Travail préparatoire séance 1: opérations en virgule fixe

Les paragraphes intitulés 'exercices' doivent être rendus en début de séance, rédigés individuellement sous forme manuscrite (pas de traitement de texte, trop facile à recopier).
Dans le cas où vous auriez rencontré des problèmes pour répondre a telle ou telle question, l'enseignant consacrera éventuellement 15 minutes à répondre à vos questions.
N'essayez pas de recopier le résultat d'une autre personne sans comprendre: si l'enseignant a un doute, il vous demandera d'effectuer pendant la séance un exercice similaire, pour vérifier si vous avez compris, ou simplement recopié. La note qui vous sera attribuée sera modifiée en fonction de la réponse à ce nouvel exercice...

1. représentation des nombres entiers en binaire signé

Les nombres entiers sont codés sur les microprocesseurs en binaire signé, sur un certain nombre de bits (valeur binaire prenant soit la valeur 0, soit la valeur 1)

Le bit de poids fort, ou bit de signe, compte négativement, et les autres bits positivement.

1.1 Passage de la représentation binaire signée à la valeur de X (exemples sur 8 bits)

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	0	1

=> $X= 0.-2^7+1.2^6 + 0.2^5 + 1.2^4+ 0.2^3 + 1.2^2+ 0.2^1 + 1.2^0=64+16+4+1=85$

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1	0	1	0	1

=> $X=1.-2^7+1.2^6 + 0.2^5 + 1.2^4+ 0.2^3 +1.2^2+ 0.2^1 + 1.2^0=-128+64+16+4+1=-43$

1.2 Passage de la valeur de X à sa représentation en binaire signé

1.2.1 exemple 1 : x=1299, sur 12 bits signé

1299 >=1024 => 1299=1024 + [1299-1024] = 1.1024 + 275
275 >=256 => 275 = 1.256 + [275-256] = 1.256+19
19 >=16 => 19 = 1.16 + [19-16] = 1.16 + 3
3 >=2 => 3 = 1.2 + [3-2] = 1.2 + 1
1 >=1 => 1 = 1.1 + [1-1] = 1.1 + 0

=> $X=1024 +256 + 16 +2 +1$
=> $X=1.[2^{10}] +1.[2^8] +1.[2^4] +1.[2^1] +1.[2^0]$

-2048 -2^{11}	1024 2^{10}	512 2^9	256 2^8	128 2^7	64 2^6	32 2^5	16 2^4	8 2^3	4 2^2	2 2^1	1 2^0
0	1	0	1	0	0	0	1	0	0	1	1

1.2.2 *exemple 2, X=-1031, sur 12 bits signé*

X=-1031<0 , le bit de signe représente -2^{11} (codage sur 12 bits signé)

=> $X=-2^{11}+[2^{11}+X]$, avec $[2^{11}+X]=[2048+X]=1017$

1017 >=512 => 1017=1.512 + [1017-512] = 1.512+505

505 >=256 => 505 = 1.256 + [505-256] = 1.256+249

249 >=128 => 249 = 1.128 + [249-128] = 1.128 +121

121 >=64 => 121 = 1.64 + [121-64] = 1.64 + 57

57 >=32 => 57 = 1.32 + [57-32] = 1.32 + 25

25 >=16 => 25 = 1.16 + [25-16] = 1.16 + 9

9 >=8 => 9 = 1.8 + [9-8] = 1.8 + 1

1 >=1 => 1 = 1.1 + [1-1] = 1.1 + 0

=> X=-2048 +512 +256 +128 + 64 +32 + 16 +8 +1

=> $X=1.[-2^{11}]+1.[2^9]+1.[2^8]+1.[2^7]+1.[2^6]+1.[2^5]+1.[2^4]+1.[2^3]+1.[2^0]$

-2048	1024	512	256	128	64	32	16	8	4	2	1
-2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	1	1	1	1	1	1	0	0	1

1.3 Exercices

-donner la valeur des nombres dont la représentation en binaire signé est donnée ci-après.

Représentation de X1 en 10 bits signé=> X1=?

1	0	1	1	0	1	1	0	0	1

Représentation de X2 en 13 bits signé=> X2=?

0	0	1	1	0	1	1	0	0	1	0	0	1

-donner la représentation en binaire signé des nombres ci-après, sur le nombre de bits demandé

Représentation de X3=2537 en 15 bits signé ?

Représentation de X4=-702 en 14 bits signé ?

-exprimer en fonction de N le plus grand nombre positif représentable sur N bits signés

-exprimer en fonction de N le plus grand nombre négatif représentable sur N bits signés

2 Opérations élémentaires sur les nombres entiers en binaire signé

2.1 négation d'un nombre (complément à 2)

Pour passer de la représentation du nombre *X* a la représentation de *-X* en binaire signé, on complémente tous les bits, et on ajoute 1.

	-16 -2^4	8 2^3	4 2^2	2 2^1	1 2^0
par exemple soit X= 10 =8+ 2 , codé sur 5 bits signé	0	1	0	1	0
en complémentant tous les bits , on obtient	1	0	1	0	1
si on ajoute 1 => +	0	0	0	0	1
on obtient alors =	1	0	1	1	0

qui est la représentation de Y=-16 +4+2 = -10 =-X

2.2 **addition de 2 nombres**

l'addition fonctionne comme l'addition de 2 nombres codés en binaire: dès lors que le résultat reste codable sur *N* bits, on obtient un résultat correct. Par contre si le résultat dépasse l'échelle de codage, on obtient un résultat dont le signe est faussé, mais qui reste exact modulo 2^N (overflow)
exemple d'addition avec overflow : 10+8, codé sur 5 bits signé.

