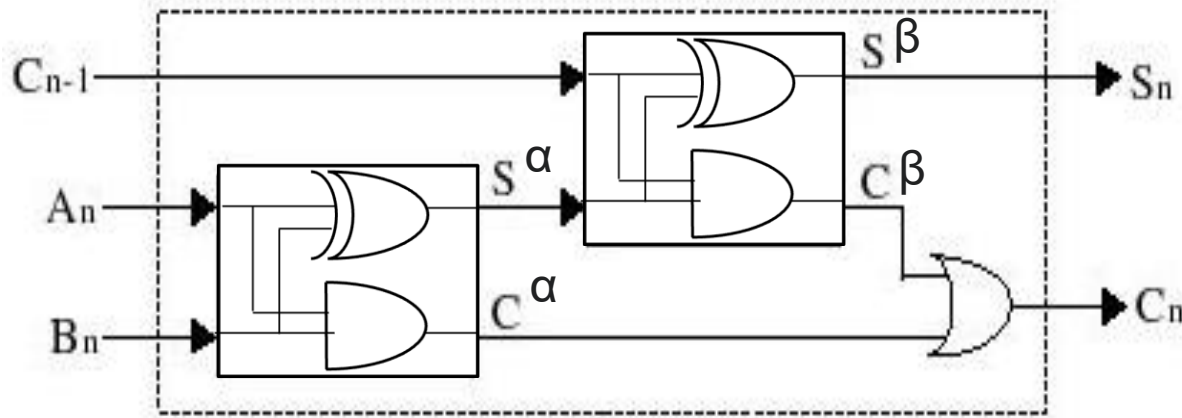
# Full Adder



VS.



# Full Adder



Nella AND del secondo HA  
(quello che esce con  $C^\beta$  e  
 $S^\beta$ ) entrano:

$C_{n-1}$

e

$S^\alpha$

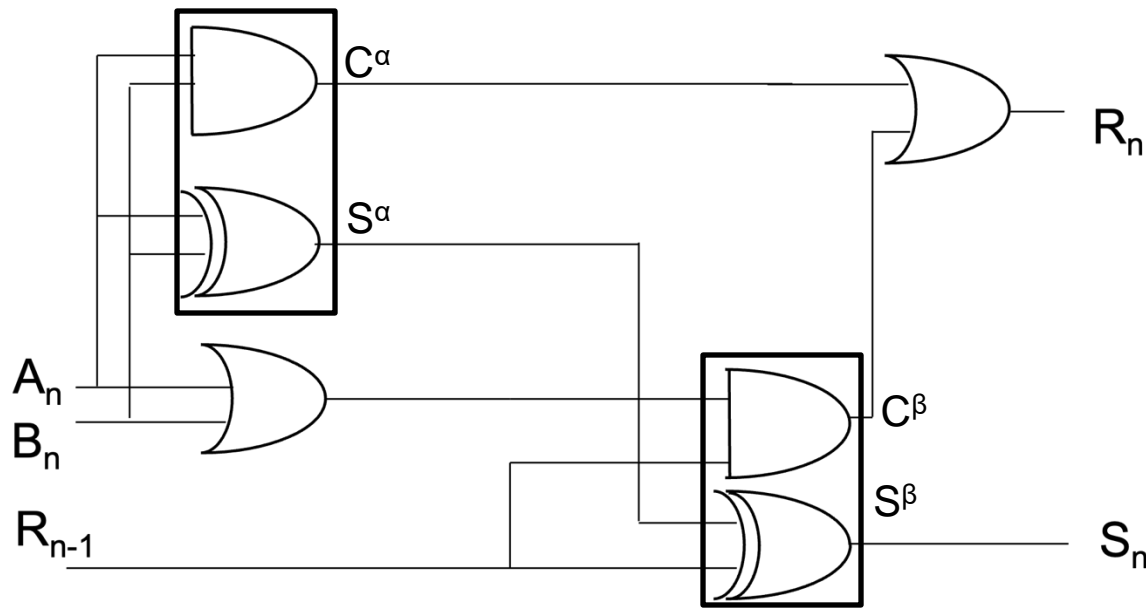
(cioè  $A_n \oplus B_n$ )

e poi  $C^\beta$  va in OR con

$C^\alpha$

(cioè  $A_n B_n$ )

VS.



