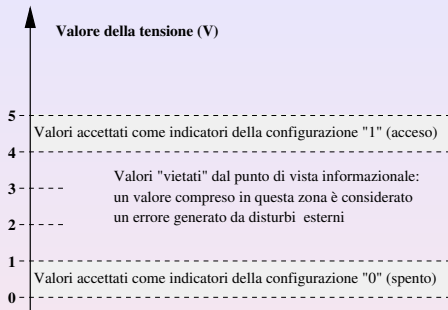


Reti logiche

Andrea Passerini
passerini@disi.unitn.it

Informatica

Codifica binaria di un segnale elettrico

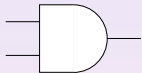


- Si introduce una tolleranza a piccoli effetti di rumore (anche in presenza di una tensione spuria di 0.5 volt le configurazioni rimangono stabili)
- Si introduce la possibilità di riconoscimento automatico della presenza di rumore (una tensione di 3 volt corrisponde ad uno stato informativo vietato)

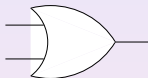
- La tecnologia a semiconduttori consente di realizzare su un'unica piastrina (*chip*) complesse strutture di componenti microscopici (*circuiti integrati*) che nel complesso realizzano *reti logiche* di dimensioni molto grandi.
- I componenti di base di tali circuiti sono detti *porte logiche* (*logic gates*) e realizzano le funzioni logiche elementari AND,OR,NOT.
- Le porte logiche si basano su due valori di tensione dei segnali elettrici, alto (H) e basso (L), che associano ai due valori di verità (alto=true=1, basso=false=0).

Rappresentazione grafica delle porte logiche

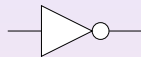
AND



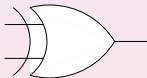
OR



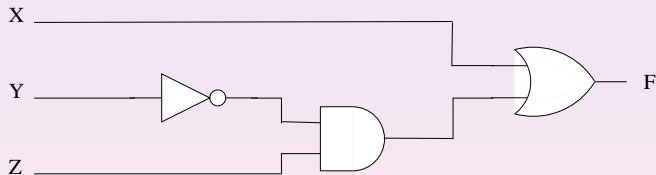
NOT



XOR



- $F = X + \bar{Y} \cdot Z$



Controllo di parità

- Per trasmettere dati su una linea con rumore, si aggiungono bit di controllo per la rilevazione di errori di trasmissione.
- Esempio classico è il *controllo di parità*: si trasmette un bit aggiuntivo (detto *bit di parità*), impostato in modo da rendere pari (*parità pari*) o dispari (*parità dispari*) il numero di bit ad 1 trasmessi.
- La porta XOR può essere vista come un generatore di parità pari o un rivelatore di parità dispari tra due ingressi:

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

- Se il numero di 1 è pari si ottiene 0
- Se il numero di 1 è dispari si ottiene 1

Controllo di parità pari

- Un circuito di parità tra più bit può essere ottenuto con più livelli di porte XOR.

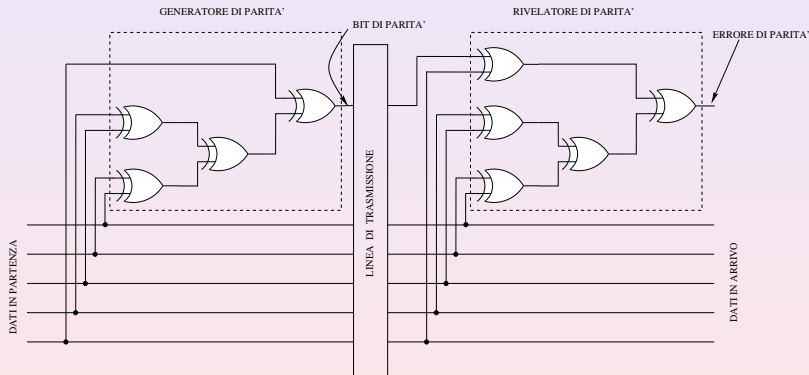


Tabella di verità per l'addizione

Ingressi

c_k	a_k	b_k	s_k	c_{k+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- c_k = riporto precedente (sul k-esimo bit)
- a_k = k-esimo bit del primo addendo
- b_k = k-esimo bit del secondo addendo

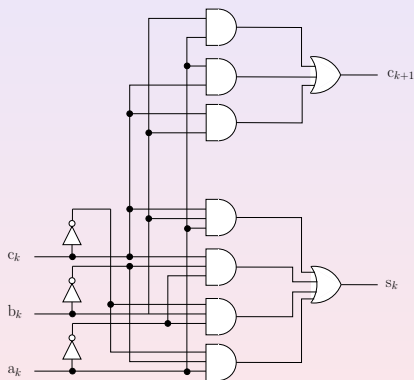
Uscite

- s_k = k-esimo bit del risultato
- c_{k+1} = riporto successivo (sul k+1-esimo bit)

Forma algebrica

- $s_k = \overline{c_k} \overline{a_k} b_k + \overline{c_k} a_k \overline{b_k} + c_k \overline{a_k} \overline{b_k} + c_k a_k b_k$
- $c_{k+1} = \overline{c_k} a_k b_k + c_k \overline{a_k} b_k + c_k a_k \overline{b_k} + c_k a_k b_k$

Circuito per l'addizione: Full Adder

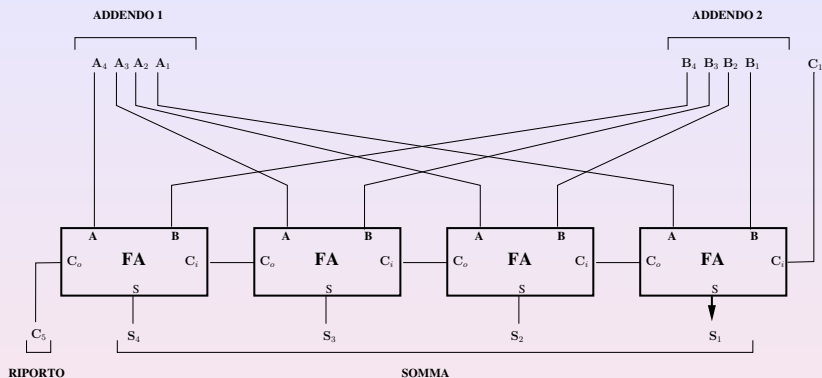


- **Forma algebrica semplificata**

- $s_k = \overline{c_k} \overline{a_k} b_k + \overline{c_k} a_k \overline{b_k} + c_k \overline{a_k} b_k + c_k a_k b_k$
- $c_{k+1} = \overline{c_k} a_k b_k + c_k \overline{a_k} b_k + c_k a_k \overline{b_k} + c_k a_k b_k = a_k b_k + b_k c_k + c_k a_k$

- Le porte a più di due ingressi sono generalizzazioni dei rispettivi operatori logici (AND, OR).
- La semplificazione consente un notevole risparmio in termini di porte logiche utilizzate.

Circuito per l'addizione: Ripple Carry Adder



- Sommatore a 4 bit ottenuto collegando in cascata 4 Full Adders.
- La somma tra numeri in codifica binaria viene dunque effettuata tramite un circuito fatto di porte logiche elementari (AND,OR,NOT).