NOTE DI RILASCIO

Le figure in questo documento sono riportate dal testo "Introduzione all'architettura dei calcolatori" Capitolo 9 . Per approfondimenti consultare il libro.

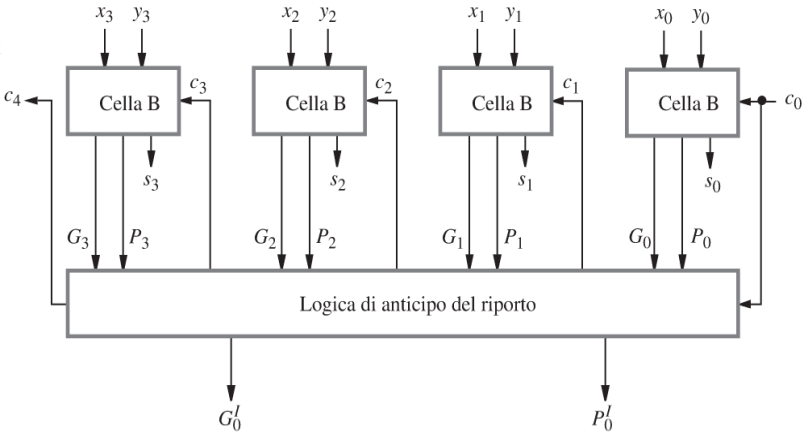
Il circuito mostrato offre solo l'operazione di somma di due numeri binari. Ecco alcuni spunti per integrare e migliorare il progetto per operazioni più complesse e interessanti:

Circuito Full-Adder a *n* bit con anticipo del riporto:

Nella somma di una quantità significativa di bit, la propagazione del riporto richiede un tempo relativamente elevato, dato che la somma di una certa cifra è corretta solo se è già avvenuta la somma di quelle che la precedono.
Per poter accelerare l'esecuzione della somma, è necessario che per ogni cifra si possa sapere, nel tempo più breve possibile, qual è il riporto generato fino allo stadio precedente.

Sapendo che l'espressione del riporto sia :
$C_{i+1} = A_iB_i + (A_i \text{ xor } B_i)C_i$ dove A e B sono i bit da sommare;
si può rappresentare come:
$c_{i+1} = G_i + P_i c_i$ dove $\mathbf{G_i = A_iB_i}$ e $\mathbf{P_i = A_i \text{ xor } B_i}$
Le espressioni Gi e Pi sono dette funzioni di generazione e propagazione per lo stadio i.

Qui un singolo Full-Adder sarà diverso da quello da me utilizzato, poiché non calcola il riporto e lo propaga bensì calcola le espressioni G_i e P_i insieme al risultato s_i $((A_i \text{ xor } B_i) \text{ xor } C_i)$



Si consideri un ***addizionatore a 4 bit*** *con anticipo del riporto.*
I riporti sono realizzati come segue:

$$\boldsymbol{C_{i+1} = G_i + P_iG_{i-1} + P_iP_{i-1}C_{i-1}}$$

I carry dell'addizionatore:

$C_1 = G_0 + P_0C_0$
$C_2 = G_1 + P_1G_0 + P_1P_0C_0$
$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0.$

Il blocco della logica di anticipo del riporto fornisce nuove funzioni di uscita de finite da G^I_k *e* P^I_k *, dove k=0 per il primo blocco da 4 bit, k =1 per il secondo da 4 bit e così via,Nel primo blocco:*

$P^I_k = P_3P_2P_1P_0;$
$G^I_k = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0.$

Per approfondimenti e schemi logici sul circuito consultare la pagina web https://www.electricaltechnology.org/2018/04/ripple-carry-and-carry-look-ahead-adder.html .

Addizionatore algebrico a *n* bit

Si definisce addizione algebrica una successione di addizioni e sottrazioni di numeri relativi, il risultato si dice somma algebrica.

L'idea è modificare il circuito addizionatore/full-adder a n bit per renderlo un addizionatore algebrico.

