

Virgule flottante

*Dieu a créé les entiers naturels, tout le
reste a été fait par l'homme* L. Kronecker



Alain GUYOT

Concurrent Integrated Systems
TIMA



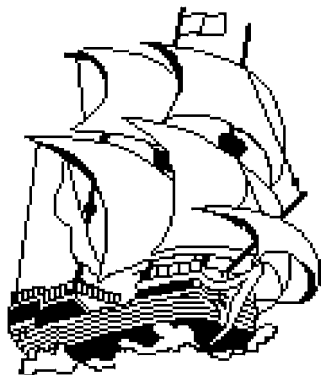
(33) 04 76 57 46 16



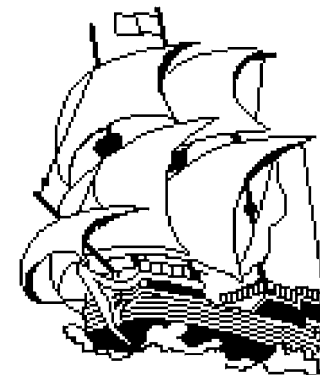
Alain.Guyot@imag.fr

<http://tima-cmp.imag.fr/Homepages/guyot>

Techniques de l'Informatique et de la Microélectronique
pour l'Architecture. Unité associée au C.N.R.S. n° B0706



Dieu a créé les entiers naturels, tout le reste a été fait par l'homme L. Kronecker



But de la virgule flottante: Représentation et calcul des nombres réels.

Représentés par des rationnels
(donc approximés avec une certaine erreur)

Avec les ordinateurs, les opérations sur les entiers sont toujours sans erreur (résultat juste) tandis que les opérations sur les réels sont souvent erronées (résultat faux).

Cela ne tient pas aux ordinateurs mais à la nature des entiers (représentables avec des chiffres) et des réels (non représentables avec des chiffres). Par exemple π ne s'écrit pas avec des chiffres mais peut se représenter comme (périmètre/diamètre) d'un cercle. Les ordinateurs représentent "au mieux" avec des chiffres les valeurs non représentables.

Essentiellement la représentation virgule flottante ne fait rien d'autre que d'automatiser le choix de l'échelle (ou la position de la virgule).

Standard pour la virgule flottante

But du "standard": assurer la portabilité des logiciels de calcul.

Problèmes d'implémentation: les opérateurs sur les réels sont assez complexes et ont une grande influence sur les performances de la machine et mobilisent une part importante des ressources.

La puissance de calcul se mesure en MFLOP (million de flottant par seconde). Actuellement de 5 à 200.

Solutions: 1- Anticipation
2- Prédiction
3- Spéculation

Standard ANSI/IEEE 754-1985 for Binary Floating-Point Arithmetic

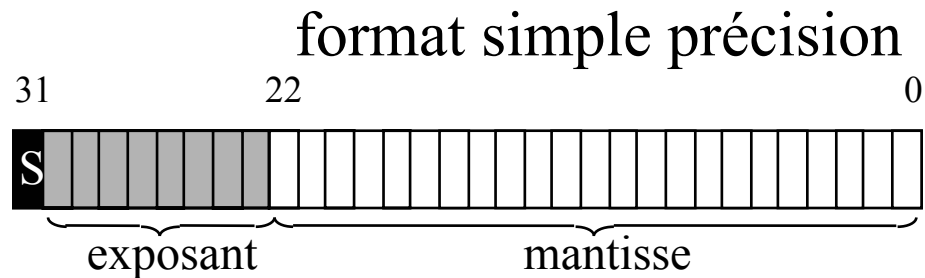
Le standard spécifie:

- 1-Les formats virgule flottante simple et double précision normalisés
- 2-Les échappement du format: ± 0 , $\pm \infty$, dénormalisé, nonnombres (NaN)
- 3-Les opérations addition, soustraction, multiplication, division, racine carrée, reste et comparaison (pas de fonction prévue)
- 4-Les conversions entre entiers et virgule flottante
- 5-Les conversions entre formats virgule flottante
- 6-Les conversions entre virgule flottante et chaîne décimales
- 7-Les modes d'arrondi (très important)
- 8-Les exceptions et leurs traitement

<http://www.loria.fr/serveurs/CCH/documentation/IEEE754>

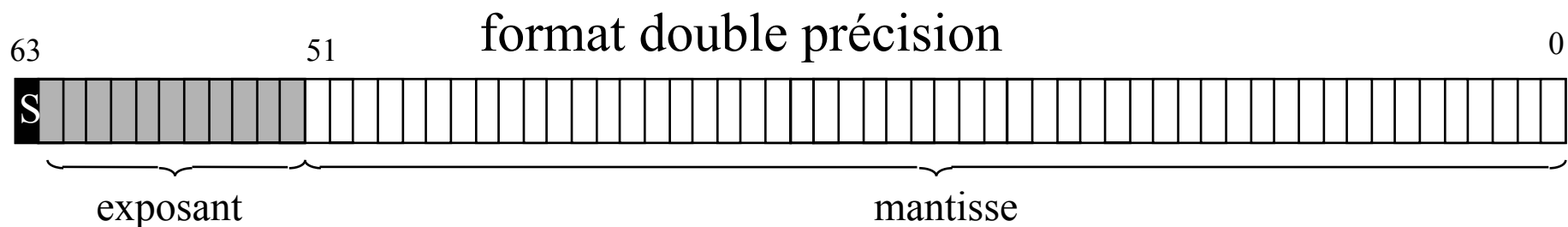
Format IEEE 754-1985 Réels Normalisés

Mantisse normalisée
 $1 \leq m < 2, \quad m = 1,f$



Calcul de la valeur de V

$$V = (-1)^{r_{31}} \times 2^{\left(\sum_{i=0}^7 r_{i+23} 2^i - 127\right)} \times \left(\frac{2^{23} + \sum_{i=0}^{22} r_i 2^i}{2^{23}} \right)$$



Champs et bits dans les champs rangés par importance décroissante

Normalisation de la mantisse (ou significande)

Avantages

1- Notation unique

$$\begin{array}{lll} 11,00 * 2^{-1} & = 3 * \frac{1}{2} & \text{non valide} \\ 1,10 * 2^0 & = 1,5 & \in [1, 2[\\ 0,11 * 2^1 & = 0,75 * 2 & \text{non valide} \end{array}$$

2 - "1" avant la virgule implicite (peut être omis ou caché)

Inconvénients

1 - La valeur "0" ne s'exprime pas

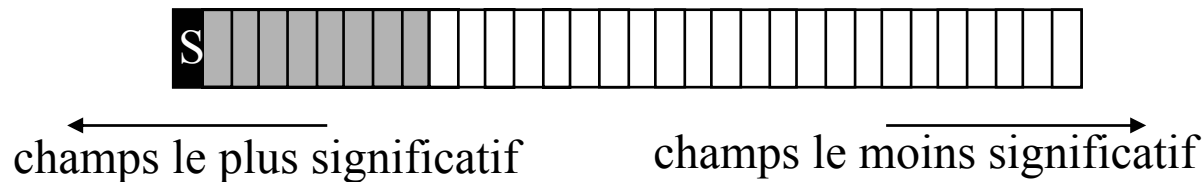
2 - Les valeur "petites" ($< 2^{\min \text{ expo}}$) ne sont pas normalisables

Représentation “biaisée” de l’exposant

Avantages

Pas de "bit de signe".

1- Comparaison de nombres:
nombres en virgule flottante comme entiers
(les champs sont par ordre de signification)



2- Comparaison d’exposant

Inconvénients

Lorsqu’on ajoute deux exposants, il faut rajouter le biais

Lorsqu’on soustrait deux exposants, il faut retrancher le biais.

Remarque: la représentation biaisée (biased) s’appelle également la représentation par excès (excess)

Format IEEE 754-1985: Limites

Longueur totale	32 bits	64 bits
mantisse + bit implicite	23 + 1 bits	52 + 1 bits
exposant	8 bits	11 bits
biais, max, min	127, +127, -126	1023, +1023, -1022
domaine approximatif	$2^{128} \approx 3,8 \times 10^{38}$	$2^{1024} \approx 9 \times 10^{307}$
précision approximative	$2^{-23} \approx 10^{-7}$	$2^{-52} \approx 10^{-15}$
plus petit nombre normalisé	$2^{-126} \approx 10^{-38}$	$2^{-1022} \approx 10^{-308}$
plus petit nombre $\neq 0$	2^{-148}	2^{-1073}

Mantisse normalisée $\Rightarrow$

- 1- Notation spéciale du **0**
- 2- Notation spéciale de nombres "dénormalisés"
- 3- Notations spéciales pour $\infty, +\infty, -\infty$
et **NaN** (Not a Number)

Le rapport entre la masse de l'univers et celle du proton est d'environ 10^{78} (Paul Dirac)

Standard IEEE 754-1985

Échappements des formats

Le standard spécifie pour les simple précision:

- 1-Si $e = 255$ et $m \neq 0$ alors v est **NaN**
- 2-Si $e = 255$ et $m = 0$ alors v est $(-1)^s \infty$
- 3-Si $0 < e < 255$ alors $v = (-1)^s 2^{e-127} (1,m)$
- 4-Si $e = 0$ et $m \neq 0$ alors $v = (-1)^s 2^{-126} (0,m)$ (*dénormalisé*)
- 5-Si $e = 0$ et $m = 0$ alors $v = (-1)^s 0$

Le standard spécifie pour les double précision:

- 1-Si $e = 2047$ et $m \neq 0$ alors v est **NaN**
- 2-Si $e = 2047$ et $m = 0$ alors v est $(-1)^s \infty$
- 3-Si $0 < e < 2047$ alors $v = (-1)^s 2^{e-1023} (1,m)$
- 4-Si $e = 0$ et $m \neq 0$ alors $v = (-1)^s 2^{-1022} (0,m)$ (*dénormalisé*)
- 5-Si $e = 0$ et $m = 0$ alors $v = (-1)^s 0$

Standard IEEE 754-1985 Algèbre d'exceptions

	a	b	a + b	a * b	a ÷ b
	0	0	0	0	NaN
	0	y	z	0	0
	0	∞	∞	NaN	0
x > 0	x	0	z	0	∞
y > 0	x	y	0, z ou ∞	0, z ou ∞	0, z ou ∞
z > 0	x	∞	∞	∞	0
	∞	0	∞	NaN	∞
	∞	y	∞	∞	∞
	∞	∞	∞	∞	NaN
	∞	$-\infty$	NaN	$-\infty$	NaN
	NaN	0	NaN	NaN	NaN
	NaN	y	NaN	NaN	NaN
	NaN	∞	NaN	NaN	NaN

Standard IEEE 754-1985 Incohérence

1: $a = L_{nn}$ (L_{nn} est le plus grand nombre représentable)

2: $b = a + a$ $b = L_{nn} + L_{nn} = 2 L_{nn}$ $b = L_{nn} + L_{nn} = \infty$

3: $c = b \div a$ $c = 2 (L_{nn} \div L_{nn}) = 2$ $c = \infty \div L_{nn} = \infty$

4: $d = 1 \div c$ $d = 1 \div 2 = 0,5$ $d = 1 \div \infty = 0$

5: $e = 1 \div (d - 0,5)$ $e = 1 \div (0,5 - 0,5) = \infty$ $e = 1 \div (0 - 0,5) = -2$

Exécution théorique

Exécution réelle

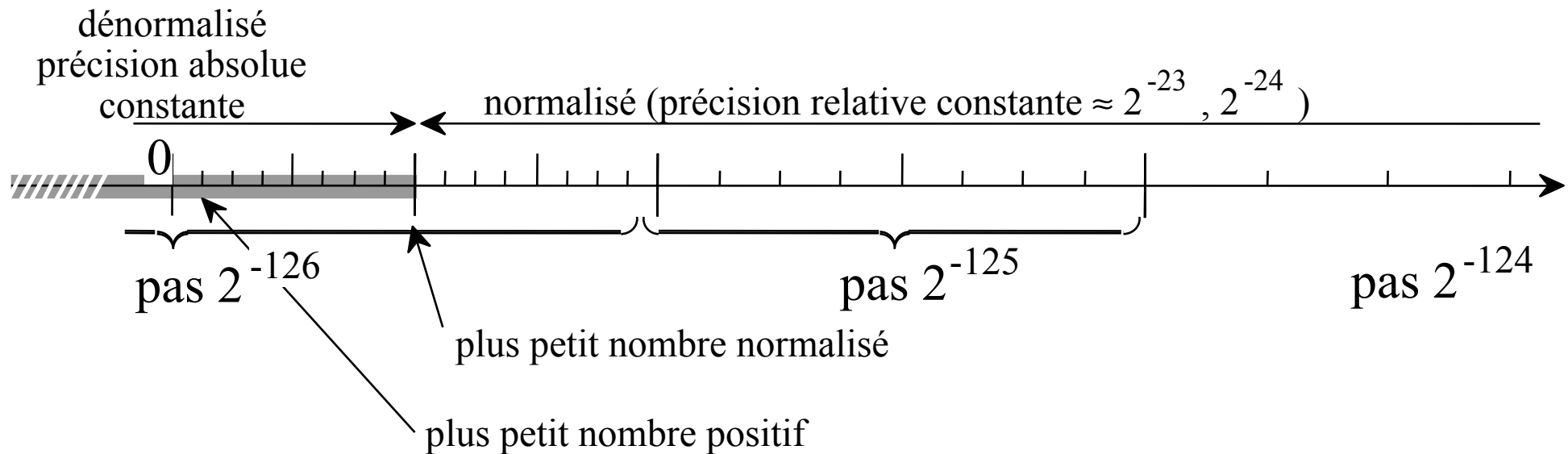
" It makes me nervous to fly an airplane since I know they are designed using floating-point arithmetic "

Alston Scott Householder (1904-1993) (un des pères de l'algorithmique numérique)

Nombres dénormalisés (1)

optionnels

But: combler le trou entre le plus petit nombre normalisé et 0.
On conserve la précision absolue de ce nombre, avec une précision relative qui se dégrade.



Les nombre dénormalisés sont seulement recommandés par la norme.
Leur usage est coûteux en délai et en matériel;
Un mode de calcul permet d'éviter leur usage pour une exécution plus rapide.

