Aritmetiche Reti Logiche

Sia A=-67 e B = 61,

a. codificare i due numeri in complemento a 2 (CA2)
b. calcolare A*B usando l'algoritmo di Booth Radix 4

(guarda sempre il numero di bit di ogni numero che scriviamo)

CODIFICA IN CA2

1. riscrivere il numero in binario normale, per fare prima scrivitelo per prima come somma di potenze di due, ignorando per un momento il segno:
   a. A= 67= 64 + 3 = 1000011
   b. B= 61 = 32+16 + 8 + 4 + 1 = 111101
2. ora consideriamo anche i segni: facciamo prima il caso più "difficile", i numeri negativi.Questi vanno invertiti (complemento a 1) e poi va sommato 1. Per finire aggiungiamo un bit a sinistra per il segno. Poi consideriamo i più facili, quelli positivi, che rimangono uguali tranne per il fatto che viene aggiunto un bit a sinistra per il segno(0 per i positivi)
   a. 67 = 1000011 → CA1: 0111100→ sommiamo 1(CA2) : 0111101 →aggiungo un bit a sinistra per il segno. Qui essendo - aggiungo 1: 10111101
   b. 61 = 111101 →CA2 : 0111101

PRODOTTO IN BOOTH RADIX 4

quando si fa il prodotto di due numeri con l'algoritmo di Booth, o 4 o 2, la logica è sempre la stessa: se devo fare A*B, vado a moltiplicare A per la codifica in Radix-N di B.

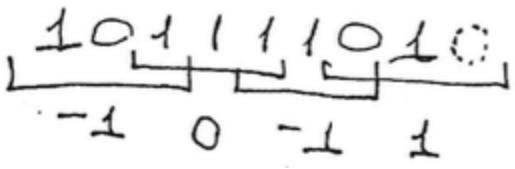
Quindi la prima cosa da fare è codificare B, poi vediamo come si moltiplica.In entrambi va aggiunto un ghost zero a destra, quindi si aggiunge uno zero in più

In radix 4 dobbiamo formare raggruppamenti da 3, in Radix2 gruppi da 2.

Supponiamo di voler fare A*B (61* (-67)).Devo per prima cosa codificare -67 in rx4:

a. Partendo dalla codifica in CA2, aggiungo ghost zero a destra: 101111010

b. Formo raggruppamenti da 3. Se non dovessi farcela a fare gruppi da 3 e un bit rimane fuori, estendo il segno(qui non serviva). A questo punto fatti i gruppi devo assegnargli il valore della codifica, in Radix 4, partendo dal bit più a sinistra, un gruppetto vale : -2*(VAL BIT PIU' A SX)+ (VALORE BIT CENTRALE)+ (VALORE BIT A DESTRA). Ad esempio 101 vale -2+0+1, oppure 111 vale -2+1+1=0 ecc...In radix 2 invece banalmente i gruppi sono da 2 e il primo bit a sx vale -1 e il secondo 1, ad esempio 11= -1+1=0; 10=-1+0; 01=0*(-1)+1...



c. A questo punto posso quasi iniziare a moltiplicare, ma consiglio di fare questo passaggio: quando trovo -1 al moltiplicatore devo riscrivere in colonna -A, se trovo 1 A, se trovo -2 devo shiftare -A di un posto, quindi parto dal bit più significativo e sposto tutto quanto a sinistra(esempio 1001→0010, il bit a sinistra in più lo scarto in radix). Quindi conviene scrivermi -A, A , A<<1.

d. adesso inizio, ricordandomi che in Radix 4 devo lasciare 2 posti a dx anzichè uno solo:

Floating Point, Eccesso 127 IEEE754

la notazione in eccesso 127 è un modo per rappresentare i numeri in floating point. Anzichè usare la virgola fissa quindi uso la virgola mobile. In virgola fissa il problema è che sono limitato come valori che posso rappresentare, invece in virgola mobile no. Per lo standard IEEE754 i numeri in float si rappresentano in questo modo

FLOAT = SEGNO(1 bit) ESPONENTE(8 bit) MANTISSA(23 bit)

In particolare l'esponente segnato li è "silenziosamente" sommato a +127. Dico silenziosamente perchè quello che metto in quegli 8 bit è considerato poi come un numero da aggiungere(o sottrarre se negativo) a 127. Quindi

> ***ESPONENTE REALE = EXP + 127***

1 BIT SEGNO	8 BIT ESPONENTE	23 BIT MANTISSA

PASSAGGIO IEEE754 E127 → DECIMALE

quindi avendo un numero in codifica floating point a singola precisione IEEE754 eccesso 127 possiamo trasformarlo in decimale cosi:

$$valore = (-1)^{SEGNO} \times 2^{ESPONENTE-127} \times (1 + MANTISSA)$$

dove segno è banalmente 0 se è +, 1 se -. Esponente è quello che c'è negli 8 bit di esponente convertito in decimale e idem per mantissa.

PASSAGGIO DECIMALE → IEEE754 E127

supponiamo di dover trasformare 86,625

1. codifichiamo 86 in binario normale: 86 (anche se era negativo perchè tanto dopo lo mettiamo nel bit apposito per il segno il -) → 86 = 84+2 = 1010110
2. adesso prendiamo la parte dopo la virgola(quindi 625) e moltiplichiamola per 2 fino a che non otteniamo 0 a destra(nella colonna destra):

⬇️		625
⬇️	1	25
⬇️	0	5
⬇️	1	0

0,625*2 = 1,25 quindi riporto 1

0,25*2 =0,5 quindi riporto 0

0,5*2= 1,0 quindi riporto 1

mi fermo perchè a destra ho ottenuto 0

⬇️ | 0 | | scrivo 0 alla fine

il numero risultante sarà 1010 preso partendo sempre dall'alto (colonna sinistra).