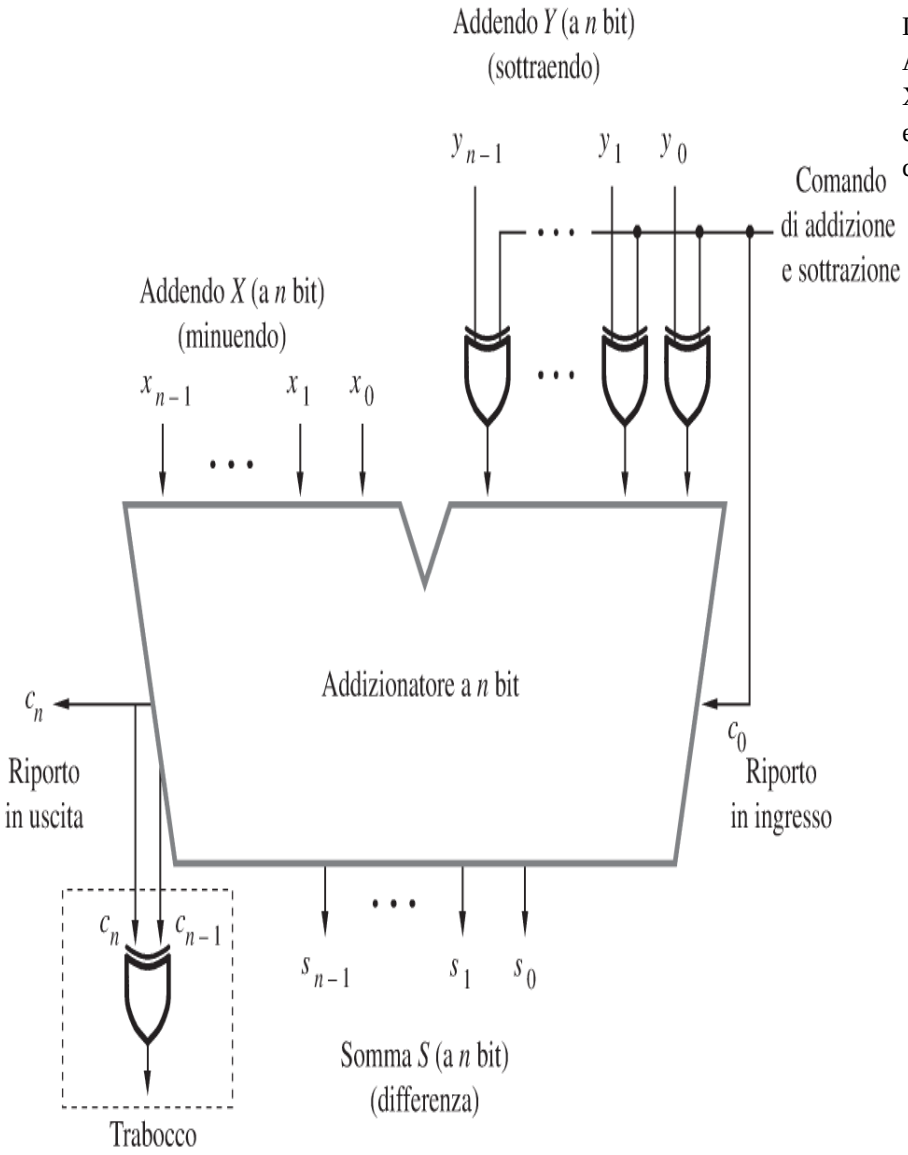
Per svolgere ciò bisogna approfondire il concetto di complemento a 2:

Il complemento a due, o complemento alla base, è il metodo più diffuso per la rappresentazione dei numeri con segno in informatica. L'espressione complemento a due viene spesso usata impropriamente per indicare l'operazione di negazione (cambiamento di segno).

La sua enorme diffusione è data dal fatto che i circuiti di addizione e sottrazione non devono esaminare il segno di un numero rappresentato con questo sistema per determinare quale delle due operazioni sia necessaria, permettendo tecnologie più semplici e con maggiore precisione; si utilizza un solo circuito, il sommatore, sia per l'addizione che per la sottrazione.

Per rappresentare l'opposto di un numero binario in complemento si negano i singoli bit: si applica cioè l'operazione logica NOT e si i aggiunge infine 1 al valore del numero trovato con questa operazione.

Detto ciò ecco uno schema dell'addizionatore algebrico:



Dove con la linea di comando Addizione/Sottrazione è collegata a uno XOR per ogni bit del secondo operando, esso definirà il lavoro che svolgerà il circuito:

- **con il valore logico 0** si otterrà la **somma**, poiché non effettuerà il complemento a 2 del numero binario. (su ogni bit dell'operando si effettua lo XOR con il valore 0 e di conseguenza avrà come uscita il bit dell'operando.)
- **con il valore logico 1** si otterrà il risultato della **sottrazione** dei due numeri. Avendo come segnale 1 si effettuerà il complemento a 2 tramite gli XOR collegati a ogni bit del sottraendo. Dopodiché si esegue la normale somma tra il minuendo e il sottraendo e l'incremento dal riporto in ingresso.

