


Division Matérielle



Alain GUYOT

Concurrent Integrated Systems
TIMA

 (33) 04 76 57 46 16

 Alain.Guyot@imag.fr

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

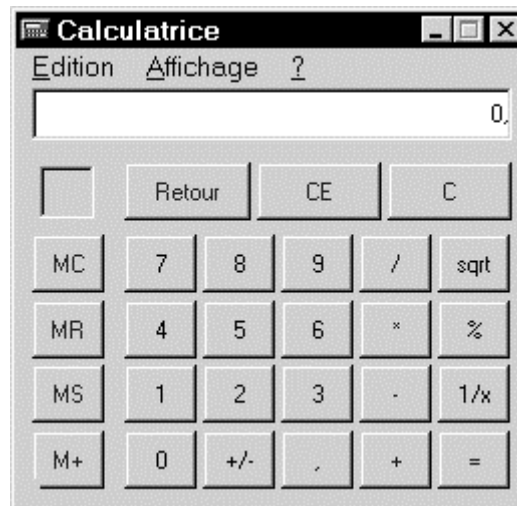
Techniques de l'Informatique et de la Microélectronique
pour l'Architecture. Unité associée au C.N.R.S. n° B0706

But

Réaliser des diviseurs combinatoires rapides

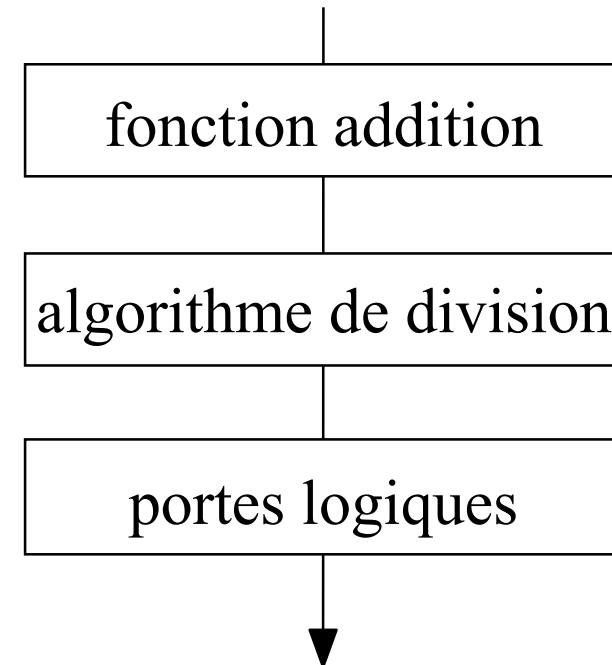
Problème

- Propagation de la retenue



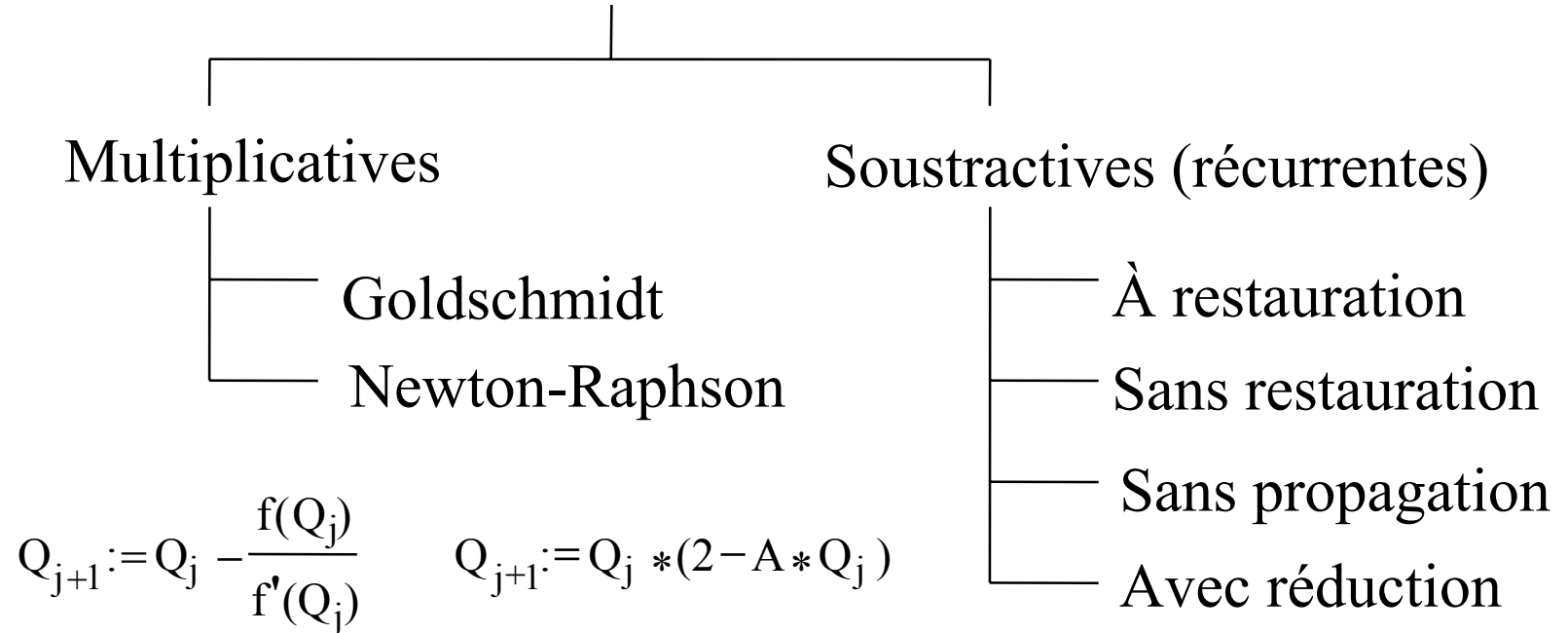
Moyen

Utiliser des additionneurs sans propagation de retenue



Généralités / Plan

Algorithmes de Division



La division de deux entier n'est en général pas un entier, en conséquence on introduit les rationnels (virgule fixe)

$$A = \sum_{i=0}^{n-1} a_i 2^{-i} = a_0, a_1 a_2 a_3 a_4 \dots a_{n-2} a_{n-1} \quad -A = \overline{A} + 2^{-n+1}$$

Division récurrente: principes

On veut calculer $Q = \frac{A}{D}$.

On va construire une suite $Q_0, Q_1, Q_2, \dots, Q_n$ et une suite $R_0, R_1, R_2, \dots, R_n$ telles que l'invariant $A = Q_j * D + R_j$ soit respecté $\forall j$.

La récurrence est :

$$Q_{j+1} = Q_j + q_{j+1} * 2^{-j-1}$$
$$R_{j+1} = R_j - q_{j+1} * D * 2^{-j-1}$$

avec comme état initial:

$$Q_0 = 0$$
$$R_0 = A$$

Quand on s'arrête, on a $Q_n = \sum_{i=0}^n q_i * 2^{-i}$

On impose que le choix des q_j soit tel que $R_j \rightarrow 0$ quand $j \rightarrow \infty$.

On aura donc une approximation Q_n de Q telle que $Q_n = \frac{A - R_n}{D}$ avec R_n petit.

Comme la valeur de Q est bornée par l'implémentation ($|Q| < 2$),
il faut que $-2 * D < A < 2 * D$ (*D doit être suffisamment grand*)

Exemple de division récurrente en décimal

On veut calculer $Q = \frac{22}{7}$.

Calcul des restes	Valeurs	Calcul des quotients	Valeurs
$R_0 := 22$	22	$Q_0 := 0$	
$R_1 := R_0 - 3 * D$	01,0	$Q_1 := Q_0 + 3$	3
$R_2 := R_1 - 0,1 * D$	00,30	$Q_2 := Q_1 + 0,1$	3,1
$R_3 := R_2 - 0,04 * D$	00,020	$Q_3 := Q_2 + 0,04$	3,14
$R_4 := R_3 - 0,002 * D$	00,0060	$Q_4 := Q_3 + 0,002$	3,142
$R_5 := R_4 - 0,0008 * D$	00,00040	$Q_5 := Q_4 + 0,0008$	3,1428
$R_6 := R_5 - 0,00005 * D$	00,000050	$Q_6 := Q_5 + 0,00005$	3,14285

On vérifie que

$22 - 0,3$	$= 21,7$	$= 7 * 3,1$
$22 - 0,02$	$= 21,98$	$= 7 * 3,14$
$22 - 0,006$	$= 21,994$	$= 7 * 3,142$
$22 - 0,0004$	$= 21,9996$	$= 7 * 3,1428$
$22 - 0,00005$	$= 21,99995$	$= 7 * 3,14285$

Exemple de division récurrente en binaire

On veut calculer $Q = \frac{10110}{111}$.

L'algorithme de division en base 2 est une transposition de l'algorithme en base 10.

Calcul des restes	Valeurs reste	Calcul des quotients	Valeurs du quotient	Valeurs du quotient
$R_0 := 10110$	10110	$Q_0 := 0$		
$R_1 := R_0 - 10 * D$	01000	$Q_1 := Q_0 + 10$	10	2
$R_2 := R_1 - 1 * D$	00001	$Q_2 := Q_1 + 1$	11	3
$R_3 := R_2 - 0,0 * D$	00001,0	$Q_3 := Q_2 + 0,0$	11,0	3
$R_4 := R_3 - 0,00 * D$	00001,00	$Q_4 := Q_3 + 0,00$	11,00	3
$R_5 := R_4 - 0,001 * D$	00000,001	$Q_5 := Q_4 + 0,001$	11,001	3,125
$R_6 := R_5 - 0,0000 * D$	00000,0010	$Q_6 := Q_5 + 0,0000$	11,0010	3,125
$R_7 := R_6 - 0,00000 * D$	00000,00100	$Q_7 := Q_6 + 0,00000$	11,00100	3,125
$R_8 := R_7 - 0,000001 * D$	00000,000001	$Q_8 := Q_7 + 0,000001$	11,001001	3,140625

3,142 578125

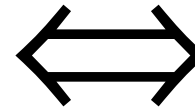
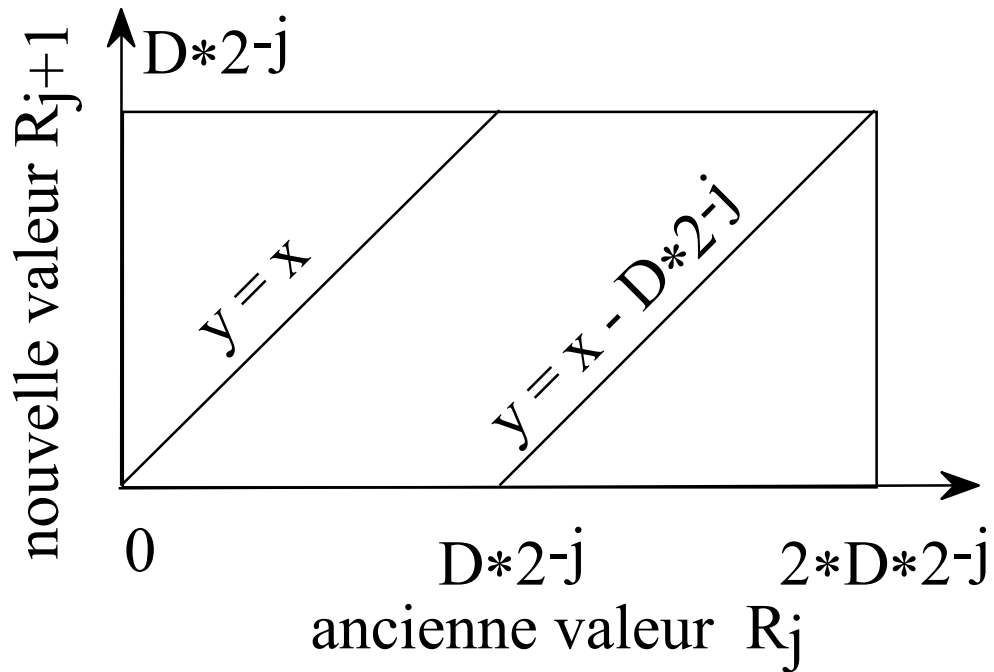
3,1428 22265

3,14285 2783

Diagramme de « Robertson »

(diviseur naïf ou à restauration)

