Architettura dei Calcolatori
Blocchi funzionali logici combinatori

Ing. Gestionale e delle Telecomunicazioni

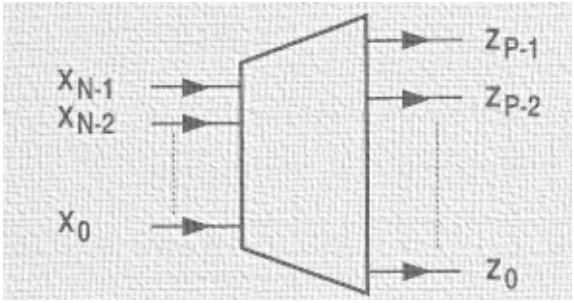
A.A. 2011/12

Gabriele Cecchetti

Blocchi Funzionali Combinatori

- Esiste una ben nota e ormai stabilizzata libreria di blocchi funzionali predefiniti di tipo combinatorio che contiene i blocchi per tutte le funzioni combinatorie di base:
  - questi blocchi appartengono alle famiglie MSI e (alcuni di essi) LSI
  - la libreria contiene anche blocchi funzionali di tipo sequenziale
- I tipici blocchi funzionali combinatori che si vedranno qui sono i seguenti:
  - decodificatore
  - codificatore
  - demultiplexer
  - multiplexer
  - comparatore
  - arbitro di priorità

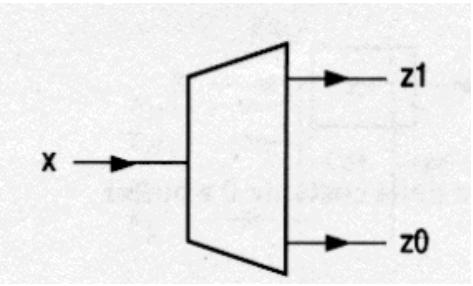
Decodificatore (decoder)

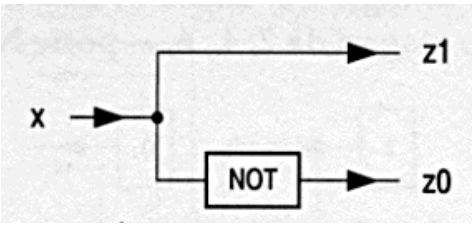


- Il blocco funzionale decodificatore ha:
  - $N \geq 1$ ingressi
  - $P = 2^N \geq 2$ uscite
- Le uscite z_p sono numerate a partire da 0 a P-1
- **Se sugli ingressi è presente il numero binario k, la k-esima uscita assume il valore 1 e le uscite restanti assumono il valore 0,** ossia

$$\forall z_p, p \in \{0,\ldots, P-1\} \Rightarrow$$
$$z_p=1 \Leftrightarrow \langle x_{N-1},\ldots,x_0\rangle \textit{ rappresenta } p,$$
$$z_p=0 \text{ altrimenti}$$

Esempio: Decodificatore 1 a 2



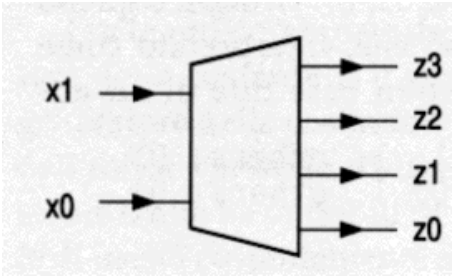


x_0	z_1	z_0
0	0	1
1	1	0

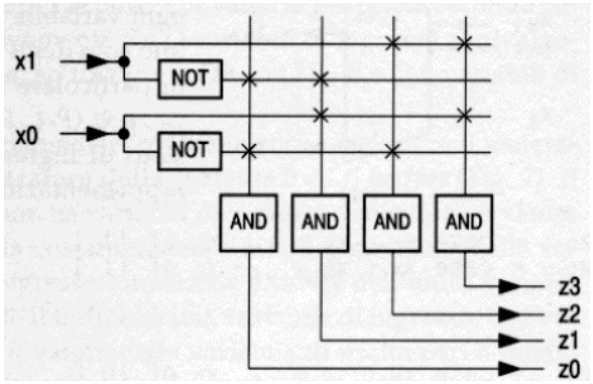
$z_0 = /x_0$

$z_1 = x_0$

Esempio: Decodificatore 2 a 4



x_1	x_0	z_3	z_2	z_1	z_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



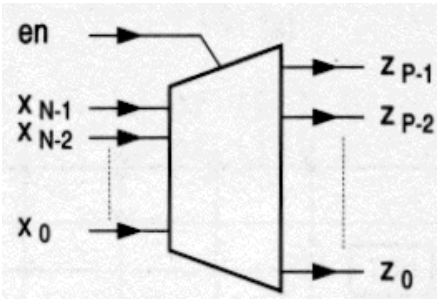
$z_0 = /x_1/x_0$

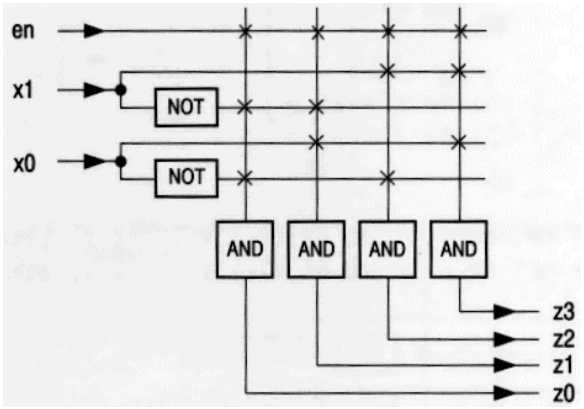
$z_1 = /x_1 x_0$

$z_2 = x_1/x_0$

$z_3 = x_1 x_0$

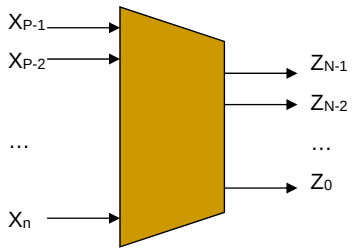
Decodificatore con abilitazione





- **E' un decoder $n \rightarrow 2^n$ dotato di un'ulteriore variabile di ingresso *en*, avente lo scopo di forzare a 0, quando essa stessa vale 0, il valore di tutte le variabili di uscita.**

Codificatore



- Il blocco funzionale codificatore ha:
  - $P = 2^N$ ingressi dati
  - N uscite
- Le uscite z_p sono numerate a partire da 0 a N-1
- **Se un solo l'ingresso *p* vale 1 (gli ingressi restanti valgono 0), le uscite rappresentano il numero binario p, ossia**:

Se x_p=1 e x_q=0 $\forall$ q≠p $\Rightarrow$ <z_{N-1},…,z_0> *rappresenta* p

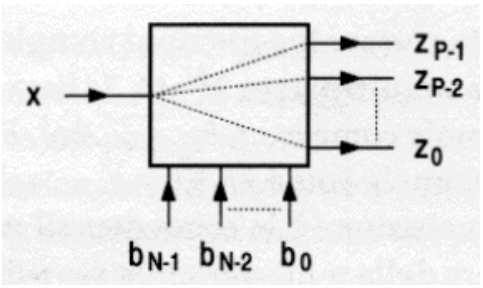
Esempio: codificatore per n=2

x_3	x_2	x_1	x_0	z_1	z_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$z_0 = x_1 + x_3$

$z_1 = x_2 + x_3$

Demultiplexer



- Il blocco funzionale demultiplexer ha:
  - 1 ingresso dati
  - N ≥ 1 variabili di comando
  - P = 2^N uscite numerata da 0 a P-1
- **Se sugli ingressi di selezione è presente il numero binario k, l'ingresso dati viene inviato alla k-esima uscita, le rimanenti sono a 0**, ossia:

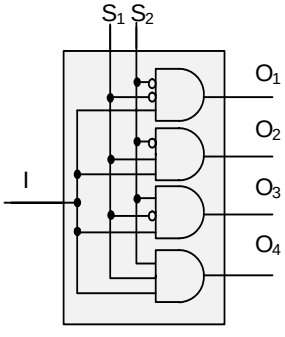
Se $<b_{N-1},\ldots,b_0>$ *rappresenta* p $\Rightarrow$ z_p=x e z_q=0 $\forall$ q≠p

