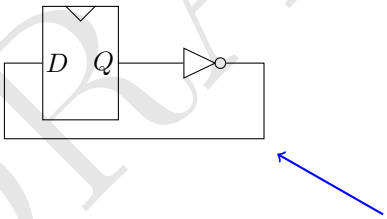
Logica e Reti Logiche
(Episodio 17: Adder/Subtractor ad anticipo di riporto)

Francesco Pasquale

9 gennaio 2025

Nello studio dei circuiti finora abbiamo sempre assunto che la propagazione del segnale lungo i fili sia pressoché "istantanea". Tuttavia, sappiamo dalla fisica che nessun segnale può propagarsi più velocemente della luce. Per quanto la velocità della luce sia enorme[1] rispetto alle velocità con cui siamo abituati a confrontarci quotidianamente, anche le frequenze di *clock* dei circuiti sono enormi (3GHz o anche più).

Esercizio 1. Si consideri il semplice circuito sequenziale qui sotto. Supponendo che il segnale si propaghi da Q a D alla velocità della luce (circa $3 \cdot 10^5$ km/s) e che il *clock* del FlipFlop sia a $3GHz$ (passa da 0 a 1 circa $3 \cdot 10^9$ volte al secondo), quanto può essere lungo, al massimo, il cavo indicato dalla freccia affiché il FlipFlop lavori correttamente?



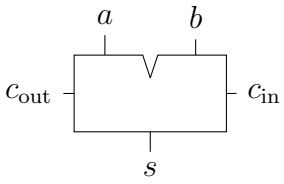
DRAFT

1 Addizionatore/Sottrattore a propagazione di riporto

Quando abbiamo iniziato a parlare di circuiti abbiamo visto come sia semplice, usando solo porte logiche elementari, costruire un circuito, il FullAdder (vedi Fig. 1), che faccia la somma di tre bit (a, b e c_{in}) restituendoci in output un bit che indica la somma e uno che indica il riporto (s e c_{out}, rispettivamente).

Esercizio 2. Ricostruire un FullAdder usando solo porte logiche elementari.

[1] Circa 300mila km al secondo



a	b	c_{in}	s	c_{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

Figura 1: Simbolo e tabella di verità di un FullAdder

Mettendo in serie k FullAdder, con $k \in \mathbb{N}$, otteniamo quindi un circuito che fa la somma in binario di numeri a k bit (vedi Fig. 2 per un *Adder* a 4 bit).

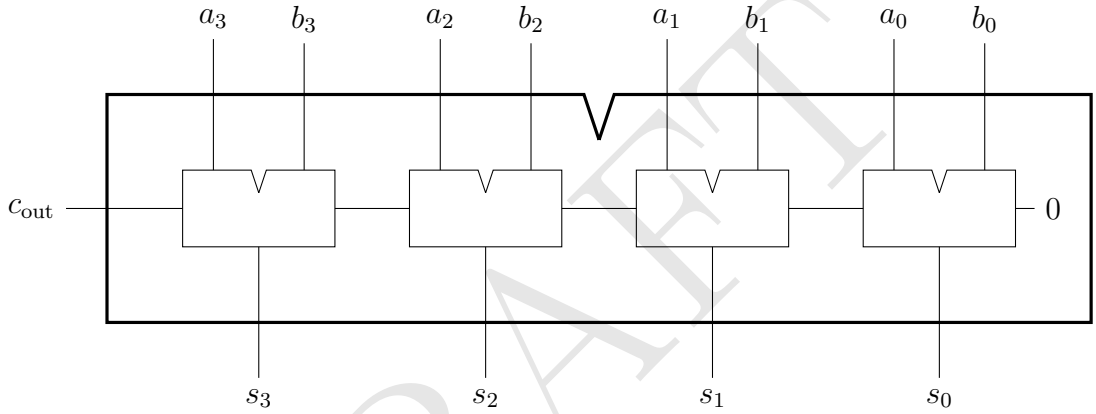


Figura 2: Un sommatore a 4 bit

Se si interpreta il bit c_{out} come il quinto bit della somma, allora il circuito in Figura 2 calcola correttamente la somma di qualunque coppia di numeri in binario a quattro bit $(a_3, a_2, a_1, a_0), (b_3, b_2, b_1, b_0)$ restituendo di fatto un numero a cinque bit $(c_{\text{out}}, s_3, s_2, s_1, s_0)$.

Interpretando le sequenze di bit come numeri in *complemento a due*, con una piccola modifica al circuito possiamo fare in modo che lo stesso circuito calcoli sia la somma che la sottrazione fra numeri in complemento a due a k bit, a seconda che il bit c_{in} sia 0 oppure 1 (vedi Fig. 2 per un *Adder/Subtractor* a 4 bit).

