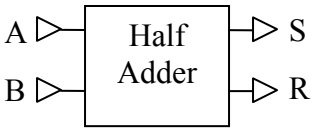
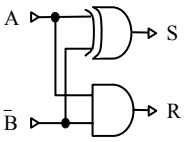
Esercitazione del 17/03/2005

1) Addizionatore Half Adder (senza riporto in ingresso):

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



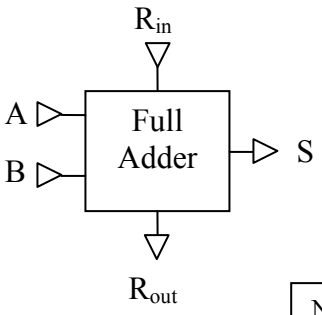


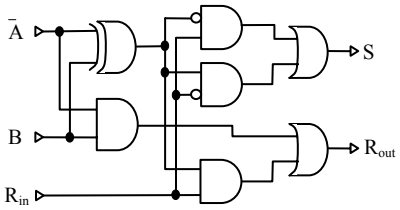
N.Porte = 2 *Cammino Critico* S = 1 , R = 1

$S = A \oplus B \qquad R = A \cdot B$

2) Addizionatore Full Adder (con riporto in ingresso):

R_{in}	A	B	S	R_{out}
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





N.Porte = 7 *Cammino Critico* S = 3 , R = 3

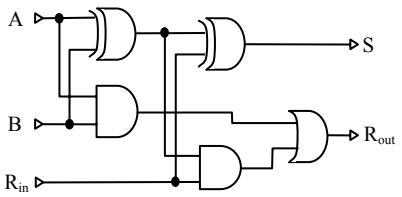
$S = {\sim}R_{in}{\sim}AB + {\sim}R_{in}A{\sim}B + R_{in}{\sim}A{\sim}B + R_{in}A\,B$
$= {\sim}\,R_{in}({\sim}AB + A{\sim}B) + R_{in}({\sim}A{\sim}B + A\,B\,)$
$= {\sim}R_{in}(A \oplus B\,) + R_{in}{\sim}\,(A \oplus B)$

$R_{out} = {\sim}R_{in}AB + R_{in}{\sim}AB + R_{in}A{\sim}B + R_{in}AB$
$= AB\,({\sim}R_{in} + R_{in}) + R_{in}\,({\sim}AB + A{\sim}B)$
$= AB + R_{in}(A \oplus B)$

Nota: **~A~B + A B** = ~A~B + A B = ~(~(~A~B + A B))
= ~(~(~A~B) ~ (A B)) = ~((~~A+~~B) (~A+~B))
=~((A+B) (~A+~B)) = ~(A(~A+~B)+B(~A+~B))
=~(A~A+A~B+B~A+B~B) = ~(A~B+B~A) = **~ (A ⊕ B)**

Semplificazione circuitale:

$S = {\sim}R_{in}(A \oplus B\,) + R_{in}{\sim}\,(A \oplus B) = R_{in} \oplus (A \oplus B\,)$

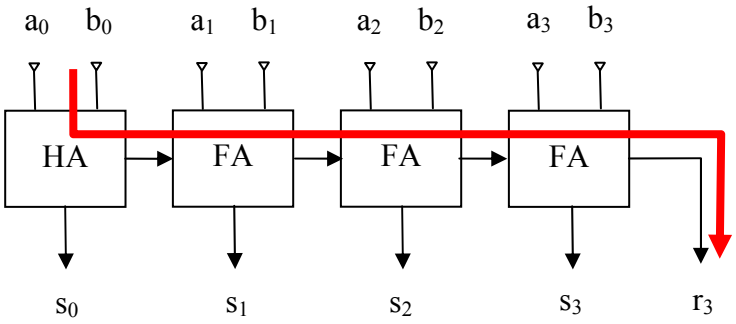


N.Porte = 5 *Cammino Critico* S = 2 , R = 3

***Nota**: le porte **AND e la OR possono essere pensate come "gate"** che lasciano "passare" il segnale presente su un terminale in funzione del segnale presente sull'altro. Diversamente **le porte XOR si comportano come "invertitori"**: il segnale presente su un terminale viene lasciato passare o invertito a seconda del segnale presente sull'altro.*