Nombres dénormalisés (2)

Un résultat "très petit" ($< 2^{\text{min expo}}$) peut être produit par:

1- Une soustraction de deux nombres "petits"

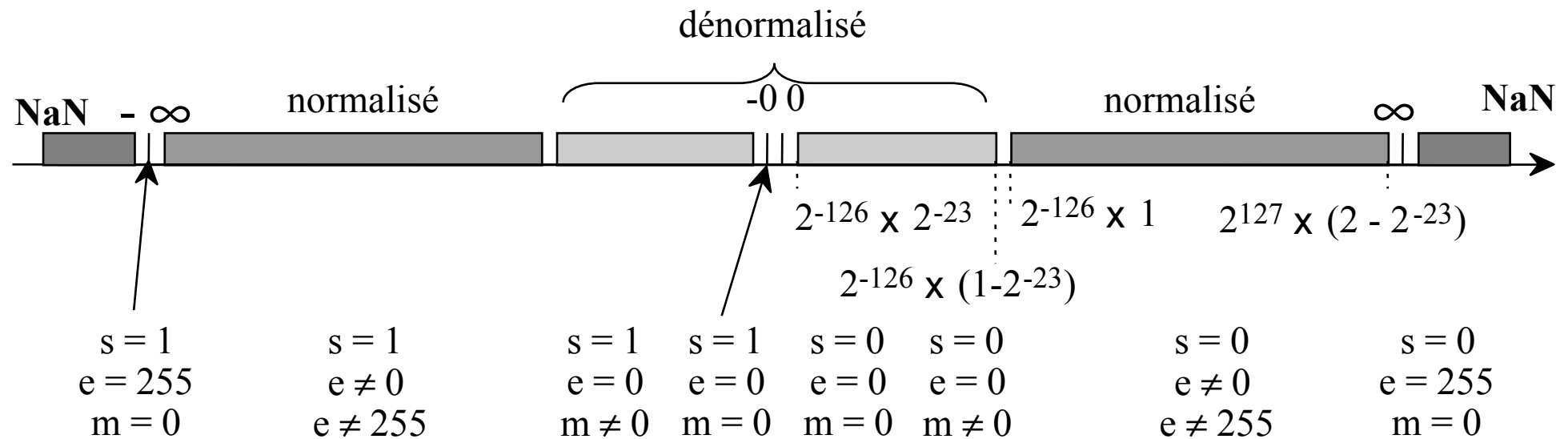
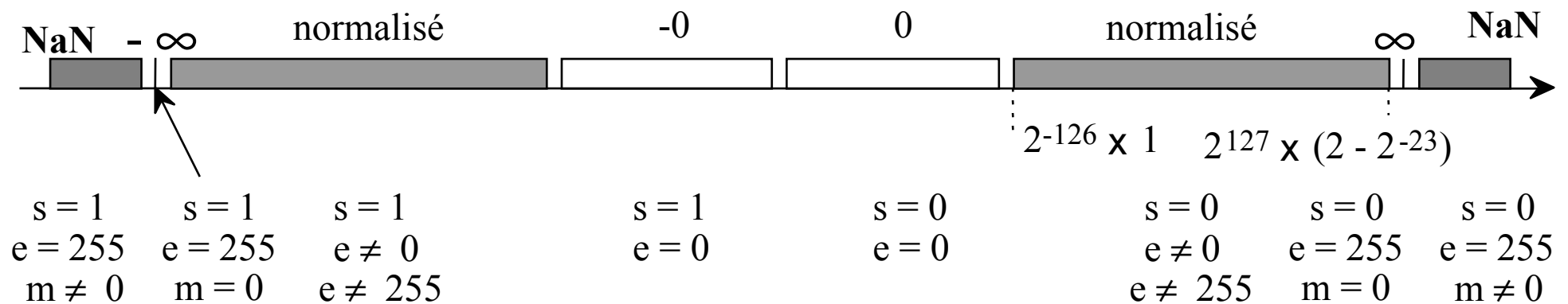
La soustraction de deux nombres "grands" donne un nombre "grand" ou zéro.

2 - Une multiplication de deux nombres "petits"

3- Une division d'un dividende "petit" par un diviseur "grand"

Le matériel pour traiter le cas 1 et les cas 2 et 3 est très différent.

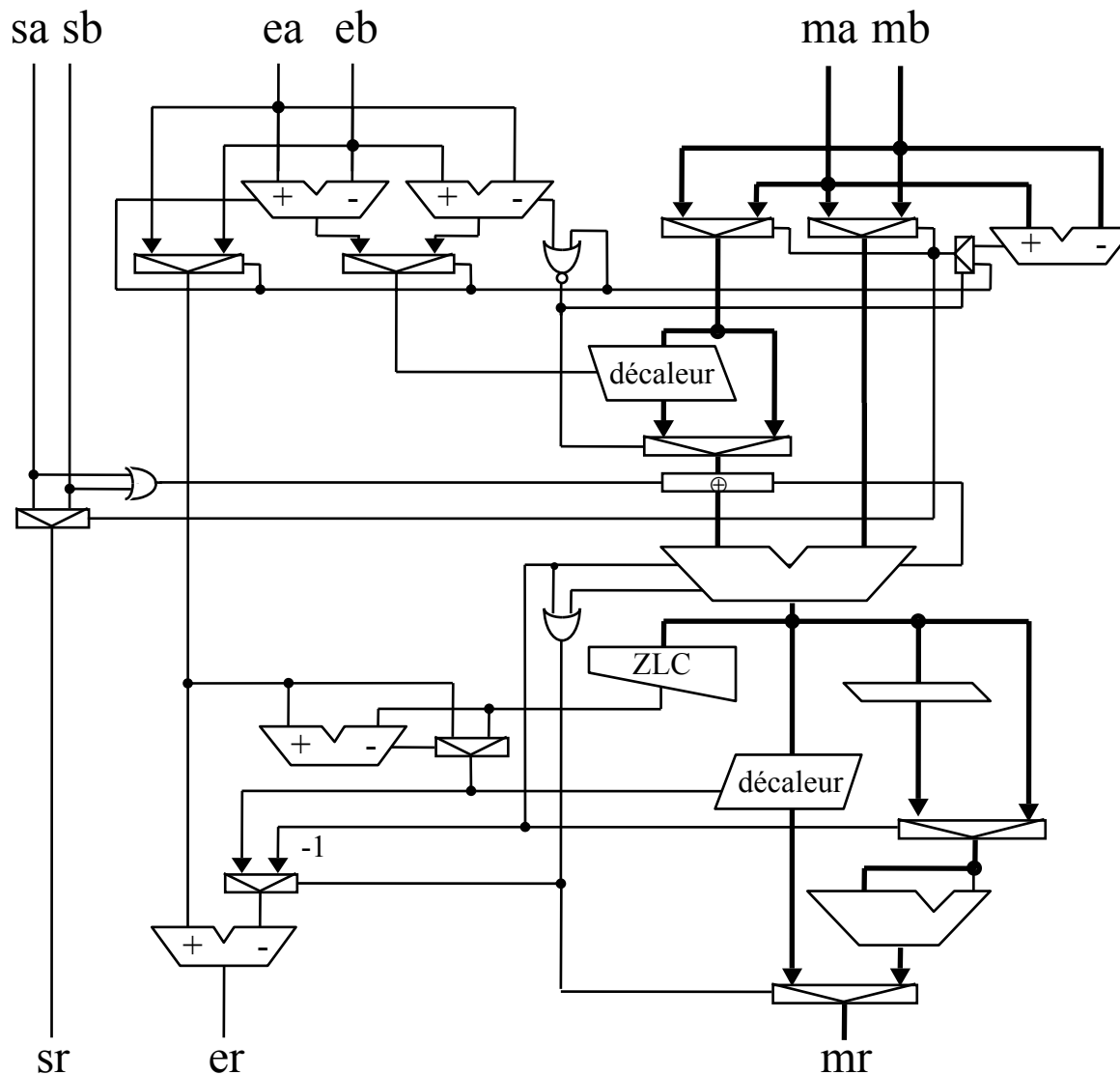
Domaines de representation du 32 bits



Exemples d'addition

Nombres Décimaux	Binaire en virgule flottante	Opérandes Alignés	Résultat normalisé
1 + 0,75	$1,0 * 2^0$ $1,1 * 2^{-1}$	$1,0 * 2^0$ <u>$0,11 * 2^0$</u> $1,11 * 2^0$	$1,11 * 2^0$
1 + 1,5	$1,0 * 2^0$ $1,1 * 2^0$	$1,0 * 2^0$ <u>$1,1 * 2^0$</u> $10,1 * 2^0$	$1,01 * 2^1$
1 - 0,75	$1,0 * 2^0$ $1,1 * 2^{-1}$	$1,0 * 2^0$ <u>$-0,11 * 2^0$</u> $0,01 * 2^0$	$1,0 * 2^{-2}$

Addition flottante



*tri du plus grand
et du plus petit*

*alignement de la mantisse
du plus petit*

*complémentation éventuelle
du plus petit*

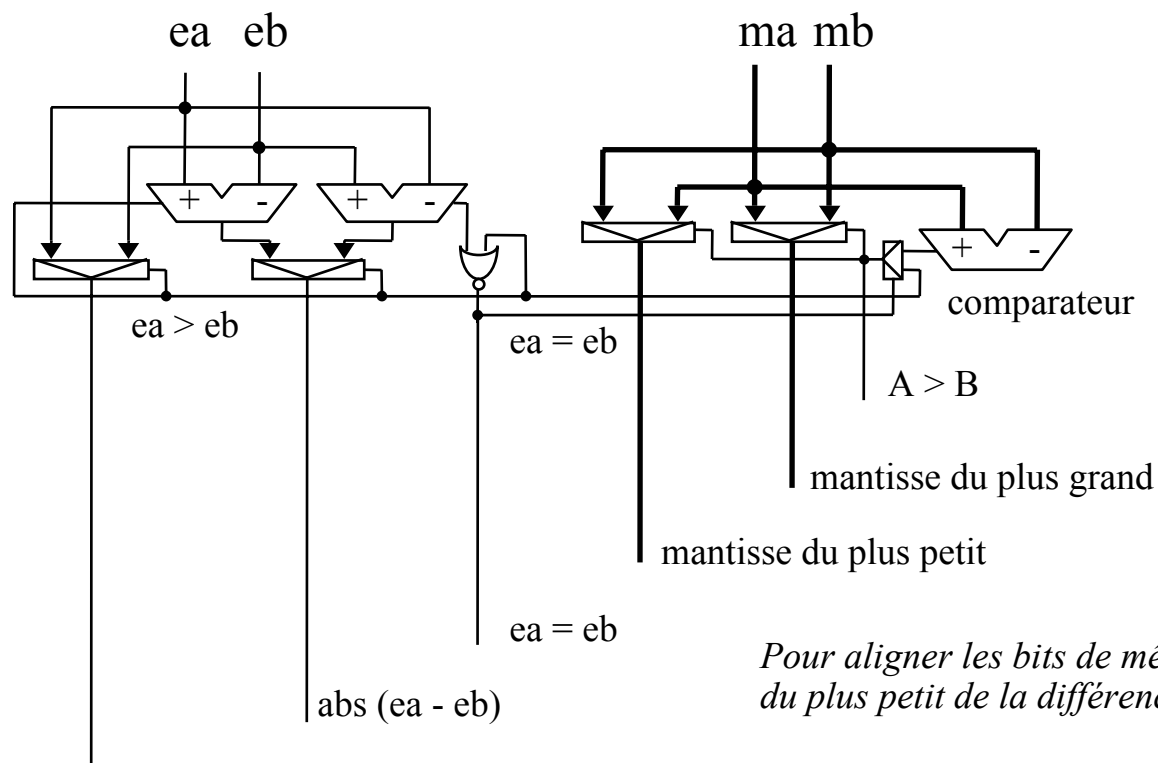
*exécution de l'opération
(addition ou soustraction)*

normalisation (si possible)

arrondi du résultat

Addition (1)

Sélection du plus grand



Le signe du résultat est le signe du plus grand

Les mantisses étant positive, il faut soustraire la plus petite de la plus grande

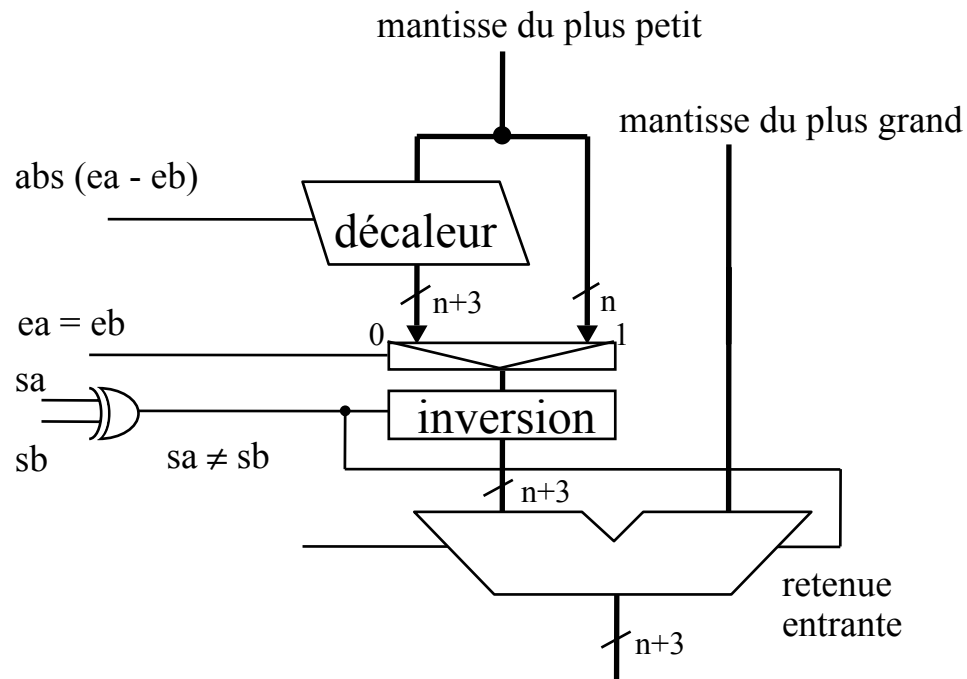
Pour aligner les bits de même poids, il faut décaler la mantisse du plus petit de la différence des exposants

$\max(ea, eb) = \text{exposant du plus grand}$

*Si A et B sont de même signe, l'exposant du résultat aura cette valeur ou cette valeur plus 1 (dépend de la retenue sortante).
Si A et B sont de signe différent, l'exposant du résultat sera compris entre cette valeur et cette valeur moins le nombre de bits de la mantisse*

Addition (2)

Alignement des mantisses



Pour aligner les bits de même poids, on décale vers la droite celui de plus petit exposant.

On garde 3 des bits sortant du décalage:

G: garde

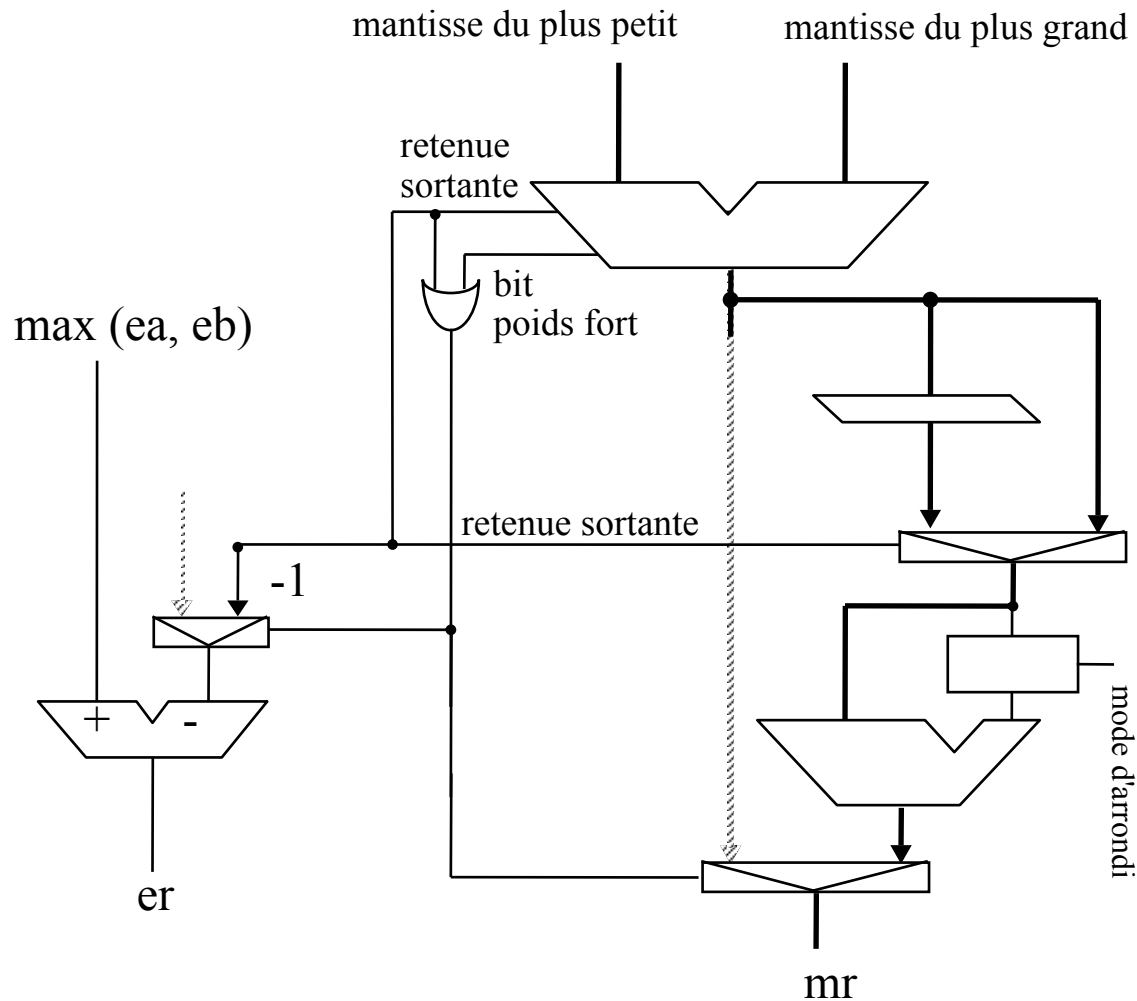
R: arrondi

C: persistant, "ou logique" de tous les bits restant.