DRAFT

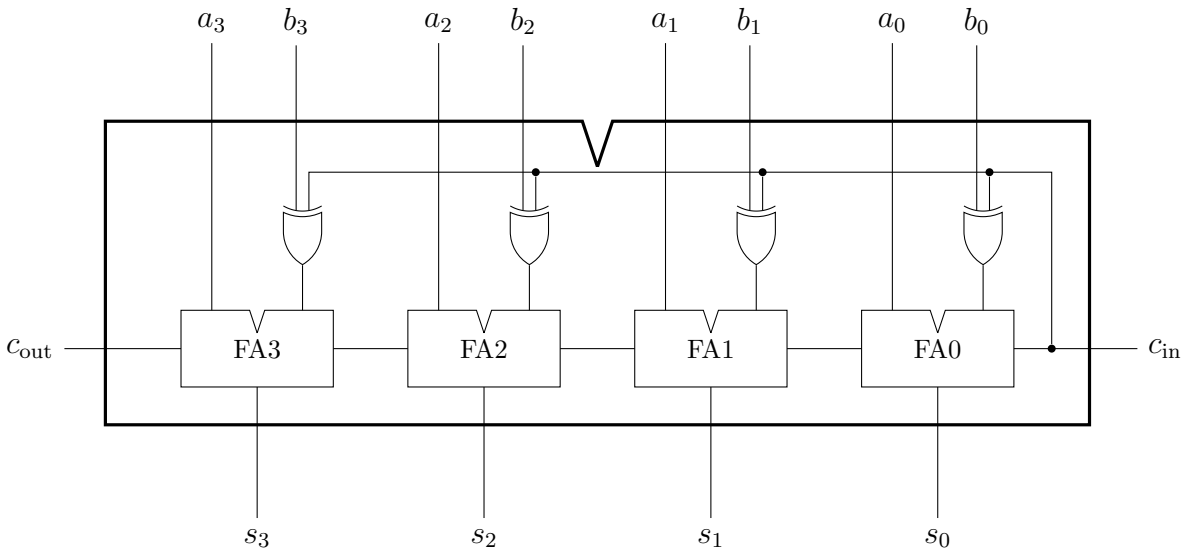


Figura 3: Un sommatore/sottrattore a 4 bit

I circuiti *Adder* e *Adder/Subtractor* costruiti così si dicono *a propagazione di riporto*. Osservate infatti, per esempio, che il FullAdder FA3 in Figura 3, per calcolare s_3 deve ricevere il riporto in uscita da FA2 che, per calcolarlo, a sua volta deve ricevere il riporto in uscita da FA1, che deve ricevere il riporto in uscita da FA0.

Se la propagazione del segnale fosse istantanea non ci sarebbe nessun problema, ma siccome i segnali non possono viaggiare più veloce della luce, questo significa che in un Adder/Subtractor a 64 bit il segnale proveniente da c_{in} impiegherebbe, per arrivare all'ultimo FullAdder, il doppio del tempo di quanto ne impiegherebbe in uno a 32 bit. Se il circuito *Adder/Subtractor* è un componente di un circuito sequenziale, passare da 32 a 64 bit quindi potrebbe costringerci a dover dimezzare la velocità del *clock*.

2 Addizionatore/Sottrattore ad anticipo di riporto

Si consideri un addizionatore/sottrattore a 16 bit e immaginiamolo diviso in 4 addizionatori a 4 bit

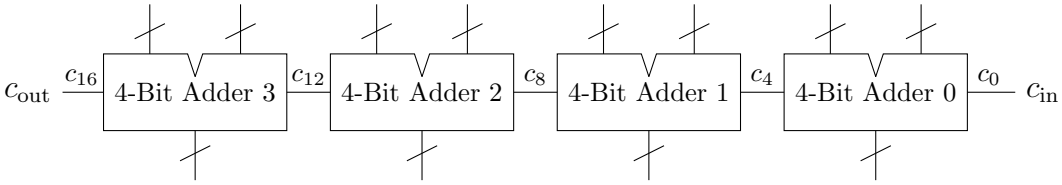


Figura 4: Un sommatore/sottrattore a 16 bit

Chiamiamo c_i il riporto in ingresso dell'i-esimo FullAdder (che è uguale al riporto in uscita dell'$(i-1)$-esimo).

Per migliorare l'efficienza del circuito, in termini di velocità di propagazione del segnale da c_0 fino a c_{16}, possiamo provare a fare in modo di *precomputare* in ogni blocco di quattro

DRAFT

bit qualcosa che ci consenta di non dover far passare i bit dei vari riporti in ognuno dei 16 FullAdder. Vediamo come fare.