	-16 -2^4	8 2^3	4 2^2	2 2^1	1 2^0
10	0	1	0	1	0
+ 8	0	1	0	0	0
= -16+2=-14	1	0	0	1	0

$-14 + 2^5 = -14+32 = 18$ (le résultat -14 est exact, modulo $2^N = 32$)

2.3 **multiplication**

La multiplication fonctionne comme la multiplication entière, en faisant attention qu'en multipliant 2 nombres codés sur *N* bits, le résultat sera codé sur 2 *N* bits
exemple : 13.11 , 13 et 11 sont codés sur 5 bits, le résultat est codé sur 10 bits

		-16 -2^4	8 2^3	4 2^2	2 2^1	1 2^0
13		0	1	1	0	1
X 11	X	0	1	0	1	1
= 13. 2^0		0	1	1	0	1
+ 13. 2^1		0	1	1	0	1
+ 13. 2^3		0	1	1	0	1
= 143		0	0	1	0	0	0	1	1	1	1
		-512	256	128	64	32	16	8	4	2	1

2.4 **multiplication ou division par une puissance entière de 2**

2.4.1 *multiplication par 2^L, avec L positif*

pour *L* positif, la multiplication par une puissance entière 2^L se ramène à un décalage à gauche de *L* bits, avec un report de zéros dans les bits de poids faible...

- cette opération peut générer un overflow (dépassement d'échelle), si le résultat théorique dépasse l'échelle de codage...
- par contre elle n'entraîne aucune perte de précision

exemple : $[-7].2^3$ codé en 8 bits signé, décalage de 3 bits à gauche

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	1	1	1	1	0	0	1	-7
1	1	0	0	1	0	0	0	-7<<3=-56

2.4.1 *division par 2^L, avec quantification par troncature: décalage à droite de L bits*

2.4.1.1 principe de codage

pour *L* positif, la division par 2^L se ramène à un décalage à droite de *L* bits, avec report du bit de signe...

- cette opération ne peut pas générer d'overflow (dépassement d'échelle)

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
X=-101	1	0	0	1	1	0	1	1				
Y=X>>4=-7	1	1	1	1	1	0	0	1	1	0	1	1

- par contre elle entraîne une perte de précision par rapport au résultat théorique (perte de la partie fractionnaire du résultat)

exemple : [-101] / 2^4 => décalage à droite de 4 bits, avec report du bit de signe dans les 4 bits de poids fort. *Les 4 bits de poids faible du résultat, correspondant à des puissances négatives de 2, sont perdus, d'où la perte de précision*

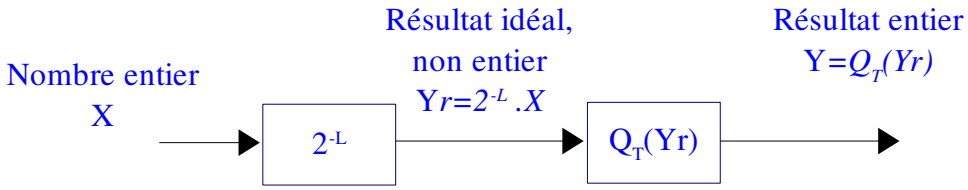
le résultat obtenu est Y=-7, alors que le résultat idéal est Yr=[-101] . 2^{-4} = -6.3125

Les bits de poids faible du résultat représentent toujours une grandeur positive <1

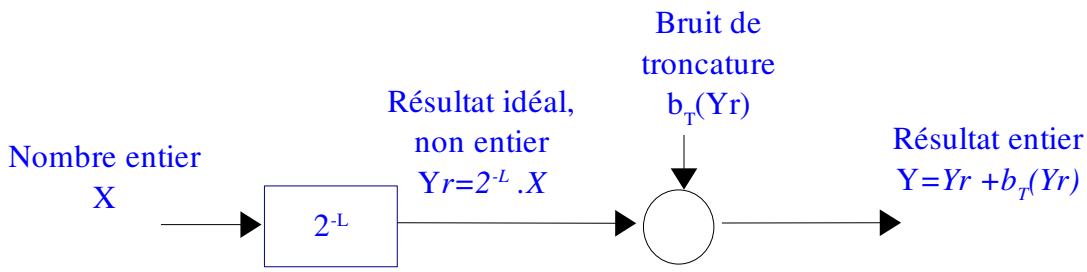
De ce fait le résultat obtenu est l'entier immédiatement inférieur ou égal au résultat idéal.

La fonction $Q_T(Yr)$ qui, à un réel Yr fait correspondre l'entier immédiatement inférieur ou égal $Q_T(Yr)$, est appelée QUANTIFICATION PAR TRONCATURE (***floor*** en anglais).

un décalage à droite de L bits correspond à une multiplication par 2^{-L}, suivi d'une quantification par troncature



On peut également remplacer la fonction $Q_T(Yr)$ par un additionneur de bruit de troncature $b_T(Yr)$, en notant : $Q_T(Yr) = Yr + b_T(Yr)$.



2.4.1.1 *Caractéristiques du bruit de troncature*

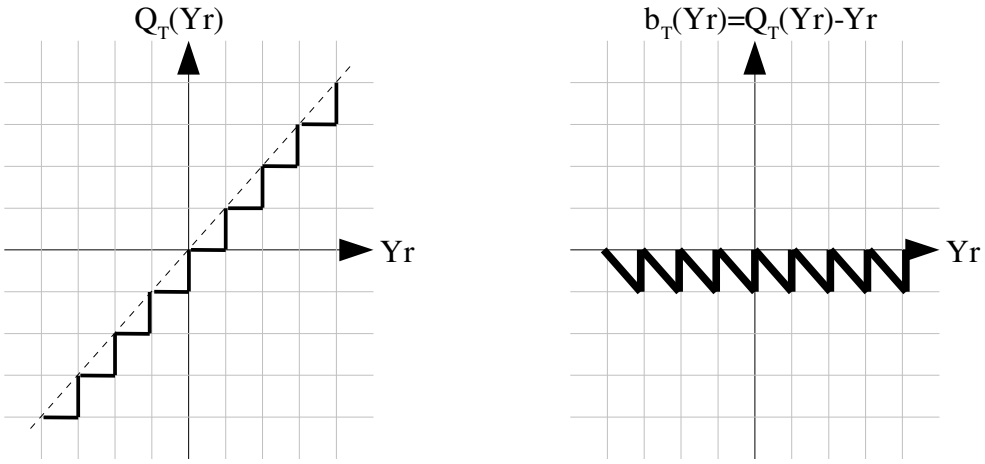


figure 1: graphe d'une quantification par troncature

1-La valeur absolue maximale du bruit de troncature b_T est égale à 1.
De plus, si on suppose que Yr est une variable aléatoire à densité de probabilité uniforme,
Le bruit de quantification b_T a alors les caractéristiques suivantes.
2- moyenne de b_T= -0.5

3-ecart-type de b_T= $\frac{1}{\sqrt{(12)}}$.

2.4.2 *division par 2^L, avec quantification par arrondi*

Il est possible d'arrondir le résultat $Y_R = X.2^{-L}$ à la valeur entière Y la plus proche, plutôt que de le tronquer. Cette troncature se nomme ***round*** *en anglais, et nous la noterons $Q_R(Yr)$, le bruit associé étant $b_R(Y_R)$.*

2.4.2.1 Caractéristiques du bruit d'arrondi

En arrondissant le résultat *Yr* à l'entier le plus proche, on fait apparaître une quantification par arrondi $Q_R(Yr)$, et un bruit de quantification b_R associé , qui a les caractérisiques suivantes :
1- valeur absolue maximale du bruit d'arrondi b_R =0.5.
avec les mêmes hypothèses sur Yr que pour le bruit de troncature
2- moyenne de b_R= 0

3-ecart-type de b_R= $\frac{1}{\sqrt{(12)}}$.

le bruit de quantification par arrondi a une <u>moyenne nulle</u>, et une v<u>aleur maximale plus faible</u> que le bruit de quantification par troncature, d'où l'intérêt d'opérer un arrondi

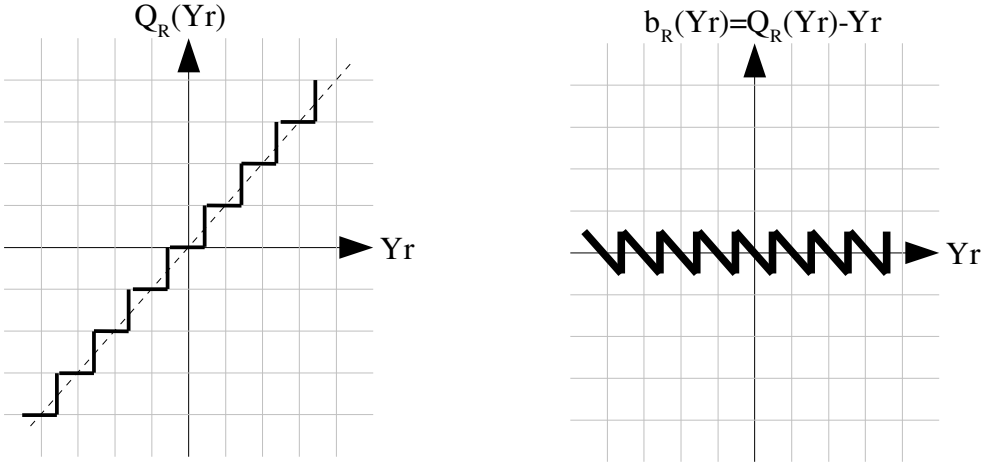


figure 2: graphe d'une quantification par arrondi

2.4.2.2 réalisation technique (parmi d'autres)

Pour réaliser une division par 2^L, avec une quantification par arrondi (le résultat Yr est arrondi à l'entier le plus proche), on peut procéder de la façon suivante :
1- on décale *X* de [*L*-1] bits à droite, avec report du signe
2- on ajoute 1 au résultat *Y* obtenu
3- on décale *Y* de 1 bit à droite
cette technique découle du fait que Arrondi(*Y*)=troncature(*Y*+0.5) =troncature((2.*Y*+1)/2)

exemple : [-101].2^{-4} avec arrondi, sur 8 bits signé =>
1- décalage à droite de 3 bits, avec report du bit de signe dans les 3 bits de poids fort.
2- On ajoute 1
3- On redécale de 1 bit à droite

le résultat final est Y=-6 , qui correspond à l'arrondi du résultat idéal Yr=-6.3125

	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
X=-101	1	0	0	1	1	0	1	1				
Y=X>>3	1	1	1	1	0	0	1	1	0	1	1	
Y=Y+1	1	1	1	1	0	1	0	0				
Y=Y>>1	1	1	1	1	1	0	1	0	0			

2.4.3 *division par 2^L, avec quantification à l'entier le plus proche de zéro*

U*ne technique très utilisée sur les DSP (Digital Signal Processor) consiste à tronquer le résultat à la valeur entière la plus proche de zéro. Cette troncature se nomme fix en anglais, et nous la noterons $Q_F(Yr)$, le bruit associé étant $b_F(Yr)$. L'intérêt de cette quantification est qu'elle correspond à une non-linéarité de gain inférieur à 1, et permet d'éviter (sous certaine conditions) l'apparition d'oscillations indésirables (cycles limite) dans les filtres numériques*

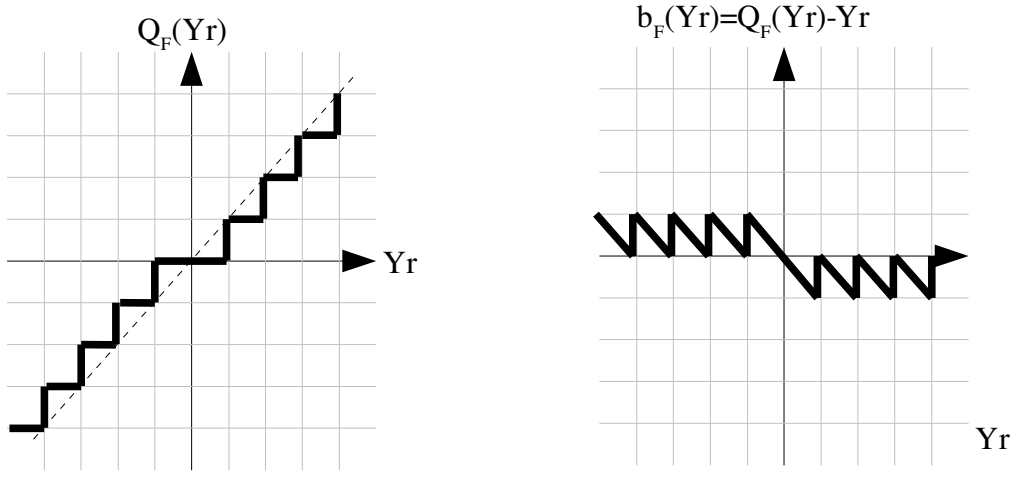


*figure 3: graphe d'une quantification de type **fix***

La réalisation technique de cette troncature peut être effectuée comme suit
1- on décale de *L* bits à droite, avec report du signe.
2- si le résultat est inférieur à 0, on ajoute 1.

2.5 exercices

-représenter en binaire l'addition [25] + [12] sur 6 bits signés. Vérifier que le résultat obtenu est faux, mais reste exact modulo 2^6

-représenter en binaire la division du nombre -13 par 2^2 , sur 5 bits signés, pour chacune des 3 méthodes de quantification du résultat : troncature Q_T , Arrondi Q_R , et Fix Q_F

3 Codage de multiplications réelles en arithmétiqe N/2N

Dans la plupart des applications de traitement de signal (filtres FIR, IIR, FFT, LMS), on doit multiplier des données (entières) par des coefficients (réels), et additionner les résultats entre eux... Pour cette raison tous les DSP (Digital Signal Processor) sont équipés d'une instruction MACC (Multiply Accumulate), qui permet d'effectuer très rapidement ce genre d'opérations. L'objet de ce paragraphe est de présenter la façon dont sont codées les multiplications d'entiers (variables) par des réels (coefficients), et la façon d'employer l'instruction MACC(facteurs d'échelle)...

3.1 Arithmétique N/2N

Un DSP (entre autres) est généralement constitué autour d'une unité de calcul en arithmétique N bits 2N bits Signée. Sans rentrer dans le détail, cela signifie que ce processeur est capable d'effectuer très efficacement

- des additions-soustractions sur des nombres entiers codés sur N ou 2N bits

- des décalages à droite (notés >>) ou à gauche (notés <<) sur N ou 2N bits

- des multiplications entières de 2 nombres codés sur N bits, pour obtenir un résultat codé sur 2N bits

lorsqu'on décrit un schéma destiné à être programmé sur un DSP ou autre, on prendra l'habitude de préciser la taille de codage des variables , sous la forme $[Nom]_{Nb\ de\ bits}$.