Nella AND del secondo HA  
(quello che esce con  $C^\beta$  e  
 $S^\beta$ ) entrano:

$C_{n-1}$

e

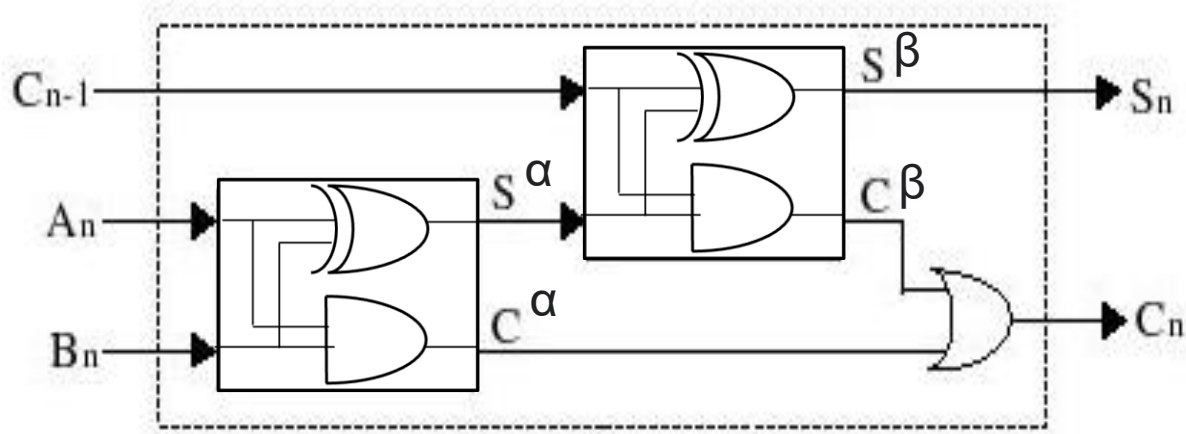
$A_n + B_n$

e poi  $C^\beta$  va in OR con

$C^\alpha$

(cioè  $A_n B_n$ )