Invariant: $0 \leq R_j \leq 2 * D * 2^{-j}$



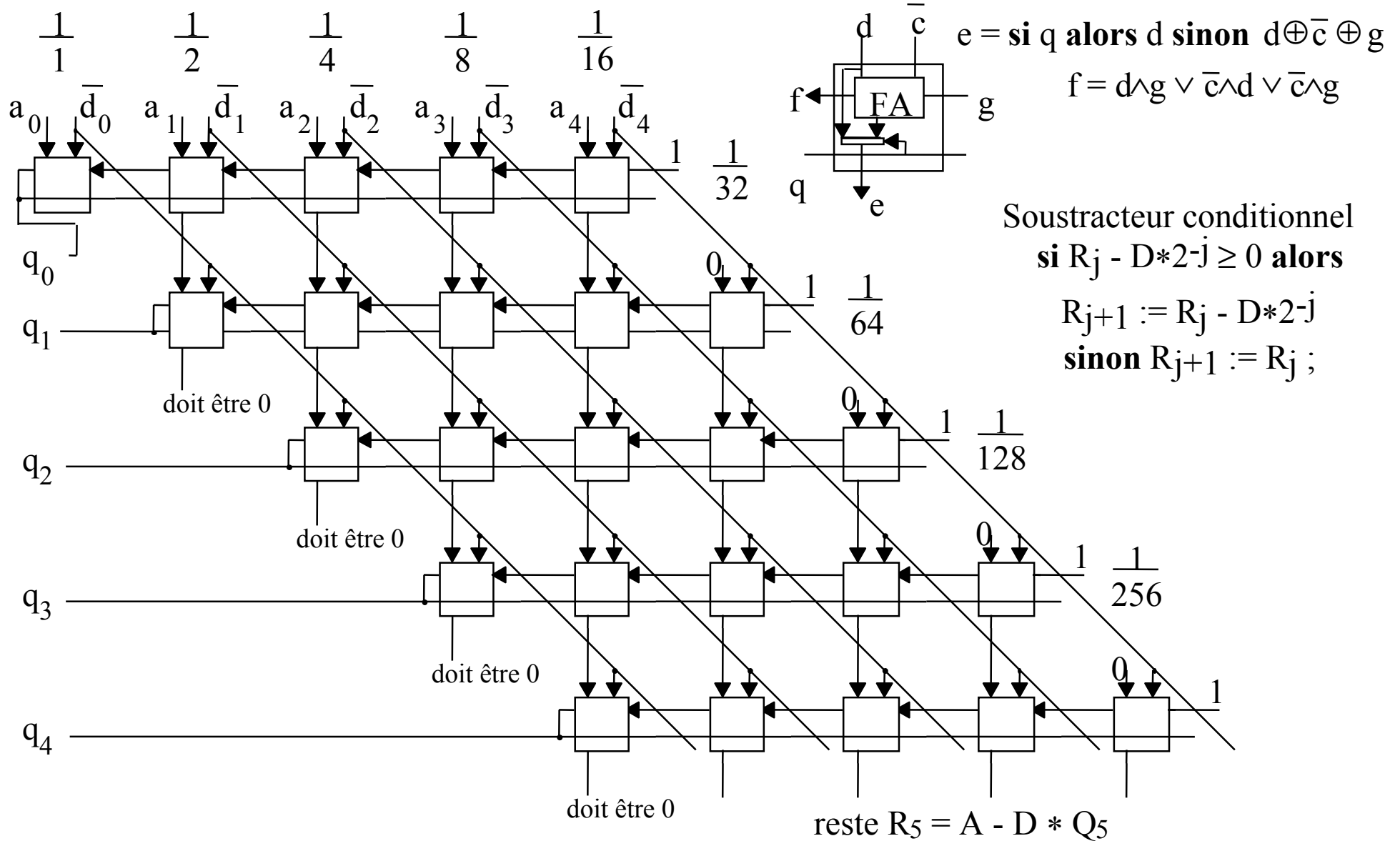
si $R_j \geq D * 2^{-j}$ **alors**

$R_{j+1} := R_j - D * 2^{-j}$

sinon $R_{j+1} := R_j$;

Récurrance

Diviseur naïf (à restauration)