3) Sommatore ad n bit (senza riporto in ingresso)

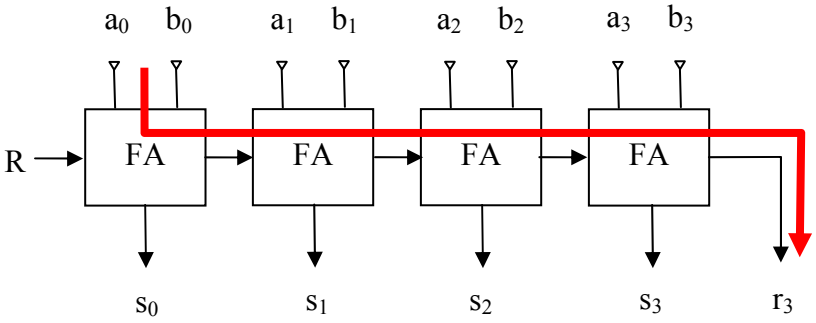
E' possibile realizzare un sommatore ad n bit usando un HA e n-1 FA collegati in cascata.



Il cammino critico è quello definito dalla propagazione dei riporti dal primo modulo all'ultimo. Il **cammino critico** quindi per questo sommatore è **1 + 3 * (n-1)** dove **n** è il numero di bit da sommare.

4) Sommatore ad n bit (con riporto in ingresso)

Normalmente si preferisce adottare un FA anche per il primo modulo. Questo permette di mettere in cascata più sommatori e di realizzare semplicemente un sottrattore binario in complemento a due (vedi più avanti).



In questo caso quindi il **cammino critico** è **3 * n**.

5) Moltiplicazioni binarie

(*Rappresentazione dell'Informazione – Conversione da base 10 a base n , slide 24-25)*

a. $1001_2 * 110_2 = ?_2$

$1001_2 * 110_2 = 1001_2 * 0_2 + 1001_2 * 10_2 + 1001_2 * 100_2$
$= 0_2 + 10010_2 + 100100_2$
$= 110110_2$

		1	0	0	1	*	
			1	1	0	=	
					0	+	0
	1	0	0	1	←	+	1
1	0	0	1	←		=	1
1	1	0	1	1	0		

La moltiplicazione binaria si esegue in maniera simile alla moltiplicazione decimale classica. Per le proprietà dell'aritmetica binaria tutto il processo si riduce ad una sequenza di shift a sinistra e somme.

Per ogni cifra del moltiplicatore si esegue un shift a sinistra del primo termine. Il termine shiftato si somma al totale solo se il corrispondente bit del secondo termine è a 1. In altri termini, ogni shift viene messo in AND con il corrispondente bit del moltiplicatore e quindi sommato.

b. $101_2 * 1101_2 = ?_2$

				1	0	1	*	
			1	1	0	1	=	
				1	0	1	+	1
					0	←	=	0
		1	0	1	←			1
		1	*1*	*^{1}0*	*0*	*1*		
	1	0	1	←				1
1	^{1}0	^{1}0	^{1}0	0	0	1		→1000001_2