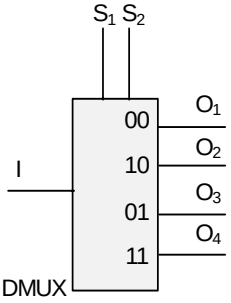
Demultiplexer a due Ingressi

Ingressi	Selezione		Uscite			
I	S_1	S_2	O_1	O_2	O_3	O_4
D	0	0	*D*	0	0	0
D	1	0	0	*D*	0	0
D	0	1	0	0	*D*	0
D	1	1	0	0	0	*D*

$O_1 = /S_1 /S_2 I$ $O_3 = /S_1 S_2 I$

$O_2 = S_1 /S_2 I$ $O_4 = S_1 S_2 I$

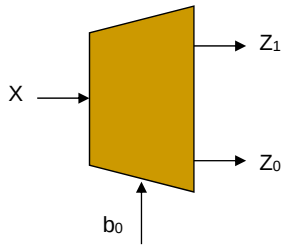


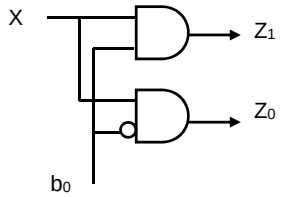


Esempio: demultiplexer per n=2

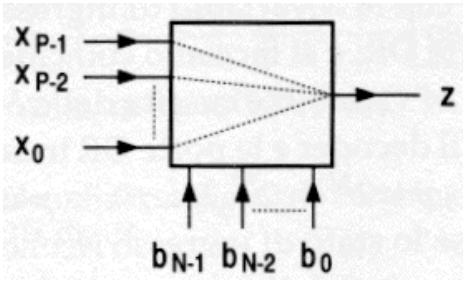
$z_0 = x/b_0$

$z_1 = xb_0$





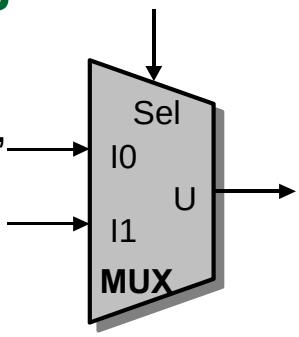
Multiplexer



- Il blocco funzionale Multiplexer ha:
  - P ingressi dati, $P = 2^N \geq 2$, numerati da 0 a $2^N - 1$
  - N variabili di comando o di selezione ≥ 1
  - 1 uscita
- **Se sugli ingressi di selezione è presente il numero binario k, il k-esimo ingresso dati viene inviato in uscita, ossia:**
  se $<b_{N-1},\ldots,b_0>$ *rappresenta* p $\Rightarrow$ z = x_p

Multiplexer a 1 ingresso

- 2 ingressi dati,
  1 ingresso di selezione,
  1 uscita.



I0	I1	Sel (Ctrl)	OUT
D1	D2	0	D1
D1	D2	1	D2

tabella di verità

I0	I1	Sel (Ctrl)	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Sel \ $I_0 I_1$	00	01	11	10
0	0	0	1	1
1	0	1	1	0

mappa di Karnaugh

$OUT = /Sel\ I_0 + Sel\ I_1$

Multiplexer a 2 Ingressi

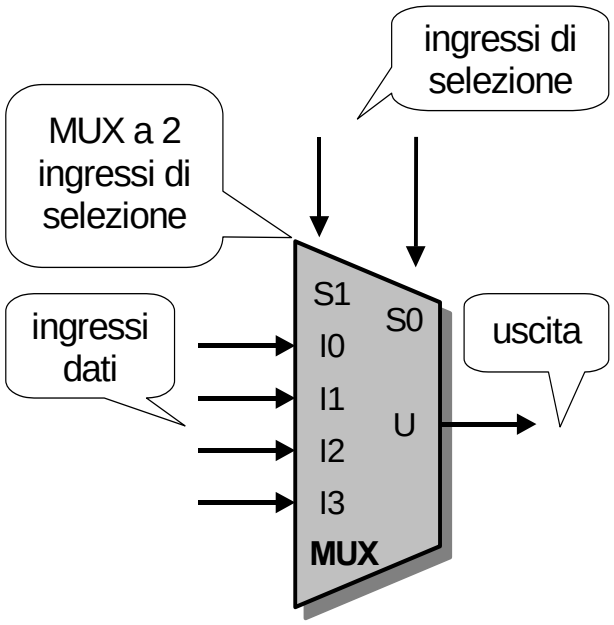
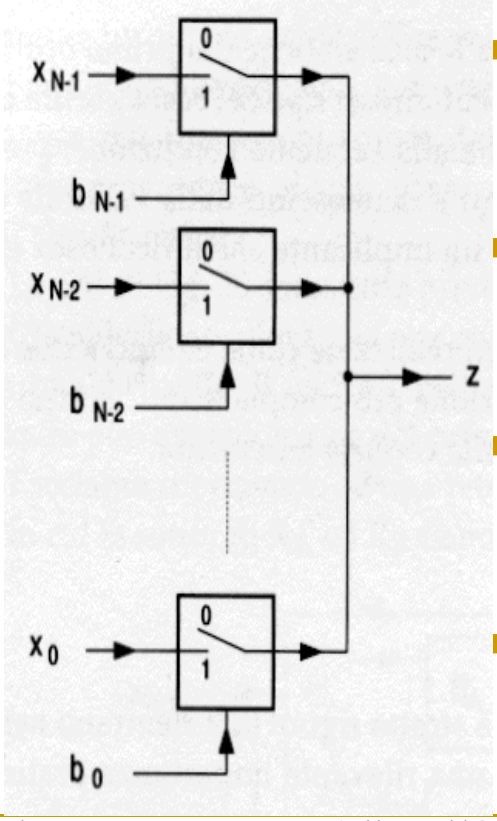


tabella di verità

# riga	S1	S0	I0	I1	I2	I3	U
0	0	0	0	X	X	X	0
1	0	0	1	X	X	X	1
2	0	1	X	0	X	X	0
3	0	1	X	1	X	X	1
4	1	0	X	X	0	X	0
5	1	0	X	X	1	X	1
6	1	1	X	X	X	0	0
7	1	1	X	X	X	1	1

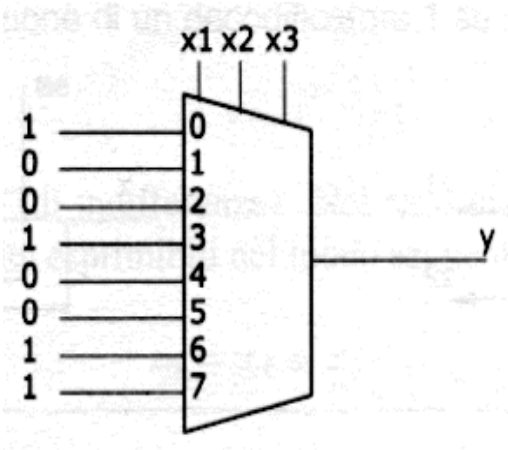
Multiplexer decodificato



- Le var. di comando sono tante quante le var. da commutare
- In ogni istante al più una var. di comando può avere valore 1.
- Quando tutte le var. di comando valgono 0, z è in alta impedenza.
- Quando la b_N=1 con N∈{N-1,N-2,…,0} ⇒ $z = x_N$

I mux per implementare le funzioni logiche

- Una qualunque funzione logica di *n* variabili può essere implementata per mezzo di un Multiplexer a 2^n vie.



Arbitro di priorità

- Risulta asserita la linea di uscita corrispondente alla linea d'ingresso di maggior priorità tra quelle asserite.

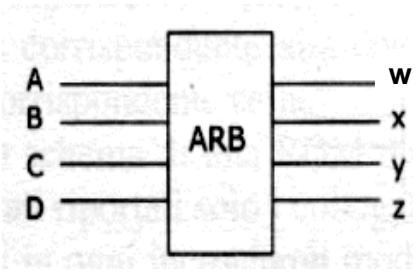
  w = A

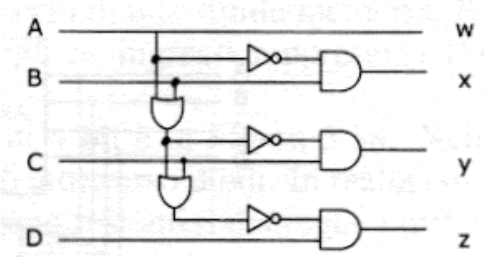
  x = B/A

  y = C/B/A = C /(B+A)

  z = D/C/B/A=D/C/(B+A)

  =D/(C+B+A)

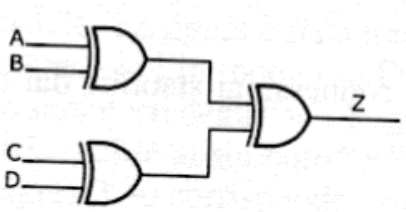




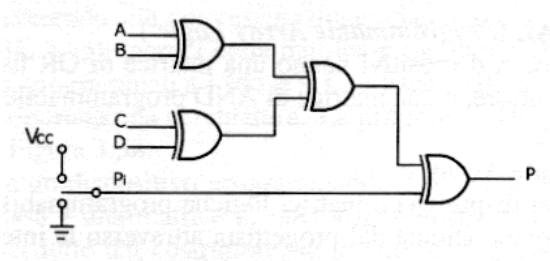
Controllore di parità

- Si usa per rilevare errori dovuti a disturbi o a malfunzionamenti degli apparati, e consiste nell'effettuare un controllo di parità: data una parola di *n* bit.
- Controllarne la parità significa determinare se è pari il numero di bit a 1 nella parola stessa.

