Laboratorio II, modulo 2
2016-2017

Elettronica digitale (1ª parte)

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

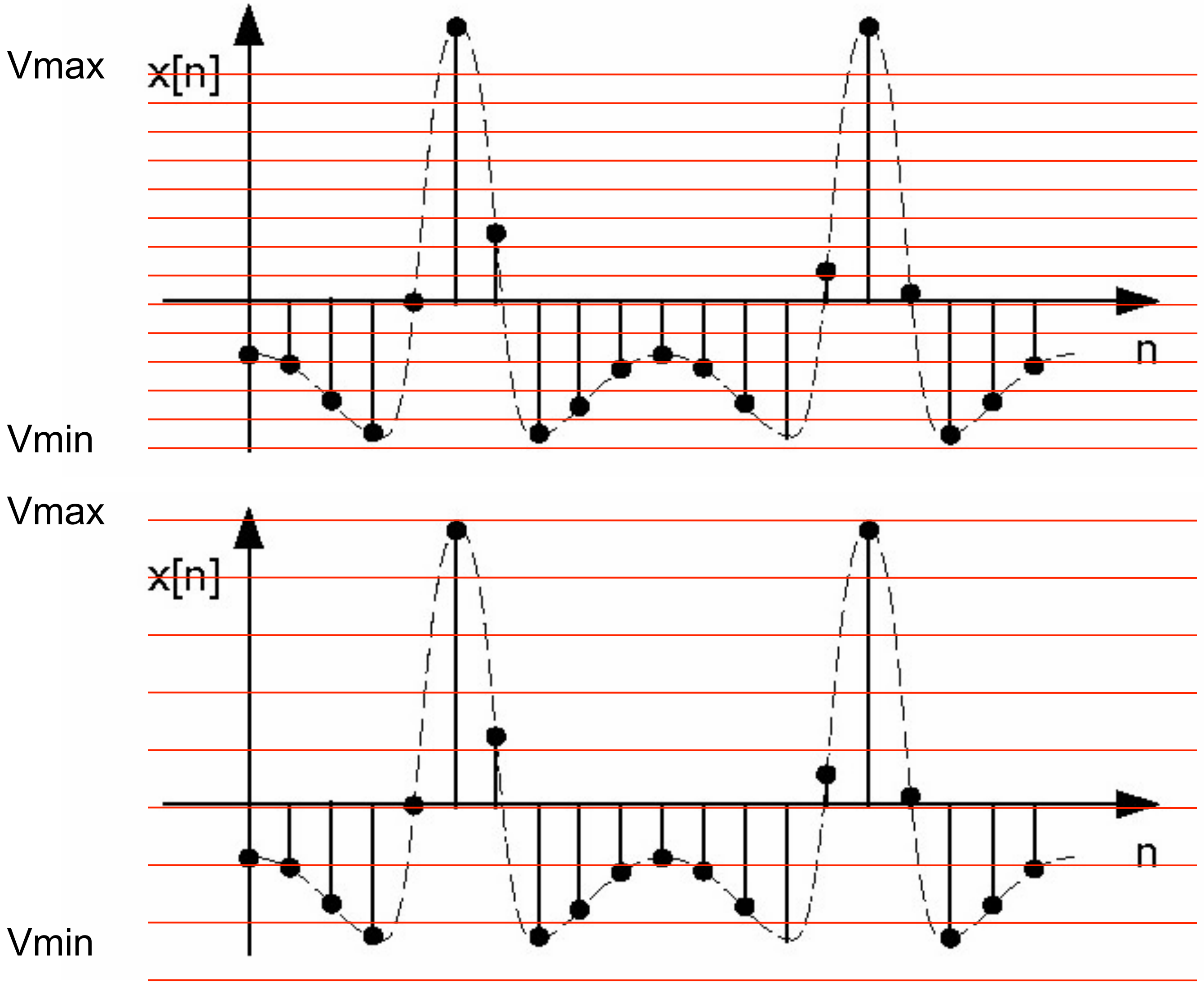
ADC (I)

- 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

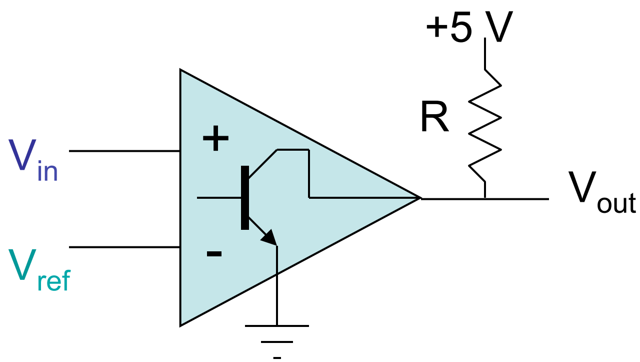
ADC (2)

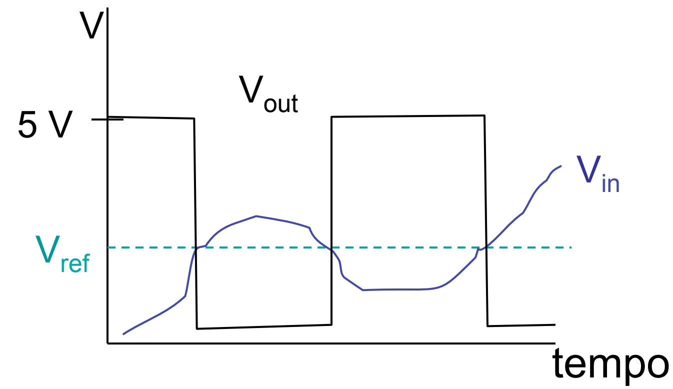


Comparatori

- 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 invertente 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 V/μs significa 2 μs per arrivare a fondo scala se alimentato ±15 V

Esempio (reale) di comparatore





- quando $V_{in} < V_{ref}$, V_{out} è "pulled-up" (attraverso il resistore di "pull-up", usualmente 1 kΩ o più)
  - questa configurazione è chiamata a "collettore aperto": l'uscita è il collettore di un transistor npn. In saturazione è tirata verso l'emettitore (ground), ma se non c'è corrente di base il collettore è tirato al voltaggio di pull-up
- 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 necessatia una maggiore precisione di "timing"

"digitale"

continuo → discreto

analogico → digitale

Stati logici solo due possibili stati

1, alto (H), vero (true)
0, basso (L), falso (false)

Algebra booleana

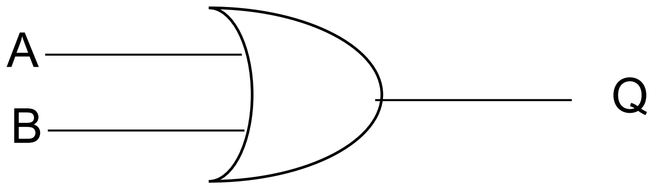
sistema matematico per l'analisi di stati logici

solo 3 funzioni logiche di base → **AND OR NOT** ← Funzioni logiche

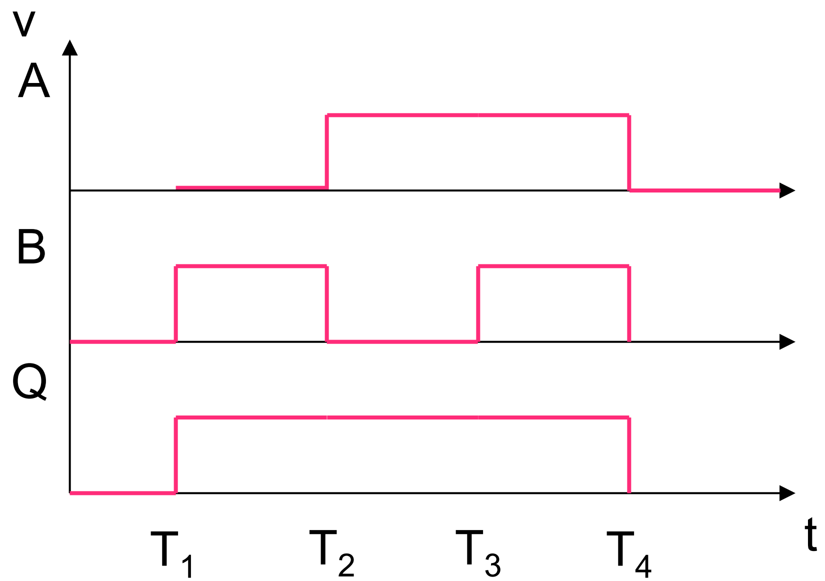
circuiti usati per la realizzazione di funzioni logiche → Porte logiche