Par exemple $[Y]_{24}$ *se lit : la variable Y, codée sur 24 bits*

3.2 multiplication d'un nombre entier par un nombre réel

3.2.1 principe et exemple

pour fixer les idées, on présentera l'exemple de la multiplication d'un entier par le coefficient $c=\sqrt{(3)}$ *, en arithmétique 16/32 bits*

Pour calculer dans un nombre entier *Y* , le résultat de la multiplication d'un nombre entier *X*, supposé variable par un coefficient réel c supposé fixe . on procède comme suit

étape 1- choix d'un facteur d'échelle adapté au coefficient

On calcule le plus grand facteur d'échelle $\lambda_c=2^{Lc}$ en puissance entière de 2 de telle façon que la quantité $[\lambda_c . c]=[2^{Lc} . c]$ reste codable sur *N* bits, une fois arrondie à l'entier le plus proche. On cherchera donc *Lc* entier le plus grand possible qui vérifie :

$2^{Lc} . |c| \le 2^{N-1}$

Pour cela on peut procéder par tâtonnement (déconseillé), ou directement raisonner sur le logarithme en base 2 de cette inéquation :

$Lc \le N-1-\log_2(|c|)$, et *Lc* entier le plus grand possible.

Par exemple pour $c=\sqrt{(3)}$ et *N*=16 , il faudra vérifier :

$Lc \le 16-1-\log_2(|\sqrt{(3)}|) \approx 14.2$ avec *Lc* entier le plus grand possible. On retiendra donc *Lc*=14

étape 2- coefficient entier C_N et décalage à droite *Lc* associé au coefficient réel C

On peut alors en déduire le coefficient entier $c_N = arrondi(2^{Lc} . c)$

Par exemple pour $c=\sqrt{(3)}$ *N=16 , Lc= 14 , on obtient*

$c_N = arrondi(2^{14} . \sqrt{(3)}) = arrondi(28377.92) = 28378$

étape 3- on peut alors passer au codage de la multiplication, sous la forme

1- $[Y]_{2N} = [X]_N \,.\, [C_N]_N$

2- $[Y]_{2N} = Q_{T,R,F}(2^{-Lc}.[Y]_{2N})$

{ $Q_{T,R,F}$ signifie : Q_T, ou Q_R, ou Q_F selon la façon dont on code le décalage à droite de *Lc* bits }

Par exemple pour $c=\sqrt{(3)}$ *N=16 , Lc= 14 ,* $c_N = 28378$

- on peut coder (quantification par troncature du résultat, voir 2.4.1)

$[Y]_{32} = [X]_{16} \,.\, [28378]_{16}$

$[Y]_{32} = [Y]_{32} >> 14$

- mais on peut aussi coder (quantification par arrondi du résultat , voir 2.4.2)

$[Y]_{32} = [X]_{16} \,.\, [28378]_{16}$

$[Y]_{32} = [Y]_{32} >> 13$

$[Y]_{32} = [Y]_{32} + 1$

$[Y]_{32} = [Y]_{32} >> 1$

3.2.2 quantification de coefficient, quantification de variable (ou de signal)

Quelle que soit la manière dont on quantifie le résultat de la multiplication par 2^{-Lc}, Le résultat Y peut s'écrire :

$[Y]_{2N} = [X]_N . [C_N]_N . 2^{-Lc} + b_{T,R,F}$ { où $b_{T,R,F}$ représente le bruit de quantification }

soit encore :

$[Y]_{2N} = [X]_N . C_q + b_{T,R,F}$ { où $C_q = C_N . 2^{-Lc}$ est le coefficient *C* quantifié, différent de *C réel*}

Par exemple pour $c=\sqrt{(3)}$ $N=16$, $Lc= 14$, $c_N=28378$, *on aura*

$[Y]_{2N} = [X]_N . C_q + b_{T,R,F}$,

{où $c_q=2^{-Lc}.c_N=2^{-14}.28378\approx1.7320557$ est différent de $c=\sqrt{(3)}\approx1.7320508$ }

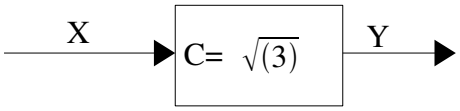
Le coefficient implicitement réalisé $c_q=2^{-Lc}.c_N$ est en général différent du coefficient réel *c* : c'est le phénomène de quantification des coefficients.

Même lorsque $c_q=c$, un bruit de quantification $b_{T,R,F}$ s'ajoute généralement au résultat de la multiplication par rapport au cas idéal : c'est la quantification des variables, ou de signal

3.2.3 les différents schémas de représentation d'une multiplication

3.2.3.1 schéma idéal

Le schéma idéal de représentation est le schéma qui ignore simultanément la quantification des coefficients, et la quantification des variables, il correspond à la représentation de la multiplication de X par le coefficient réel *C*, sans ajout de bruit

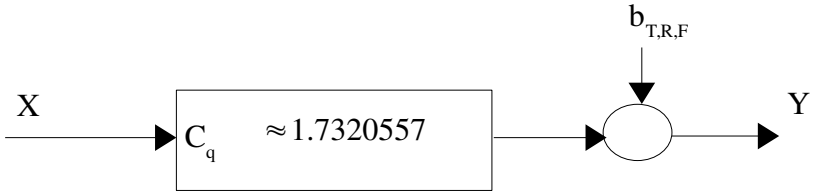


3.2.3.2 Schéma d'analyse linéaire

Le schéma d'analyse linéaire est le schéma qui ignore les détails de programmation, mais fait apparaître les dégradations par rapport au schéma idéal, à savoir