On soustrait toujours la mantisse du plus petit de la mantisse du plus grand

Pour aller plus vite, on peut ne pas attendre de savoir quelle est la mantisse du plus grand lorsque les exposants sont égaux en doublant l'additionneur et en choisissant le résultat positif après une soustraction.

Addition (3) - Normalisation du résultat

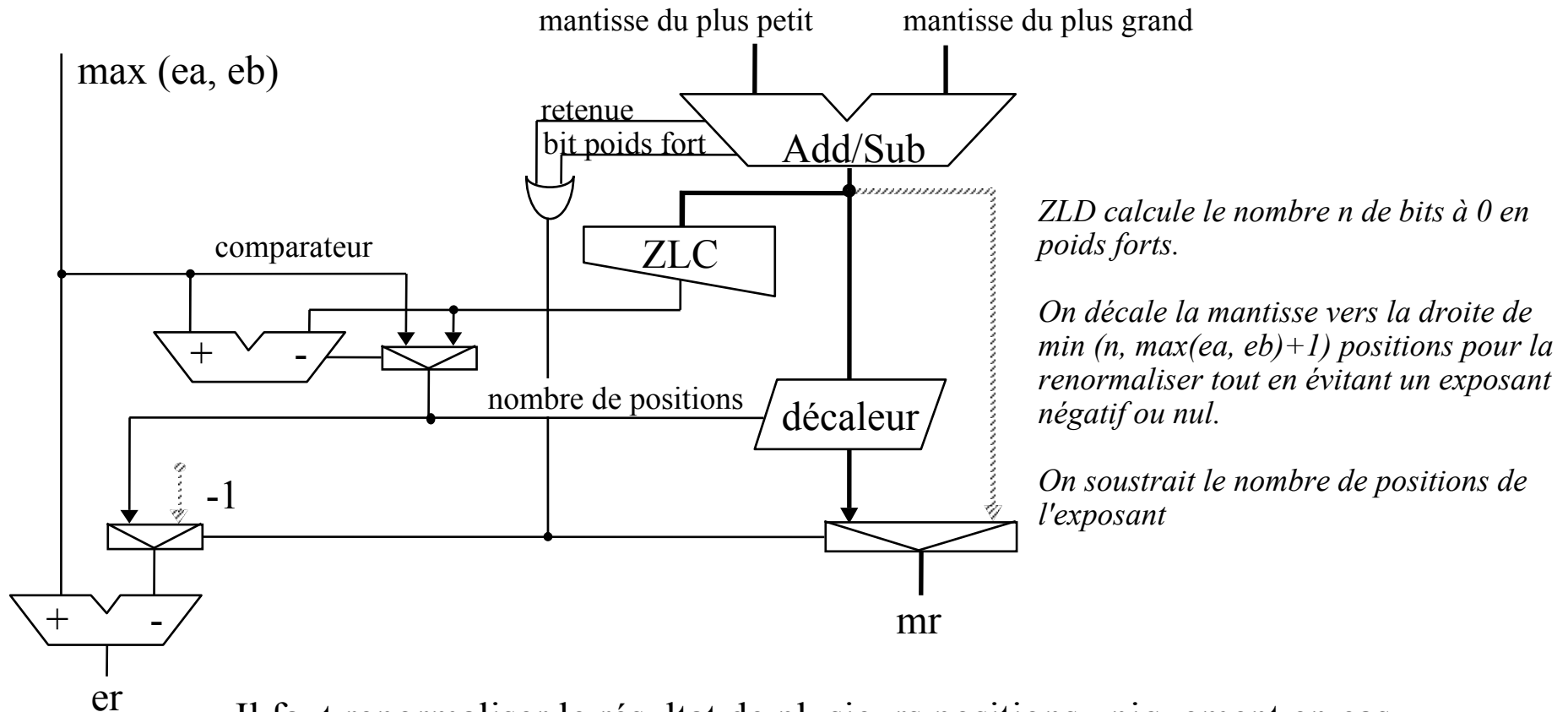


Si la retenue sortante vaut 1 alors décaler le résultat d'une position à droite.

Si la retenue vaut 0 et le bit poids fort vaut 1 alors le nombre est normalisé ($1 \leq n < 2$)

Le résultat doit être arrondi suivant le mode choisi

Addition (3') - Normalisation du résultat



Il faut renormaliser le résultat de plusieurs positions uniquement en cas de soustraction de nombre de même exposant.

Dans ce cas il n'y a pas d'arrondi

Pour aller plus vite, on peut prédire le nombre de Zéros en poids forts à partir des opérandes de la soustraction et non à partir du résultat de la soustraction

Arrondi

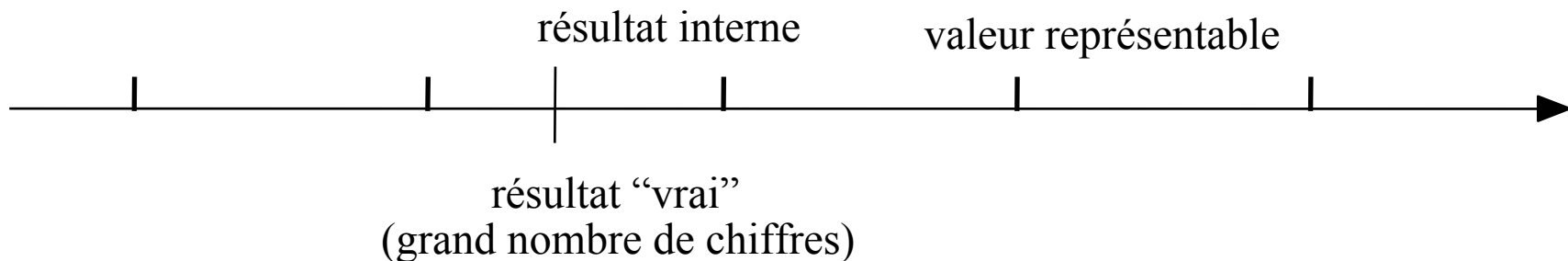
On effectue des opérations arithmétiques sur des réels représentables en format IEEE.

Le résultat d'une addition, d'une soustraction ou d'une multiplication peut être évalué sans erreur, mais le résultat peut ne pas être représentable en format IEEE.

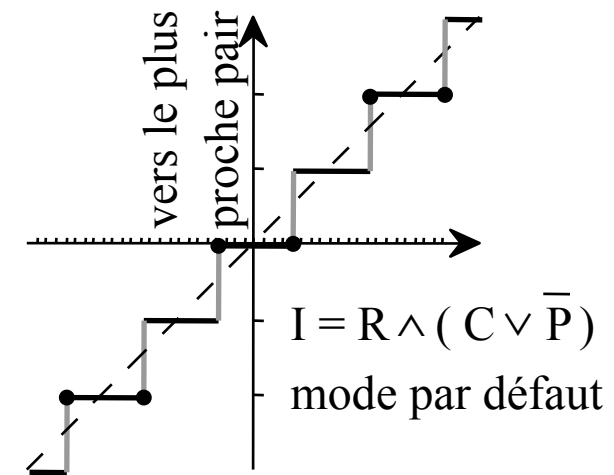
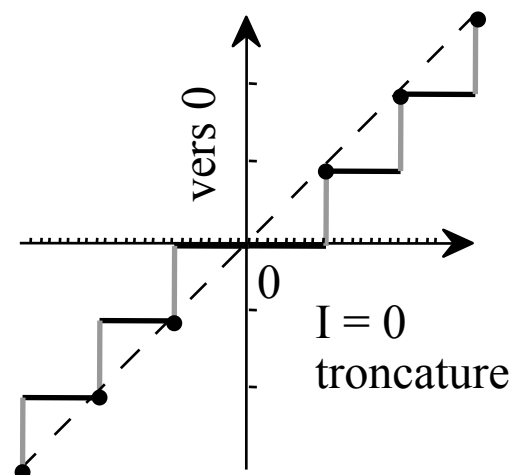
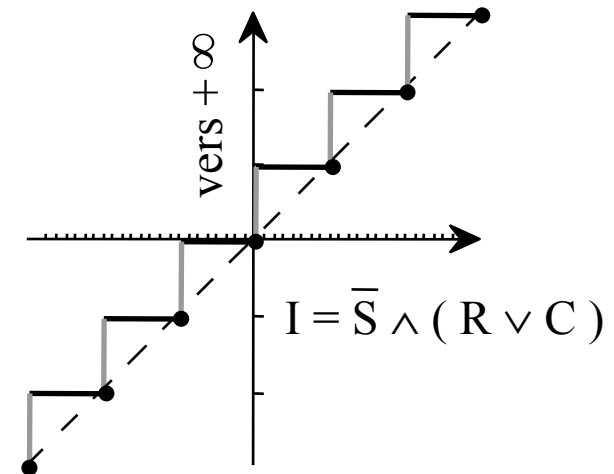
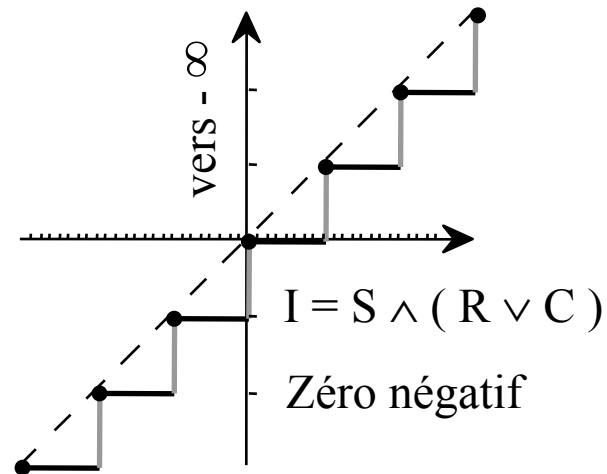
Le résultat d'une division ou d'une extraction de racine carrée peut ne pas être évaluable sans erreur sur un nombre fini de bits, (c'est pourquoi il y a un reste).

L'arrondi est destiné à trouver une représentation non exacte mais cependant acceptable.

De nombreux travaux ont porté sur la génération et la propagation des erreurs d'arrondi.



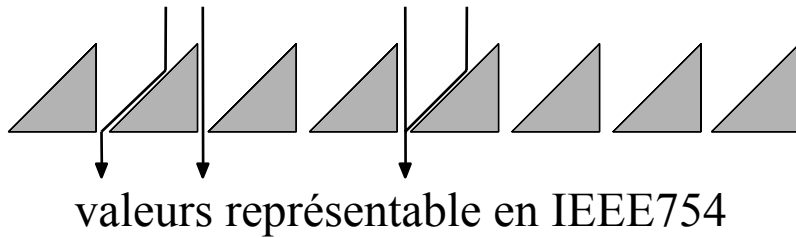
Modes d'arrondi (1)



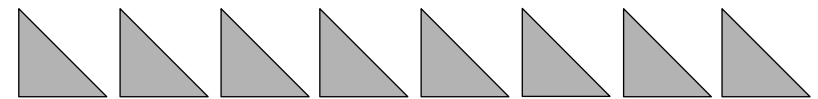
Modes d'arrondi (2)

vers $-\infty$

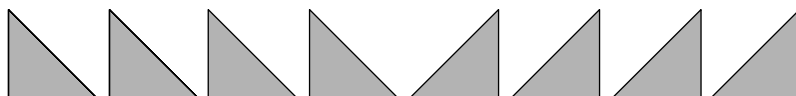
Résultat de calcul



vers $+\infty$



vers 0



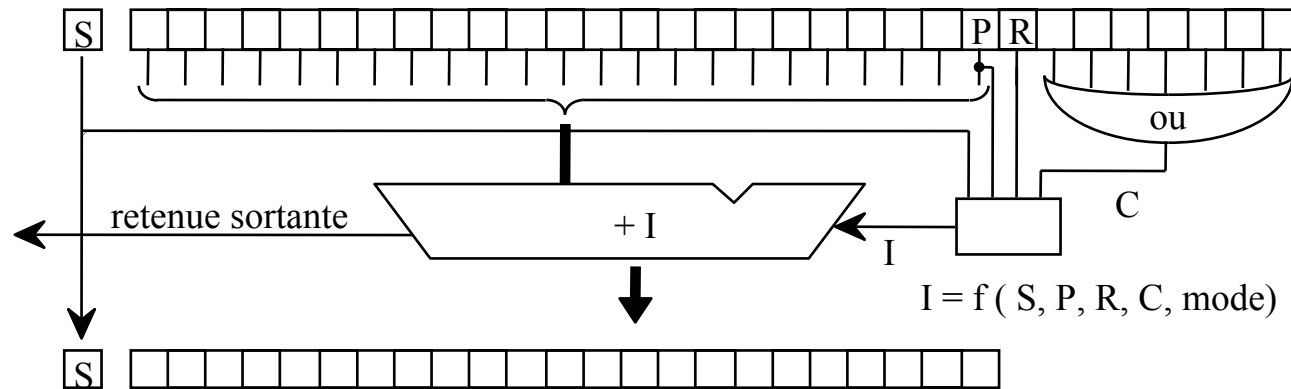
vers le plus proche



Matériel pour l'arrondi

Principe de l'arrondi

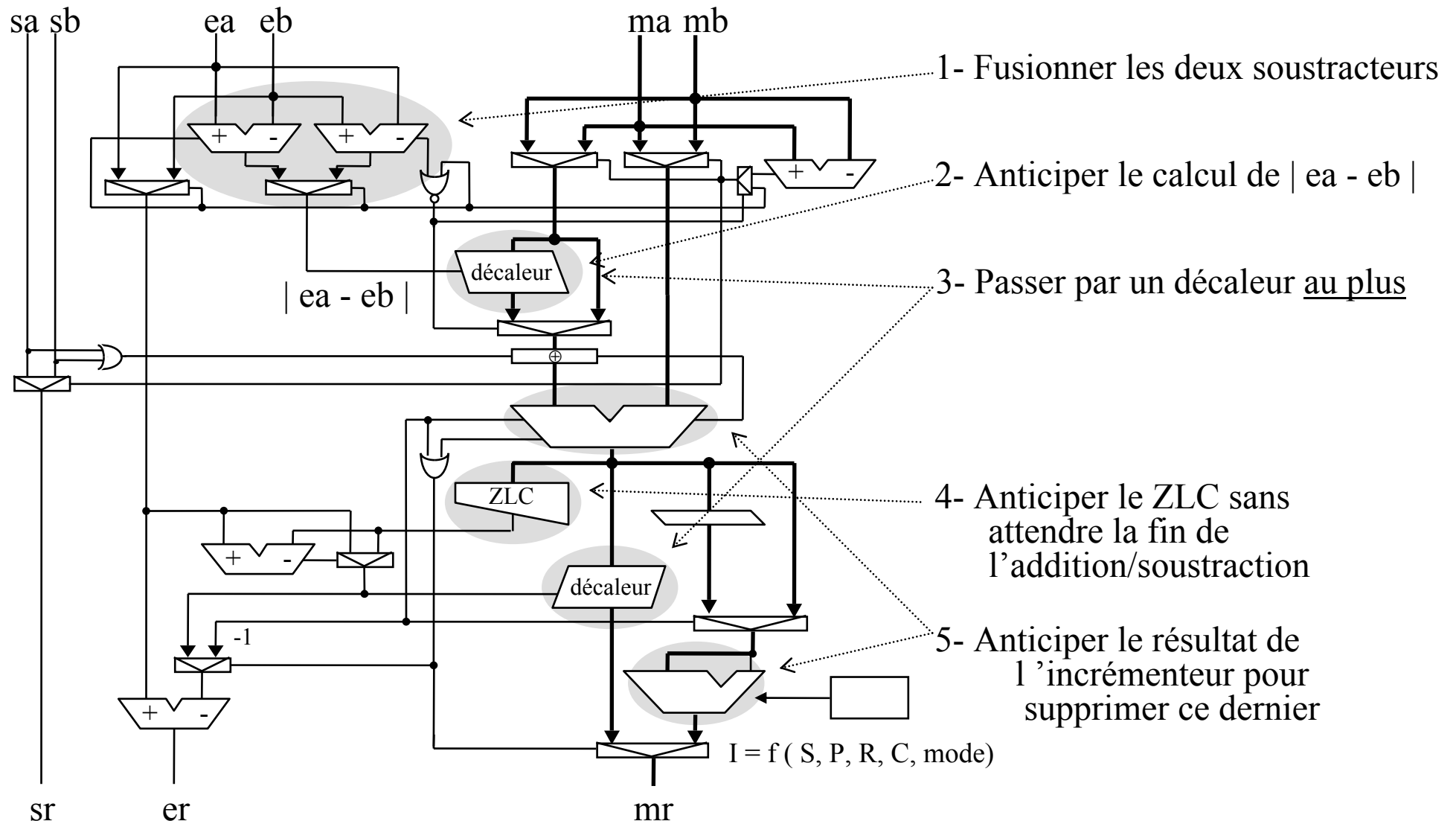
- 1- Le résultat est calculé en précision infinie (sans perte d'information)
- 2- Le résultat précédent est traité pour tenir dans un format virgule flottante.



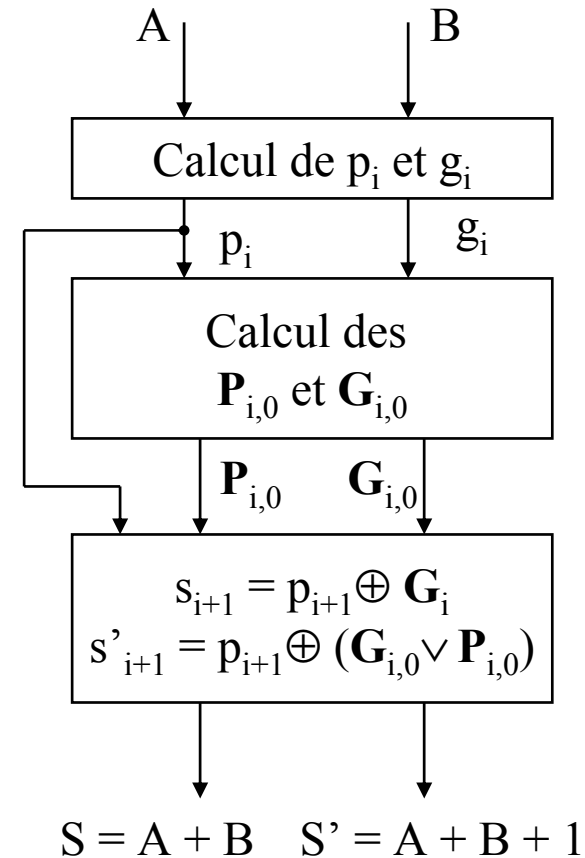
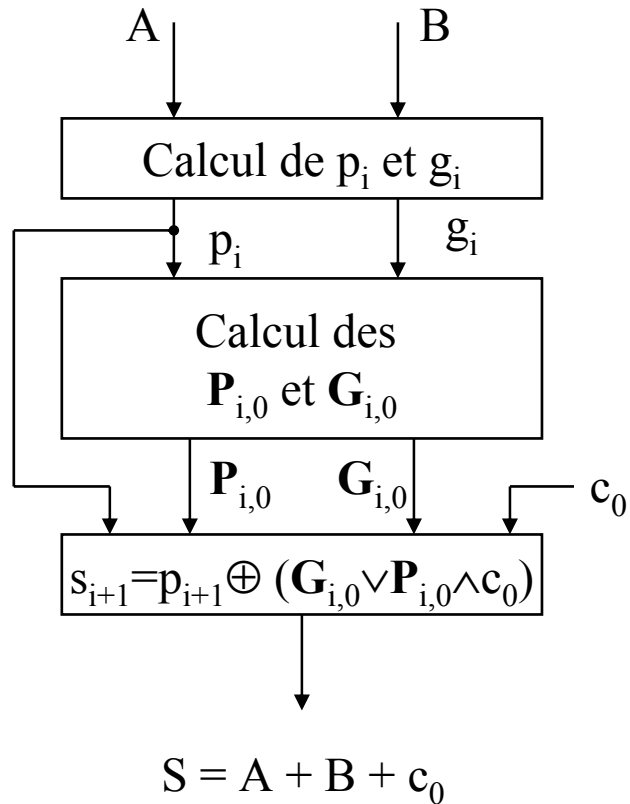
Si l'arrondi provoque une retenue sortante $\neq 0$ alors le résultat est 2

mode d'arrondi	valeur de I
vers $+\infty$	$I = \bar{S} \wedge (R \vee C)$
vers $-\infty$	$I = S \wedge (R \vee C)$
vers 0	$I = 0$
vers + proche	$I = R \wedge (C \vee P)$

Amélioration de l'Addition Flottante



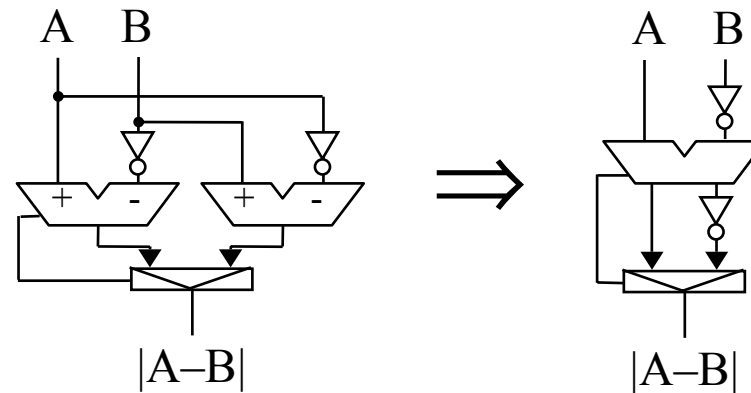
Addition à résultats multiples



On peut calculer simultanément $S = A + B$ et $S' = A + B + 1$

On peut étendre à plus de 2 sorties

Valeur absolue de la différence



Pour calculer $|A-B|$ on calcule simultanément $A-B$ et $B-A$ avec un additionneur à 2 sorties.

Soit $D = A + \overline{B} + 1 = A - B$.

Soit $S = \overline{A + \overline{B}} = \overline{A - B - 1} = -(A - B - 1) - 1 = B - A$.

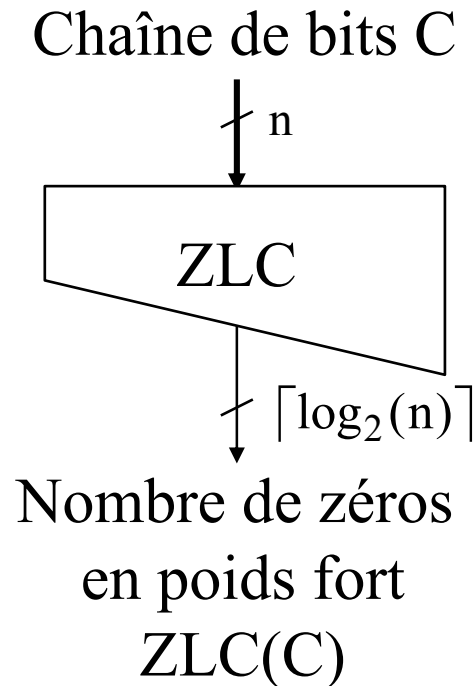
(on vérifie facilement que $D + S = 0$)

On calcule $p_i = a_i \oplus \overline{b_i}$ et $g_i = a_i \wedge \overline{b_i} \quad \forall i$

puis on calcule $G_{i,0}$ et $P_{i,0}$ avec un assemblage adéquat de cellules de Brent et Kung.

On a alors $d_i = p_i \oplus (G_{i,0} \vee P_{i,0})$ et $s_i = \overline{p_i} \oplus \overline{G_{i,0}}$

Comptage des zéros (1)



Soit une chaîne de bits C

Long(C) = longueur de la chaîne C (nombre de bits)

ZLC(C) = nombre de bits à zéro en poids forts de C

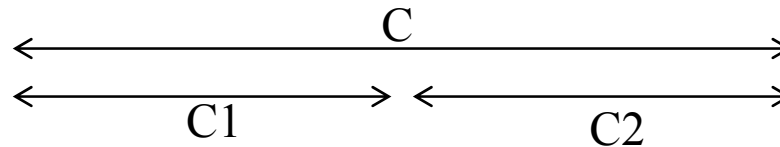
Comptage par dichotomie:

On coupe la chaîne C en deux chaînes C1 et C2.

Si ZLC(C1) < Long(C1) **alors** ZLC(C) = ZLC(C1)

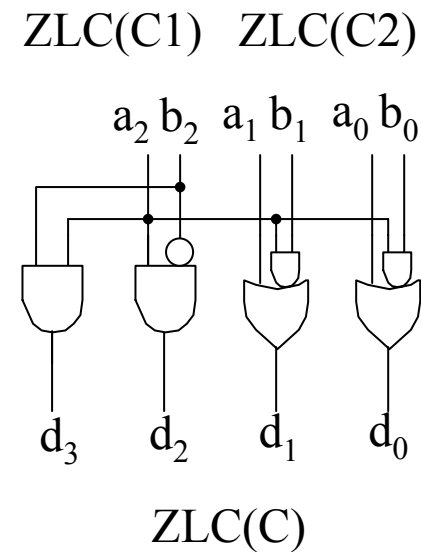
sinon ZLC(C) = Long(C1) + ZLC(C2)

Pour remplacer la comparaison ($<$) par le test de 1 bit
et remplacer l'addition ($+$) par une concaténation
les longueurs sont des puissances de 2.



Comptage des zéros (2)

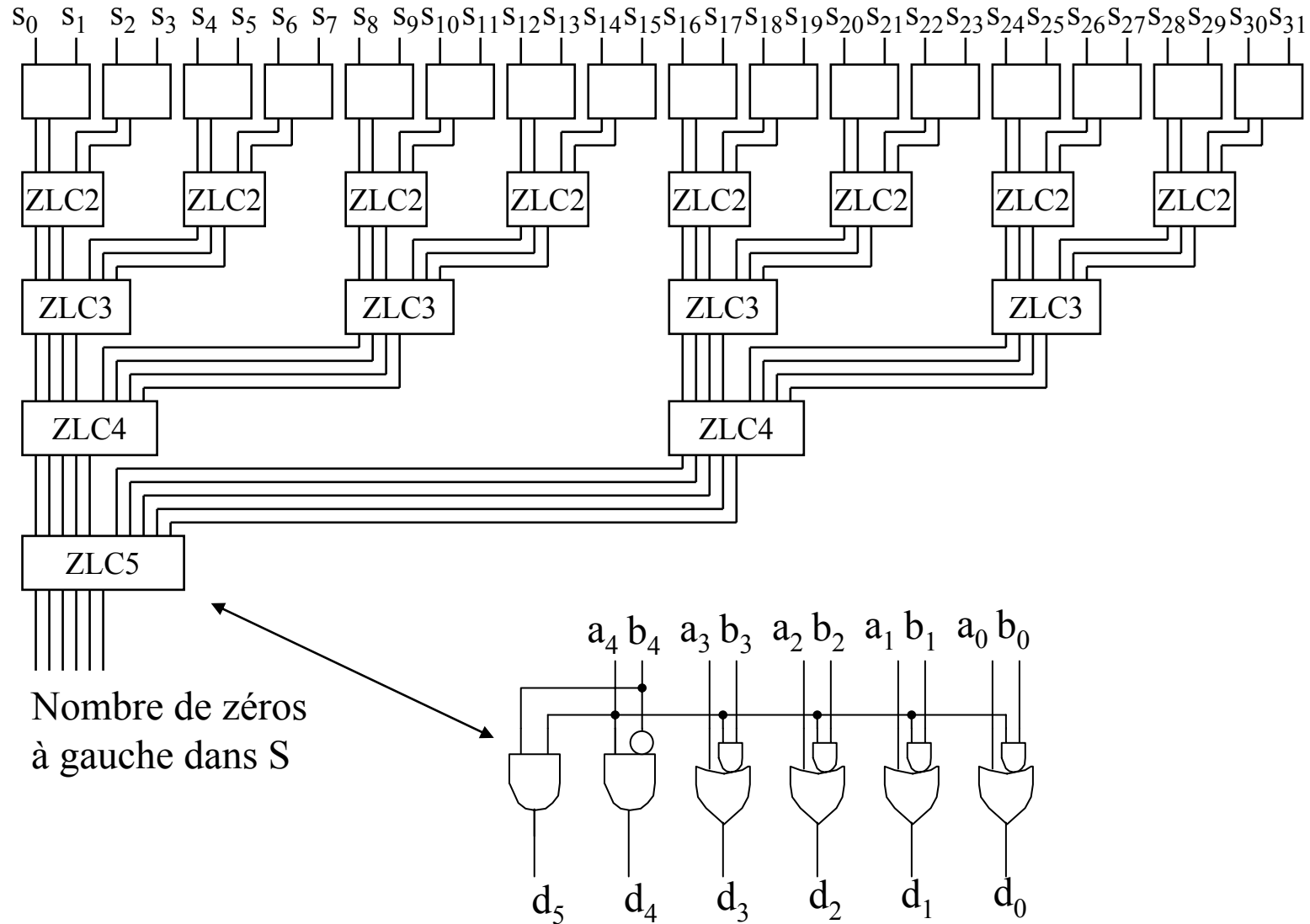
C		ZLC (C1)	ZLC (C2)	ZLC (C)
C1	C2	$a_2 a_1 a_0$	$b_2 b_1 b_0$	$d_3 d_2 d_1 d_0$
1 x x x	x x x x	0 0 0	x x x	0 0 0 0
0 1 x x	x x x x	0 0 1	x x x	0 0 0 1
0 0 1 x	x x x x	0 1 0	x x x	0 0 1 0
0 0 0 1	x x x x	0 1 1	x x x	0 0 1 1
0 0 0 0	1 x x x	1 0 0	0 0 0	0 1 0 0
0 0 0 0	0 1 x x	1 0 0	0 0 1	0 1 0 1
0 0 0 0	0 0 1 x	1 0 0	0 1 0	0 1 1 0
0 0 0 0	0 0 0 1	1 0 0	0 1 1	0 1 1 1
0 0 0 0	0 0 0 0	1 0 0	1 0 0	1 0 0 0



Si $\text{ZLC}(C1) < \text{Long}(C1)$ **alors** $\text{ZLC}(C) = \text{ZLC}(C1)$
sinon $\text{ZLC}(C) = \text{Long}(C1) + \text{ZLC}(C2)$

avec $\text{Long}(C1) = \text{Long}(C2) = 2^i$

Arbre de comptage des zéros

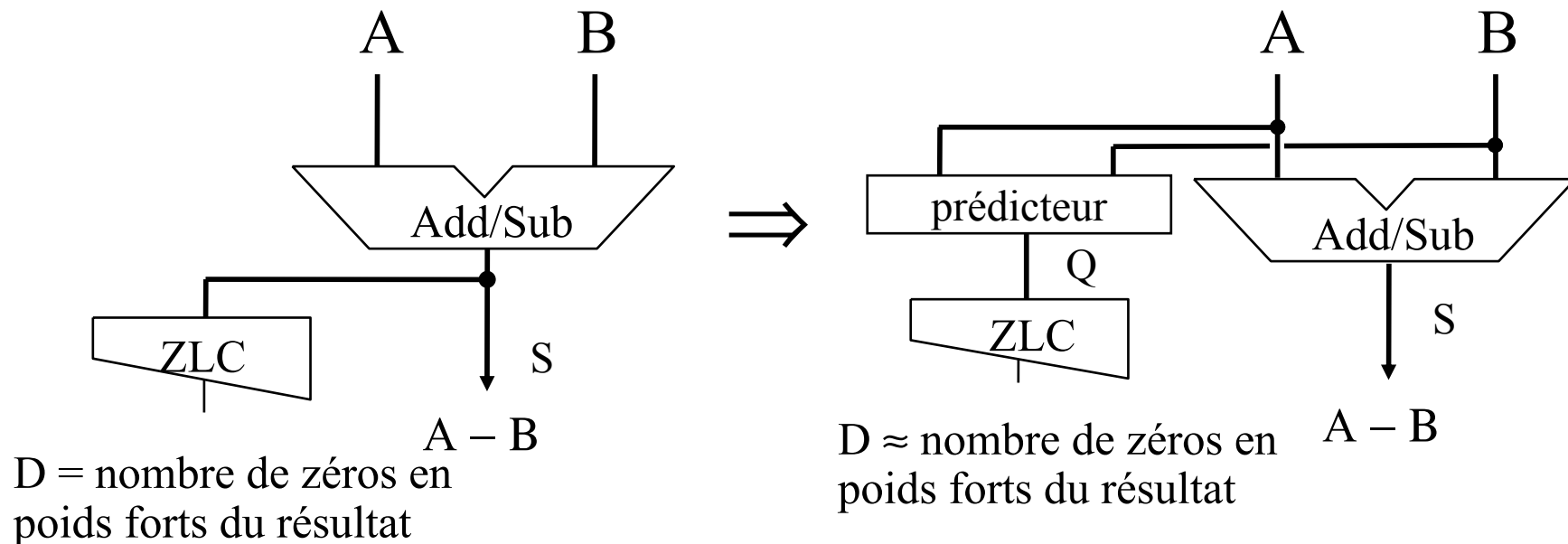


Anticipation du comptage des zéros (1)

Problème: prédire le nombre de zéros en poids forts de $S = A - B$ sans calculer S .

On veut une chaîne Q :

- 1- avec les mêmes zéros poids forts que S
- 2- avec un temps de calcul très court



On n'a pas de prédicteur à la fois précis et rapide

Anticipation du comptage des zéros (2)

On peut prédire certaines
bits s_i de S à partir de p_i, g_i, k_i et $p_{i+1}, g_{i+1}, k_{i+1}$
mais pas tous (p_{i+1} à droite)

k_i	k_{i+1}	$s_i = 0$
k_i	p_{i+1}	$s_i = x$
k_i	g_{i+1}	$s_i = 1$
p_i	k_{i+1}	$s_i = 1$
p_i	p_{i+1}	$s_i = x$
p_i	g_{i+1}	$s_i = 0$
g_i	k_{i+1}	$s_i = 0$
g_i	p_{i+1}	$s_i = x$
g_i	g_{i+1}	$s_i = 1$

On calcule la chaîne
 Q comme ci-dessous
(on choisi des valeurs pour les x)

k_i	k_{i+1}	$q_i = 0$
k_i	p_{i+1}	$q_i = 1$
k_i	g_{i+1}	$q_i = 1$
p_i	k_{i+1}	$q_i = x$
p_i	p_{i+1}	$q_i = 0$
p_i	g_{i+1}	$q_i = 0$
g_i	k_{i+1}	$q_i = 0$
g_i	p_{i+1}	$q_i = 1$
g_i	g_{i+1}	$q_i = 1$

n'appartient pas à
 $p^* g k^*$
on choisi 0

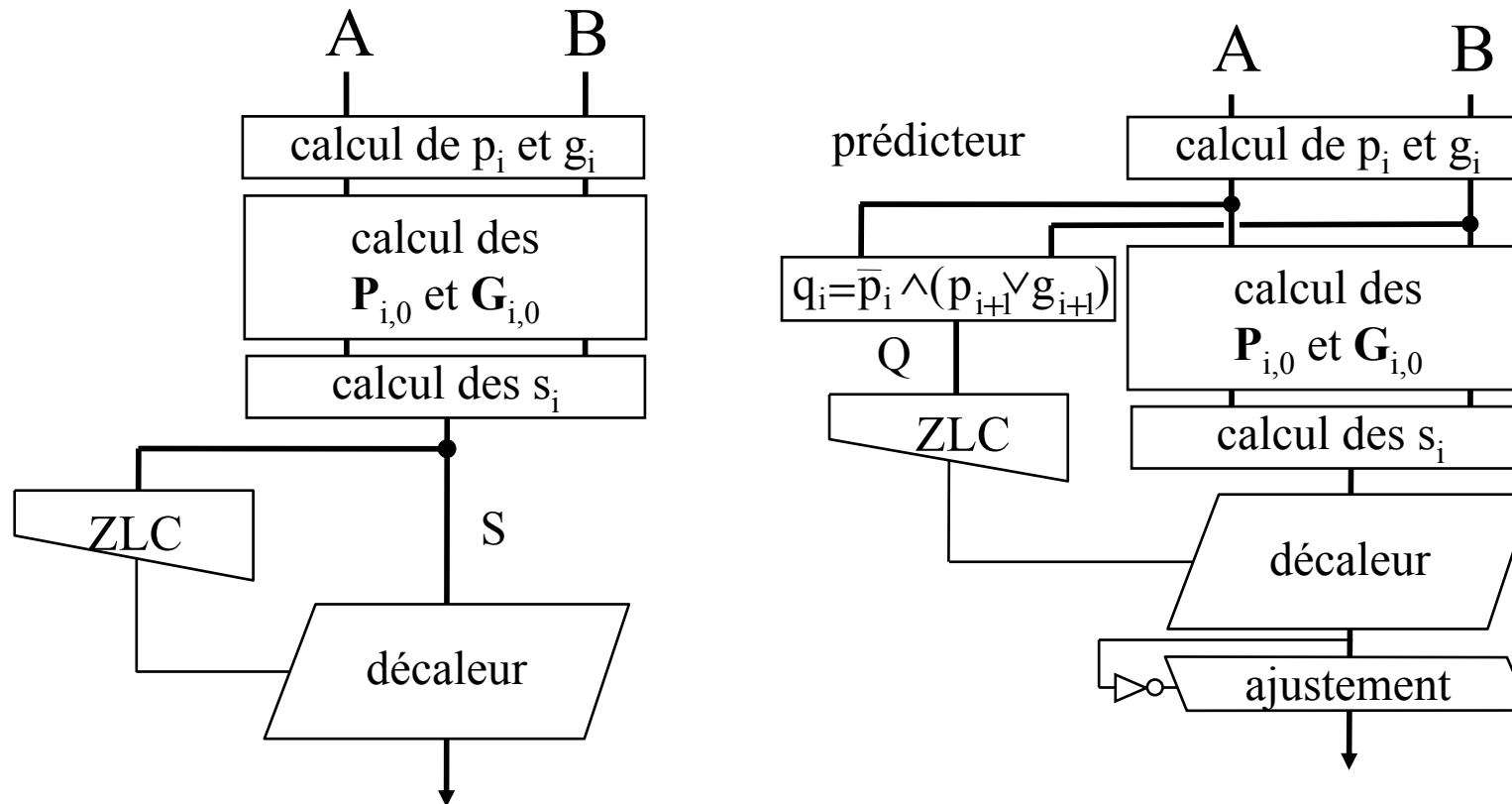
$$q_i = (\overline{p_i} \wedge \overline{k_{i+1}})$$

$$= (\overline{a_i} \wedge \overline{b_i} \vee a_i \wedge b_i)$$

$$\wedge (a_{i+1} \vee b_{i+1})$$

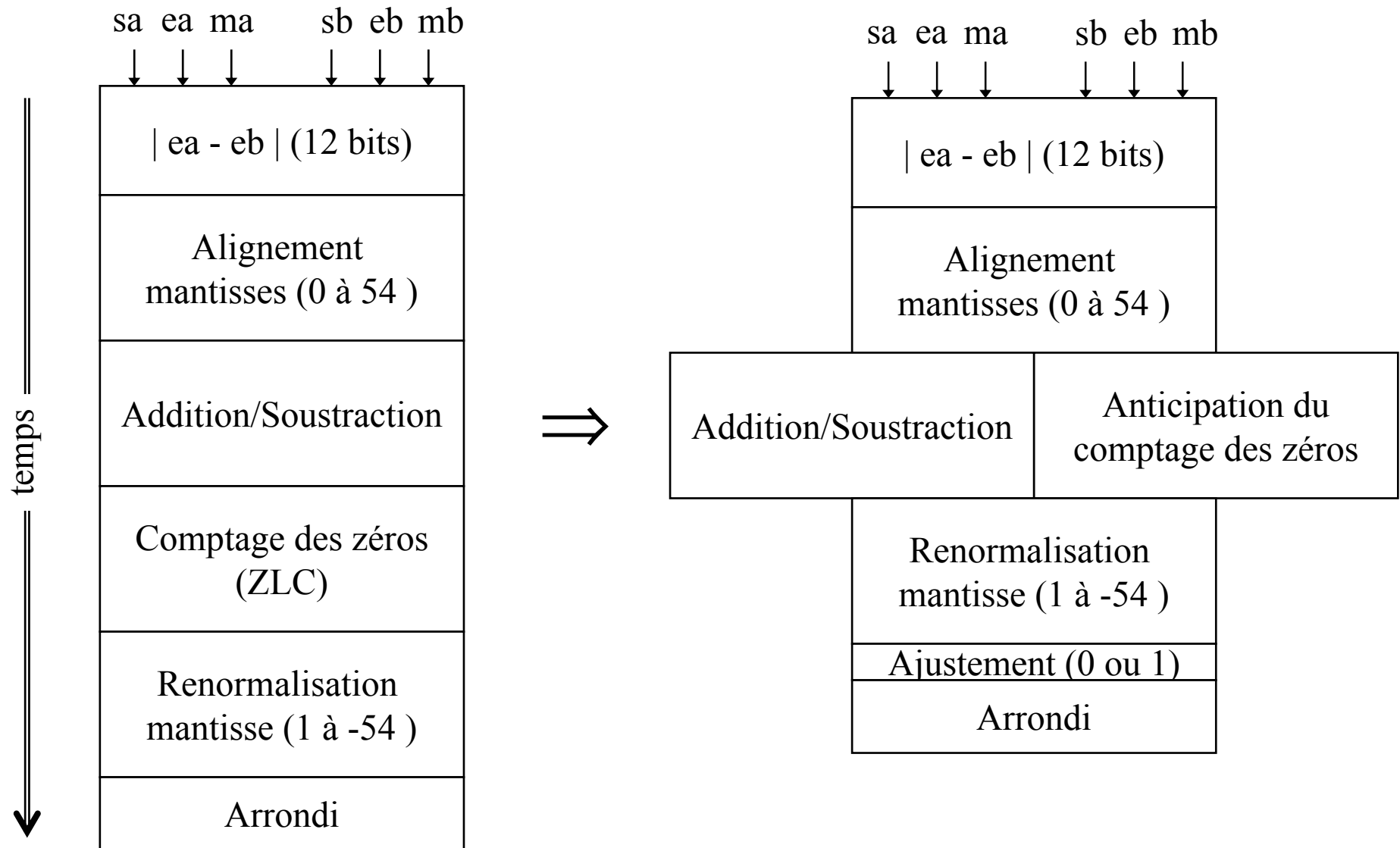
Alors la chaîne $S = A - B$, $S \geq 0$ et la chaîne Q ont le même nombre de bits à zéro dans la chaîne $p^* g k^*$, c'est à dire le même nombre de zéros en poids forts à 1 près.

Anticipation du comptage des zéros (3)



L'imprécision du prédicteur demande un étage d'ajustement
 $ZLD(S) \leq ZLD(S') \leq ZLD(S) + 1$

Anticipation du comptage des zéros (4)



Erreur de prédiction

Une chaîne de propagation de retenue ($\neq p^*$) commence soit par p^*k , soit par p^*g .
Le début de chaîne p^*k produit une retenue sortante à 0.
Le début de chaîne p^*g produit une retenue sortante à 1.

Soit $S = A - B$ avec $A > B$, $S > 0$.
Alors la chaîne de retenues commence par p^*g .
Soit D' la longueur de la sous-chaîne $p^*g k^*$.

Soit D le nombre de zéros en tête de S .
Si la chaîne commence par $p^*g k^*p^* k$ alors $D = D'$
Si la chaîne commence par $p^*g k^*p^* g$ alors $D = D' - 1$

En résumé:
On sait que la chaîne de propagation commence par p^*g
On veut la longueur de la chaîne $p^*g k^*$
On veut savoir si la chaîne commence par $p^*g k^*p^*k$.

Prédiction de l'ajustement (1)

Transcodage

$$\bar{p}_i \bar{k}_{i+1} \Rightarrow q_i$$

$$p_{i-1} k_i \Rightarrow n_i$$

$$\text{autre} \Rightarrow z_i$$

Chaîne à reconnaître

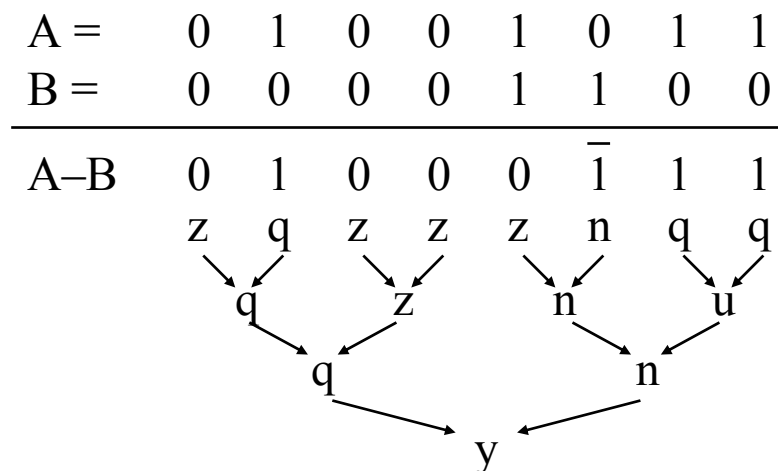
$p^* g k^* p^* k$

transcodée en

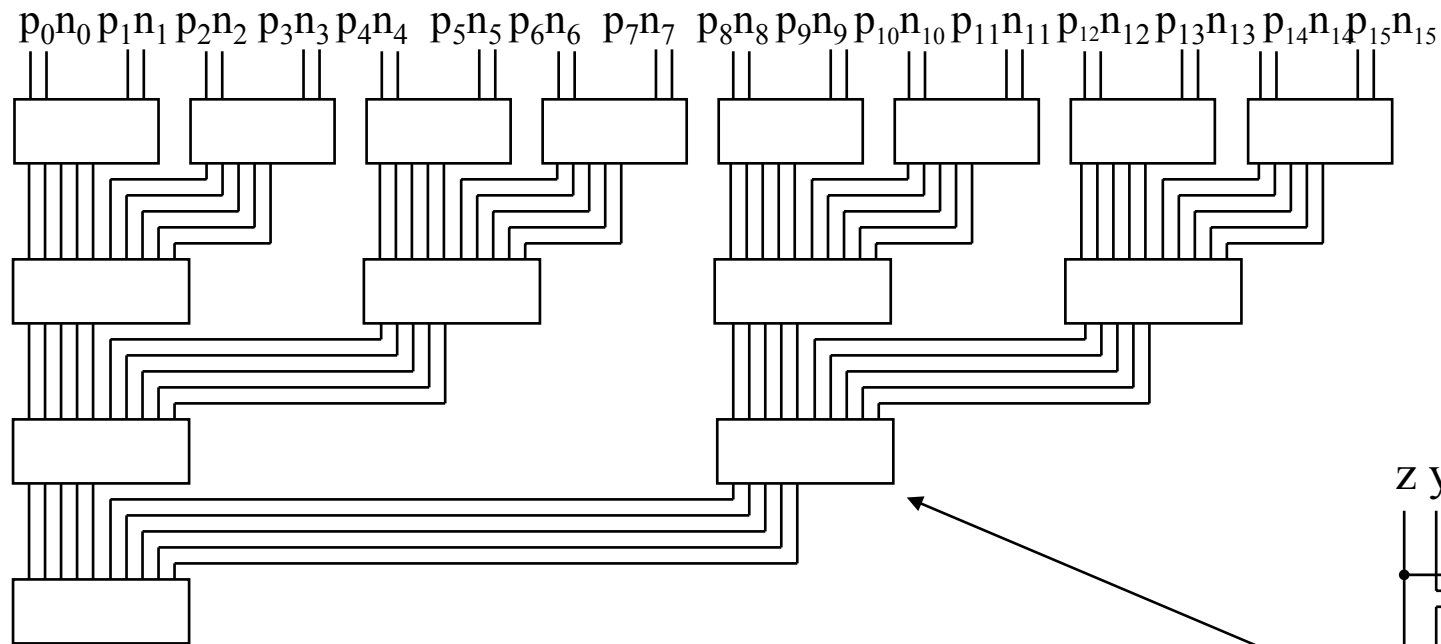
$z^* q z^* n$

Concaténation de chaînes

		chaîne de droite				
chaîne de gauche		z	q	n	y	u
	z	z	q	n	y	u
	q	q	u	y	u	u
	n	n	n	n	n	n
	y	y	y	y	y	y
	u	u	u	u	u	u

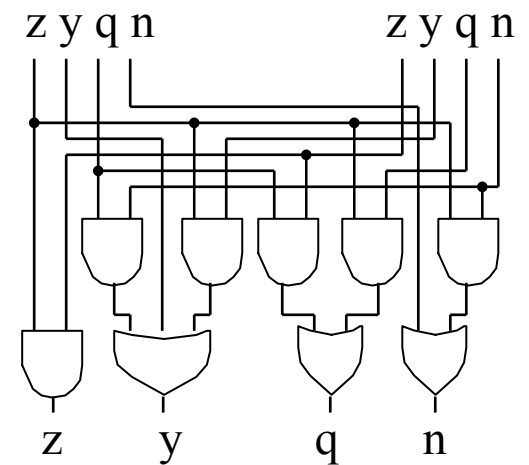


Prédiction de l'ajustement (2)

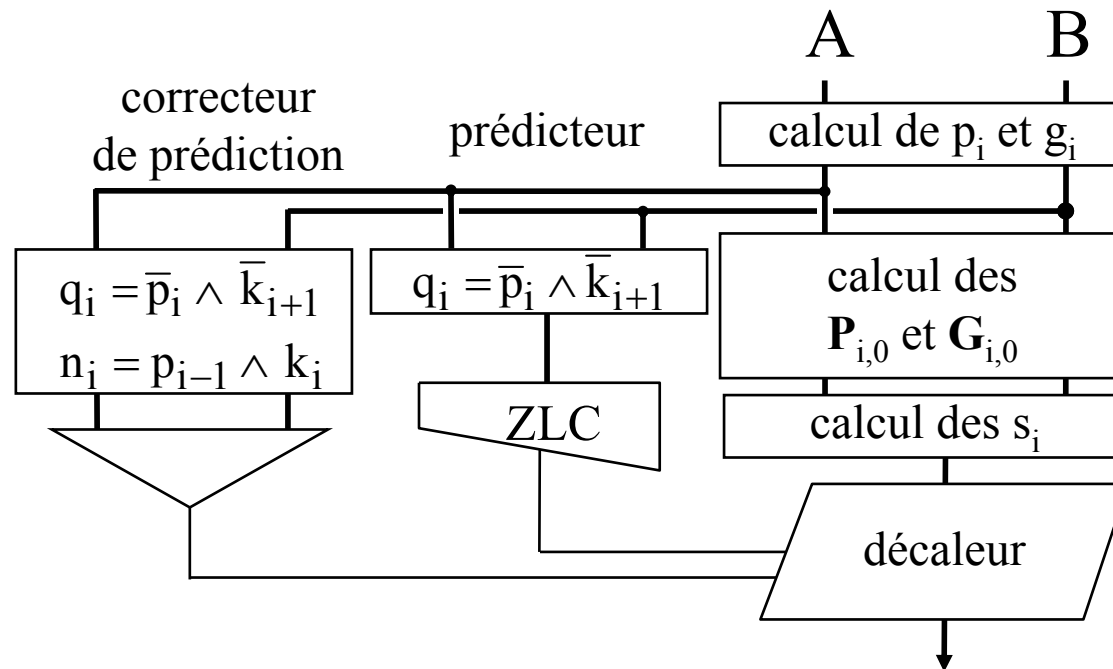


	z	q	n	y	u
z	z	q	n	y	u
q	q	u	y	u	u
n	n	n	n	n	n
y	y	y	y	y	y
u	u	u	u	u	u

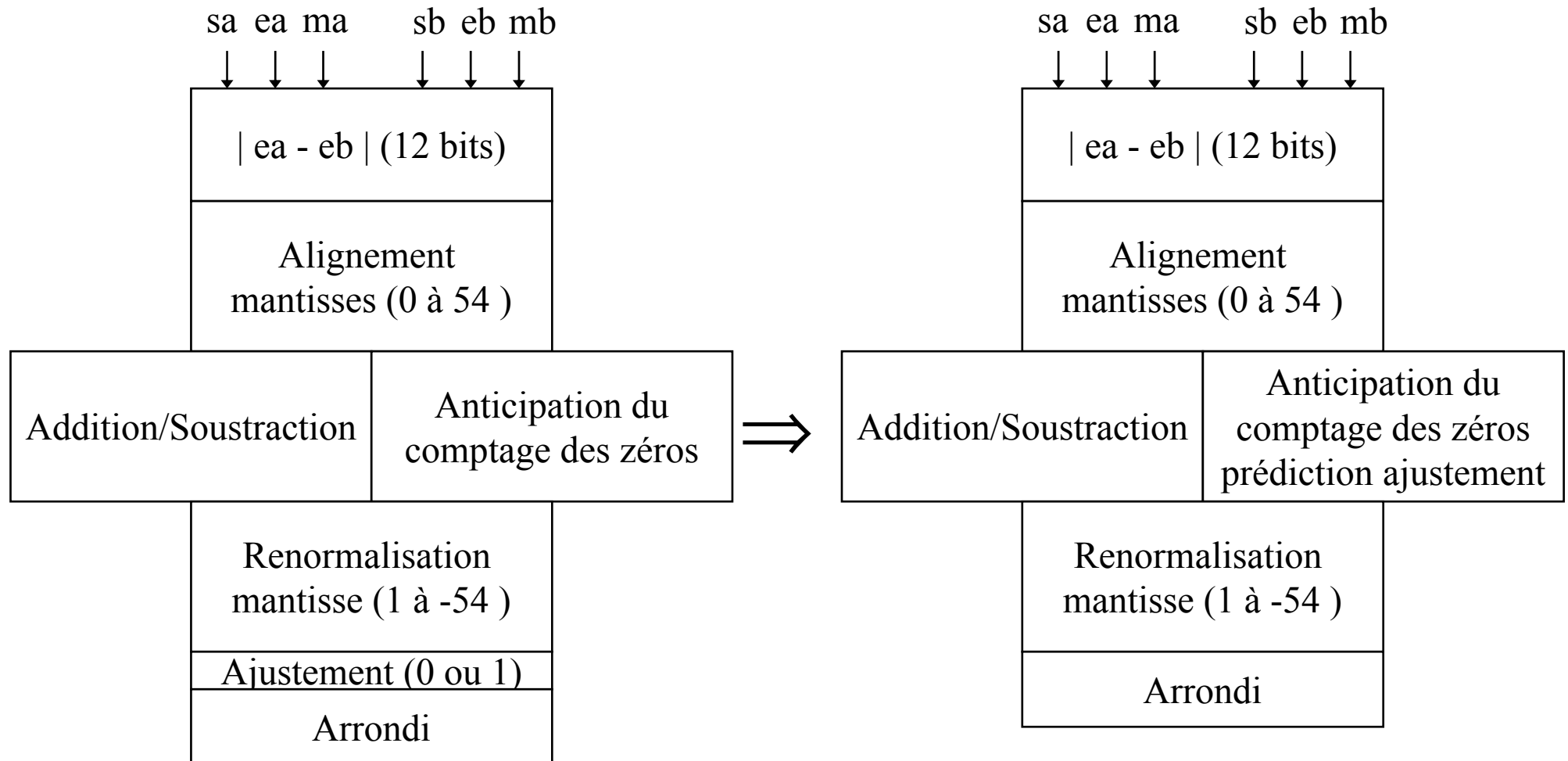
$$\begin{aligned}
 z &= z z \\
 y &= q n \vee y \phi \vee z y \\
 q &= q z \vee z q \\
 n &= n \phi \vee z n
 \end{aligned}$$



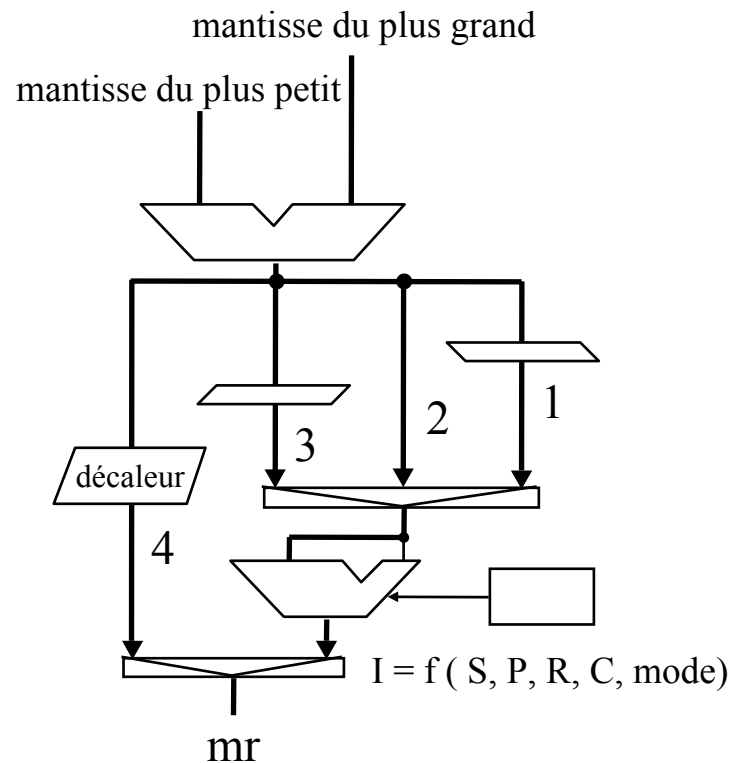
Prédiction de l'ajustement (3)



Prédiction de l'ajustement (4)



Renormalisation et arrondi de la mantisse



Pour être renormalisé, le résultat doit être soit

- 1- décalé à droite de 1 position (addition)
- 2- déjà normalisé (addition/soustraction)
- 3- décalé à gauche de 1 position (soustraction)
- 4- décalé à gauche de 2 positions ou plus (soustraction)

Dans le cas 4, $|e_a - e_b| \leq 1$, donc le décalage introduit à droite 1 zéro au moins.

Seuls les cas 1, 2 et 3 peuvent provoquer une propagation de retenue pour l'arrondi.

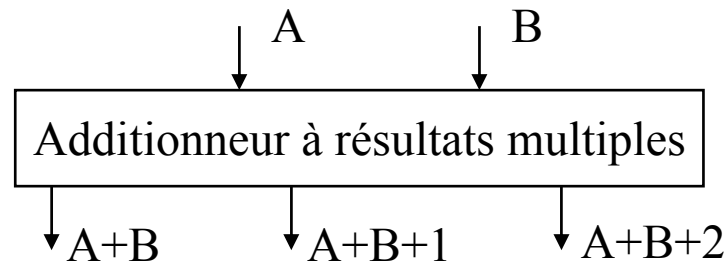
On peut « précalculer » chacun de ces résultats avec un additionneur à résultats multiples

Arrondi spéculatif

On commence par réduire les 3 cas à 2 cas en décalant systématiquement de une position à gauche les soustractions

On précalcule

- 1- la mantisse du résultat,
- 2 -la mantisse du résultat +1 lsb (cas 2)
- 3 -la mantisse du résultat +2 lsb (cas 1)



Si l'arrondi du résultat est de

- 1- sans ajout (troncature)
- 2 -de une position à droite puis ajout
- 3 -de 0 position puis ajout

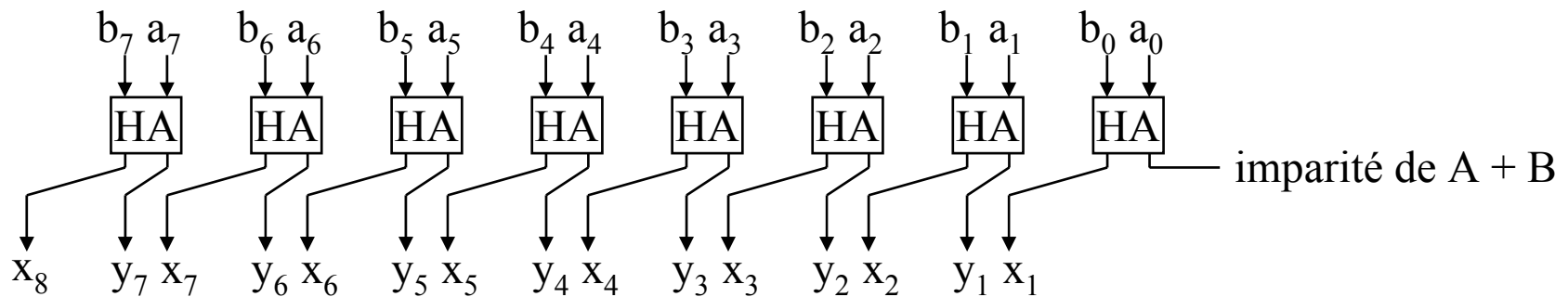
Alors le résultat est précalculé, sinon il n'y a pas besoin d'addition puisqu'il n'y a jamais d'ajout.