Porte logiche di base - OR

OR



$Q = A+B$



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

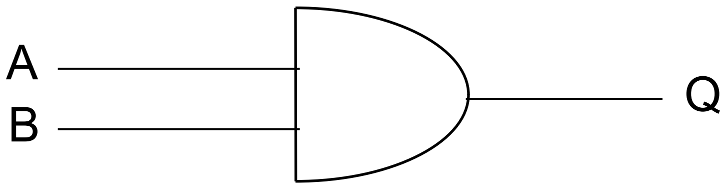
$$A+B+C = (A+B)+C = A+(B+C)$$

$$A+B = B+A$$

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

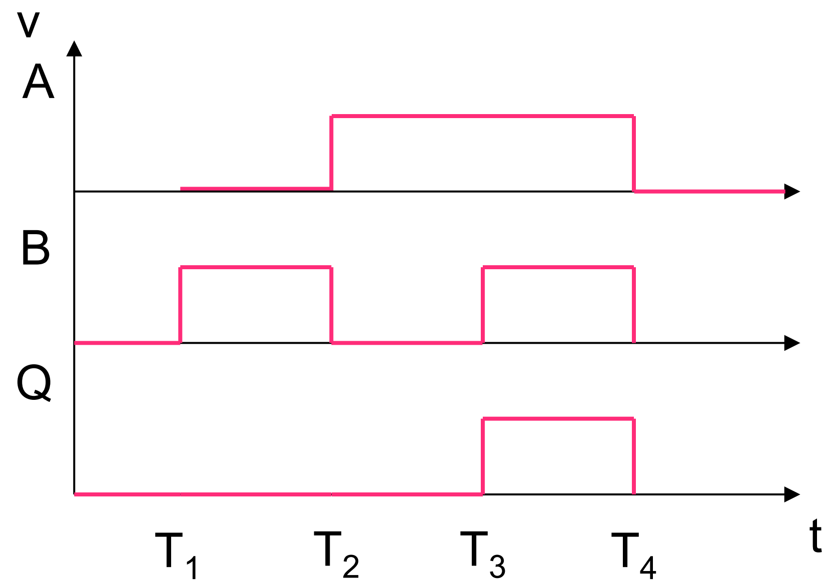
Porte logiche di base - AND

AND



$$Q = A \cdot B$$

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



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

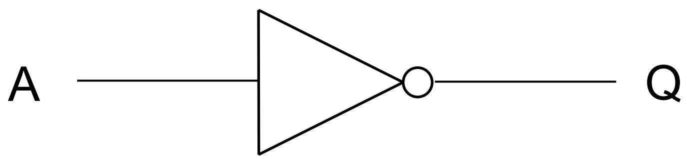
$$A \cdot B = B \cdot A$$

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

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

Porte logiche di base - NOT

NOT



A	Q
0	1
1	0

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

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

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

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

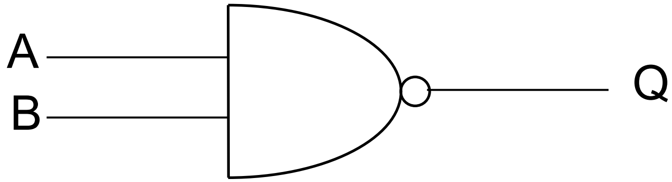
sapendo che

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

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

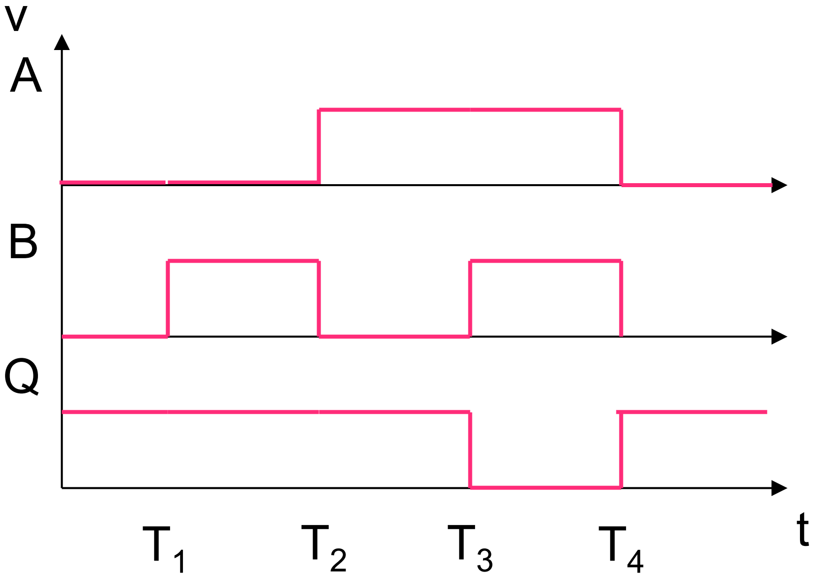
Porte logiche di base - NAND

NAND



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

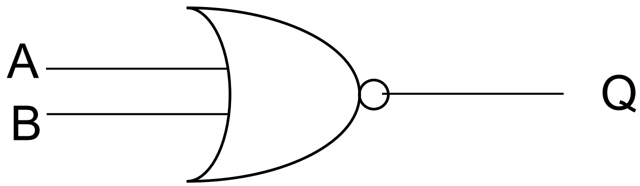
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



porta universale

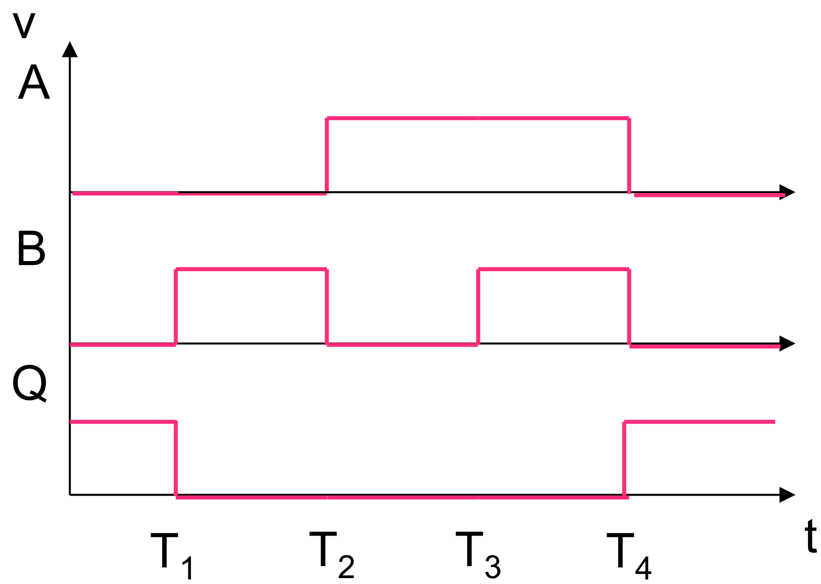
Porte logiche di base - NOR

NOR



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

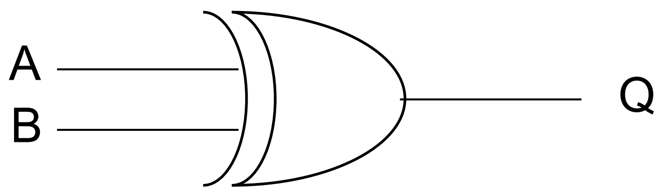
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