Esempio: controllore di parità per n=4 bit

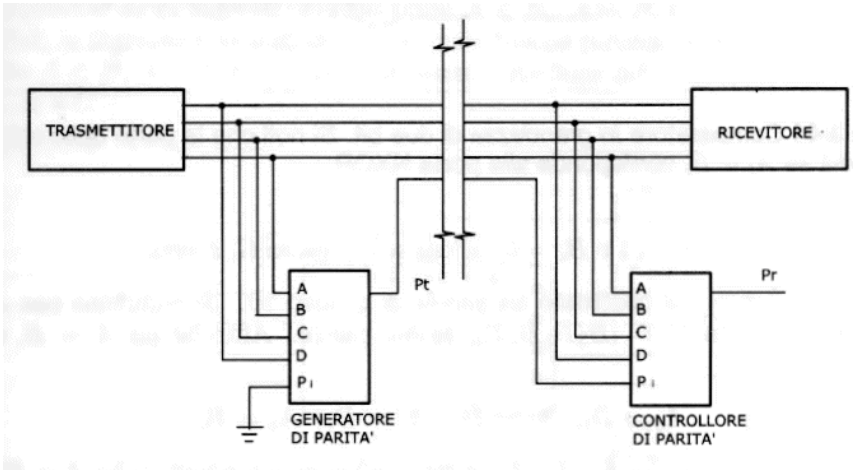


- Questo circuito calcola la parità di 4 bit, per mezzo delle porte XOR (per le quali vale la proprietà associativa)
- Se è dispari il n. di bit in ingresso a 1 allora z=1



- Questo circuito calcola la parità pari o dispari a seconda che Pi sia collegato a Vcc (1) oppure a massa (0)

Esempio di controllo di parità



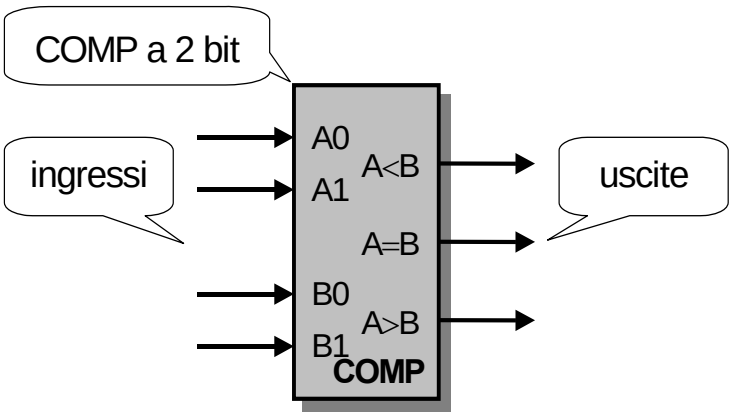
- Il tx. forma una parola di 4 bit + 1 bit di parità.
- Il rx. controlla per mezzo di XOR se il bit di parità ricevuto e la parola hanno la stessa parità. Se la parità è:
  - dispari Pr vale 0 in caso di errore, 1 altrimenti;
  - pari Pr vale 1 in caso di errore, 0 altrimenti.

Comparatore digitale

- Il blocco funzionale comparatore ha:
  - due gruppi A e B d'ingressi da n ≥ 1 bit ciascuno;
  - tre uscite: minoranza A < B, uguaglianza A = B e maggioranza A > B

- Il blocco confronta i due numeri binari A e B da n bit presenti sui due gruppi di ingressi, e attiva (a 1) l'uscita corrispondente all'esito del confronto.

Comparatore digitale a due bit



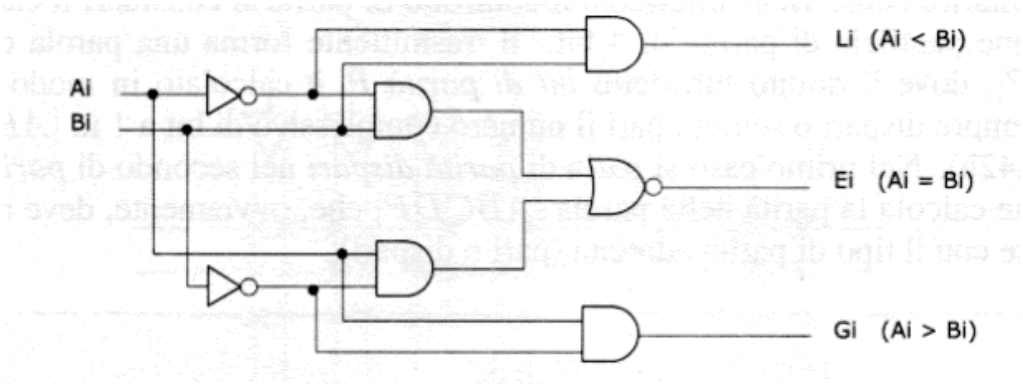
confrontatore di numeri a due bit

tabella di verità

# riga	A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	0	1	0
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	0	0	1
9	1	0	0	1	0	0	1
10	1	0	1	0	0	1	0
11	1	0	1	1	1	0	0
12	1	1	0	0	0	0	1
13	1	1	0	1	0	0	1
14	1	1	1	0	0	0	1
15	1	1	1	1	0	1	0

Estendere quindi al caso per *n* bit.

Comparatore digitale per un bit



- $G_i = A_i /B_i$
- $E_i = A_iB_i + /A_i/B_i$
- $L_i = /A_iB_i$

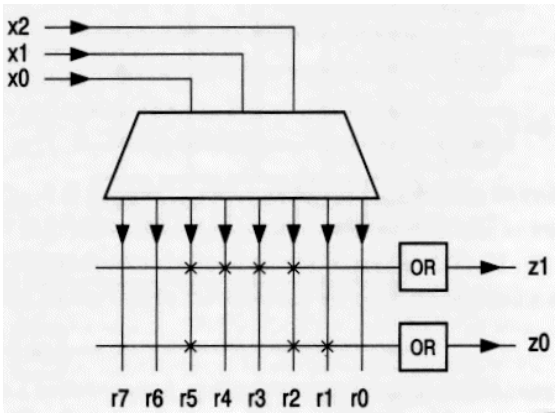
Modello strutturale per reti combinatorie

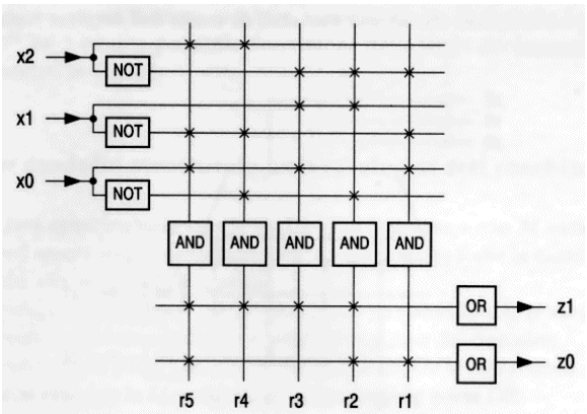
Concetto ed esempio

Modello strutturale per le reti combinatorie

- Una rete combinatoria con N var. d'ingresso può essere implementata, quale che sia la legge F che la caratterizza, in accordo alla seguente procedura:
  1. **decodificatore N→2^N** a cui colleghiamo in ingresso le *N var. d'ingresso* della rete
  2. **M porte OR collegate alle M uscite** della rete
  3. si **interconnette il decodificatore e le porte OR in accordo alla seguente regola**:
     sia r_p, p=2^N-1,2^N-2, …,0, la *p*-esima variabile di uscita del decoder; **se e solo se lo stato di ingresso riconosciuto da r_p è fra gli stati d'ingresso riconosciuti dalla *m*-esima var. di uscita z_m della rete**, *m*=*M*-1,*M*-2,…,0, **allora r_p deve essere una della var. d'ingresso della porta OR che ha come var. di uscita z_m**

