OverFlow

Il modo più semplice per verificare la condizione di overflow durante una operazione di somma binaria a precisione finita con segno consiste nel verificare se il segno del risultato è coerente con il segno degli operandi. Se il segno degli operandi è diverso allora non si verifica mai l'overflow. Se il segno dei due operandi è lo stesso, allora anche il segno del risultato deve essere uguale al segno degli operandi, altrimenti si è verificato un overflow. Definiamo la tabella di verità definita da questa funzione:

A=Sa	B=Sb	C=Sr	Y=overflow
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

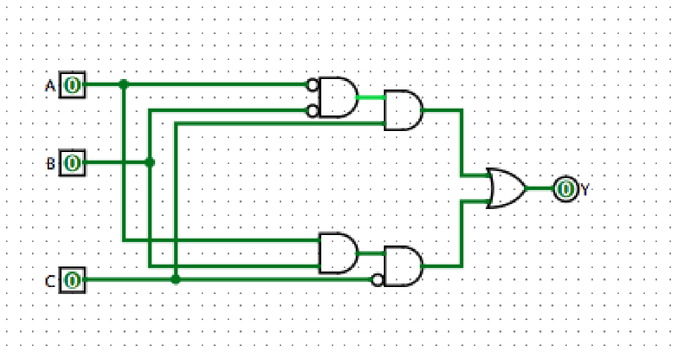
Gli ingressi sono 3, segno dei due operandi (Sa e Sb) e segno del risultato (Sr), quindi ho $2^3=8$ righe contenenti tutte le possibli combinazioni degli ingressi. La forma canonica più conveniente per sintetizzare una funzione è la SOP in quanto sono presenti due **1** contro i sei **0** che definiscono la POS. Calcoliamo la SOP:

SOP = ~A~BC + AB~C

Calcoliamo per completezza anche la POS:

POS = (A+B+C)(A+~B+C)(A+~B+~C)(~A+B+C)(~A+B+~C)(~A+~B+~C)

Il costo della SOP è pari al numero di porte a due ingressi usate nella formula: 2 AND per i 2 min-termini più 1 per l'operazione di OR, totale 5. Il cammino critico è 2 per attraversare i min-termini più 1 per eseguire la OR, quindi 3. Sintetizziamo il circuito della SOP:



Per i fini del corso, eventuali raggruppamenti di termini comuni all'interno della SOP non vengono considerati nel calcolo del costo.

Il costo si può ricavare anche dal circuito usando solo porte a 2 ingressi e valutando numero di porte usate; il cammino critico si ottiene individuando il cammino più lungo tra ingressi ed uscita, in questo caso il cammino da A (o B indifferentemente) verso l'uscita.

Il cammino critico può essere calcolato anche cercando una parentesizzazione ottimale della funzione data, cioè raggruppando i termini secondo la giusta precedenza in modo da minimizzare il numero di parentesi necessarie, in questo caso 3 livelli di parentesi:

$$\mathrm{SOP} = \{[(\sim A \sim B)C] + [(AB)\sim C]\}$$

Calcoliamo la mappa di Karnaugh:

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="4">BC</td></tr>
<tr><td>00</td><td>01</td><td>11</td><td>10</td></tr>
<tr><td rowspan="2">A</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
</table>

Per ottenere la mappa dividiamo le tre variabili in ingresso in due gruppi, in questo caso A e BC, quindi mettiamo sull'intestazione delle righe e delle colonne tutte le possibili combinazioni dei gruppi di variabili avendo cura che due configurazioni adiacenti differiscano di un solo bit (in pratica considerando un ordinamento naturale, la configurazione 11 si scambia con la configurazione 10). Raggruppiamo poi gli 1 della mappa in modo che gli 1 del raggruppamento coincidano con un implicante della funzione. Poiché non è possibile raggruppare in nessun modo i due 1 presenti nella mappa si conclude che non esiste una forma più compatta della SOP usando solo AND,OR e NOT. La disposizione vagamente a scacchiera della mappa di Karnaugh e la disposizione a specchio nella tabella di verità suggeriscono che può esistere (ma in generale non è certo) una forma più compatta che usi anche porte XOR.
Riconsideriamo la definizione data all'inizio di overflow. Quando i segni degli operandi sono discordi non si verifica overflow. Si consideri la seguente funzione:

$$F1(A,B,C) = \sim(A \text{ XOR } B)$$

La funzione F1 vale 1 quando i segni dei due operandi sono uguali, condizione necessaria (ma non sufficiente) affinché si verifichi l'overflow. La seconda condizione necessaria (ma non sufficiente) affinché si verifichi l'overflow è che il segno del risultato differisca da quello di uno dei due operatori, esempio A. Si consideri la seguente funzione:

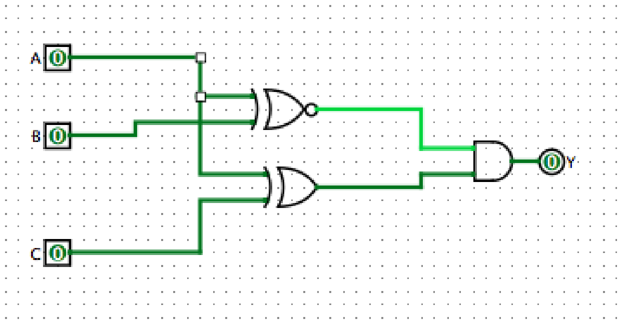
$$F2(A,B,C) = (A \text{ XOR } C)$$

La funzione F2 vale 1 quando il segno di A differisce dal segno del risultato.
Entrambe le condizioni definite da F1 e F2 sono condizioni necessarie; perchè si verifichi l'overflow devono valere entrambe (condizione necessaria e sufficiente). Se una sola di queste condizioni non si verifica, allora non si verifica nemmeno l'overflow. In altre parole se una delle due funzioni vale 0 allora anche la funzione overflow vale 0. Se al contrario entrambe valgono 1 allora la la funzione overflow vale 1. In termini algebrici si può scrivere:

$$\text{overflow}(A,B,C) = F1(A,B,C) \text{ AND } F2(A,B,C) = \sim(A \text{ XOR } B)\ (A \text{ XOR } C)$$

La funzione data ha costo 3 e cammino critico 2.

Sintetizziamo il circuito:



Per verificare che la funzione sia effettivamente quella cercata ricalcoliamo la tabella di verità partendo dalla formula trovata:

A=Sa	B=Sb	C=Sr	A xor B	~(A XOR B)	A XOR C	F1. F2
0	0	0	0	1	0	0
0	0	1	0	1	1	1
0	1	0	1	0	0	0
0	1	1	1	0	1	0
1	0	0	1	0	1	0
1	0	1	1	0	0	0
1	1	0	0	1	1	1
1	1	1	0	1	0	0

La tabella ricavata coincide con quella calcolata all'inizio. Proviamo a dimostrare l'uguaglianza attraverso le proprietà dell'algebra booleana:

overflow(A,B,C) = ~(A XOR B).(A XOR C)	
= ~(~AB+A~B).(~AC+A~C)	;sostituisco alla XOR il suo sviluppo SOP
= [~(~AB).~(A~B)].(~AC+A~C)	;De Morgan più volte
= [((~~A)+~B).(~A+(~~B)].(~AC+A~C)	;
= [(A+~B).(~A+B)].(~AC+A~C)	; ~(~X)=X

da notare che la formula all'interno delle parentesi quadre è la forma POS della XNOR

= [(A+~B).~A+(A+~B).B)].(~AC+A~C)	; Z(X+Y)=ZX+ZY più volte
= [A~A+~B~A+AB +~BB].(~AC+A~C)	;
= [0+~B~A+AB +0].(~AC+A~C)	;X~X=0
= [~B~A+AB].(~AC+A~C)	;X+0=X

da notare che la formula all'interno delle parentesi quadre è la forma SOP della XNOR

=[~B~A+AB].~AC+[~B~A+AB](A~C)	; Z(X+Y)=ZX+ZY più volte
=~B~A~AC+AB~AC+~B~AA~C+ABA~C	;
=~A~BC+0+0+AB~C	; XY=YX, X~X=0, X0=0
=~A~BC+AB~C	; X+0=X

che è la forma SOP calcolata all'inizio.