Porte logiche di base - XOR

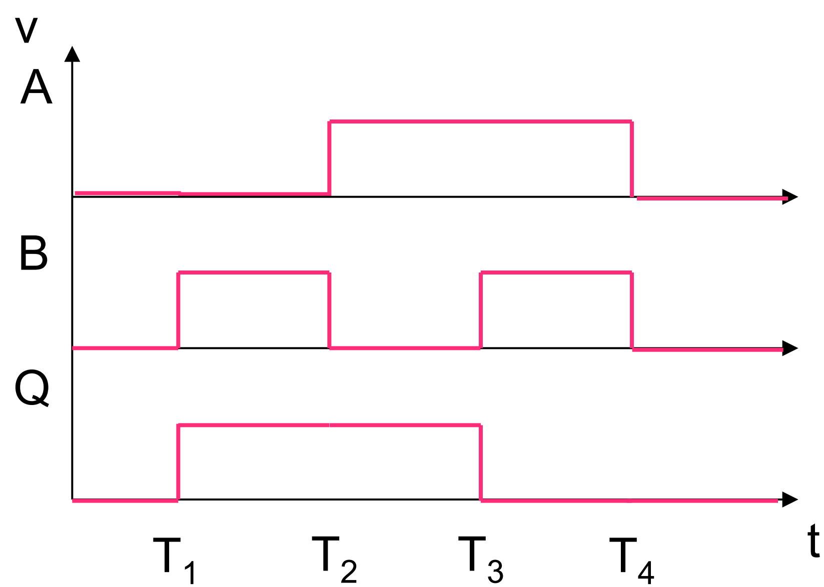
XOR

OR esclusivo



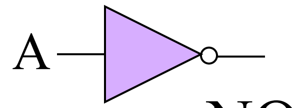
$Q = A \oplus B$

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



Manipolazione dati

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



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

XOR

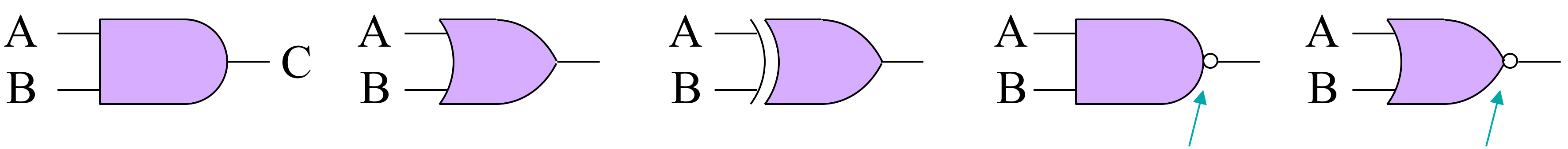
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

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

Algebra booleana

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

Teoremi di De Morgan

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

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

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

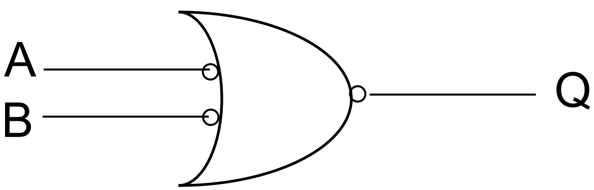
Il complemento dell' OR di più variabili logiche è dato dall' AND dei complementi

Algebra Booleana

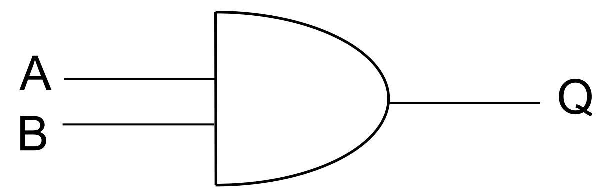
Un circuito AND per logica positiva funziona come un OR per logica negativa

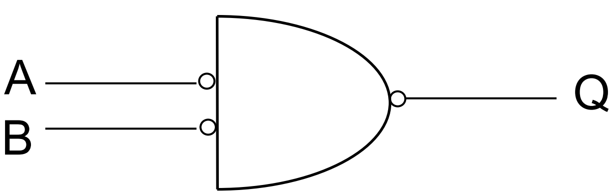
non è necessario usare i tre circuiti di base

bastano due OR e NOT oppure AND e NOT

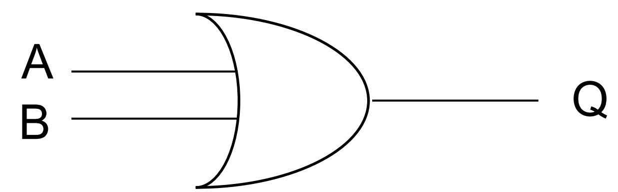


$$\overline{\overline{A}+\overline{B}} \Leftrightarrow A \cdot B$$





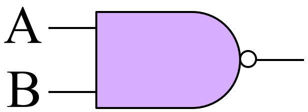
$$\overline{\overline{A}\cdot\overline{B}} \Leftrightarrow A + B$$