Esempio di modello strutturale per reti combinatorie per N=3 e M=2



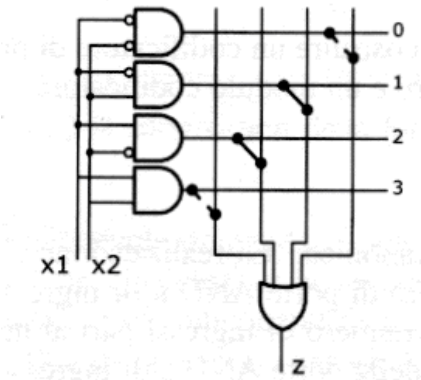


Memorie

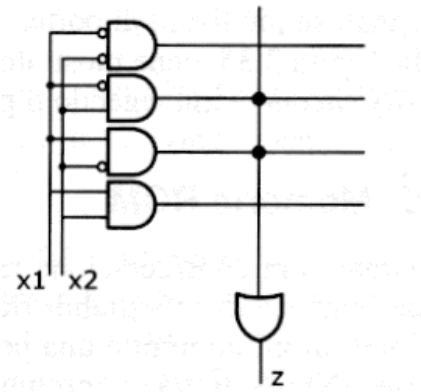
ROM, PROM, EPROM, EEPROM, PLA, PAL, FPGA

Memoria ROM (1/2)

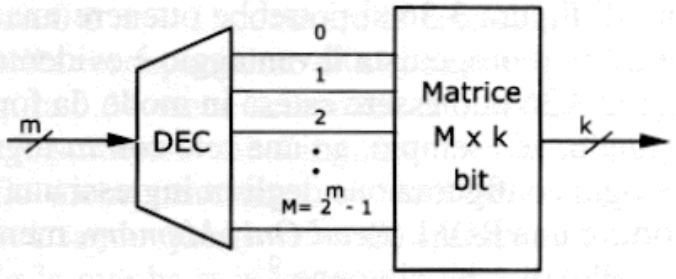
- Considerata la forma canonica SP di una funzione F di *m* variabili, per realizzarla occorrono porte a *m* ingressi pari al numero dei *mintermini* e una porta OR con numero d'ingressi pari al numero delle porte AND.



- Fissare i contatti tra le uscite delle porte AND e gli ingressi della porta OR equivale a programmare il contenuto della rete.

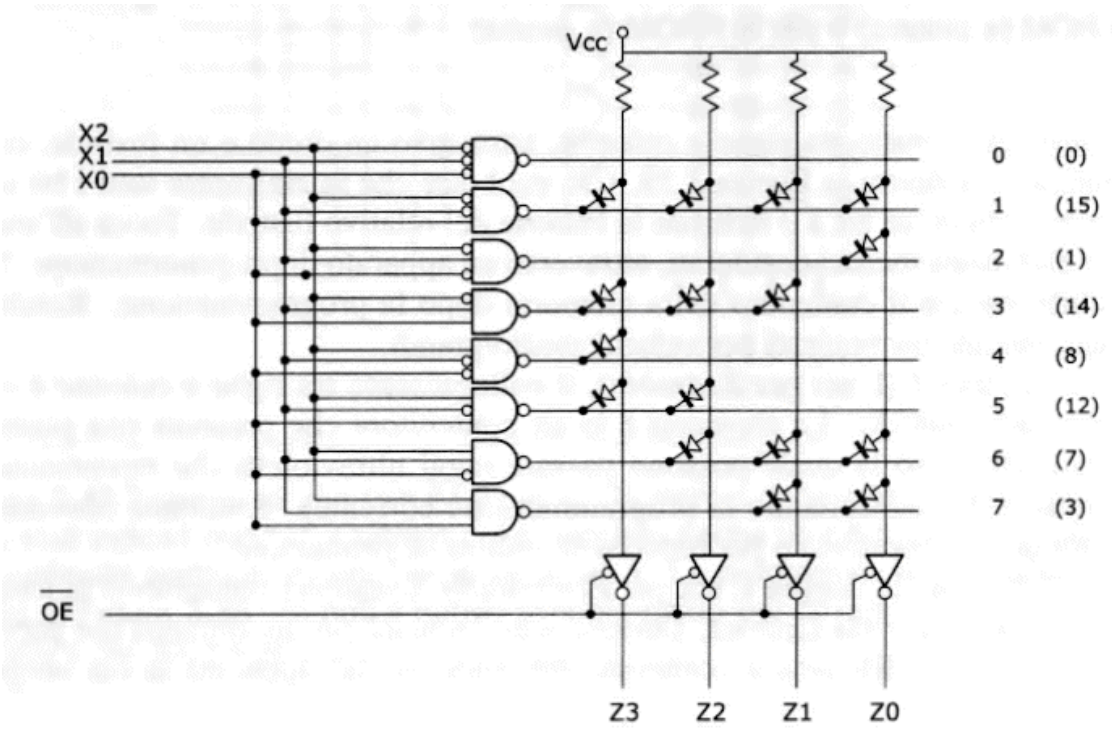


Memoria ROM (2/2)



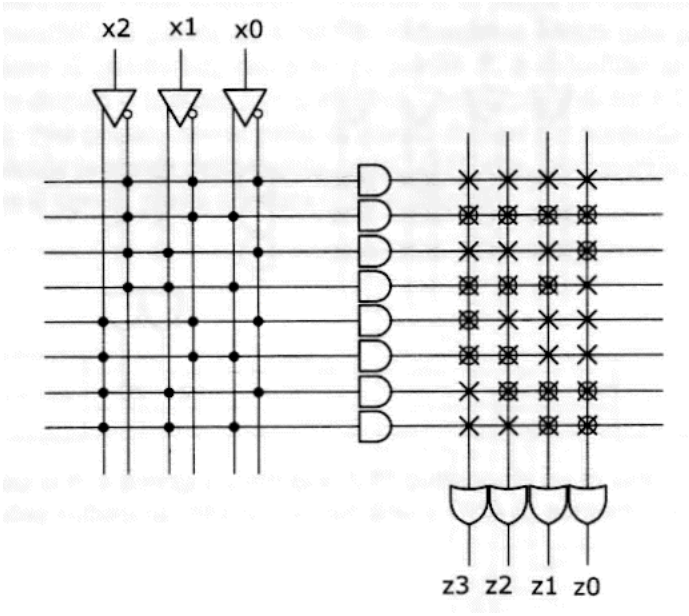
- In generale, possiamo realizzare una memoria ROM (Read Only Memory) avente *m* ingressi e *k* uscite.
- Questa memoria ha 2^m celle ognuna ampia *k* bit.

Esempio di memoria ROM



- Memoria ROM da 8 posizioni di 4 bit.

PROM (*Programmable Read Only Memory*)



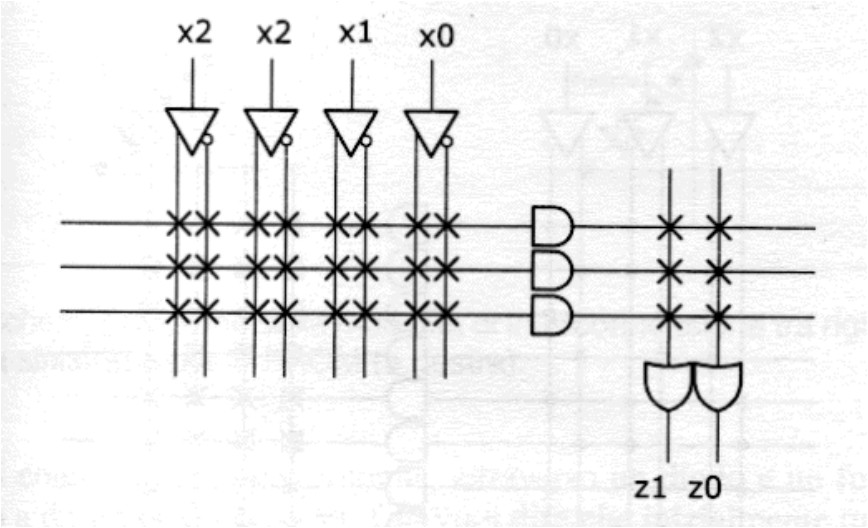
- La matrice ha inizialmente tutti i punti di contatto tra righe e colonne attraverso un diodo e un fusibile (bit a 1).
- Il programmatore inserisce gli 0 negli incroci desiderati attraverso un apparato di programmazione.

EPROM, EEPROM

- **EPROM**: si programmano come le PROM, ma **è possibile cancellare la programmazione corrente esponendo la piastrina al quarzo**, posta in superficie, **ai raggi ultravioletti**.
  Sono programmabili migliaia di volte.

- **EEPROM**: **si programmano e si cancellano per via elettrica**, di solito all'interno dello stesso apparecchio nel quale sono utilizzate.
  Sono programmabili centinaia di migliaia di volte.

PLA



- E’ un dispositivo in cui è possibile programmare sia la matrice AND che la matrice di OR.

PAL, Gate Array

- **PAL**: ha la *matrice di OR fissa*, con connessioni stabilite dal costruttore, e una *matrice di AND programmabile*.

- **GATE ARRAY**: danno la possibilità al progettista di programmare le interconnessioni tra *macrocelle* e *macrofunzioni*, già disponibili nel *gate array*.

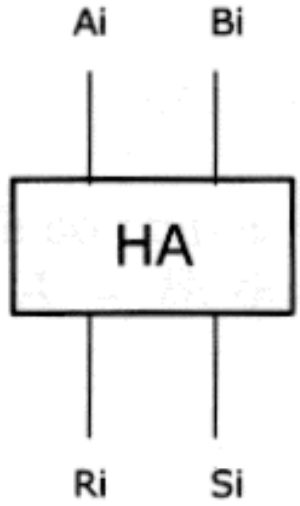
Blocchi aritmetici combinatori

Semisommatore, sommatore completo,
Sommatore con *Look-Ahead Carry Generator*
Unità aritmetico logiche (ALU)

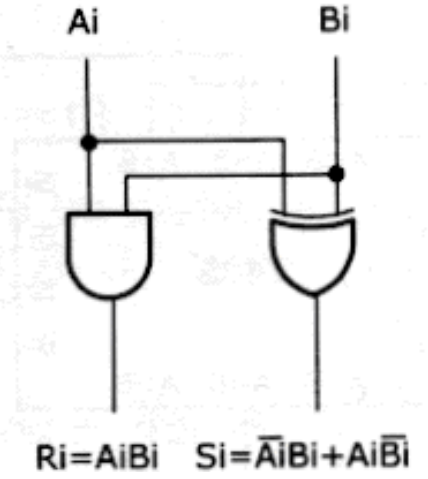
Sommatore (Addizionatori)

- Il circuito (combinatorio) base per il calcolo dell'addizione è l'addizionatore a un bit ("full adder" o "addizionatore completo").
- Ha in ingresso:
  - due addendi a_i e b_i, da un bit ciascuno
  - il riporto in ingresso r_i, da un bit
- Ha in uscita:
  - la somma s_i, da un bit
  - il riporto in uscita r_{i+1}, da un bit
- Esso opera per numeri naturali, ma funziona anche senza modifiche per numeri in complemento a due, giacché l'algoritmo di addizione è il medesimo.
- La versione senza riporto in ingresso si chiama "half adder" o "semiaddizionatore".

Semisommatore (*Half adder*)

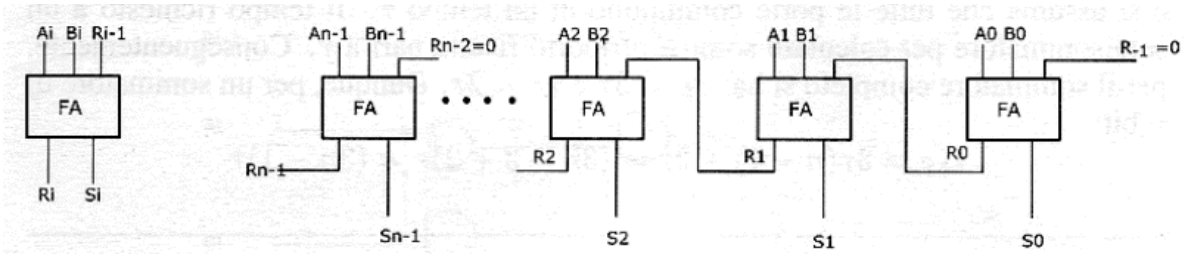


Ai	Bi	Si	Ri
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Nota. Il semisommatore non tiene conto di un eventuale riporto in ingresso

Somma di due interi A e B



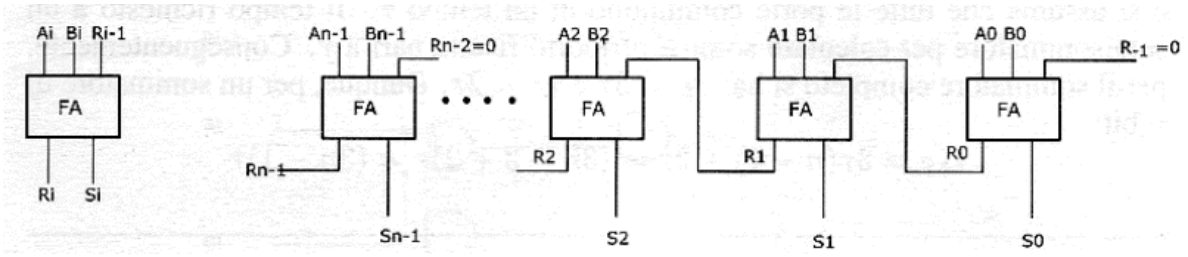
- $A = \{A_{n-1},\ldots,A_0\}$
- $B = \{B_{n-1},\ldots,B_0\}$
- $A + B = S$, ove $S = \{S_{n-1},\ldots,S_0\}$, e
- l'eventuale riporto è R_n

Tempo necessario per la somma

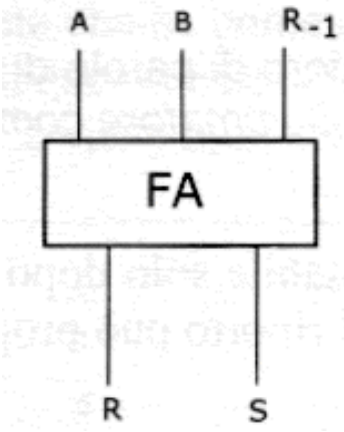
- La rete vista per la somma può produrre un risultato dopo che per ogni FA è divenuto stabile il riporto in ingresso R_i.
- Nel peggiore dei casi:

$\Delta_R = n\tau_R$ per R_{n-1}

$\Delta_S = (n-1)\tau_R + \tau_S$ per S



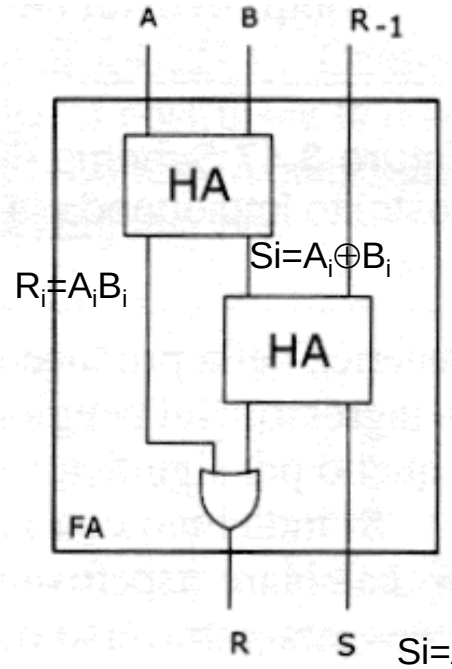
Come è fatto il *Full Adder* ?



Ai	Bi	Ri-1	Si	Ri
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

- Se consideriamo la sua tabella di verità ricaviamo che:
- $S_i = A_i \oplus B_i \oplus R_{i-1}$
- $R_i = A_iB_iR_{i-1} + /A_iB_iR_{i-1} + A_i/B_iR_{i-1} + A_iB_i/R_{i-1} =$
  $= A_iB_i (R_{i-1}+/R_{i-1}) + (/A_iB_i+A_i/B_i)R_{i-1} =$
  $= A_iB_i + (A_i \oplus B_i)R_{i-1}$

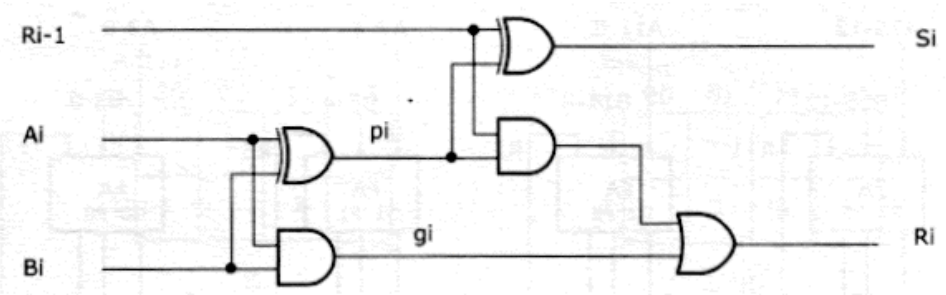
Sintesi del *Full Adder*



$Si=Ai \oplus Bi \oplus R_{i-1}$

$R_i=A_iB_i + (Ai \oplus Bi)R_{i-1}$

Somma con calcolo anticipato del riporto



- Considerata la rete del *Full-Adder* si ponga

  $p_i = A_i \oplus B_i$ e $g_i = A_iB_i$

  Dunque
- $S_i = A_i \oplus B_i \oplus R_{i-1} = p_i \oplus R_{i-1}$
- $R_i = A_iB_i + (A_i \oplus B_i)R_{i-1} = g_i + p_iR_{i-1}$

Calcolo anticipato del riporto per n=4 bit

- $R_0 = g_0 + p_0 R_{-1}$
- $R_1 = g_1 + p_1 R_0 = g_1 + p_1 g_0 + p_1 p_0 R_{-1}$
- $R_2 = g_2 + p_2 R_1 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 R_{-1}$
- $R_3 = g_3 + p_3 R_2 =$

  $= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 R_{-1}$

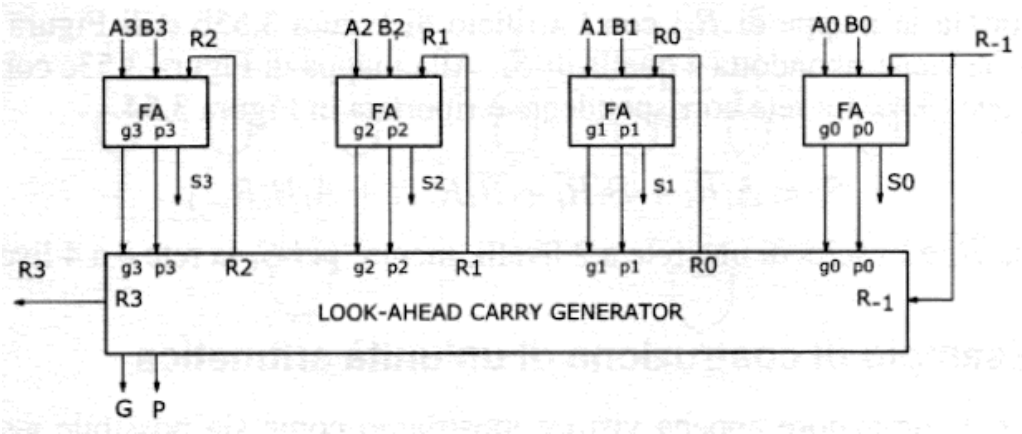
posto $G = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$ e

$P = p_3 p_2 p_1 p_0$

si ha: $R_3 = G + P R_{-1}$

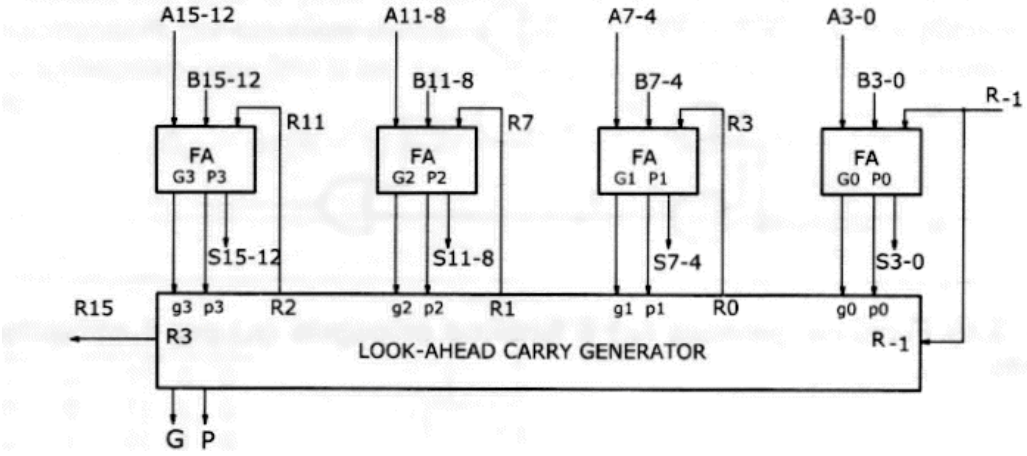
- G e P sono ottenuti tramite una rete combinatoria che ha in ingresso solo A e B e quindi possono essere calcolati immediatamente,

Sintesi del calcolo anticipato del riporto per n=4 bit



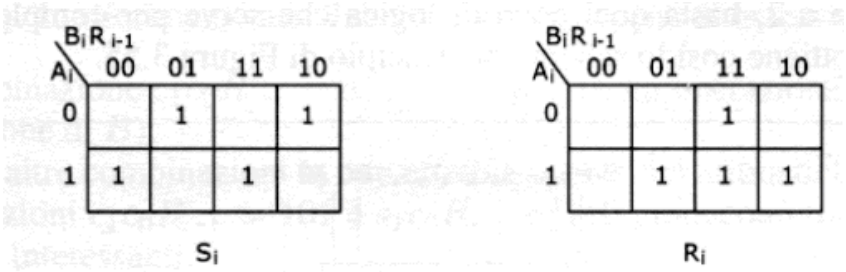
- Schema del sommatore di parole di 4 bit con calcolo anticipato del riporto.

Sintesi del calcolo anticipato del riporto per n=16 bit



- Si ottiene iterativamente utilizzando 4 volte il precedente schema.

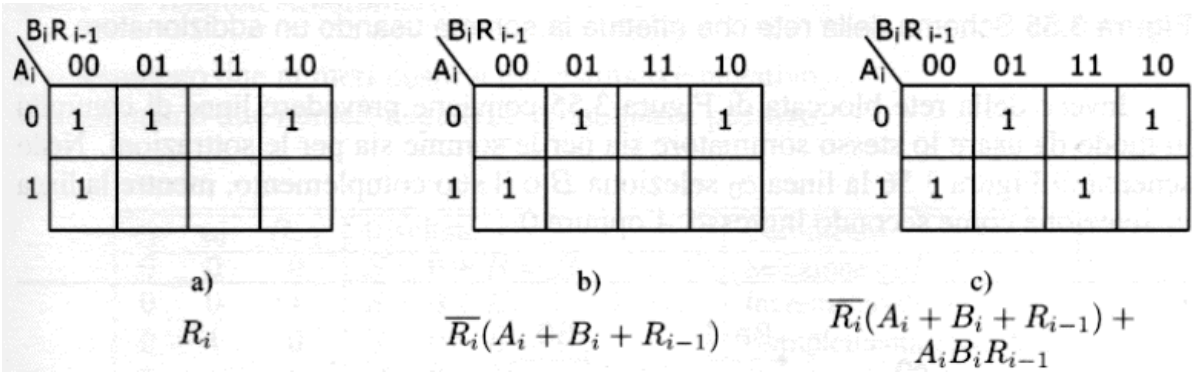
Realizzazione alternativa del sommatore completo (1/3)



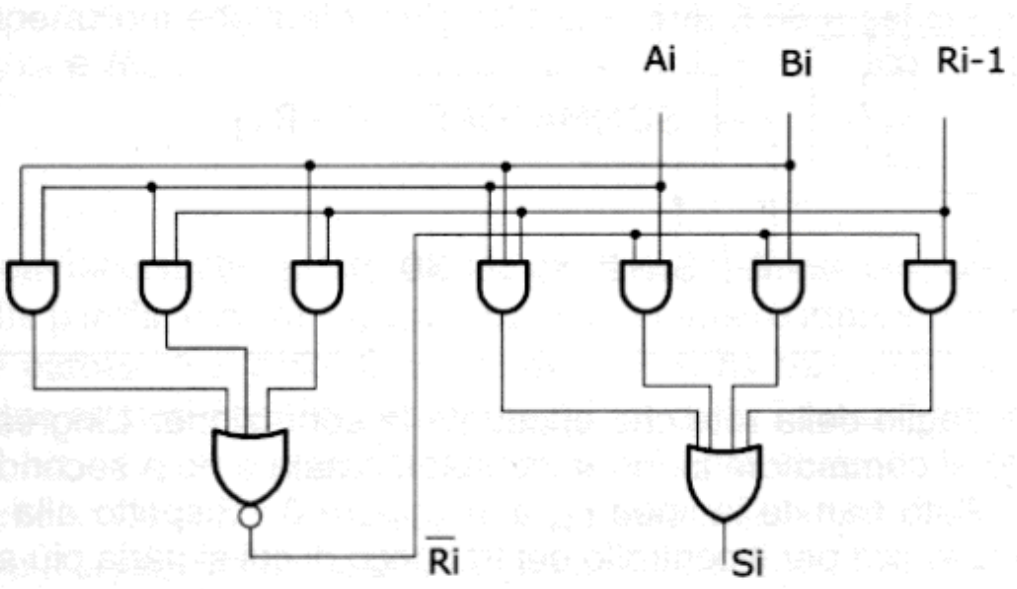
- Consideriamo sommatore completo
- S_i e R_i hanno le seguenti mappe di Karnaugh