Consideriamo il generico FullAdder i-esimo, per $i = 0, \ldots, 15$ e osserviamo che:

- Se $a_i b_i = 1$ allora il FullAdder *genera* il bit di riporto $c_{i+1} = 1$ indipendentemente se il suo riporto in entrata c_i sia 0 o 1;
- Se $a_i \oplus b_i = 1$ allora il FullAdder *propaga* il bit di riporto: $c_{i+1} = c_i$.

Quindi possiamo scrivere $c_{i+1} = a_i b_i + (a_i \oplus b_i) c_i$.

Esercizio 3. Osservare che $a_i b_i + (a_i \oplus b_i) c_i$ è equivalente a $a_i b_i + (a_i + b_i) c_i$

Se per comodità chiamiamo $g_i = a_i b_i$ (*generazione* del riporto) e $p_i = a_i + b_i$ (*propagazione* del riporto), dall'esercizio precedente abbiamo che deve valere la relazione di ricorrenza

$$c_{i+1} = g_i + p_i c_i \tag{1}$$

Applicando la relazione precedente con $i = 7$ abbiamo, per esempio, che il riporto in uscita dal secondo *adder* a 4-bit è $c_8 = g_7 + p_t c_7$, e applicandola ricorsivamente con $i = 6$ abbiamo che

$$\begin{aligned} c_8 &= g_7 + p_7 c_7 \\ &= g_7 + p_7(g_6 + p_6 c_6) = g_7 + p_7 g_6 + p_7 p_6 c_6 \end{aligned}$$

Esercizio 4. Istanziando l'equazione di ricorrenza (1) con $i = 7, 6, 5, 4$ verificare che si può scrivere il riporto c_8 in uscita dal secondo adder a 4-bit in funzione del riporto c_4 in entrata nello stesso *adder* in questo modo

$$\begin{aligned} c_8 &= g_7 + p_7(g_6 + p_6(g_5 + p_5(g_4 + p_4 c_4))) \\ &= g_7 + p_7(g_6 + p_6(g_5 + p_5 g_4)) + p_7 p_6 p_5 p_4 \, c_4 \\ &= A + B \, c_4 \end{aligned}$$

Dove nell'ultima uguaglianza ho chiamato

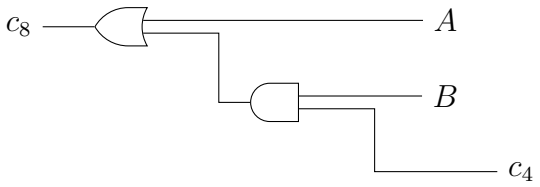
$$\begin{aligned} A &= g_7 + p_7(g_6 + p_6(g_5 + p_5 g_4)) \\ B &= p_7 p_6 p_5 p_4 \end{aligned}$$

Esercizio 5. Osservare che i due bit A e B dell'esercizio precedente sono funzioni di (g_7, g_6, g_5, g_4) e (p_7, p_6, p_5, p_4) che a loro volta sono funzioni di (a_7, a_6, a_5, a_4) e di (b_7, b_6, b_5, b_4). Quindi A e B possono essere calcolate nel secondo Adder a 4-bit senza dover aspettare l'arrivo di c_4.

Esercizio 6. Progettare un circuito che prenda in input (g_7, g_6, g_5, g_4) e (p_7, p_6, p_5, p_4) e restituisca in output A e B.

Una volta *precomputati* A e B, nel momento in cui arriva il riporto in entrata c_4 l'adder a 4-bit può calcolare il riporto c_8 in uscita usando solo una porta AND e una porta OR

DRAFT



In questo modo, quando arriva il riporto in entrata c_4, che è necessario ai quattro FullAdder contenuti nell'Adder 1 per calcolare (s_7, s_6, s_5, s_4), l'Adder 1 può *anticipare* (ossia, calcolare tramite un circuito più breve, che quindi richiederà meno tempo) il riporto c_8 all'Adder 2, che a sua volta lo userà per calcolare $(s_{11}, s_{10}, s_9, s_8)$ e per *anticipare* c_{12} all'Adder 3.

Esercizio 7. Descrivere in un *Hardware Description Language* un circuito che prenda in input (a_7, a_6, a_5, a_4) e (b_7, b_6, b_5, b_4) e restituisca in output A e B.

La procedura vista in questa sezione per costruire un adder a 16 bit *precomputando* in ogni blocco a 4 bit ciò che serve per propagare il riporto all'adder a quattro bit successivo si può iterare: per esempio, per costruire un *Adder* a 64 bit si può dividerlo in blocchi da 16 bit e precomputare ciò che serve per fare in modo che il riporto si propaghi da un blocco al successivo senza dover passare attraverso tutti i FullAdder. Un circuito *Adder/Subtractor* costruito in questo modo si chiama *ad anticipo di riporto*.

DRAFT