Zeste de savoir

Les nombres flottants logarithmiques

12 août 2019

Table des matières

1. Rappels 1
   1.1. Nombres flottants 1
   1.2. Nombres à virgule fixe 2
2. Nombres logarithmiques 3
   2.1. Encodage 3
   2.2. Pourquoi "logarithmiques" ? 3
3. Unité de calcul logarithmique 4
   3.1. Multiplication et division 4
   3.2. Addition et soustraction 5
   3.3. Simplifications 6

Il existe divers formats de nombres à virgule utilisés en informatique. Le plus utilisé d'entre eux est le format à virgule flottante. Mais il en existe deux autres : la virgule fixe, et les nombres flottants logarithmiques. Dans cet article, nous allons voir les nombres flottants logarithmiques, et montrer comment fabriquer une unité de calcul pour ces nombres flottants.

1. Rappels

Mais avant toute chose, nous allons faire un rappel sur les nombres flottants et les nombres à virgule fixe : nous en aurons besoin dans ce qui va suivre, pour bien comprendre ce que sont les nombres flottants logarithmiques.

1.1. Nombres flottants

L'écriture d'un nombre flottant en binaire est basée sur son écriture scientifique. Cela permet de coder beaucoup plus de valeurs qu'un nombre en virgule fixe, à nombre de bits égal. Pour rappel, en décimal, l'écriture scientifique d'un nombre consiste à écrire celui-ci comme un produit entre un nombre et une puissance de 10. Ainsi, un nombre x aura une écriture scientifique en base 10 de la forme :

$a \times 10^{Exposant}$

Notre nombre a ne possède qu'un seul chiffre à gauche de la virgule : on peut toujours trouver un exposant tel que ce soit le cas. En clair, en base 10, sa valeur est comprise entre 1 (inclus) et 10 (exclu).

En binaire, c'est à peu près la même chose, mais avec une puissance de deux. L'écriture scientifique binaire d'un nombre consiste à écrire celui-ci sous la forme :

$a \times 2^{exposant}$

Le nombre a ne possède toujours qu'un seul chiffre à gauche de la virgule, comme en base 10. Toutefois, en binaire, seuls deux chiffres sont possibles : 0 et 1. Le chiffre de a situé à gauche de la virgule est donc soit un zéro ou un 1.

Pour stocker cette écriture scientifique avec des zéros et des un, il nous faut stocker la partie fractionnaire de notre nombre a, qu'on appelle la mantisse, et l'exposant. On rajoute souvent un bit de signe qui sert à calculer le signe du nombre flottant : ce bit vaut 1 si ce nombre est négatif et vaut 0 si notre flottant est positif.

Bit de signe	Exposant	Mantisse
0	0011 0001	111 0000 1101 1001

Mais parlons un peu de cette mantisse. Vous croyez surement que l'ensemble de cette mantisse est stockée dans notre nombre flottant. Et bien rien n'est plus faux : seule la partie fractionnaire est stockée dans nos nombres flottants : le chiffre situé à gauche de la virgule n'est pas stocké dans la mantisse.

Ce bit est stocké dans notre nombre flottant de façon implicite et peut se déduire en fonction de l'exposant : on ne doit pas le stocker dans notre nombre flottant, ce qui permet d'économiser un bit. Il est souvent appelé le bit implicite dans certains livres ou certaines documentations. Dans la majorité des cas, il vaut 1, et ne vaut 0 que dans quelques rares exceptions : les flottants dénormaux.

Après avoir stocké notre mantisse, parlons de l'exposant. Sachez que celui-ci peut être aussi bien positif que négatif : c'est pour permettre de coder des nombres très petits. Mais notre exposant n'est pas codé avec les représentations de nombres entiers qu'on a vues au-dessus. A la place, notre exposant est stocké en lui soustrayant un décalage prédéterminé. Pour un nombre flottant de n bits, ce décalage vaut $2^{n-1} - 1$.