Realizzazione alternativa del sommatore completo (2/3)

- Esprimiamo R_i in modo tale da ricondurci ad S_i



Realizzazione alternativa del sommatore completo (3/3)

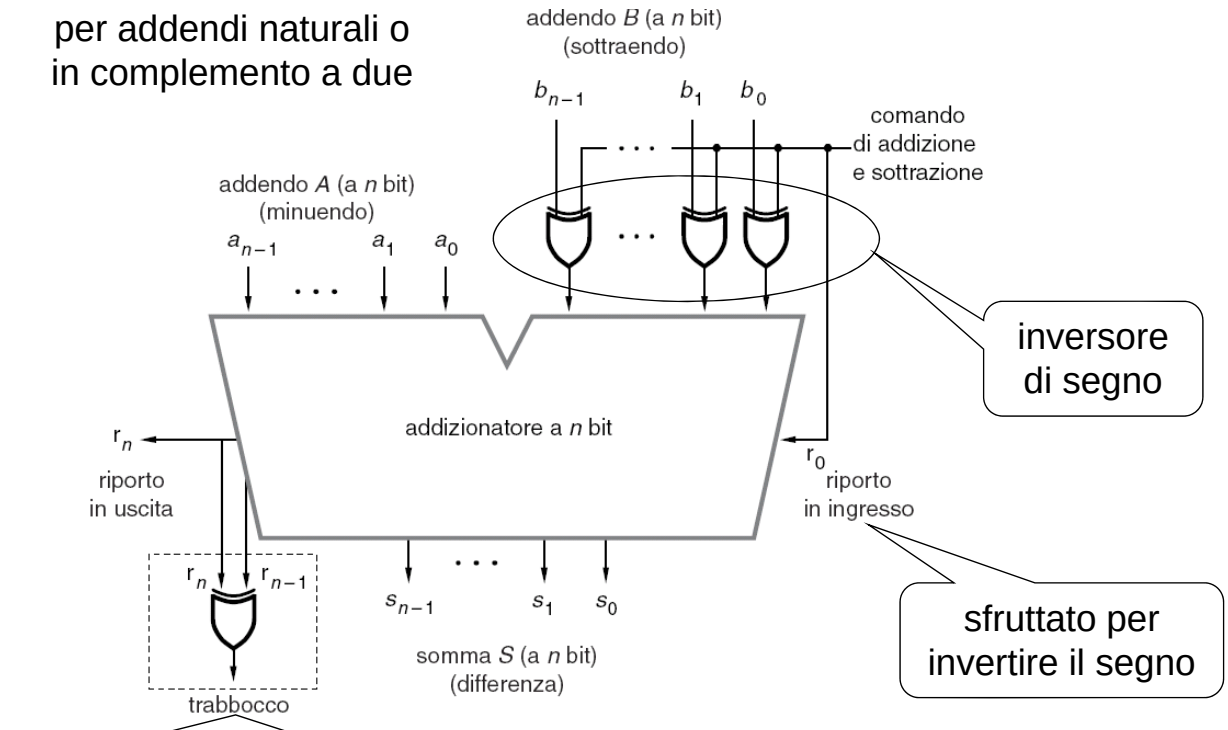


Addizione algebrica

- Dato che in complemento a due addizione e sottrazione sono sostanzialmente operazioni unificate, il circuito che le calcola è fondamentalmente uno solo.
- Nella forma più semplice, l'addizionatore algebrico è a propagazione di riporto (come quello naturale), salvo permettere di invertire (in C_2) il sottraendo, a comando.
- Il circuito ha:
  - due ingressi A e B a *n* bit (addendi o minuendo e sottraendo)
  - un ingresso di comando (addizione o sottrazione)
  - un'uscita S a *n* bit (somma)
  - il riporto in uscita e la segnalazione di trabocco in uscita
- L'addizionatore algebrico funziona sia per numeri naturali sia per numeri in complemento a due, senza bisogno di distinguere.

Schema logico di addizionatore algebrico

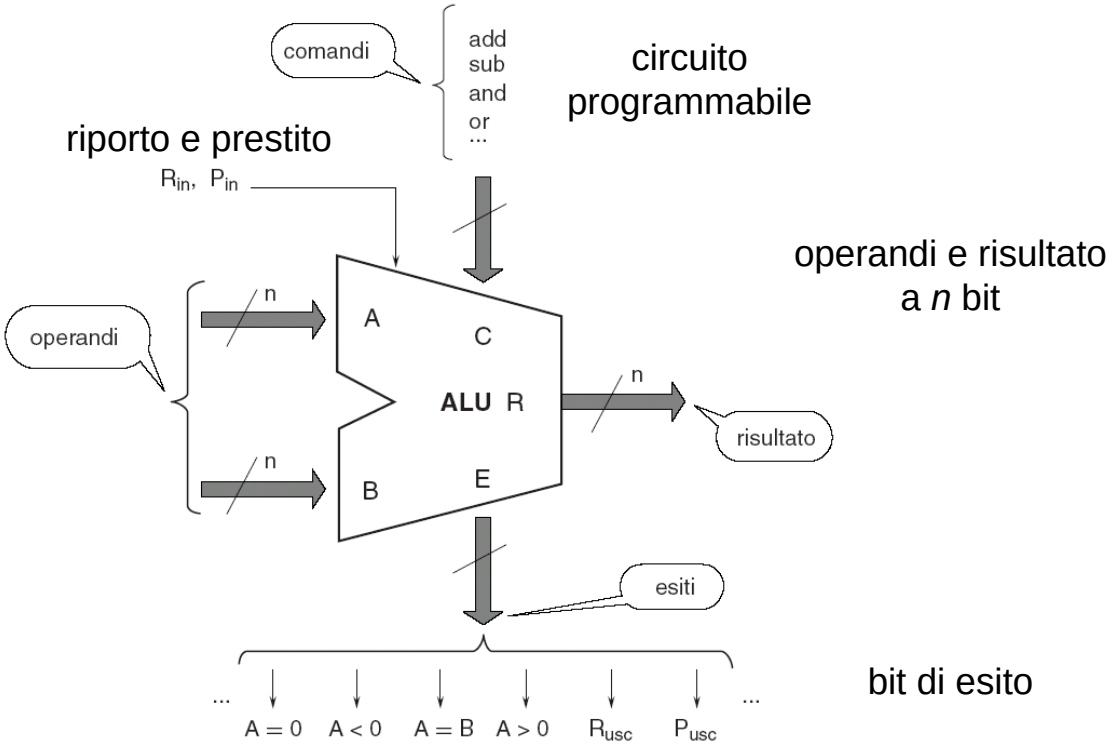
per addendi naturali o
in complemento a due



Unità Aritmetica-Logica (ALU)

- Il circuito (combinatorio) algebrico completo è l'unità aritmetica-logica (*Arithmetic-Logic Unit*, ALU).
- È un dispositivo programmabile, capace di eseguire un repertorio di operazioni (per numeri interi):
  - aritmetiche (addizione, sottrazione, moltiplicazione, ecc)
  - logiche (negazione, somma e prodotto logico bit a bit, ecc)
  - di confronto (uguale, diverso, maggiore, ecc)
- Il repertorio di operazioni varia secondo la ALU.
- Esistono anche ALU per numeri reali.