Additionneur calculant $A+B$, $A+B+1$, $A+B+2$

On précalcule sans propagation de retenue 2 nombres X et Y tels que:

si $A + B$ est pair **alors** $X + Y = A + B + 1$

si $A + B$ est impair **alors** $X + Y = A + B$

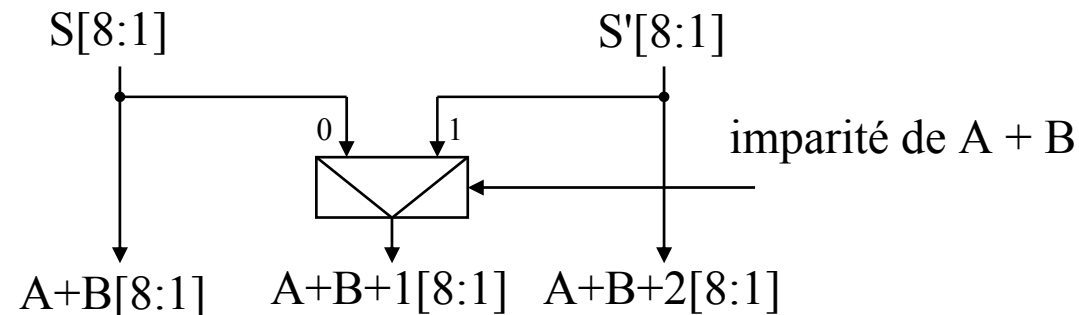


Puis on calcule $S = X + Y$ et $S' = X + Y + 1$ par un additionneur à 2 sorties S et S'

si $A + B$ est pair **alors** $S = A + B + 1$ (impair), $S - 1$ (pair) = $A + B$, $S' = A + B + 2$

si $A + B$ est impair **alors** $S = A + B$, $S' = A + B + 1$ (pair), $S' + 1$ (impair) = $A + B + 2$

Circuit calculant
les 8 bits poids
forts de $A+B$,
 $A+B+1$, $A+B+2$



Addition/Soustraction spéculative (1)

On spécule que $|ea - eb| > 1$ **Eloigné**

Alignement

- 1 - calculer $n = |ea - eb|$ sur 12 bits
(détermine si la spéculation est correcte)
- 2 - décaler la mantisse du plus petit de
 n positions à droite
- 2' - Si $n > 54$ ou $n \leq 1$ alors abandon.

Normalisation

- 3 - Le nombre de position est donné par les 3 premiers bits du résultat avant normalisation
- 4 - La mantisse du résultat est décalé:
soit 1 position à droite (addition)
soit 0 position (addition ou soustraction)
soit 1 position à gauche (soustraction)

On spécule que $|ea - eb| \leq 1$ **Proche**

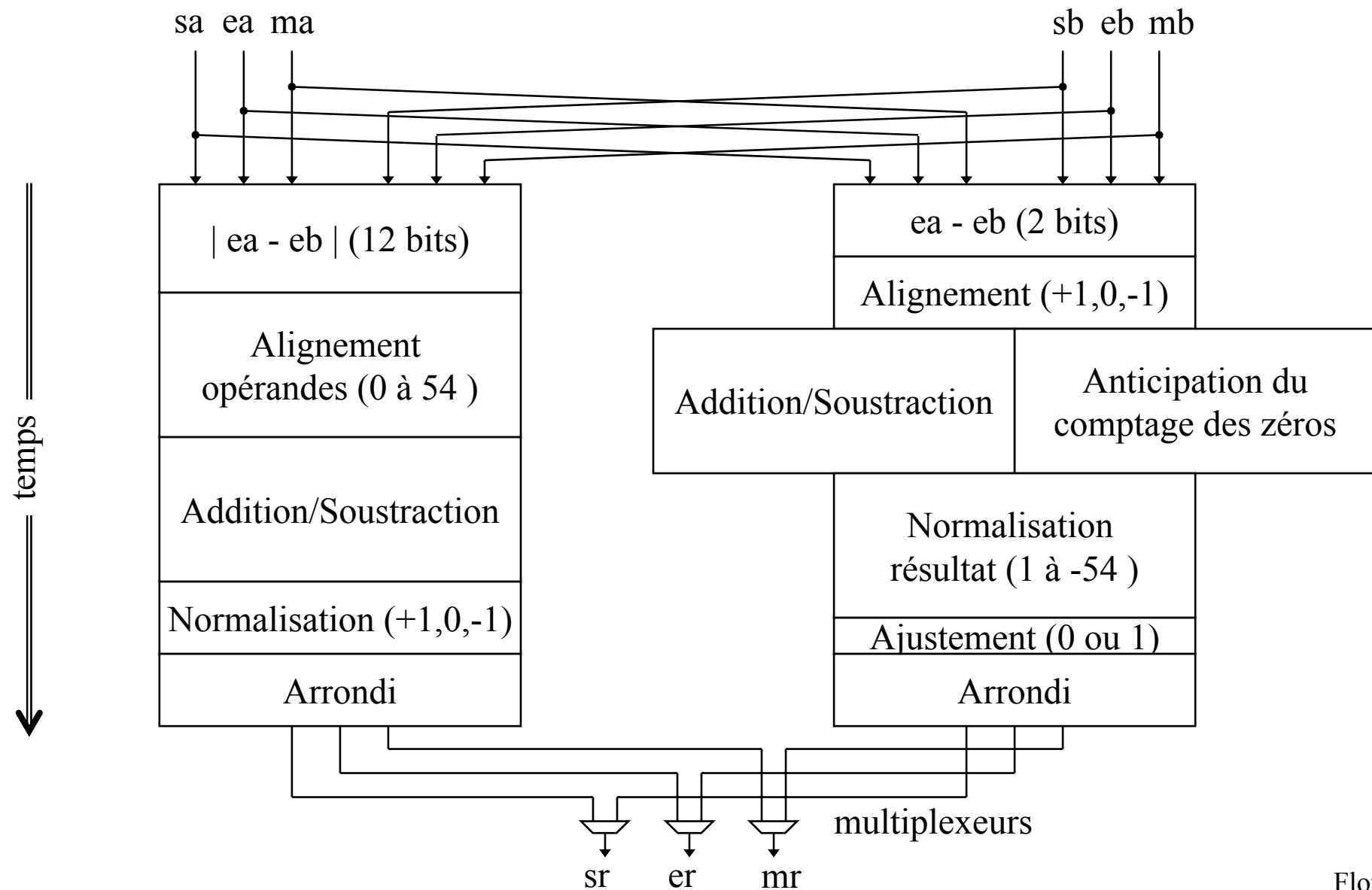
Alignement

- 1 - le nombre de positions est donné par les
2 derniers bits de ea et de eb
- 2 - décaler la mantisse du plus petit de
0 ou 1 position à droite

Normalisation

- 3 - pour une addition, le résultat est décalé
de 0 ou 1 position à droite
- 3' - pour une soustraction,
3'-1 calculer $n =$ nombre de 0 en
poids forts
3'-2 décaler la mantisse du résultat de n
positions à gauche (introduire n zéros)

Addition/Soustraction spéculative (2)



Calcul spéculatif de (ea - eb)

On spécule que $|ea - eb| \leq 1$

On veut calculer $(ea - eb) \in \{-1, 0, 1\}$

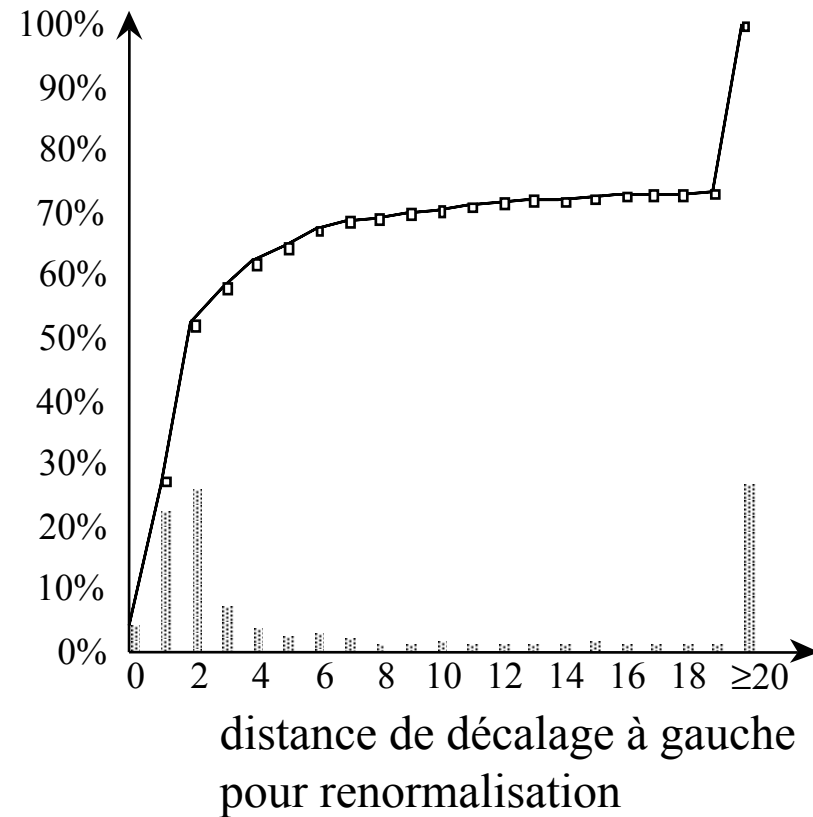
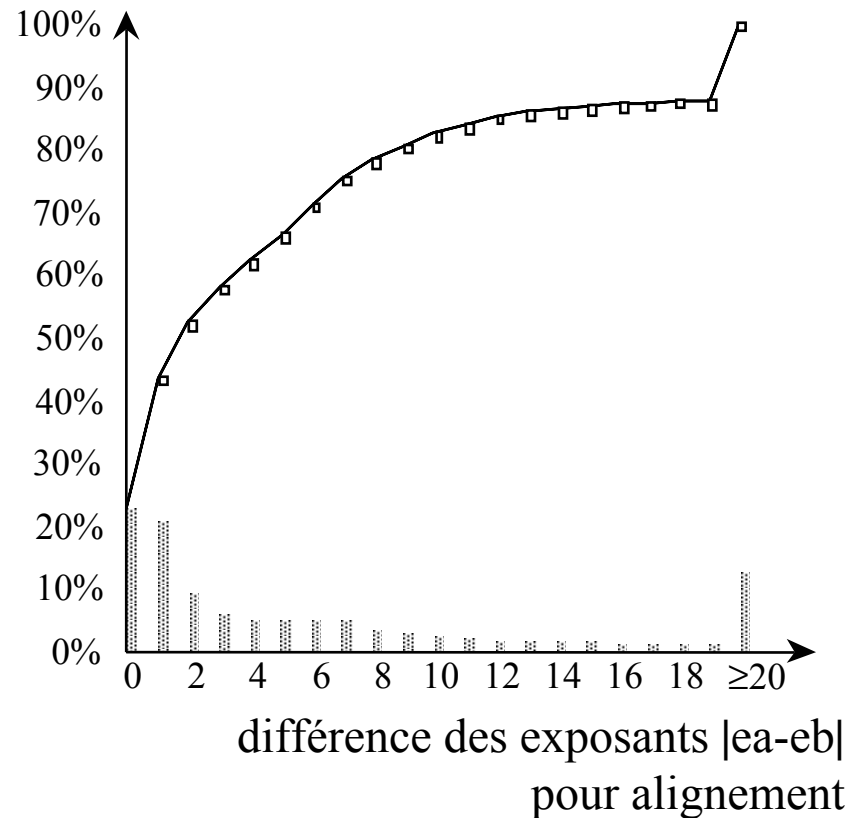
On ne tient compte que des 2 derniers
bits de ea et de eb pour ce calcul

$\Rightarrow$ Le calcul spéculatif de $(ea - eb)$ est rapide

Remarque: dans les cases « $\notin \{-1, 0, 1\}$ », on peut mettre ϕ puisque cette spéculation sera abandonnés.

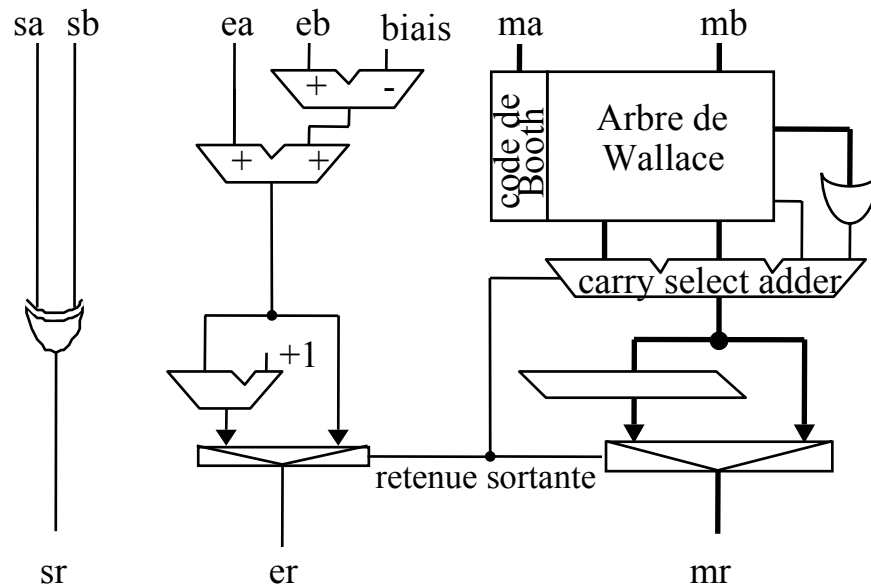
ea	eb	ea - eb
00	00	0
00	01	-1
00	10	$\notin \{-1, 0, 1\}$
00	11	1
01	00	1
01	01	0
01	10	-1
01	11	$\notin \{-1, 0, 1\}$
10	00	$\notin \{-1, 0, 1\}$
10	01	1
10	10	0
10	11	-1
11	00	-1
11	01	$\notin \{-1, 0, 1\}$
11	10	1
11	11	0

Répartition des opérandes



Statistiquement, les deux branches de l'addition spéculative sont prises ($\approx 44\%$, $\approx 56\%$)

Multiplication flottante



Multiplication des mantisses

Les bits poids faible servent pour l'arrondi

Ajout des exposants

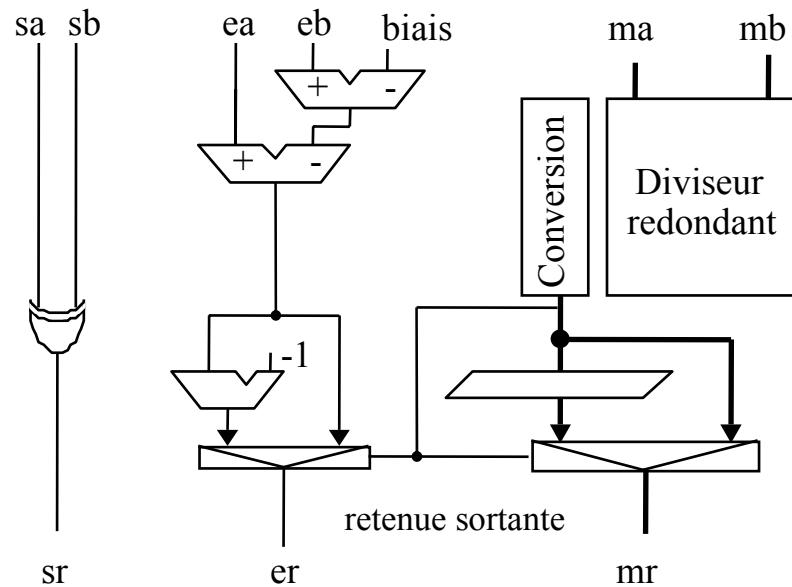
Si la retenue sortante vaut 1 alors décaler le résultat d'une position à droite

L'arbre de Wallace étant difficile à implémenter, on utilise des variantes
On utilise aussi les arbres binaires (dits CS)

Seuls les sorties poids fort du multiplieur, en "carry save" sont additionnés.
On fait le Ou des autres sans les additionner pour l'arrondi.