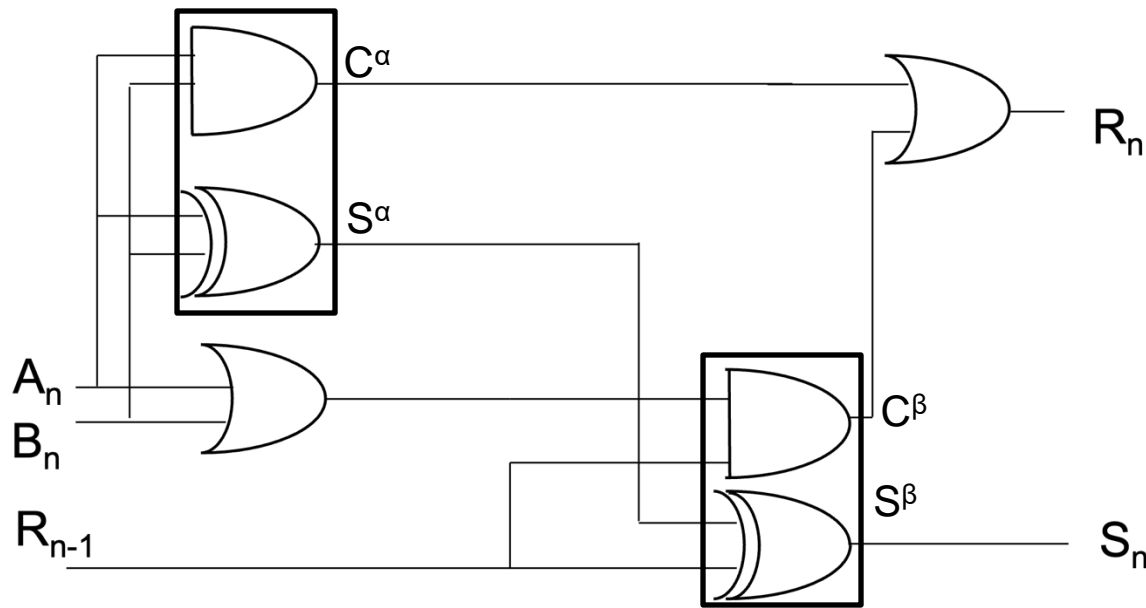
# Full Adder



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

$$C_{n-1}(A_n + B_n) + A_n B_n$$

VS.



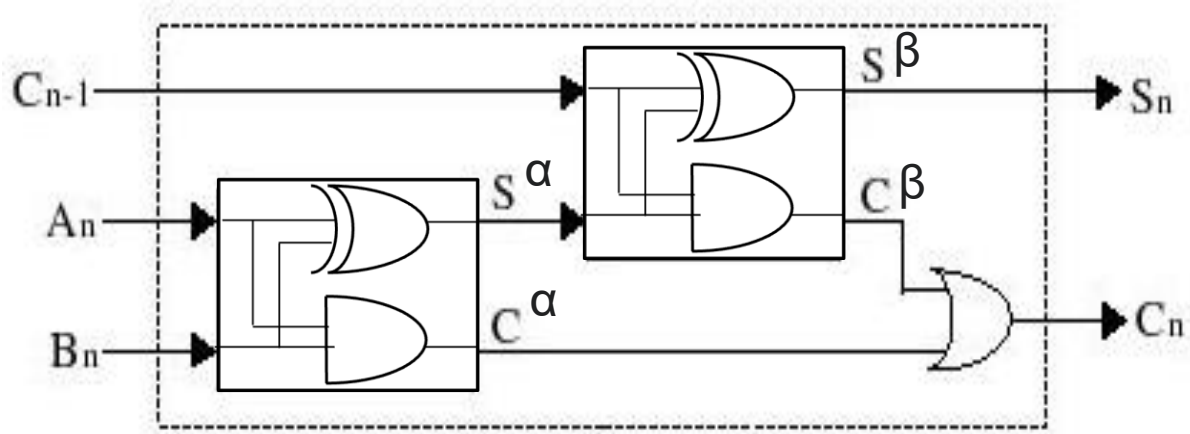
l'unica differenza potrebbe esserci se  $A_n$  e  $B_n$  sono entrambi 1.

In questo caso:

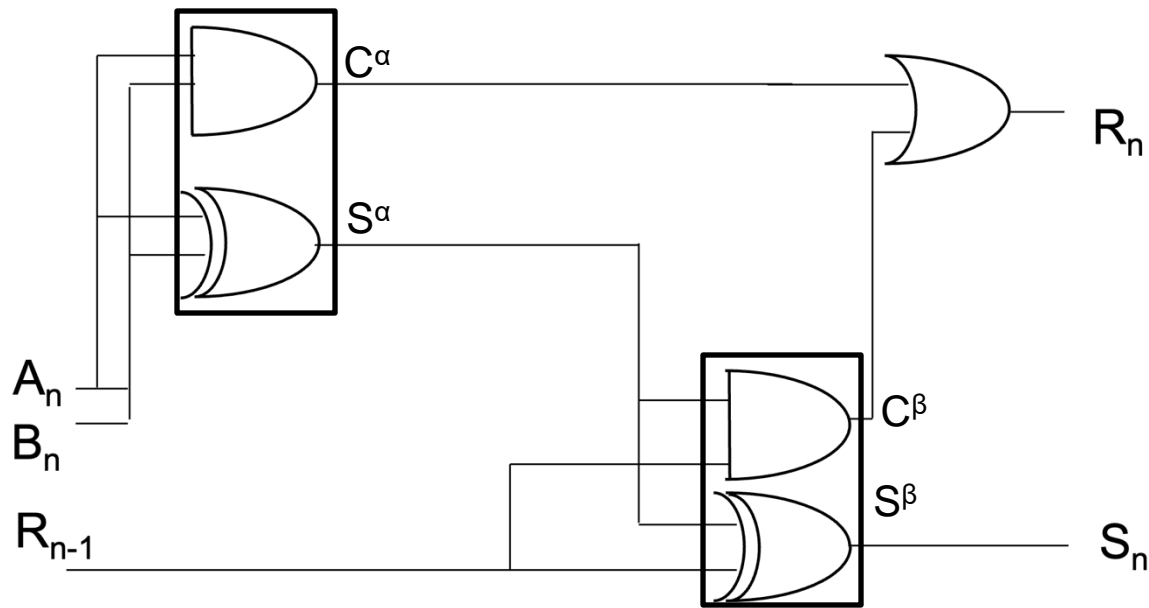
$$A_n B_n = C^\alpha = 1$$

a questo punto, dato che questo è in OR non è importante con cosa lo sia

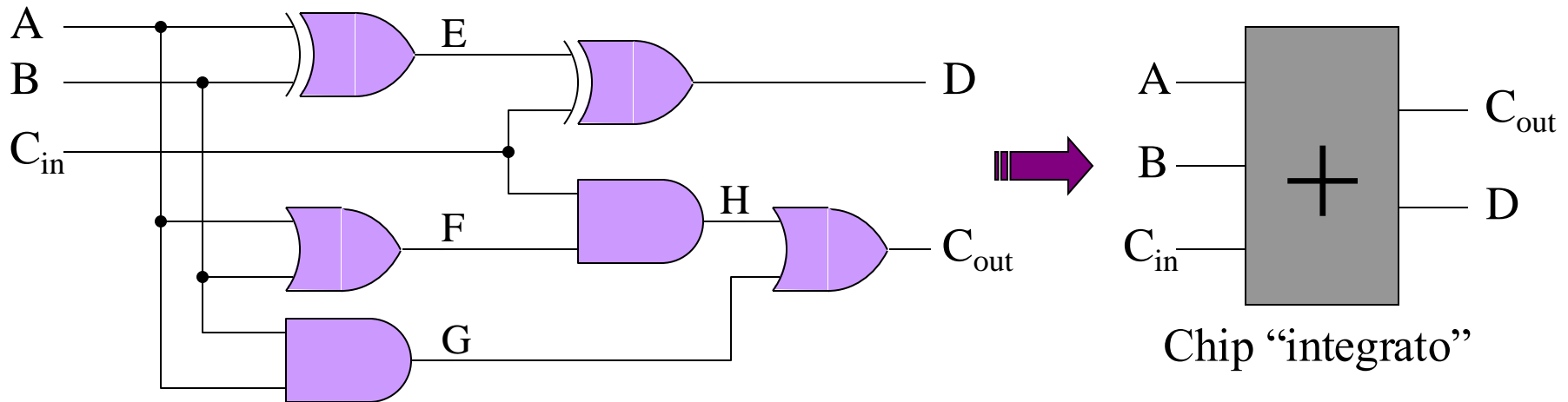
# Full Adder



VS.



# Aritmetica binaria vs transistor



Input			Intermedi				Output	
A	B	$C_{in}$	E	F	H	G	D	$C_{out}$
0	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	1	0
1	0	0	1	1	0	0	1	0
1	1	0	0	1	0	1	0	1
0	0	1	0	0	0	0	1	0
0	1	1	1	1	1	0	0	1
1	0	1	1	1	1	0	0	1
1	1	1	0	1	1	1	1	1

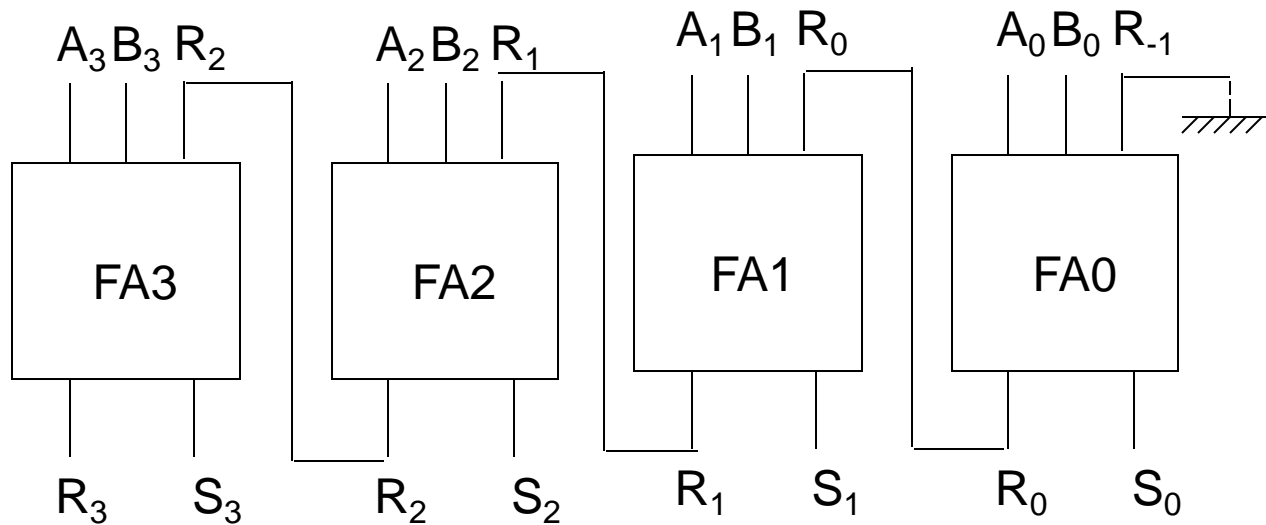
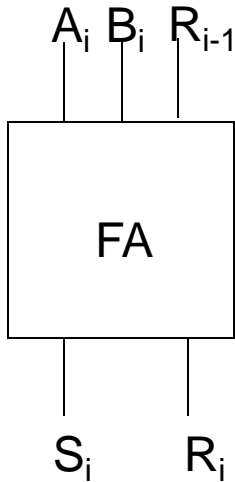
- ogni cifra richiede 6 porte
- ogni porta ha ~ 6 transistor
- ~ 36 transistor per cifra