Circuito moltiplicatore sequenziale a *n* bit:

Si può creare un moltiplicatore di due numeri in circuito sequenziale che usa solo un addizionatore a *n* bit.

Per creare ciò si ha bisogno di:

- *1 full-adder a n bit ;*
- *1 Multiplatore:lo denominiamo MUX;*
- *3 registri : li denominiamo A , Q (a scorrimento) e M.*
- 1 flip-flop: lo denominiamo C.

I passi da eseguire sono:

1. Inizializza a zero un registro A
2. Inizializza a zero il flip-flop C per il riporto
3. Salva nei registri Q ed M moltiplicatore e moltiplicando
4. Se il bit meno significativo di Q vale 1

   •Somma A ed M

   •Memorizza il risultato in A
5. Scorrimento a destra del registro [C; A; Q] di una posizione
6. Ripeti dal punto 4 per n volte
7. Preleva il risultato della moltiplicazione dai registri [A; Q]

Descrizione Approfondita

La descrizione che segue è riprodotta dal testo citato, Capitolo 9, Paragrafo 9.3.2

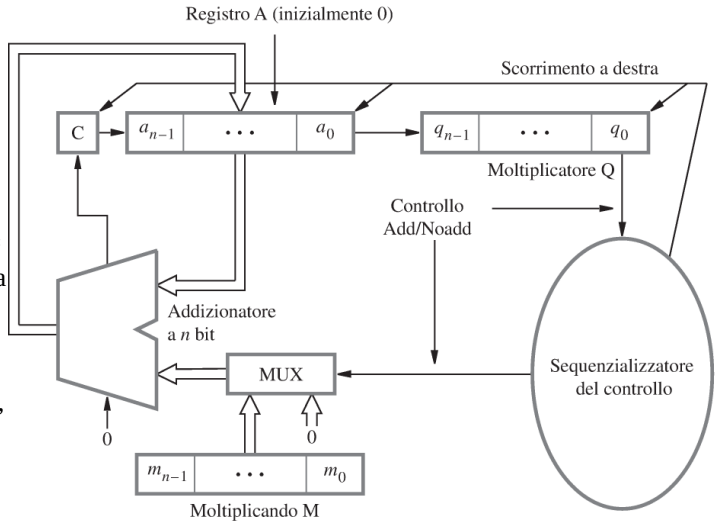
Questo circuito effettua la moltiplicazione usando n volte un singolo addizionatore a n bit.

I registri A e Q sono registri a scorrimento,concatenati come mostrato in figura.

Insieme , essi contengono il prodotto parziale PPi mentre il bit q_i del moltiplicatore genera il segnale Add/Noadd. Questo segnale fa sì che il multiplatore MUX selezioni 0 quando q_i = 0 , o selezioni il moltiplicando M quando q_i=1 , da sommare a PPi per generare PPi+1.

Il prodotto è calcolato in *n* cicli. Il prodotto parziale cresce in lunghezza di un bit per ciclo a partire dal vettore iniziale,PP0, di *n* zeri nel registro A.

Il riporto in uscita dall'addizionatore è memorizzato nel flip-flop C , mostrato in figura.



All'inizio il moltiplicatore è caricato nel registro Q, il moltiplicando nel registro M, C e A sono azzerati. Alla fine di ogni ciclo ,C ,A ,Q scorrono a destra di una posizione per permettere al prodotto parziale di crescere man mano che il moltiplicatore scorre fuori dal registro Q.

A causa di questo scorrimento , il bit q_i del moltiplicatore appare nella posizione meno significativa di Q per generare il segnale Add/Noadd al momento giusto ,iniziando con q_0 durante il primo ciclo, q_1 durante il secondo ciclo e così via.

Dopo che sono stati usati , i bit del moltiplicatore vengono scartati dall'operazione di scorrimento a destra . Si noti che il riporto in uscita dall'addizionatore è il bit più a sinistra di PP(i+1) , e che deve essere tenuto nel flip-flop C per scorrere a destra assieme i contenuti A e Q. Dopo *n* cicli , la metà di ordine alto del prodotto si trova nel registro A e la metà di ordine basso è nel registro Q.

La figura mostra come la sistemazione dell'hardware realizzerebbe la moltiplicazione.

Esempio: