

Corso di Calcolatori Elettronici I

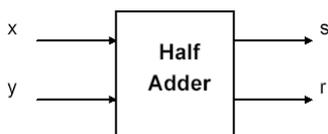
Macchine combinatorie: addizionatori binari

Prof. Roberto Canonico



Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica
e delle Tecnologie dell'Informazione
Corso di Laurea in Ingegneria Informatica
Corso di Laurea in Ingegneria dell'Automazione

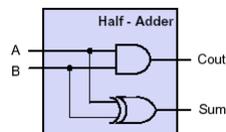
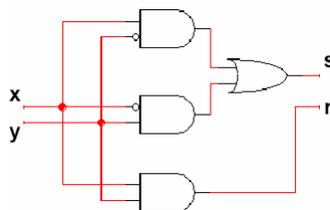
Half Adder



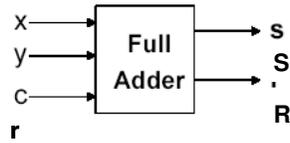
x	y	s	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \bar{X}Y + X\bar{Y} = X \text{ xor } Y$$

$$r = XY$$



Full Adder (1/2)



X	Y	r	S	R
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

S

r \ xy	00	01	11	10
0		1		1
1	1		1	

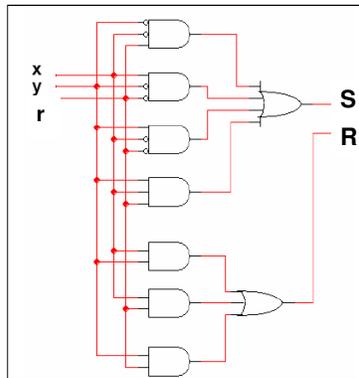
R

r \ xy	00	01	11	10
0			1	
1		1	1	1

Full Adder (2/2)

$$S = \bar{x} \cdot \bar{y} \cdot r + \bar{x} \cdot y \cdot \bar{r} + x \cdot \bar{y} \cdot \bar{r} + x \cdot y \cdot r$$

$$R = \bar{x} \cdot y \cdot r + x \cdot \bar{y} \cdot r + x \cdot y \cdot \bar{r} + x \cdot y \cdot r = x \cdot y + x \cdot r + y \cdot r$$



Addizionale binario

- E' possibile isolare il fattore $(a \oplus b)$
- Rielaborando le precedenti espressioni è possibile ottenere le seguenti espressioni per l'addizionale completo:

$$S = (a \oplus b) \oplus r = P \oplus r$$

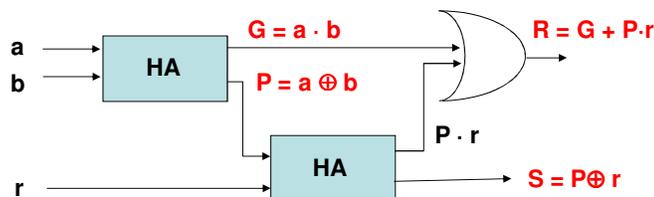
$$R = a \cdot b + r \cdot (a \oplus b) = G + r \cdot P$$

Addizionale binario

- Pertanto, un addizionale completo può essere ottenuto a partire da due semiaddizionatori:

$$S = (a \oplus b) \oplus r = P \oplus r$$

$$R = a \cdot b + r \cdot (a \oplus b) = G + r \cdot P$$



Addizionatore binario: riporto

- Le diverse componenti dell'espressione di R assumono un significato particolare:
 - $\mathbf{G = a \cdot b}$ “**riporto generato**”: indica la creazione di un riporto all'interno dell'addizionatore binario
 - $\mathbf{P = a \oplus b}$ “**riporto propagato**”: indica se, in presenza di un riporto in ingresso, lo stesso verrà propagato in uscita
 - Il riporto in uscita può quindi essere espresso come $\mathbf{R = G + P \cdot r}$

Addizionatori binari

$$n_i = \overline{r_i}$$

Non-riporto

Indica assenza di riporto in ingresso ($r=0$)

$$K_i = \overline{a_i} \cdot \overline{b_i}$$

Riporto “ucciso” (“killed”)

Indica che, indipendentemente dalla presenza di un riporto entrante, il riporto in uscita sarà comunque zero

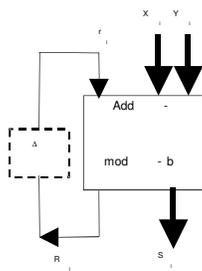
$$N_i = K_i + P_i \cdot n_i$$

Propagazione del non-riporto

Indica assenza di riporto in uscita ($R=0$)