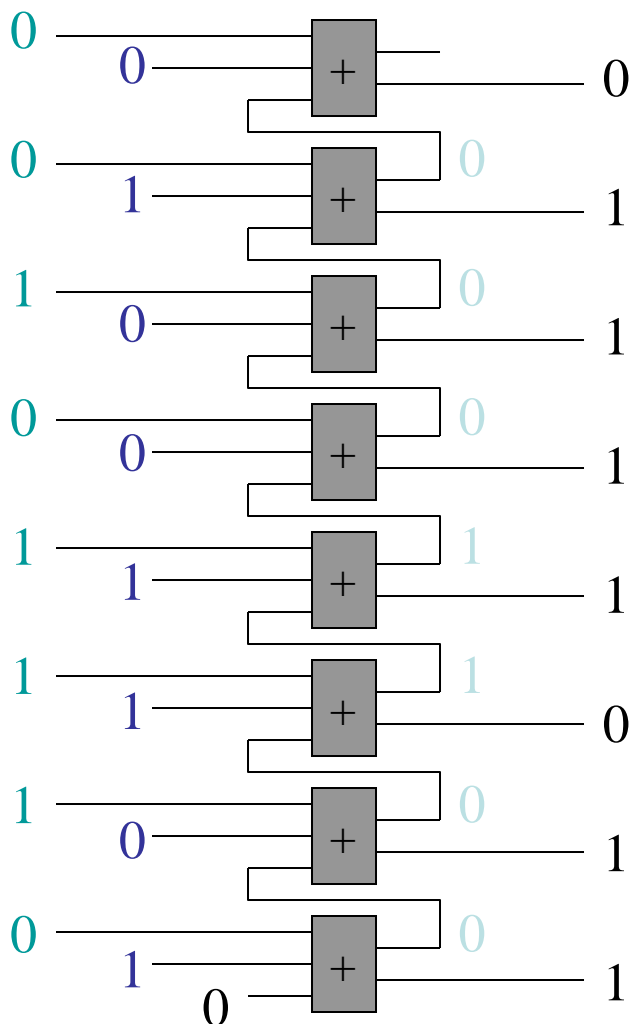
# Full Adder

3 input e 2 output

Una somma di 4 bit può essere eseguita in parallelo usando 4 Full Adders



# Aritmetica binaria a 8 bit (in cascata)



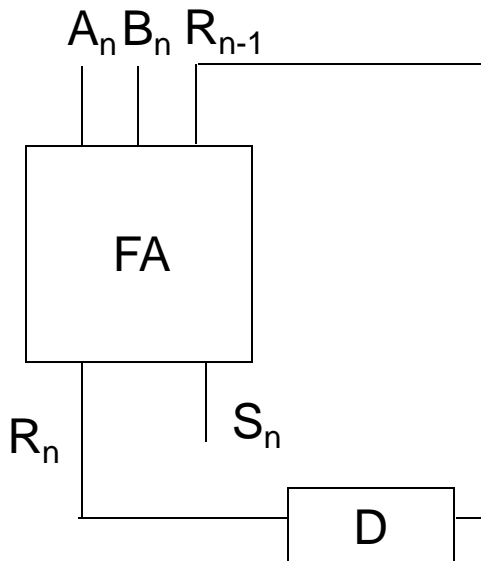
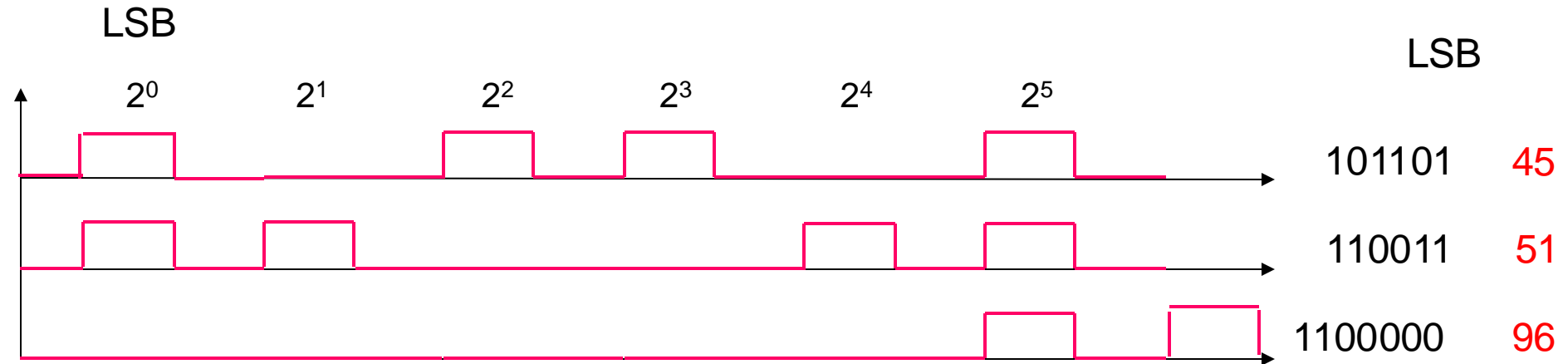
MSB = Most Significant Bit

$$\begin{array}{r}
 00101110 = 46 \\
 + 01001101 = 77 \\
 \hline
 01111011 = 123
 \end{array}$$

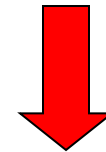
- il riporto è utilizzato nel successivo stadio
- somma di due numeri a 8 bit
- ~300 transistor per fare questa operazione di base. Poi ci sono  $-$ ,  $\times$ ,  $/$ , etc...

LSB = Least Significant Bit

# Somma seriale



Una unità di ritardo in più  
 $D = T$  fra gli impulsi



impulso di riporto in tempo  
con i bit da sommare