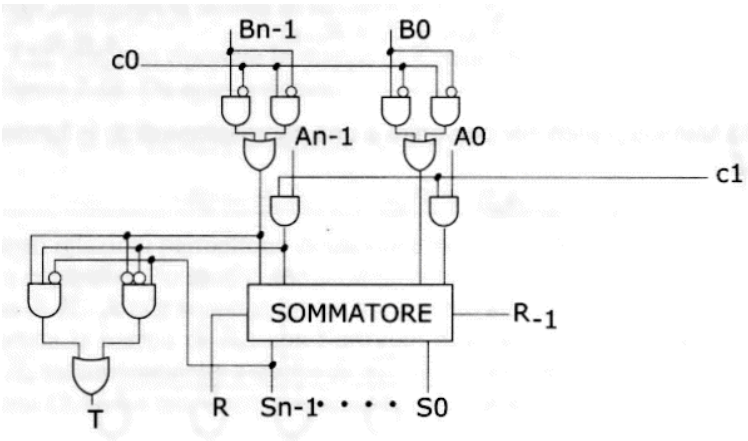
Schema logico di una ALU

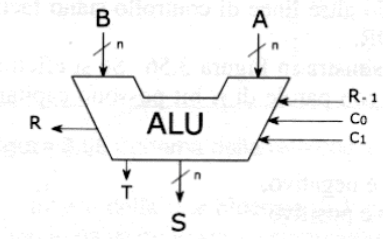


Possibili funzioni di una ALU

# riga	comando	operazione	risultato	esito
0	add	somma A e B	$A + B + R_{in}$	riporto in uscita R_{usc}
1	sub	sottrae B da A	$A - B - P_{in}$	prestito in uscita P_{usc}
2	pass A	A passa in uscita	A	-
3	pass B	B passa in uscita	B	-
4	zero	annulla uscita	0	-
5	shift left A	A scorre a sx	2 A	bit più significativo di A
6	shift right A	A scorre a dx	A / 2	bit meno significativo di A
7	null	confronta A con 0	-	$A = 0$
8	compare	confronta A con B	$A <=> B$	$A < B$, $A = B$, $A > B$
9	multiply	prodotto di A e B	$A \times B$	trabocco
10	divide	divisione A / B	A / B	divisione per 0?
...	...	...	...	...

Esempio di implementazione di una ALU





- Rete che effettua somma, sottrazione, incremento ed altre operazioni a seconda del valore delle variabili di comando c1 e c0.

Fenomeni transitori nelle reti combinatorie

Alee

Eliminazione delle alee

Fenomeni transitori nelle reti combinatorie

- Avvengono quando viene cambiato lo stato d'ingresso
- Sono dovuti al fatto che le reti non sono ideali (porte e fili non ideali)
- Questi fenomeni transitori possono provocare *Alee*, cioè:
  - oscillazione temporanea di una o più var. di uscita
  - produzione di alcuni stati di uscita indesiderati
  - non produzione di alcuni stati di uscita desiderati

Come annullare le Alee ? (1/3)

Quando uno stato di ingresso $X(i)$ viene rimosso e sostituito da uno stato $X(i+1)$ occorre che:

1) la **rete sia pilotata in modo fondamentale**, cioè $X(i+1)$ venga presentato solo quando la rete è a regime;
2) **non vi siano transizioni multiple in ingresso**, cioè $X(i)$ e $X(i+1)$ differiscano al più per una sola var.;
3) la funzione F caratterizzante la rete sia tale che **$F\{X(i)\}$ e $F\{X(i+1)\}$ differiscano al più nel valore di una variabile**.

Come annullare le Alee ? (2/3)

- La prima condizione evita che il transitorio che si accende quando viene presentato $X(i+1)$ si accavalli al transitorio, non ancora concluso, che si era acceso quando era stato presentato $X(i) \Rightarrow$ rischio di non produrre correttamente $F\{X(i)\}$.
- La seconda condizione evita che la rete riceva una raffica di stati di ingresso spuri quando lo stato $X(i)$ viene rimosso.
- La terza condizione evita che la rete generi una raffica di stati di uscita spuri.

Come annullare le Alee ? (3/3)

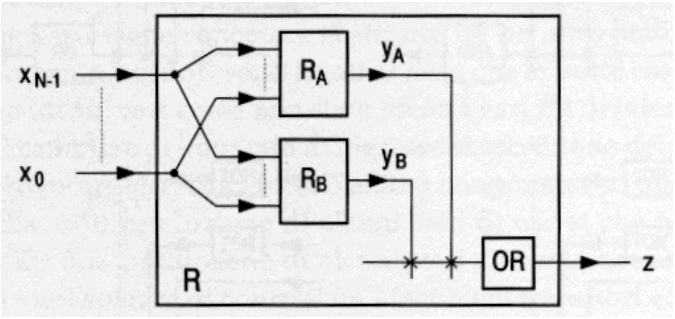
- Poiché una rete combinatoria con M var. di uscita può essere vista come M reti indipendenti ciascuna con una sola var. di uscita:
  - Il problema di eliminare le alee in una qualunque rete la cui legge F soddisfa la condizione 3) può essere rimandato a quello di **eliminare le alee in reti combinatorie che abbiano N var. di ingresso ed una sola var. di uscita** e che siano pilotate dal mondo esterno consistentemente con quanto richiesto dalle condizioni 1) e 2).

Alee statiche e dinamiche

- Si dice che una rete combinatoria con N var. di ingresso ed una var. di uscita presenta un'***alea del primo ordine*** se esiste almeno uno stato di ingresso che pur applicato alla rete nel pieno rispetto delle condizioni 1) e 2) provoca più transizioni spurie della var. di uscita, mentre non dovrebbe provocarne alcuna (caso dell'***alea statica***) o soltanto una (caso dell'***alea dinamica***).

Alee statiche del primo ordine

- Le reti R_A e R_B sono prive di alee del primo ordine.
- La rete R non può avere alee del primo ordine ma solo alee statiche e solo sul livello 1



- Ciò accade se ad uno stato di ingresso *X*(*i*) cui corrisponde ad es.
  $y_A=1, y_B=0 \quad \Rightarrow z=1$
- è seguito nel tempo da uno stato d'ingresso *X*(*i*+1) cui corrisponde
  $y_A=0, y_B=1 \quad \Rightarrow z=1$
- l'uscita *z* dovrebbe rimanere 1,invece può andare temporaneamente a 0 se R_A è più veloce di R_B, ossia:
- $y_A=1, y_B=0 \quad \Rightarrow z=1$
- $y_A=0, y_B=0 \quad \Rightarrow z=0$
- $y_A=0, y_B=1 \quad \Rightarrow z=1$

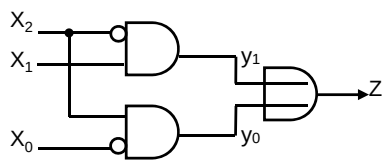
Alee statiche del primo ordine – soluzione (1/2)

- Per eliminare tali alee occorre implementare R in modo ridondante, assicurando che *se X(i) ed X(i+1) sono due stati di ingresso adiacenti e prevedono entrambi z=1*,
  allora ci sia una sottorete RC che non sia influenzata dalla var. di ingresso che commuta nel passare da *X*(*i*) a *X*(*i*+1) e la cui var. di uscita y_C rimanga costantemente pari a 1.

Alee statiche del primo ordine – soluzione (2/2)

- Per eliminare allora le alee statiche del primo ordine sul livello 1 da una rete realizzata come circuito di tipo SP basta partire da una lista di copertura che soddisfi anche alla seguente condizione:
  - presi due qualunque stati di ingresso adiacenti e riconosciuti dalla var. di uscita della rete, **la lista deve prevedere un *implicante che li riconosca entrambi*.**

Esempio pratico di rimozione delle alee statiche del primo ordine (1/2)



x_2	x_1	x_0	y_1	y_0	z
0	1	0	1	0	1
0	1	1	1	0	1
0	*1*	*0*	*1*	*0*	*1*
1	*1*	*0*	*0*	*1*	*1*
1	0	0	0	1	1
1	1	1	0	1	1

- Di questa rete controllo tutti gli ingressi per cui z=1
- Li ordino considerando le coppie di ingressi adiacenti
- Controllo se le var. intermedie possono provocare alee

Esempio pratico di rimozione delle alee statiche del primo ordine (2/2)

- Le alee le vedo anche per mezzo delle mappe di Karnaugh
- Si noti che passando dalla cella 010 alla cella 110 si *passa da un implicante all'altro* $\Rightarrow$ **ciò può provocare un'alea**!
- Per eliminarla bisogna **introdurre un ulteriore implicante ridondante** che copra contemporaneamente le due caselle

x_0 \ x_2x_1	00	01	11	10
0	0	1	1	1
1	0	1	0	0

x_0 \ x_2x_1	00	01	11	10
0	0	1	1	1
1	0	1	0	0