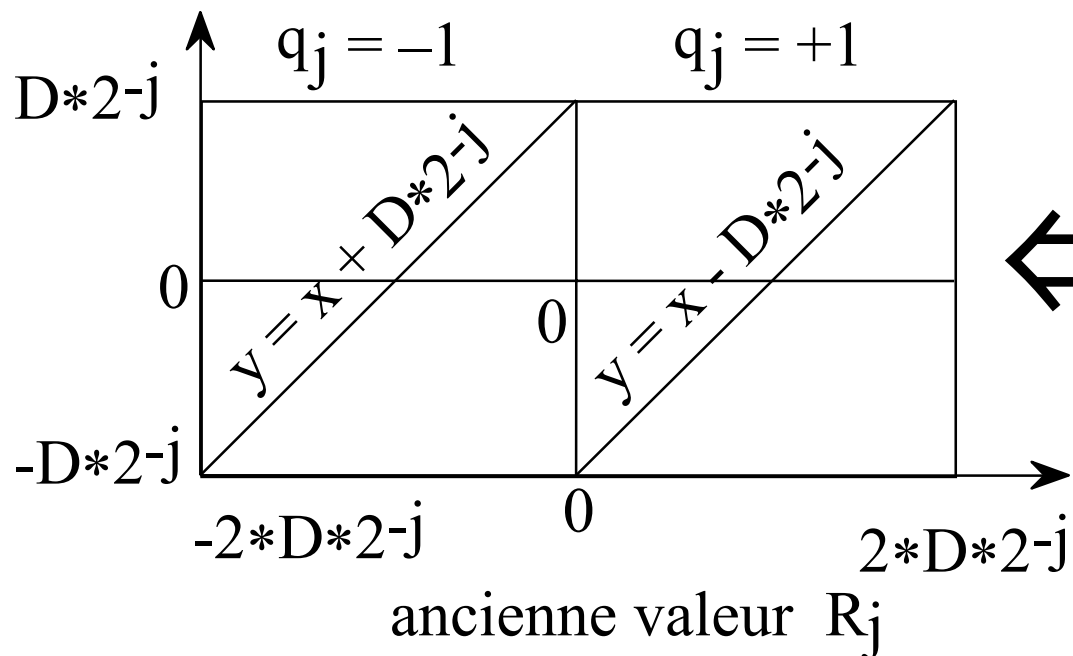


Exemple de division récurrente sans restauration

Calcul des restes	Valeurs du reste (signé)		Calcul des quotients	Valeurs
$R_0 := 10110$	10110	22	$Q_0 := 0$	
$R_1 := R_0 - 10 * D$	$\bar{0}1000$	8	$Q_1 := Q_0 + 10$	10
$R_2 := R_1 - 1 * D$	$0\bar{0}001$	1	$Q_2 := Q_1 + 1$	11
$R_3 := R_2 - 0,1 * D$	$00\bar{1}01,1$	-2,5	$Q_3 := Q_2 + 0,1$	11,1
$R_4 := R_3 + 0,01 * D$	$000\bar{1}1,01$	-0,75	$Q_4 := Q_3 - 0,01$	11,01
$R_5 := R_4 + 0,001 * D$	$00000\bar{,}001$	0,125	$Q_5 := Q_4 - 0,001$	11,001
$R_6 := R_5 - 0,0001 * D$	$00000,\bar{1}011$	-0,3125	$Q_6 := Q_5 + 0,0001$	11,0011
$R_7 := R_6 + 0,00001 * D$	$00000,0\bar{1}101$	-0,09375	$Q_7 := Q_6 - 0,00001$	11,00101
$R_8 := R_7 + 0,000001 * D$	$00000,00\bar{0}001$	0,015625	$Q_8 := Q_7 - 0,000001$	11,001001