Tutta la logica con la sola NAND

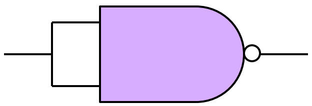
NAND

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



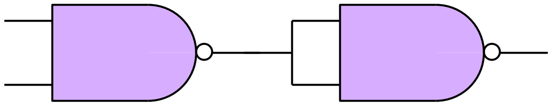
NOT

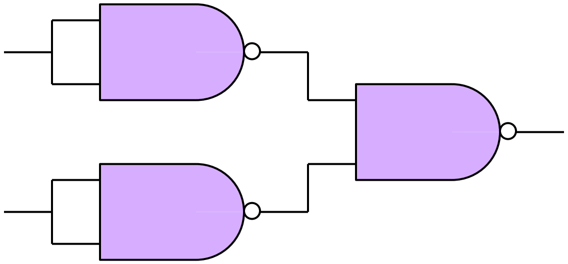
A	C
0	1
1	0



AND

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



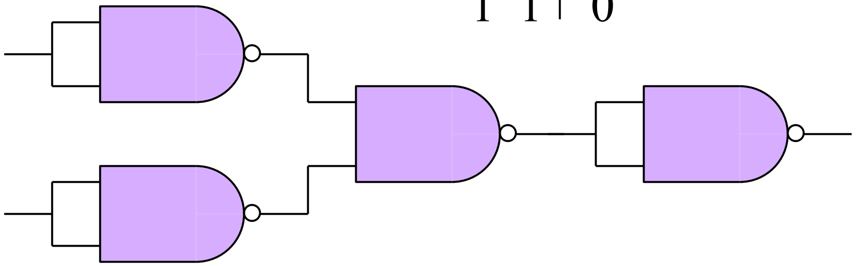


OR

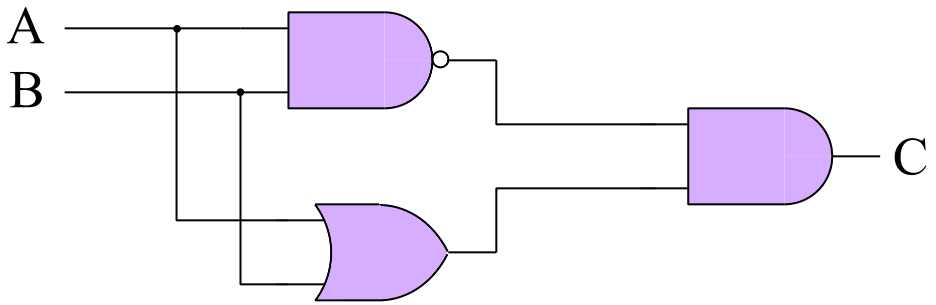
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

NOR

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



Tutta la logica con la sola NAND



XOR = (A NAND B) AND (A 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 negato

Aritmetica

- sommiamo due numeri binari:

  00101110 = 46

  + 01001101 = 77

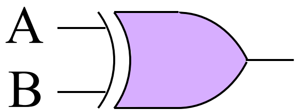
  01111011 = 123

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

  0 + 0 = 0; 0 + 1 = 1 + 0 = 1; 1 + 1 = 10 (2): (0, riporto 1);

  1 + 1 + (1 di riporto) = 11 (3): (1, riporto 1)

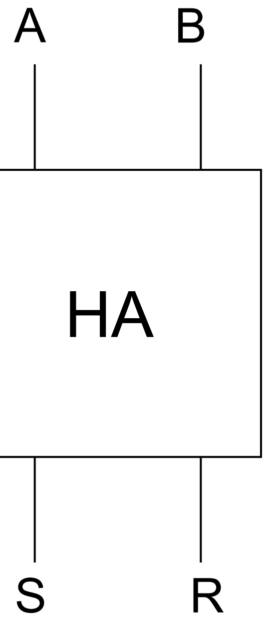
- proviamo ad associare una porta logica a queste “regole”
  – essendo una somma pensiamo subito alla OR
  – il caso “’1+1=0 (riporto1)’ si adatta meglio alla XOR
  – ancora manca le gestione del riporto



XOR

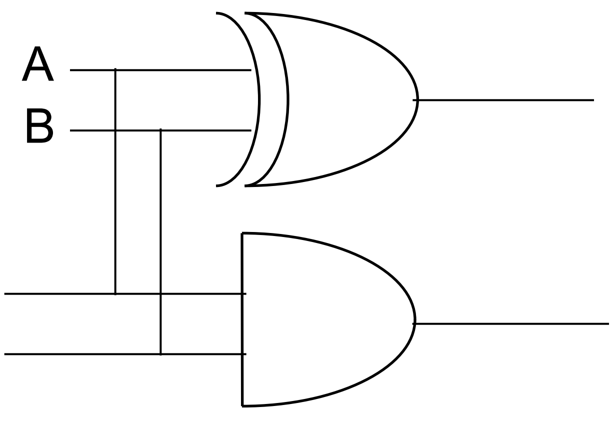
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