-le coefficient implicitement programmé Cq, différent du coefficient C idéal

- le bruit de quantification de signal $b_{T,R,F}$



3.2.3.3 Schéma d'implémentation

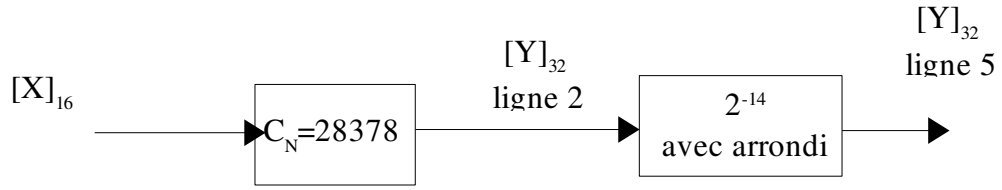
-le schéma d'implémentation est le schéma qui permet la programmation de la multiplication, on y détaille :

- les tailles de codage des variables
- les coefficients entiers, et décalages associés
- le type de quantification utilisé

si par exemple le code de la multiplication ressemble au bout de langage C suivant:

```
long  int Y=code_mul(short int X) {
    Y=28378*X;          // ligne 2
    Y=Y>>13;            // ligne 3
    Y=Y+1;              // ligne 4
    Y=Y>>1;             // ligne 5
    return Y;
}
```

le schéma d'implémentation ressemblera alors au schéma suivant



3.2.4 exercices

codage de multiplications élémentaires

en vous inspirant de 3.2.1 , écrire le code des multiplications suivantes, en arithmétique 16/32 bits : Y=-0.262 . X, Y=-5.45 .X, Y= $\sqrt{2}$ .X, avec quantification du résultat par troncature

pour chacune de ces 3 multiplications, préciser

1- la valeur du coefficient entier c_N codé sur *N* bits

2- la valeur du décalage à droite *Lc*

3- la valeur du coefficient c_q implicitement programmé

4- l'erreur relative réalisée $e_{rel}=\left|\frac{c-c_q}{c}\right|$ entre le coefficient programmé c_q et le coefficient *c*

c	c_N	L_c	c_q	e_{rel}
-0.262				
-5.45				
$\sqrt{(2)}$				

3.2.5 **cas de plusieurs additions – multiplications, notion de facteur d'échell**e

3.2.5.1 **additions multiplications en parallèle, schéma naïf**

supposons que nous ayons à réaliser plusieurs additions multiplications , conformément au schéma idéal suivant :

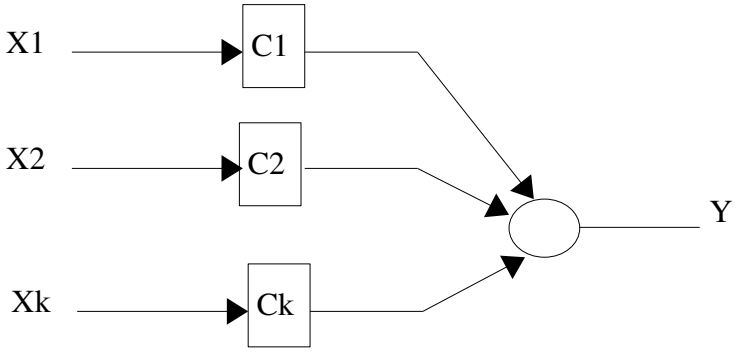


figure 4: additions-multiplications en parallèle

On peut imaginer coder chacun des coefficients Ci à sa propre échelle Lci , en utilisant la technique présentée en 3.2.1 , ce qui conduira alors au schéma d'analyse suivant

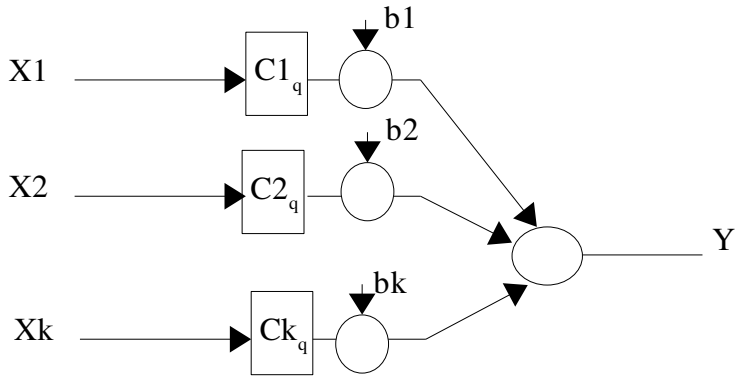


figure 5: schéma d'analyse d'additions-multiplications en parallèle

Sur ce schéma d'analyse, on s'apercoit que l'on fait apparaître *k* bruits de quantification de signal b1,..,bk , qui a priori seront indépendants, et viendront entacher la sortie Y d'une erreur qui pourra être assez importante...

3.2.5.2 évaluation de l'erreur sur Y due aux bruits

On peut évaluer les caractéristiques de l'erreur sur Y due aux bruits de la façon suivante .
1-On exprime Y uniquement en fonction des bruits, en annulant les autres entrées
2-On en déduit la valeur maximum, ou la moyenne, ou la puissance de Y.
Ici on a par exemple :
1-Y=b1+b2+..+bk { en annulant les autres entrées }
2- et donc on a

$$max(|Y|)=max(|b_1|)+max(|b_2|)+..+max(|b_k|)$$

$$E(Y)=E(b_1)+E(b_2)+..+E(b_k)$$

et, en supposant b_1, b_k indépendants , à densité de probabilité uniforme:

$$variance(Y)=variance(b_1)+variance(b_2)+..+variance(b_k)=\frac{k}{12}$$

, soit encore

$$ecart-type(Y)=\sqrt{\frac{k}{12}}$$

3.2.5.3 **additions multiplications en parallèle, schéma retenu**

pour obtenir une erreur plus faible, on peut (ou plutôt on doit) opérer les décalages à droite de remise à l'échelle en sortie de l'additionneur. La vision théorique de cela consiste à dire que l'on va multiplier l'entrée par un facteur d'échelle de signal (et non pas de coefficient) $\lambda=2^L$, avec *L* le plus grand possible, puis diviser la sortie globale Y par $\lambda=2^L$. Cette technique va atténuer l'effet des bruits b_1,.., b_k d'un facteur $\lambda=2^L$, tout en laissant inchangée la relation entrée-sortie.