c. $11101_2 * 1011_2 = ?_2$

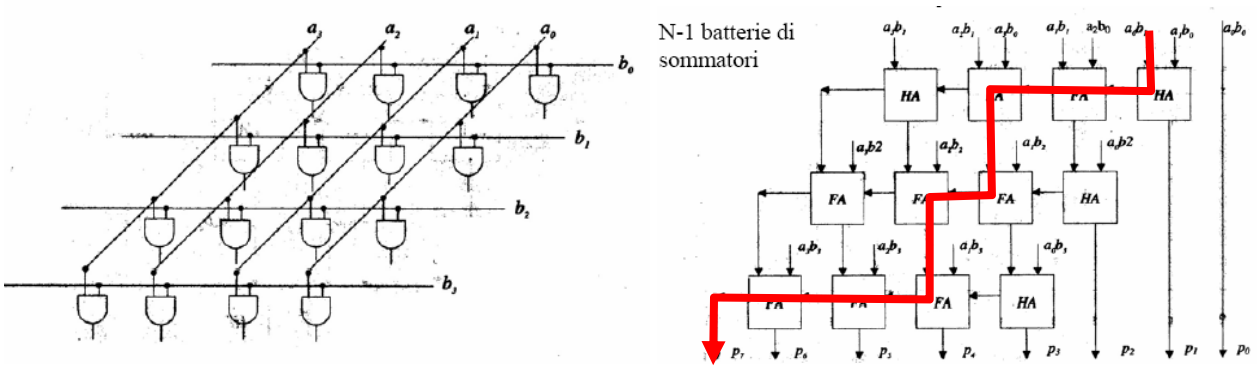
				1	1	1	0	1	*	
					1	0	1	1	=	
				1	1	1	0	1	+	1
			1	1	1	0	1	←	=	1
		1	*^{1}0*	*^{1}1*	*^{1}0*	*1*	*1*	*1*	+	
						0	←			0
	1	1	1	0	1	←				1
1	^{1}0	^{1}0	1	1	1	1	1	1		→100111111_2

d. $10111_2 * 111_2 = ?_2$

			1	0	1	1	1	*	
					1	1	1	=	
			1	0	1	1	1	+	1
		1	0	1	1	1	←	=	1
	1	*^{1}0*	*^{1}0*	*^{1}0*	*^{1}1*	*^{1}0*	*1*	+	
	1	0	1	1	1	←			1
1	^{1}0	1	^{1}0	^{1}0	^{1}0	0	1		→10100001_2

Il circuito della moltiplicazione può essere visto come una matrice di AND seguita da una matrice di sommatori FA-HA. Ogni bit b_i del moltiplicatore si comporta come un gate per la corrispondente parola del moltiplicando opportunamente shiftata a sinistra di i posizioni.

Nota: L'operazione di shift a sinistra equivale in algebra binaria ad una moltiplicazione per 10_2, analogamente lo shift a destra equivale ad una divisione per 10_2.



Nota: HA è più rapido di FA:

HA → *Cammino Critico* S = 1 , R_{out} = 1

FA → *Cammino Critico* S = 2 , R_{out} = 3

quindi il percorso critico a parità di moduli attraversati è quello che attraversa un maggior numero di FA.

Il percorso più lungo è quello che propaga gli effetti dei bit a_0 e b_0 su r_n. Per propagare il riporto verso sinistra occorrono 3 passaggi, per eseguire un livello di sommatorie occorrono 2 passaggi.

*Nota: la lunghezza del risultato di una moltiplicazione sarà al più la somma della lunghezza dei due termini. Es: 8 bit * 8 bit →16 bit $1111\ 1111_2 * 1111\ 1111_2 = 1111\ 1110\ 0000\ 0001_2$*

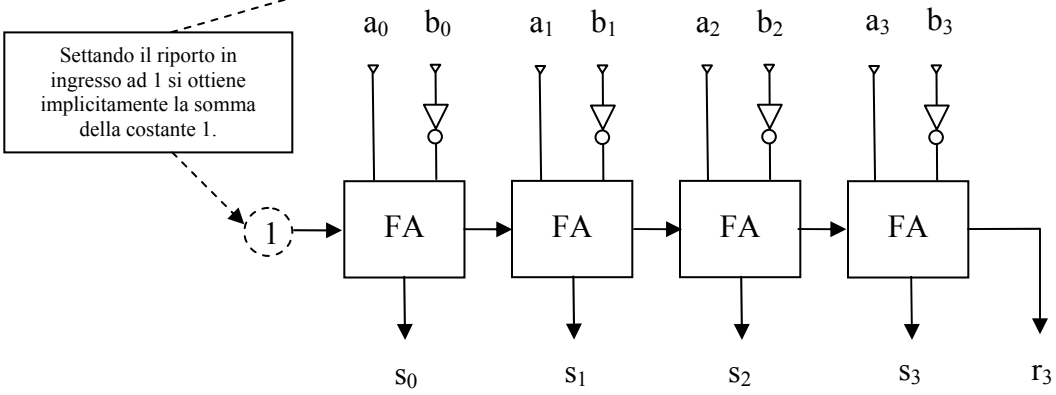
6) Sottrattori ad n bit

In binario la sottrazione ad n bit (con segno) può essere realizzata con una somma secondo la regola:

$$\mathbf{A - B = A + \sim B + 1}$$

dove **~B** è l'inversione bit a bit del secondo termine.