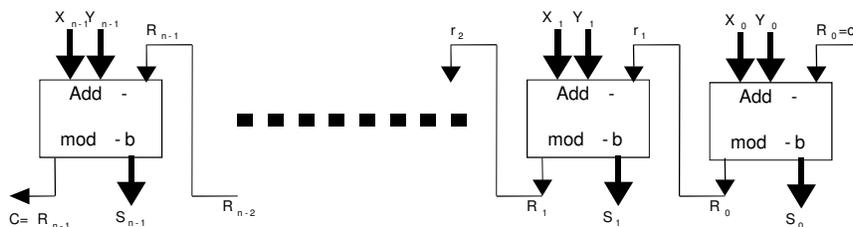
Addizionatori seriali

- Usa un unico addizzatore operante sulla singola cifra
- Opera in momenti successivi su cifre diverse degli addendi
- Richiede un blocco "con memoria"
- E' lento rispetto ad addizionatori che lavorano in parallelo sulle diverse cifre degli addendi



Addizzatore binario parallelo

- Opera sulle cifre degli addendi in parallelo
- ...anche se il riporto deve propagarsi attraverso l'intera struttura
- Richiede un numero maggiore di risorse rispetto all'addizzatore seriale



Addizionatore parallelo: tempo di risposta

- Gli addizionatori ottenuti collegando in cascata n addizionatori di cifra sono anche chiamati addizionatori a propagazione del riporto (*carry-ripple* o *carry-propagate*)
- ε = tempo di risposta di uno stadio
- Allo stadio i , il riporto uscente:
 - o è **generato** o è **ucciso** o è **propagato**
- Tempo di ritardo complessivo: **Limite inferiore** ε (in tutti gli stadi il riporto è generato o ucciso)
- Tempo di ritardo complessivo: **Limite superiore** $n\varepsilon$ (un riporto entrante nel primo stadio che è propagato in tutti gli stadi)
- Tempo di ritardo complessivo = $k\varepsilon$ ($k \leq n$), dove k è la più lunga catena di condizioni di propagazione.

Adder con anticipo del riporto

- Normalmente chiamati addizionatori *carry lookahead*
- Per ogni stadio i , dal $(k+1)$ -esimo al $(k+j)$ -esimo, r_i si ottiene direttamente dai bit degli addendi X ed Y e dal riporto entrante nella catena (invece che dal riporto uscente R_{i-1})

$$r_{k+1} = G_k + r_k P_k$$

$$r_{k+2} = G_{k+1} + G_k P_{k+1} + r_k P_k P_{k+1}$$

.....

$$r_{k+j} = G_{k+j-1} + G_{k+j-2} P_{k+j-1} + \dots + G_k P_{k+1} \dots P_{k+j-1} + r_k P_k P_{k+1} \dots P_{k+j-1}$$

Adder carry lookahead

$$r_{k+j} = G_{k+j-1} + G_{k+j-2}P_{k+j-1} + \dots + G_k P_{k+1} \dots P_{k+j-1} + r_k P_k P_{k+1} \dots P_{k+j-1}$$

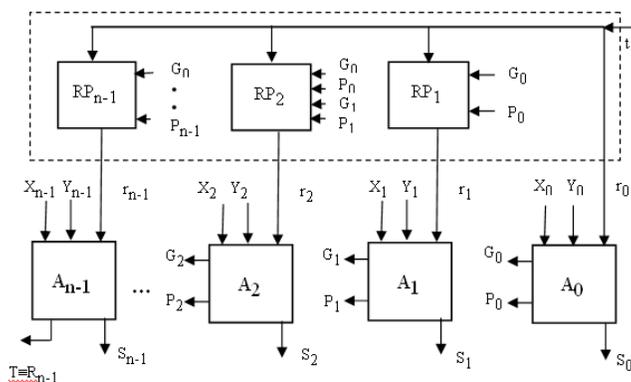
r_{k+j} è alto se è verificata la condizione di generazione nell'ultimo stadio

...oppure se è verificata la condizione di generazione G_k e se questa viene propagata dagli stadi dal $(k+1)$ -esimo fino all'ultimo

...oppure è pari al riporto entrante r_k se questo viene propagato in tutti gli stadi

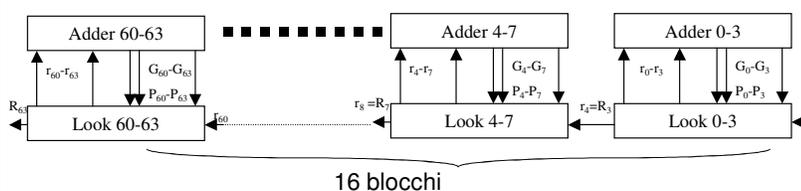
Addizionatori carry lookahead

- L'espressione precedente può essere realizzata nella maniera riportata in figura



Addizionatori carry lookahead

- L'idea di base negli addizionatori carry lookahead è quella di calcolare la relazione tra r_k ed r_{k+j} separatamente per ogni gruppo di cifre $k+1 \dots k+j$
- Una rete a livello superiore valuta la propagazione del carry tra *gruppi* di cifre
- Ad esempio, per un adder a 64 bit suddiviso in blocchi di quattro cifre (bit), la parte di "look" lungo cui si propaga il riporto è lunga $64/4=16$ stadi



Addizionatori carry lookahead