1.2. Nombres à virgule fixe

La méthode de la virgule fixe consiste à émuler nos nombres à virgule à partir de nombre entiers. Un nombre à virgule fixe est donc codé par un nombre entier proportionnel à notre nombre à virgule fixe. Pour obtenir la valeur de notre nombre à virgule fixe, il suffit de diviser l'entier servant à le représenter par un nombre constant, fixé une bonne fois pour toute.

Par exemple, pour coder 1,23 en virgule fixe, on peut choisir comme "facteur de conversion" 1000. L'entier permettant de coder 1,23 sera alors 1230. La représentation en virgule fixe était utile du temps où les ordinateurs n'intégraient pas de circuits capables de travailler directement sur des nombres à virgule flottante. Cette méthode n'est presque plus utilisée, et vous pouvez l'oublier sans problème.

Pour se simplifier la vie, les nombres en virgule fixe utilisent un facteur de conversion qui est une puissance de deux. Cela permet ainsi de simplifier les calculs.

2. Nombres logarithmiques

Dans les nombres flottants logarithmiques, la mantisse est fixée une fois pour toute. Oui, vous avez bien lus : tous les nombres logarithmiques ont exactement la même mantisse. Seul leurs exposants changent. Un nombre logarithmique est donc une spécialisation d'un nombre flottant.

La mantisse est choisie pour avoir la valeur 1. C'est ainsi, tous les nombres logarithmiques sont des puissances de deux. Attention toutefois : l'exposant est ici un nombre fractionnaire, et pas un entier. Ce qui permet de représenter pas mal de nombres de taille diverses.

2.1. Encodage

Pour rappel, un nombre flottant s'écrit ainsi :

$a \times 2^{exposant}$

Sauf que comme dit plus haut, la mantisse est fixée à 1, ce qui donne :

$2^{exposant}$

Dans ces conditions, pas besoin de préciser la mantisse dans le nombre. La base 2 reste implicite, comme elle l'était pour les nombres flottants ou à virgule fixe. Un nombre logarithmique est donc composé :

— d'un bit de signe ;
— d'un exposant.

Bit de signe	Exposant
0	0011 0001

Vu que l'exposant est fractionnaire, celui-ci peut être stocké de deux manières différentes :

— sous la forme d'un nombre à virgule fixe ;
— comme pour les flottants, avec un biais.

2.2. Pourquoi "logarithmiques" ?

Prenons un nombre logarithmique, noté A. On sait que :

$A = 2^{exposant}$

Prenons le logarithme en base 2 des deux cotés de l'équation :

$log_2(A) = log_2(2^{exposant})$

Ce qui donne :

$log(A) = exposant$

Voici pourquoi l'on parle de nombres logarithmiques : ces nombres sont représentés en machine par leur exposant en base deux.

3. Unité de calcul logarithmique

Maintenant, nous allons fabriquer une unité de calcul pour les flottants logarithmiques. Nous allons commencer par voir les deux opérations de base : la multiplication et la division.

3.1. Multiplication et division

Ces nombres logarithmiques ont une propriété assez intéressante : les multiplications et divisions entre nombres logarithmiques sont très simples à effectuer. Pour comprendre pourquoi, il faut se souvenir d'un théorème de mathématique sur les logarithmes : le logarithme d'un produit est égal à la somme des logarithmes.

$log(A \times B) = log(A) + log(B)$

Dans ces conditions, une multiplication entre deux flottants logarithmiques se transforme en une simple addition d'exposants.

Le même raisonnement peut être tenu pour la division. Dans les calculs précédents, il suffit de se rappeler que diviser par N, c'est multiplier par $\frac{1}{N}$.

$log(\frac{A}{B}) = log(A \times \frac{1}{B})$

En appliquant le théorème du dessus, on trouve :

$log(A \times \frac{1}{B}) = log(A) + log(\frac{1}{B})$

Là, il faut se rappeler que $log(\frac{1}{B}) = -log(B)$. En remplaçant dans l'expression du dessus, on trouve :

$log(A \times \frac{1}{B}) = log(A) - log(B)$.

La division s'est transformée en simple soustraction.

Dans ces conditions, une unité de calcul logarithmique devant effectuer des multiplications et des divisions est constituée d'un simple additionneur/soustracteur et de quelques (ou plusieurs, ça marche aussi) circuits pour corriger le tout.

Nous allons d'abord voir comment sont gérés les flottants en virgule fixe, puis ceux avec un biais.

3.1.1. Virgule fixe

Si l'exposant est géré en virgule fixe, cela signifie que chaque exposant est multiplié par un facteur de conversion. Donc, quand on veut additionner les deux exposants, voici ce qui se passe :

$(exposant_1 \times conversion) + (exposant_2 \times conversion)$

Ce qui se factorise en :

$(exposant_1 + exposant_2) \times conversion$

Et oui, il n'y a rien à faire : on obtient bien le bon résultat dès le départ.

3.1.2. Gestion du biais

Mais quand il y a présence d'un biais, on n'est pas aussi chanceux. Dans les cas de flottants gérés avec un biais, voyons ce que donne l'addition de deux exposants :

$(exposant_1 + biais) + (exposant_2 + biais)$

En effectuant l'addition telle quelle, le biais est compté deux fois. On doit donc le soustraire après l'addition.

Même chose pour la soustraction :

$(exposant_1 + biais) - (exposant_2 + biais)$

Ce qui donne :

$exposant_1 - exposant_2$

Or, le résultat attendu devrait être : $exposant_1 - exposant_2 + biais$ Il faut rajouter le biais pour obtenir l'exposant correct. On a donc besoin de deux additionneurs/soustracteurs pour implémenter la multiplication et la division.

3.2. Addition et soustraction

Pour l'addition et la soustraction, la situation est beaucoup plus corsée. Il n'y a pas vraiment de méthodes pour calculer le logarithme d'une somme. Dans ces conditions, la seule solution est de pré-calculer celle-ci. Notre unité de calcul va donc incorporer une mémoire ROM, dans laquelle on stockera le résultat de la somme de deux nombres logarithmiques. En envoyant les deux nombres concaténés sur le bus d'adresse, on récupère ainsi l'exposant de leur somme.

Seul problème : la taille de cette ROM est tout simplement gigantesque pour des flottants de 32 à 64 bits. Pour des nombres de 16 bits, cela passe relativement bien : une mémoire de 32 kilo-octets suffit. Mais pour du 32 bits, on atteint 16 gibi-octets. Dans ces conditions, on doit ruser pour diminuer la taille de la ROM en utilisant diverses propriétés mathématiques.

L'idée est de transformer notre addition en une opération plus simple, qui peut se pré-calculer plus facilement. La table ainsi obtenue devra être plus petite.

On veut :

$log(x + y)$

Ce qui est équivalent à :

$log(x + x \times \frac{y}{x})$

Ce qui vaut :

$log(x \times (1 + \frac{y}{x}))$

Vu que le logarithme d'un produit est égal à la somme des logarithmes, on a :

$log(x) + log(1 + \frac{y}{x})$

Le terme de gauche est le nombre logarithmique. Le logarithme de droite peut se pré-calculer facilement, et donne une table vraiment petite.

Dans ces conditions, l'addition se traduit en :

— un circuit qui divise les deux opérandes et leur ajoute 1 : un diviseur couplé à un additionneur suffit ;
— une table qui prend en entrée le résultat de l'additionneur, et fournit le terme de droite ;
— et un autre additionneur pour le résultat.

3.3. Simplifications

Pour implémenter les quatre opérations, on a donc besoin :

— de deux additionneurs/soustracteur et d'un diviseur pour l'addition/soustraction ;
— de deux autres additionneurs/soustracteur pour la multiplication et la division ;
— et d'une ROM.

Je ne sais pas si vous avez remarqués, mais Il est tout-à-fait possible de mutualiser les additionneurs pour la multiplication et l'addition. En rajoutant quelques multiplexeurs, on peut faire en sorte que le circuit puisse se configurer pour les additionneurs servent soit pour la multiplication, soit pour l'addition. On économise en peu de circuits.