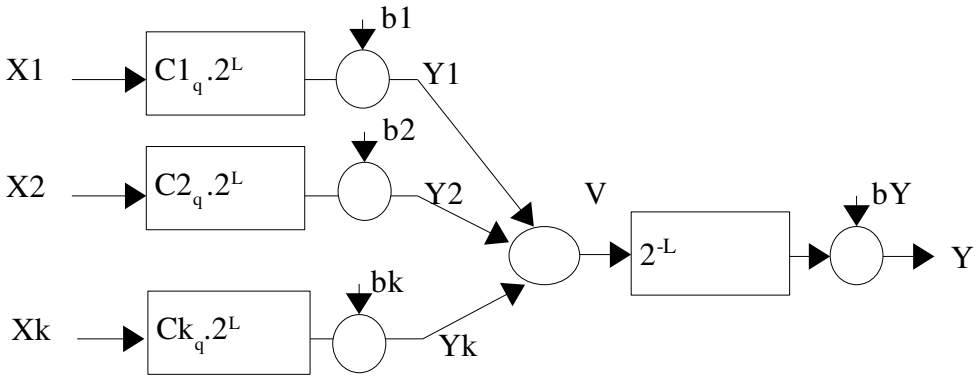


figure 6: schéma d'analyse avec facteur d'échelle $\lambda = 2^L$

On doit alors chercher le plus grand *L* possible pour que le calcul puisse s'effectuer sans dépassement d'échelle.
Pour cela il suffit que la variable *V* en sortie de l' additionneur ne dépasse pas la pleine échelle de codage: les dépassements en interne des variables Y1, Yk sont sans importance, du moment qu'à la fin les bits de poids fort de leur somme ne soient pas significatifs...

En supposant que les Xi sont codés sur *N* bits , et que *V* est codée sur 2*N* bits on doit donc trouver *L* vérifiant : $\underbrace{2^{N-1}}_{X_{max}} \cdot \sum_{i=1}^{k} |cq_i| . 2^L < \underbrace{2^{2N-1}-1}_{V_{max}}$,

soit encore : $2^L \leq \frac{V_{max}}{\left[\sum_{i=1}^{k} |cq_i|\right] . X_{max}}$:

exemple :(on reprend les mêmes coeffs qu'en 3.2.4)
c1=-0.262, c2=-5.45,c3= $\sqrt{2}$, codage sur 14 bits =>

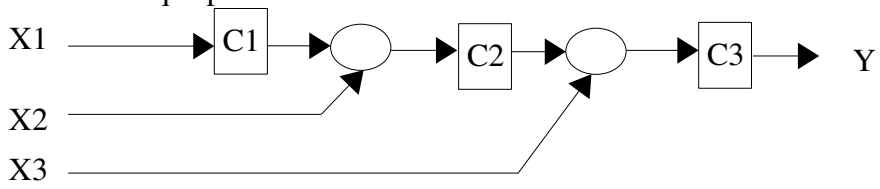
$2^L \leq \frac{2^{28-1}-1}{\left[0{,}262+5{,}45+\sqrt{(2)}\right] . 2^{14-1}} \approx 2290$ <=> $L \leq \log_2(2290) \approx 11.2$, on retiendra *L*=11.

on choisira donc un facteur d'échelle global $2^L = 2^{11} = 2048$ qui permettra de diviser par 2048 l'effet des erreurs de quantification de signal *b*1,*b*2,*b*3 sur la sortie Y.

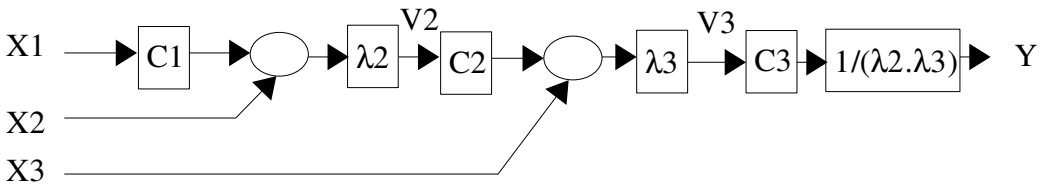
avec un tel facteur d'échelle, en arithmétique 14/28, on n'a aucun risque de dépassement de pleine échelle sur la variable V, donc le résultat sur Y ne présentera pas non plus de dépassement d'échelle.

3.2.5.1 **Cas de schémas quelconques, facteurs d'échelle et variables internes(dangereuses)..**

Dans beaucoup d'applications, il arrive que l'on ait à effectuer des additions multiplications en des points variés, tel que par exemple sur le schéma ci-après, avec les mêmes valeurs de coefficients que précédemment...