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

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

Espressione booleana corrispondente alla tabella di verità

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

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

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

$$R_n = \left(\overline{A_n}B_nR_{n-1} + A_nB_nR_{n-1}\right) + \left(A_n\overline{B_n}R_{n-1} + A_nB_nR_{n-1}\right) + \left(A_nB_n\overline{R_{n-1}} + A_nB_nR_{n-1}\right)$$

$$R_n = \left(\overline{A_n} + A_n\right)B_nR_{n-1} + \left(\overline{B_n} + B_n\right)A_nR_{n-1} + \left(\overline{R_{n-1}} + R_{n-1}\right)A_nB_n$$

$$R_n = B_nR_{n-1} + A_nR_{n-1} + A_nB_n = \boxed{A_nB_n + \left(A_n + B_n\right)R_{n-1}}$$

Full Adder

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

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

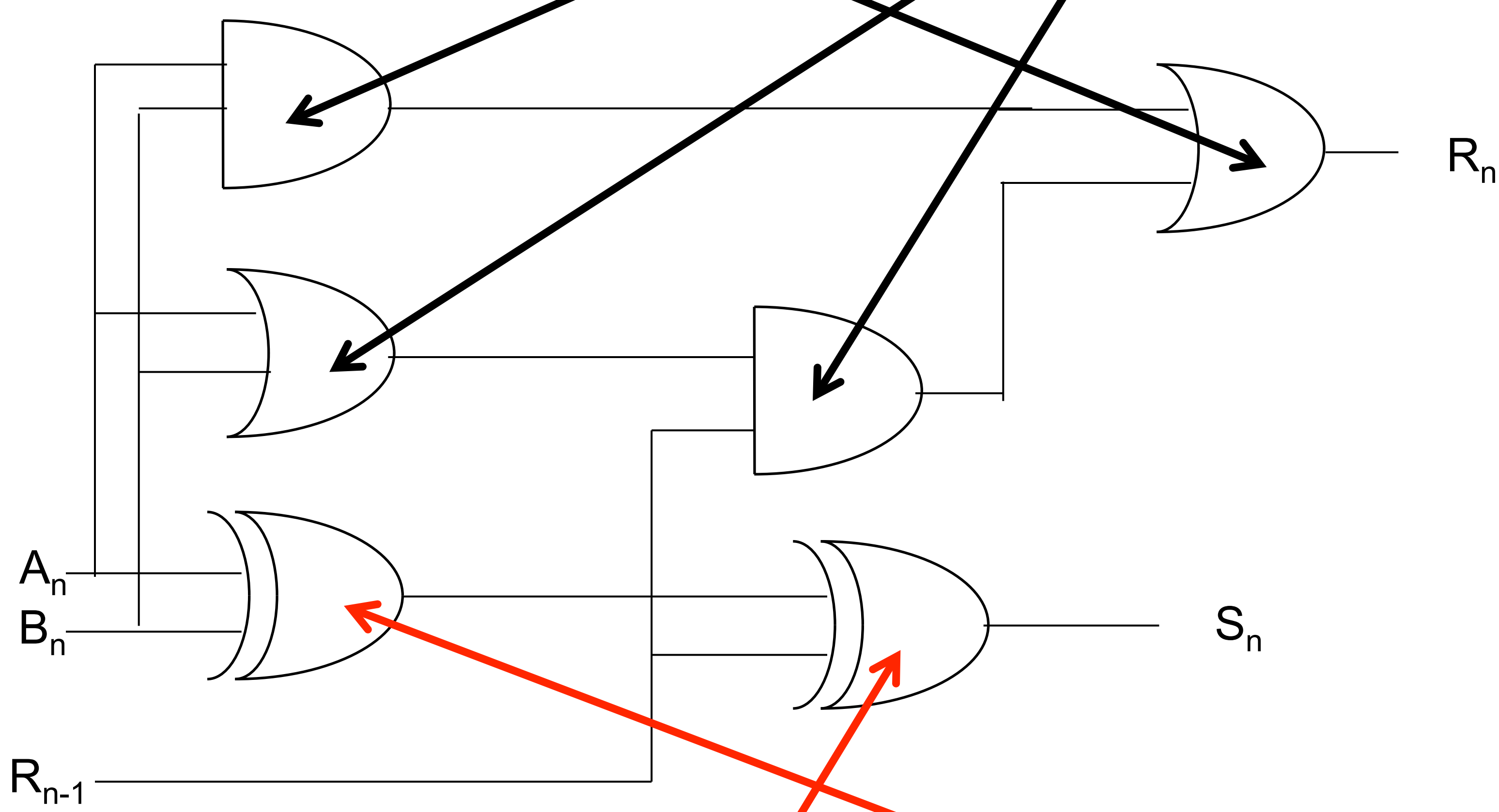
quindi

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

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

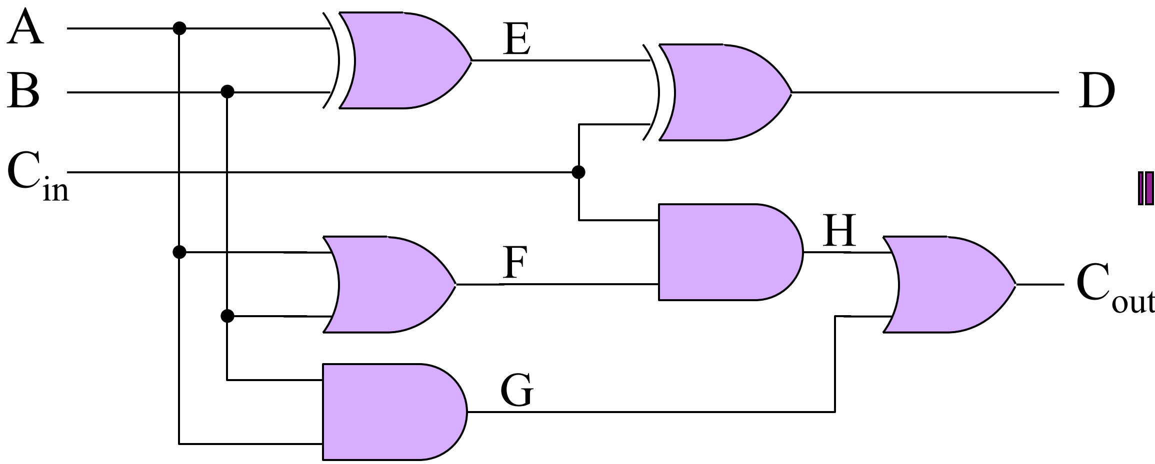
Full Adder - circuito

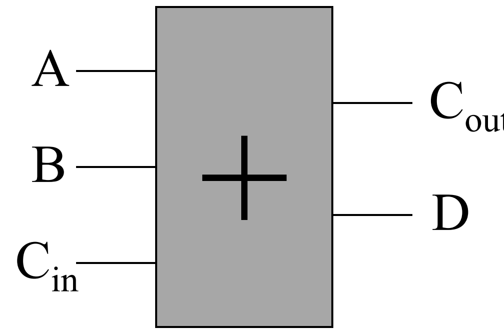
$$R_n = A_n B_n + \left(A_n + B_n\right) R_{n-1}$$



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

Aritmetica binaria vs transistor



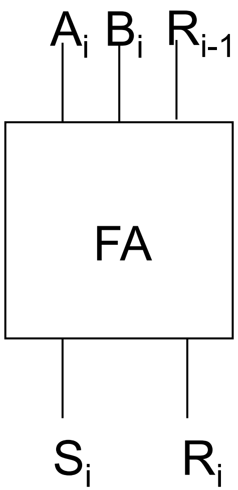


Chip “integrato”

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

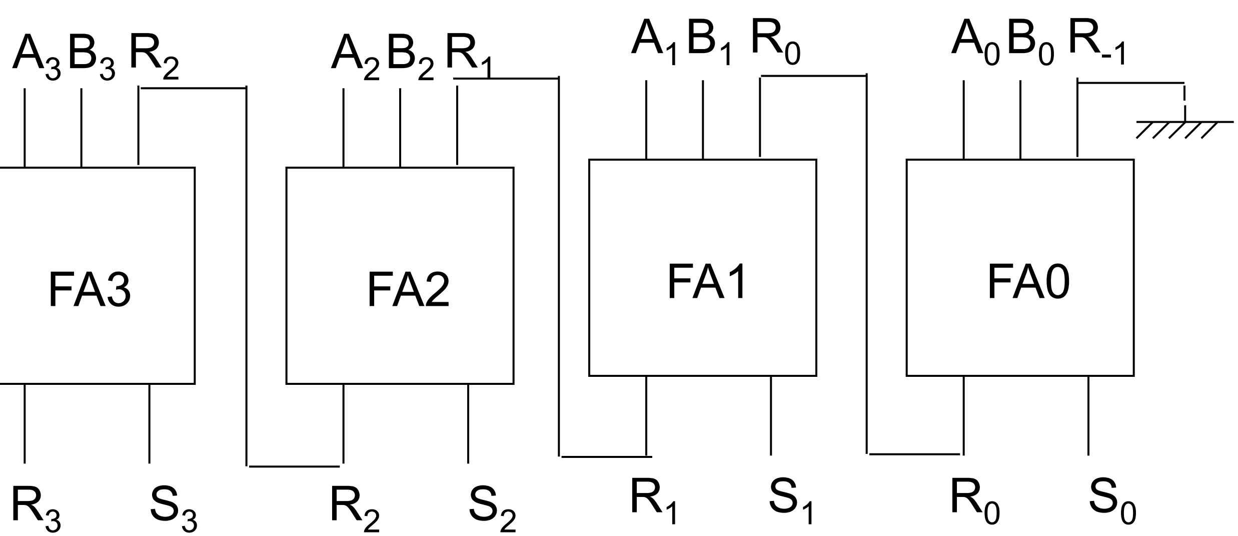
- ogni cifra richiede 6 porte
- ogni porta ha ~ 6 transistor
- ~ 36 transistor per cirfra