Attenzione: può capitare che non si arrivi mai a 0 a destra, in quel caso basta prendere quei bit come periodici, ad esempio se vedo che sono in un loop e a sinistra si ripetono i bit 1101 per esempio, scrivo la prima parte non periodica normalmente, supponiamo che sia 010, poi aggiungo quella periodica: il numero sarà

$$010\overline{1101}$$

quindi sui 23 bit della mantissa scriviamo 010 a partire da sinistra(dal più significativo), poi ripetiamo per tutti i bit rimanenti la sequenza 1101.

3. quindi adesso riscrivo il numero completo(86.625):

   86.625 = 1010110,1010

4. portiamo sulla prima cifra la virgola, così da ottenere il nostro esponente da portare in notazione eccesso 127:

   1010110,1010 → 1,0101101010 → ci siamo spostati di 6 posti a sinistra

   adesso abbiamo ottenuto anche la **mantissa**, cioè ciò che c'è dopo la virgola.

5. adesso abbiamo quindi il nostro numero nella forma 1,0101101010 × 2^6.
   L'esponente sarà 127+6 = 133 = 128+4+1 = **10000101**

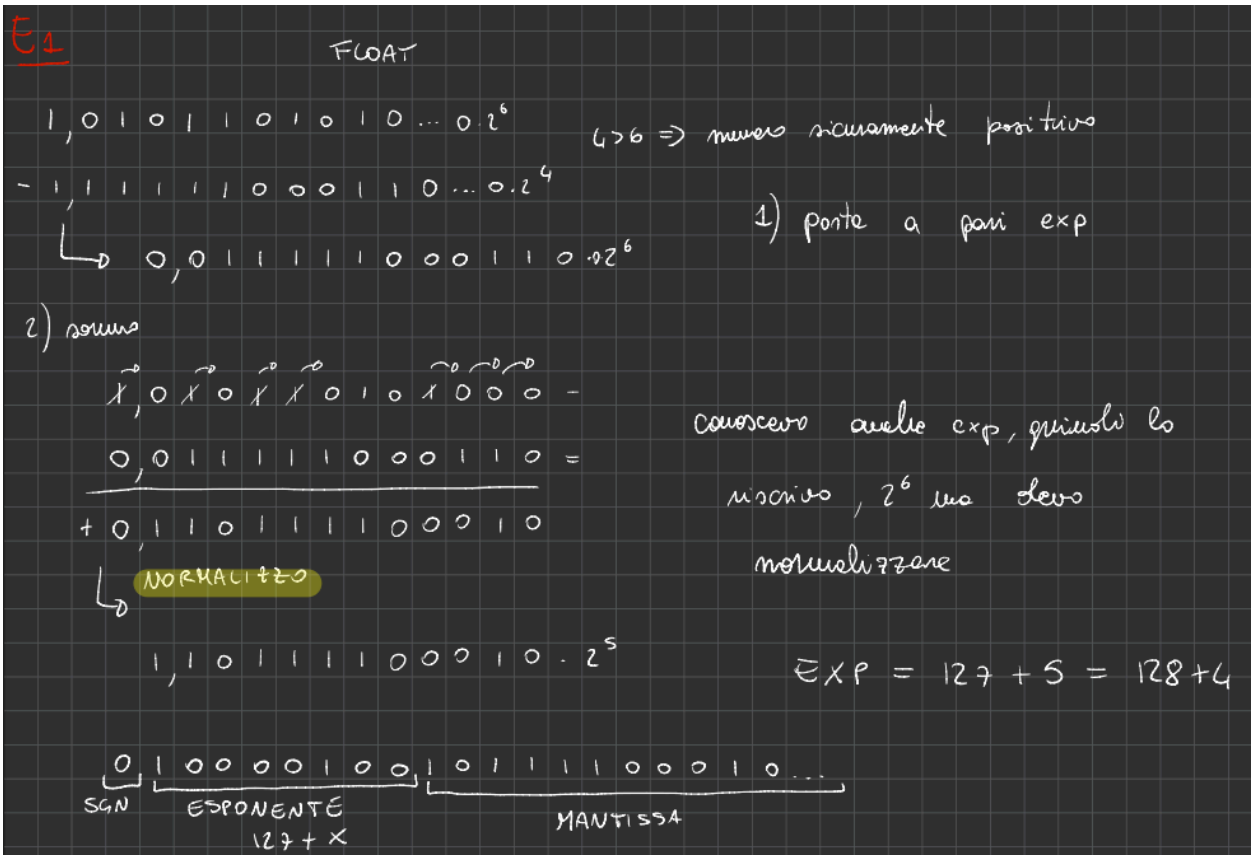
(ricorda che per scrivere in binario un numero, ad esempio 128, devi considerare che 128 è 2^7, ma siccome il primo bit in binario vale 2^0=1, serviranno sempre 7 bit + 1 =8 bit)

1 BIT SEGNO	8 BIT ESPONENTE	23 BIT MANTISSA
0	**10000101**	01011010100000000000000

ho esteso quindi la mantissa(ricorda che la mantissa si scrive da sinistra verso destra)

Metto 0 sul segno perchè sappiamo che il numero è positivo(86,625).

SOTTRAZIONE/ADDIZIONE FLOATING POINT A SINGOLA PRECISIONE



sottrazione e addizione si eseguono normalmente, ma alla fine il risultato va normalizzato, quindi si porta la virgola sul primo uno.RICORDA: prima di scrivere un numero in IEEE754 eccesso 127 devi sempre **normalizzare il numero**, quindi portare la virgola sul primo uno, in modo da avere 1.MANTISSA x 2^E dove E è l'esponente in eccesso 127.