# Manipolazione di numeri in rappresentazione binaria

Rappresentazione generica in Fixed Point:

Si supponga di dover rappresentare su di un BUS ad 8 bit dei numeri che rappresentino con la migliore precisione possibile dei dati di temperatura misurati all’interno di una cella frigorifera compresi tra -17 e 24 gradi. Come sfruttare al meglio i bit a disposizione?

1. Innanzitutto si nota che si devono rappresentare numeri sia positivi che negativi, per cui sembra idonea una rappresentazione in modalità “signed” ovvero uno degli otto bit verrà impiegato come bit di segno

2. Attraverso gli altri sette bit si possono rappresentare 127 numeri positivi e 128 numeri negativi oltre allo zero.

3. Per rappresentare il numero più alto in valore assoluto (24) si necessita quindi di almeno 5 bit per la parte intera.

4. Rimangono pertanto a disposizione ulteriori 2 bit per rappresentare la parte frazionaria



Numeri rappresentabli:

Secondo la rappresentazione appena stabilita:

1. si potranno rappresentare tutti i numeri compresi tra

(-32)10 : (100000,00)2

e

(31.75)10 : (011111,11)2

2. saranno rappresentabili 127 numeri positivi e 128 numeri negativi

3. lo zero sarà rappresentato con 000000,00

4. il più piccolo numero positivo, nonché step tra due numeri consecutivi sarà

(0,25)10 : (000000,01)2

5. il più piccolo numero negativo (In valore assoluto) rappresentabile

(-0,25)10 : (111111,11)2

Rappresentazione in Fixed Point di un numero specifico:

In questo contesto come andrebbe rappresentato ad esempio il numero -11,724?

0. Si inizi con analizzare la rappresentazione dell’omologo numero positivo: 11,724

1. Il numero 11 sui 5 bit della parte intera viene rappresentato come 01011 infatti:

Operando attraverso divisioni successive per 2 ed analizzando i resti si trova:



Dove “msb”e “lsb”: less significant bit : most significant bit.   
Ma poiché è previsto vi siano 5 bit assegnati alla parte intera (pù il bit di segno) il bit più significativo viene posto a zero.

2. La parte frazionaria si analizza per moltiplicazioni successive e memorizzando la parte intera.



E, come capita abbastanza comunemente, la procedura restituisce un valore con infinite cifre decimali (periodico) 0.101110…. che quindi andrà arrotondato al valore più prossimo composto di 2 cifre decimali. In questo caso 0.11, che rappresenta in realtà il valore 0.75, introducendo pertanto un errore di arrotondamento pari a 0.026.

Si noti peraltro che un eventuale troncamento a 2bit avrebbe rappresentato la parte frazionaria con 0.10 equivalente a 0.5 e quindi con un errore di approssimazione pari a 0.224.

Il numero completo 11,724 trova pertanto la sua migliore rappresentazione nel numero binario:

001011,11 = 11,75

Per ottenere ora la rappresentazione del numero negativo basta effettuare il complemento a 2 del numero binario appena ottenuto complementando ciascun bit e sommando 1 nella posizione meno significativa.

001011,11 + 0,01 = 110100,00 +0,01 = (110100,01)2 = (-11.75)10

casting:

Supponendo che il suddetto numero binario debba essere ora fatto confluire in un altro BUS dove però la rappresentazione sia sempre di tipo signed ma in formato (5,3), quale sarebbe la sua rappresentazione?

1. I due bit più significativi sono uguali (entrambi 1) quindi si può sopprimere il bit più significativo senza alterare il valore

2. il bit meno significativo può essere riempito di default con uno zero

110100,01 🡪 10100,010 = (-11.75)10

E supponendo la medesima operazione fosse fatta verso un BUS in formato (10,4) ?

1. i bit più significativi in eccedenza riprendono il bit di segno.

2. i bit meno significativi vengono posti a zero.

110100,01 🡪 1111110100,0100 = (-11.75)10

E supponendo la medesima operazione fosse fatta verso un BUS in formato (4,2) ?

1. I bit interi a disposizione non sono sufficienti per rappresentare il valore desiderato. Infatti i numeri rappresentabili con questa rappresentazione sono compresi tra -8 e 7,875, quindi è presente una situazione di overflow. Si può scegliere se rappresentare i valori fuori dal range con il “fondo scala” (nel nostro caso con 1000.000 = (-8)10 oppure con una opportuna segnalazione di errore su una line dedicata.

Tale errore di overflow può essere individuato in quanto i BIT della parte intera da sopprimere NON coincidono col bit più significativo residuo che verrebbe ad assumere, nella nuova rappresentazione, la funzione di bit di segno.:

110100,01

0100,01 = (4,25)10

Come riprova si noti che la parola

111100,01 = (-3,75)10

Potrebbe essere rappresentata senza errori anche utilizzando un bus che utilizzi solo 4 bit per la parte intera.