Le matériel nécessaire au calcul d'un résultat dénormalisé n'est pas inclus dans ce schéma.

Division flottante



Division des mantisses

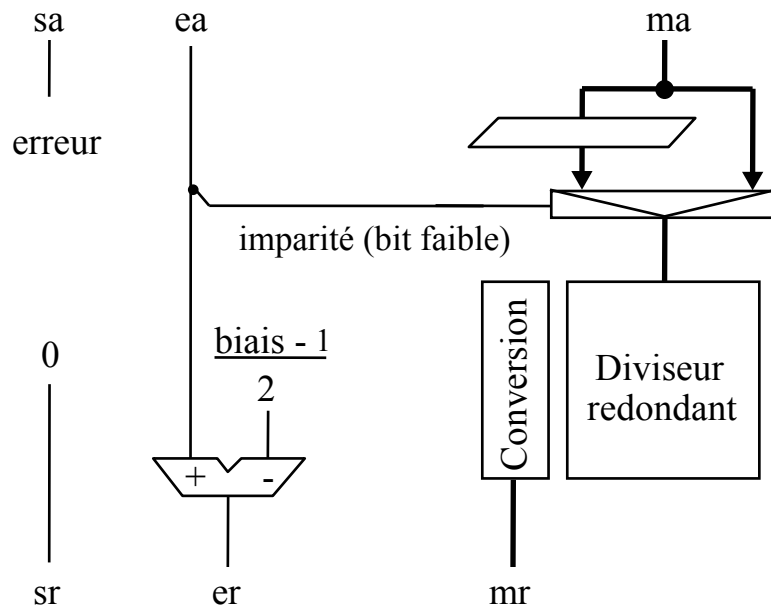
Différence des exposants

*Si le quotient est < 1 alors le décaler
de une position à gauche*

La norme précise que le reste a le même signe que le dividende.

Le matériel nécessaire à la production d'un résultat dénormalisé

Racine Carrée flottante



Le biais est toujours impair

L'exposant d'un carré est pair

Avec des modifications mineures, un diviseur peut calculer la racine carrée

Les racines sont ≥ 0 sauf $\sqrt{-0} = -0$ (Standard IEEE)

Le matériel nécessaire à la production d'un résultat dénormalisé n'est pas inclus dans ce schéma.

Fonctions élémentaires flottante

Les fonctions élémentaires sont transcendantes

Il n'est donc pas possible de prédire le nombre de bits à calculer pour un arrondi au plus près pour une mantisse de taille fixée

En conséquence la norme IEEE 754 ne prévoit rien pour le calcul des fonctions élémentaires

Conclusion

Les principales techniques d'accélération du calcul en virgule flottantes sont:

- Parallélisme (vu)

- Anticipation (vu)

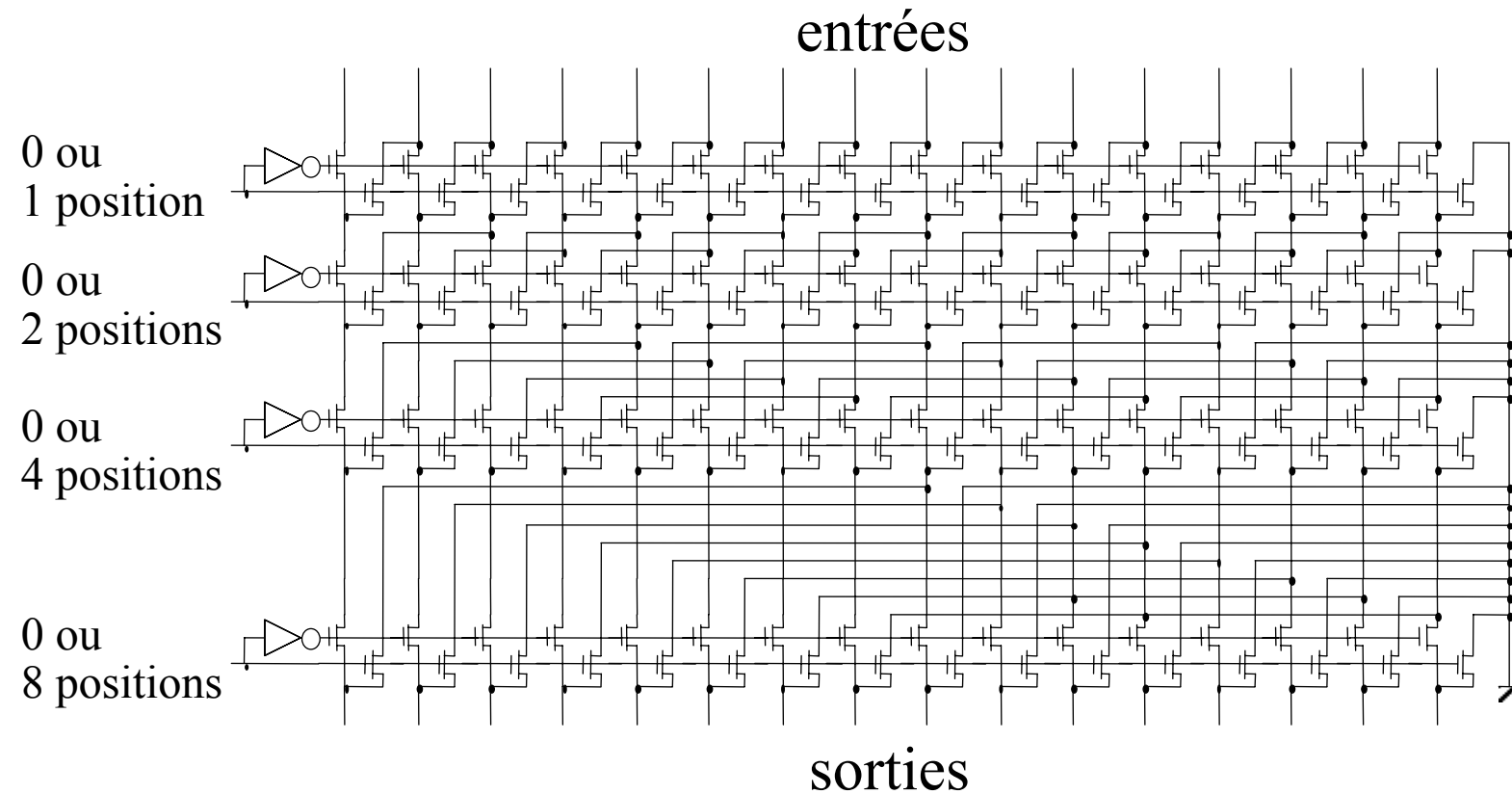
- Exécution spéculative (spécule sur la différence des exposants) (vu)

- Pipe-line (non vu)

- Réinjection en redondant (retenue non propagée) (non vu)

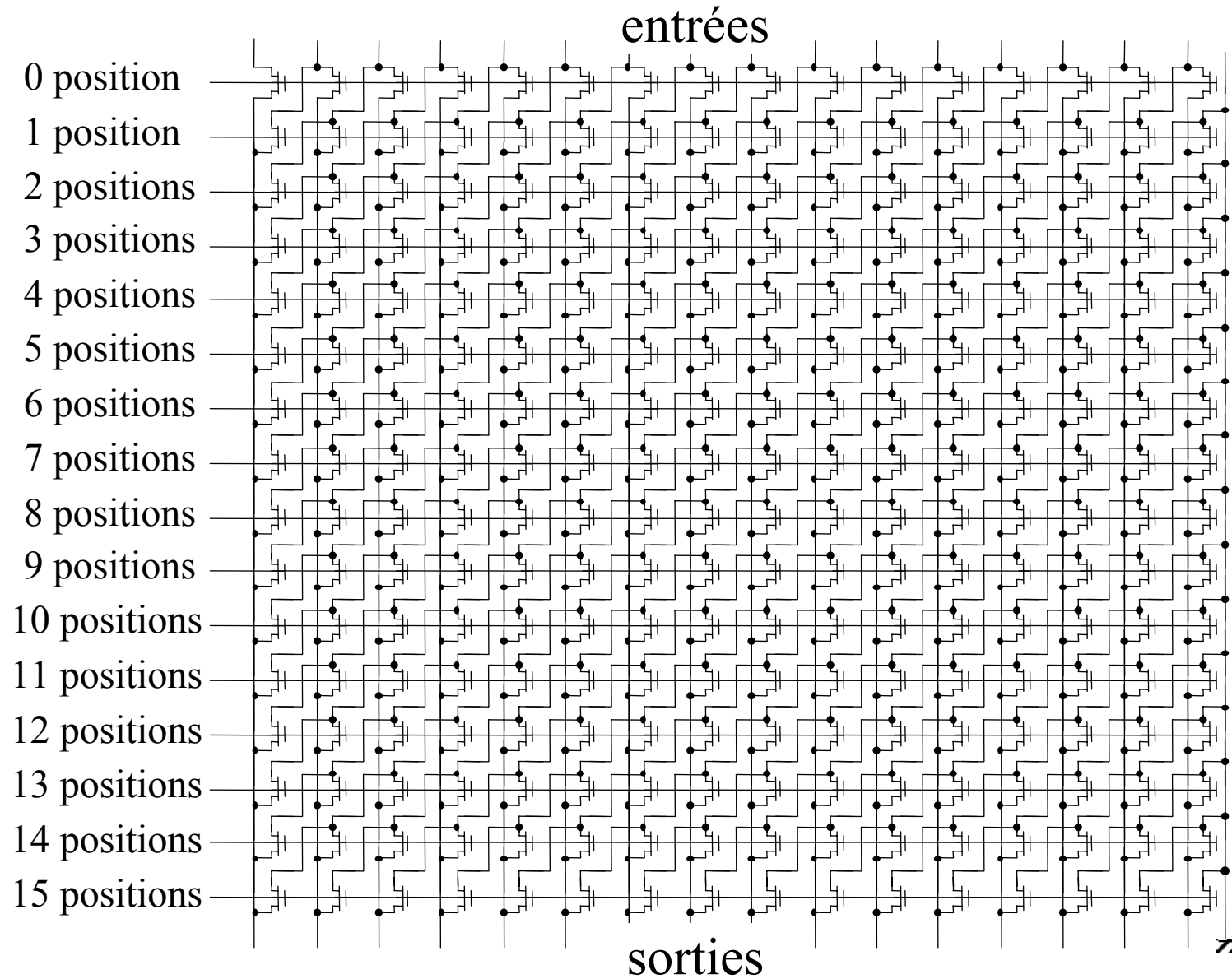
- Autosynchrone (sans horloge) (non vu)

Décaleur logarithmique



$$s_i := \text{si } i+k \geq n \text{ alors } e_{i+k} \text{ sinon } 0$$

Décaleur en barillet

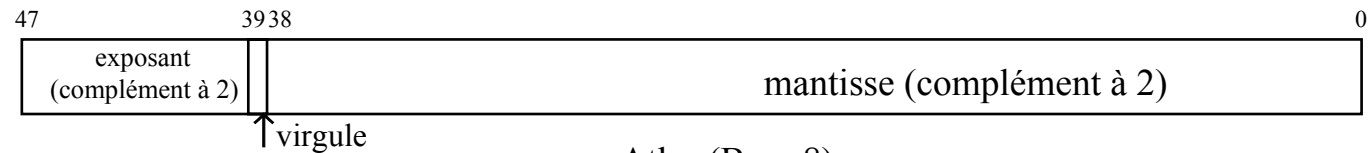


Autres formats virgule flottante

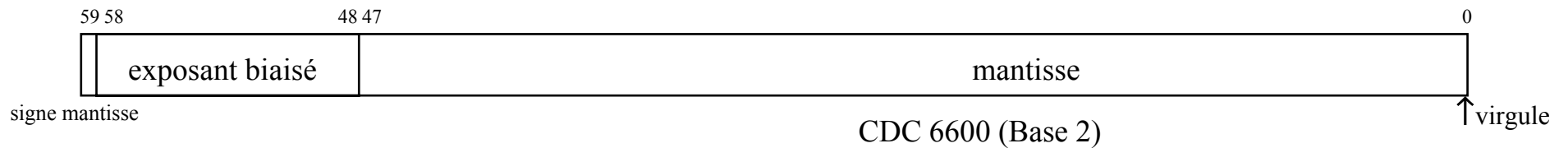
63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0



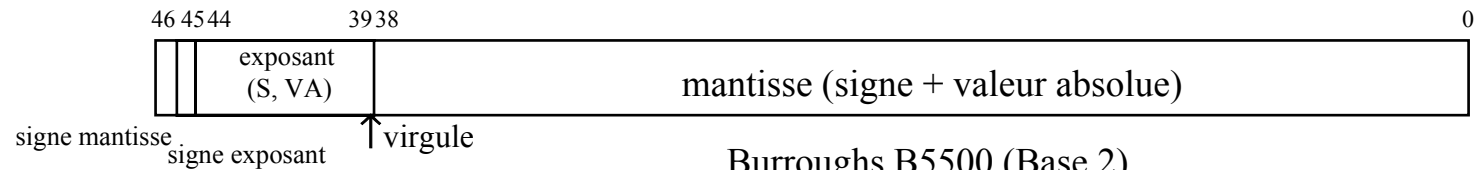
MIL-STD-1750A (Base 2)



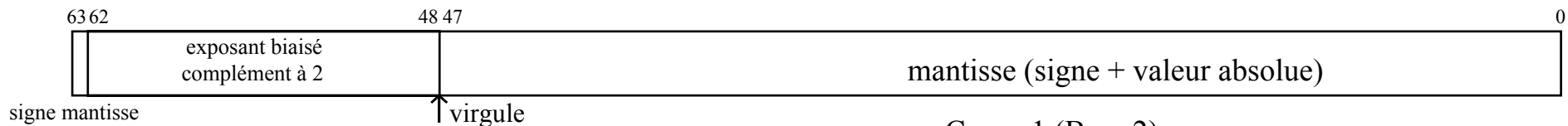
Atlas (Base 8)



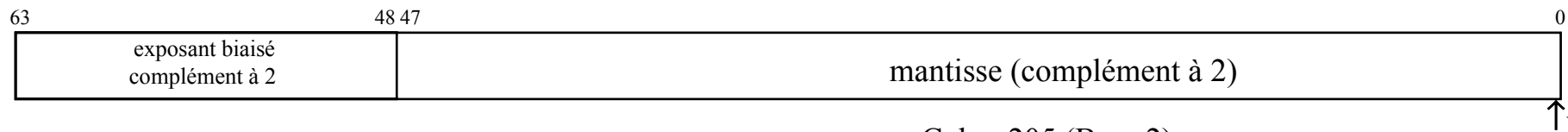
CDC 6600 (Base 2)



Burroughs B5500 (Base 2)



Cray - 1 (Base 2)



Cyber 205 (Base 2)

Flottant 210