Full Adder

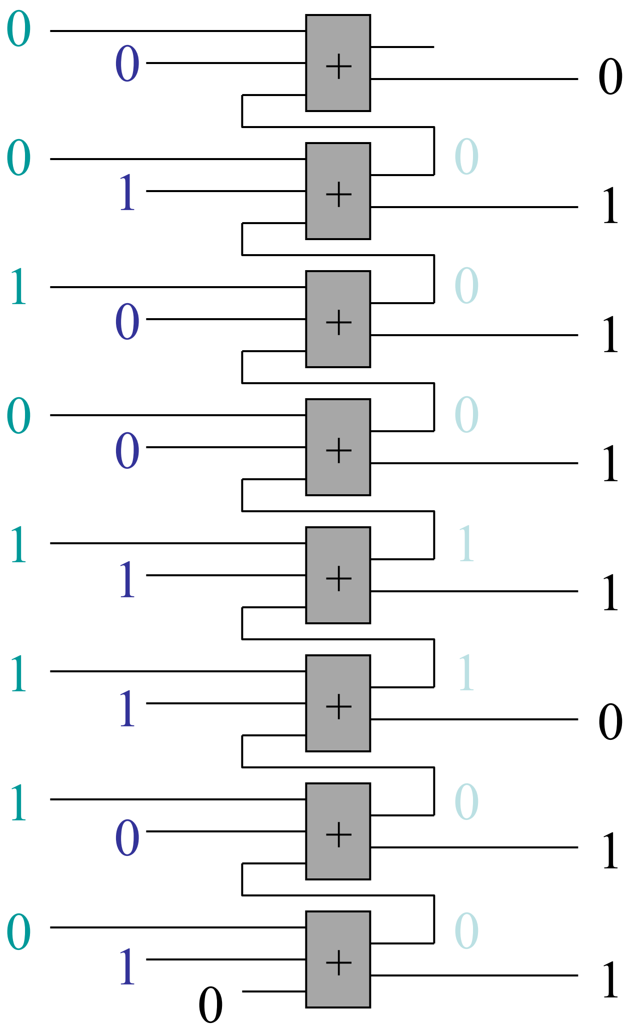


3 input e 2 output

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



Aritmetica binaria a 8 bit (in cascata)



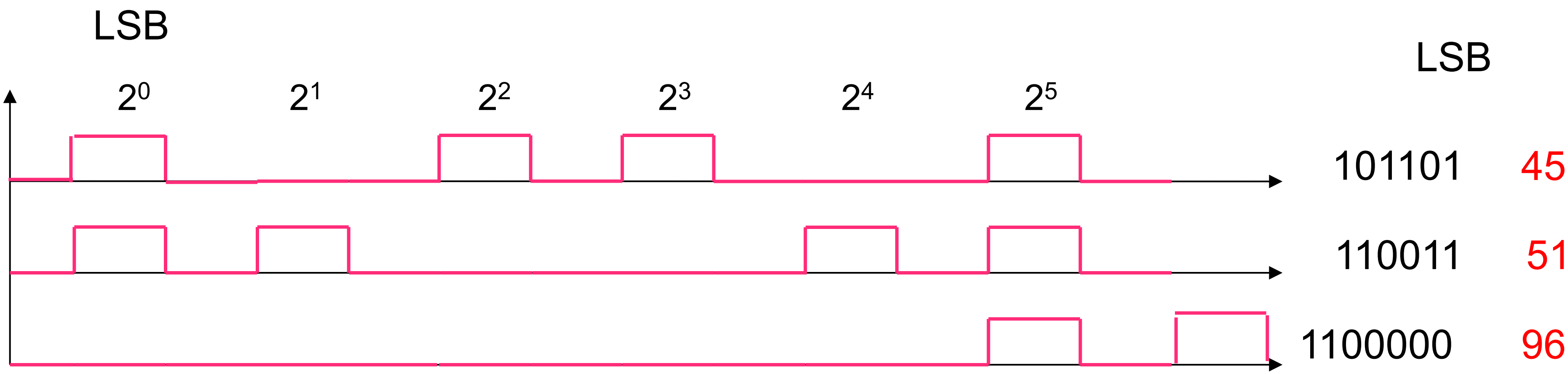
MSB = Most Significant Bit

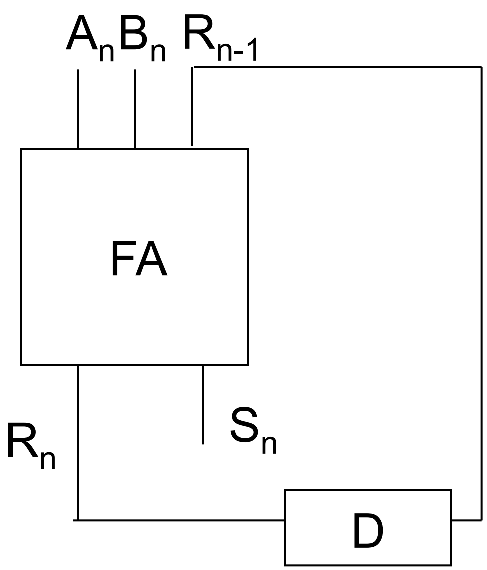
```
      11
  00101110  =  46
+ 01001101  =  77
  --------
  01111011  =  123
```

- 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 –, ×, /, 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