Dans ce cas, avant de passer au codage, on doit repérer l'échelle de codage imposée pour les différentes variables, et en déduire la valeur et l'emplacement des facteurs d'échelle en appliquant essentiellement la règle suivante :
les variables à l'entrée des multiplications ne doivent pas dépasser la pleine échelle de codage sur N bits. (ces variables sont appelées *variables internes* du schéma)
Ici, les *2 variables internes* sont les variables V2 et V3 *à l'entrée des coefficients C2 et C3...*
On doit donc prévoir des facteur d'échelle en puissance entière de 2 : $\lambda_2 = 2^{L2}$, $\lambda_3 = 2^{L3}$ avant ces variables, pour assurer qu'au niveau du codage, elles resteront codables sur l'échelle *N* bits....
pour minimiser l'effet des bruits sur la sortie Y, ces facteurs d'échelle devront être aussi grands que possible., tout en veillant à ce que les variables V2 et V3 restent codables sur l'échelle *N* bits.



3.2.6 Exercices

3.2.6.1 Codage de multiplications en parallèle

On veut programmer le schéma en 3.2.5.3 en arithmétique 16/32 bits
- M*ontrer que le plus grand facteur d'échelle* 2^L *que l'on peut utiliser si l'on veut que la variable V reste codable sur 2.N bits, sachant que X1,X2 et X3 sont codés sur 16 bits, correspond à L= 13...*
techniquement, pour implémenter le facteur d'échelle 2^L, on procède comme suit,
Au lieu de programmer
des multiplications par c_N, suivies par des décalages à droite L_c,
on programme
des multiplications par c_N, suivies par des décalages à droite $L_c - L$
- écrire le programme correspondant au schéma en 3.2.5.3 , lorsque L =13. (les quantifications étant effectuées par troncature). Les seules opérations autorisées sont : Additions sur N ou 2N bits, décalages à droite >> , multiplication de 2 nombres entiers codés sur N bits, résultat sur 2N bits... pour chacune des instructions, préciser le nombre de bits de codage des variables ....
- en raisonnant sur le schéma d'analyse en 3.2.5.3 , déterminer l'erreur maximale sur Y, dûe aux bruits b1,b2,b3,bY {voir 3.2.5.2 }
- montrer les bruits b1,b2,b3 ont une influence négligeable sur Y, relativement à bY...

3.2.6.2 **optimisation du code,même facteur d'échelle *L* pour tous les coefficients**
Les coefficients sont à présent quantifiés en imposant Lc =L_{commun}*=12 , indépendamment de la valeur du coefficient c. Compléter le tableau suivant correspondant à ce cas*

c	c_N	L_c=12	c_q	e_{rel}
-0.262				
-5.45				
$\sqrt{(2)}$				

-Comparer ce tableau à celui en 3.2.4 , et en déduire si cette technique présente un intérêt

quant à la précision des coefficients ci...
- écrire le programme correspondant au schéma en 3.2.5.3 , lorsque L =Lc= L_{commun}=12. et en déduire si cette technique semble présenter un intérêt quant à la complexité du code...
- Déterminer l'erreur maximale sur Y, dûe au seul bruit bY. Cette erreur a -t-elle été sensiblement modifiée par rapport à celle obtenue en 3.2.6.1 ?..

3.2.6.3 codage des multiplications pour un schéma quelconque

Vous allez ici procéder à la programmation en arithmétique 16/32 bits du schéma en 3.2.5.1 .
- sachant que les variables Xi sont codées sur 16 bits, déterminer l'expression des variables internes Vi, sur le schéma de principe avec facteur d'échelle (sans tenir compte de la quantification des coefficients c1,c2,c3...

partie 1: détermination des facteurs d'échelle de signal

- en déduire la plus grande valeur de λ_2 que l'on peut retenir pour que V2 reste toujours codable sur 16 bits.
- sachant que $\lambda_2 = 2^{L2}$, avec L2 entier, en déduire la plus grande valeur de L2, puis de $\lambda_2 = 2^{L2}$, que l'on peut retenir...
- pour cette valeur de $\lambda_2 = 2^{L2}$, en déduire la plus grande valeur de λ_3 que l'on peut retenir, pour que V3 reste toujours codable sur l'échelle 16 bits...
- sachant que $\lambda_3 = 2^{L3}$, avec L3 entier, en déduire la plus grande valeur de L3, puis de $\lambda_3 = 2^{L3}$, que l'on peut retenir...

partie 2: écriture du programme

*N*ormalement, on devrait commencer par dessiner le schéma d'analyse, pour évaluer le bruit sur Y, lié aux bruits de quantification de signal...
Pour que vous compreniez plus facilement où apparaissent ces bruits, on vous demande d'écrire tout d'abord le programme....
- Ecrire le programme correspondant au schéma de principe avec facteurs d'échelle, en employant les coefficients entiers C_N et décalages *Lc* du tableau en 3.2.4 ... Lorsque cela est possible, regrouper les décalages à droite en un seul décalage global (c'est le cas par exemple en sortie de la multiplication par le coefficient c3)...

partie 3: analyse des erreurs

-dessiner le schéma d'implémentation correspondant à votre programme, puis en déduire le schéma d'analyse
- en déduire l'erreur de sortie due aux bruits de quantification, ainsi que ses caractéristiques {valeur maximum, moyenne, écart-type , voir 3.2.5.2 }