Adottando un sommatore con riporto in ingresso è possibile realizzare un sottrattore binario a n bit in questo modo:



7) Problema dell'overflow

Nel caso che il risultato di un addizione con segno ecceda il limite di rappresentazione della dimensione della parola si verifica un errore di segno (Overflow).

Esempio:

$19_{10} + 15_{10} = 34_{10}$ se eseguita in aritmetica binaria con segno a 6 bit dà come risultato:

1	*1*	*1*	*1*	*1*		*R*
0	**1**	**0**	**0**	**1**	**1**	+
0	**0**	**1**	**1**	**1**	**1**	=
1	**0**	**0**	**0**	**1**	**0**	

$0\ 10011_2 + 0\ 01111_2 = 1\ 00010_2$ che in rappresentazione in complemento a due è un numero negativo , precisamente - $11110_2 = -30_{10}$.

Analogamente si verifica lo stesso effetto di riporto errato sul bit di segno quando si sommano due numeri negativi in valore assoluto troppo grandi:

$-17_{10} + -19_{10} = -35_{10}$ *se eseguita in aritmetica binaria con segno a 6 bit dà come risultato:*

$-17_{10} = -\ 10001_2$ (M&S) = 101111_2 (CA2)

$-19_{10} = -\ 10011_2$ (M&S) = 101101_2 (CA2)

1	*1*	*1*	*1*	*1*	*1*		*R*
	1	**0**	**1**	**1**	**1**	**1**	+
	1	**0**	**1**	**1**	**0**	**1**	=
~~**1**~~	**0**	**1**	**1**	**1**	**0**	**0**	

$1\ 01111_2 + 1\ 01101_2 = 0\ 11100_2$ che in rappresentazione in complemento a due è un numero positivo , precisamente + $00100_2 = 4_{10}$.

Nel caso di somme miste, negativo con positivo e viceversa questo problema non si pone poichè il risultato in valore assoluto sarà sempre al più grande come il più grande in valore assoluto dei due termini sommati.

Circuito per rilevare l'overflow:

Da quanto visto sopra ne segue che l'overflow si verifica solo nei seguenti due casi:

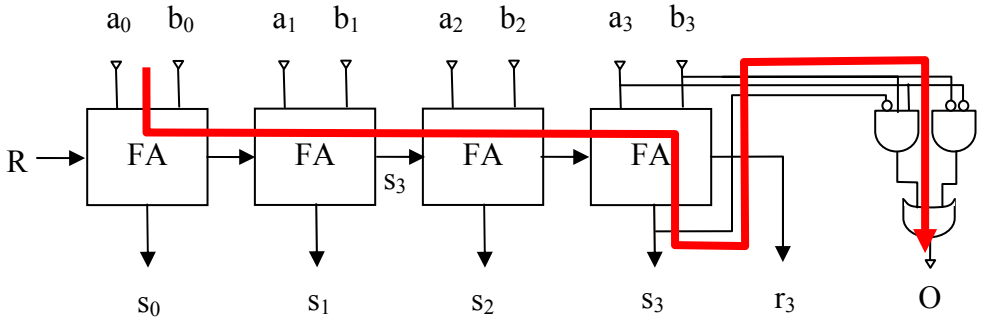
Segno di A = 0 e Segno di b = 0 e Segno del risultato = 1 oppure

Segno di A = 1 e Segno di b = 1 e Segno del risultato = 0

Il circuito che realizza questo controllo sintetizzera la seguente forma tabellare:

r_{n-1}	a_{n-1}	b_{n-1}	Overflow
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	0
1	1	1	0

In forma SOP si può scrivere: $O = \sim r_{n-1}\, a_{n-1}\, b_{n-1} + r_{n-1} \sim a_{n-1} \sim b_{n-1}$



Cammino critico: 3(n-1) +2 +2

8) ALU: Arithmetic Logic Unit:
Vedi appunti lezione

9) Sommatori veloci: Carry Look-Ahead.
Vedi appunti lezione