Division sans restauration

Invariant: $-2 \cdot D \cdot 2^{-j} \leq R_j \leq 2 \cdot D \cdot 2^{-j}$



si $R_j \geq 0$ alors

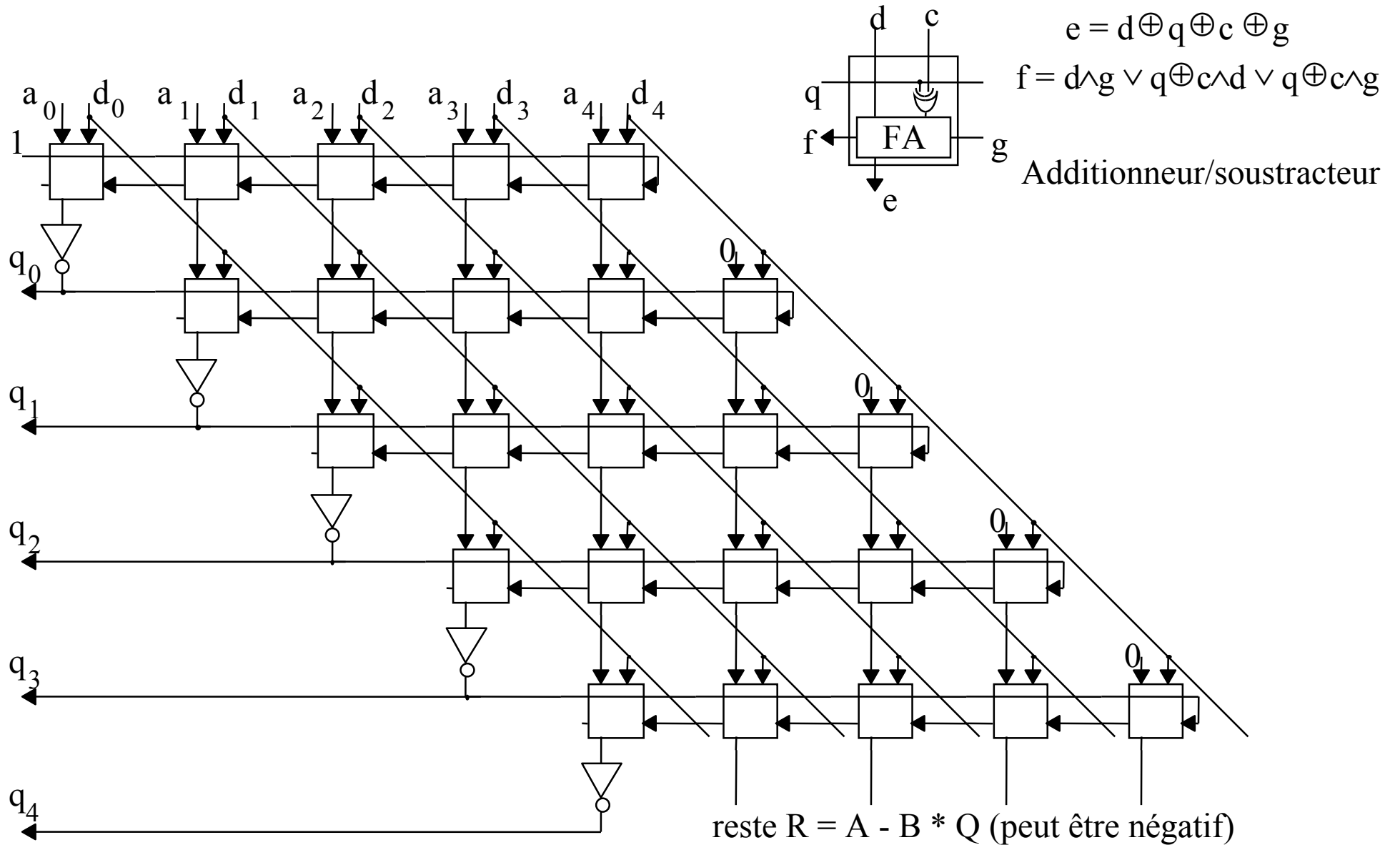
$$R_{j+1} := R_j - D \cdot 2^{-j}$$

sinon

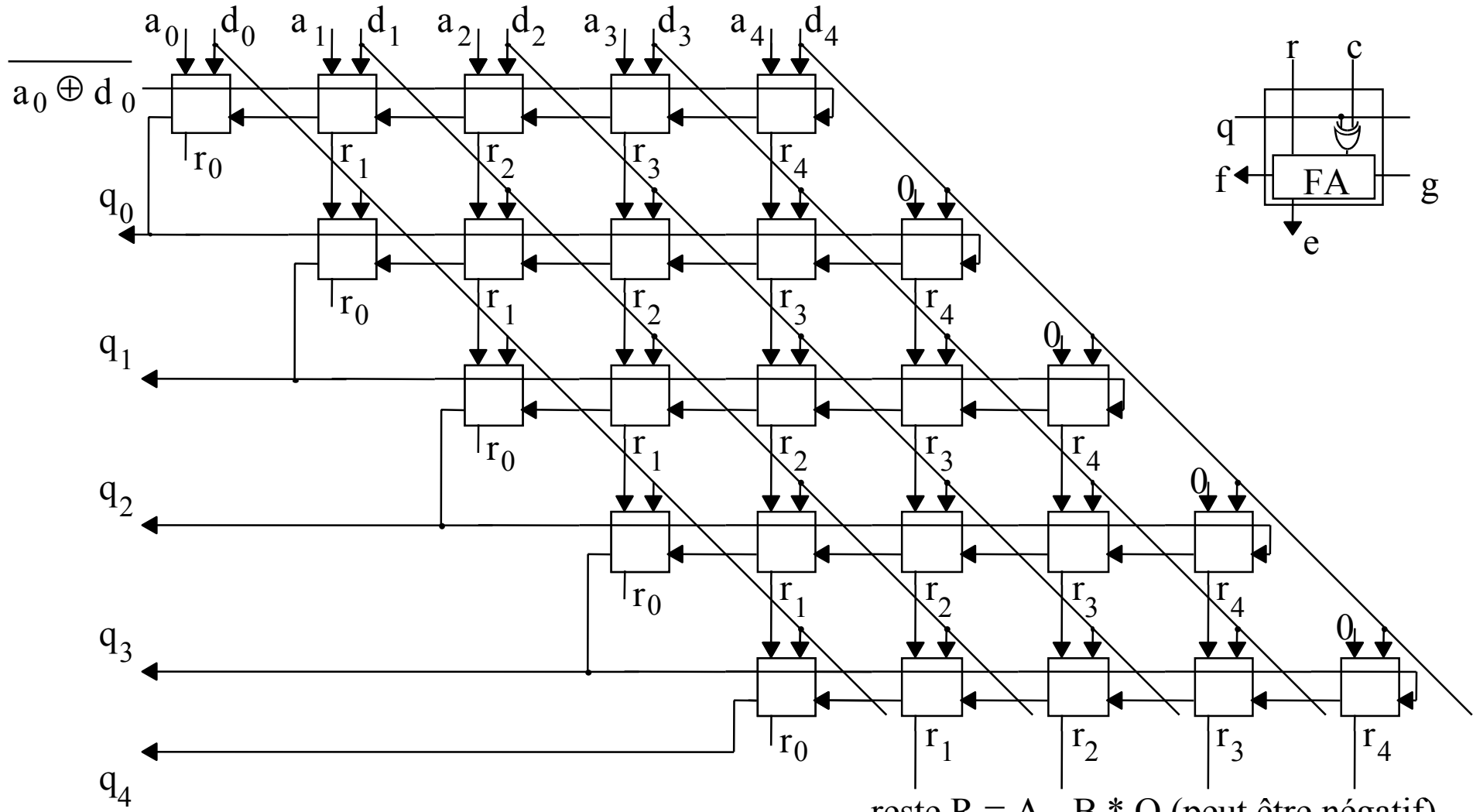
$$R_{j+1} := R_j + D \cdot 2^{-j};$$

Récurrence

Diviseur sans restauration (2)



Division signée



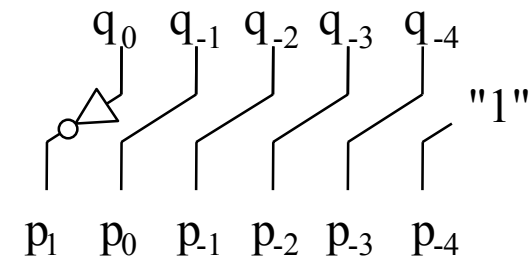
reste $R = A - B * Q$ (peut être négatif)

Conversion du quotient

Le quotient Q s'écrit $Q = \sum_{i=0}^{n-1} q_i * 2^{-i}$ avec $q_i \in \{-1,+1\}$. Pour le convertir on le réécrit:

$$\begin{aligned}
 Q &= \sum_{i=0}^{n-1} q_i * 2^{-i} = \sum_{i=0}^{n-1} q_i * 2^{-i} + \sum_{i=0}^{n-1} 2^{-i} - \sum_{i=0}^{n-1} 2^{-i} \\
 &= \sum_{i=0}^{n-1} (q_i + 1) * 2^{-i} - \sum_{i=0}^{n-1} 2^{-i} \\
 &= 2 * \sum_{i=0}^{n-1} p_i * 2^{-i} - 2 + 2^{-n} \\
 &= 2 * (p_0 - 1 + \sum_{i=1}^{n-1} p_i * 2^{-i} + 2^{-n-1}) \\
 &= 2 * (\underbrace{-\overline{p_0} * 2^0 + \sum_{i=1}^{n-1} p_i * 2^{-i}}_{p_i \in \{0,1\}} + 2^{-n-1})
 \end{aligned}$$

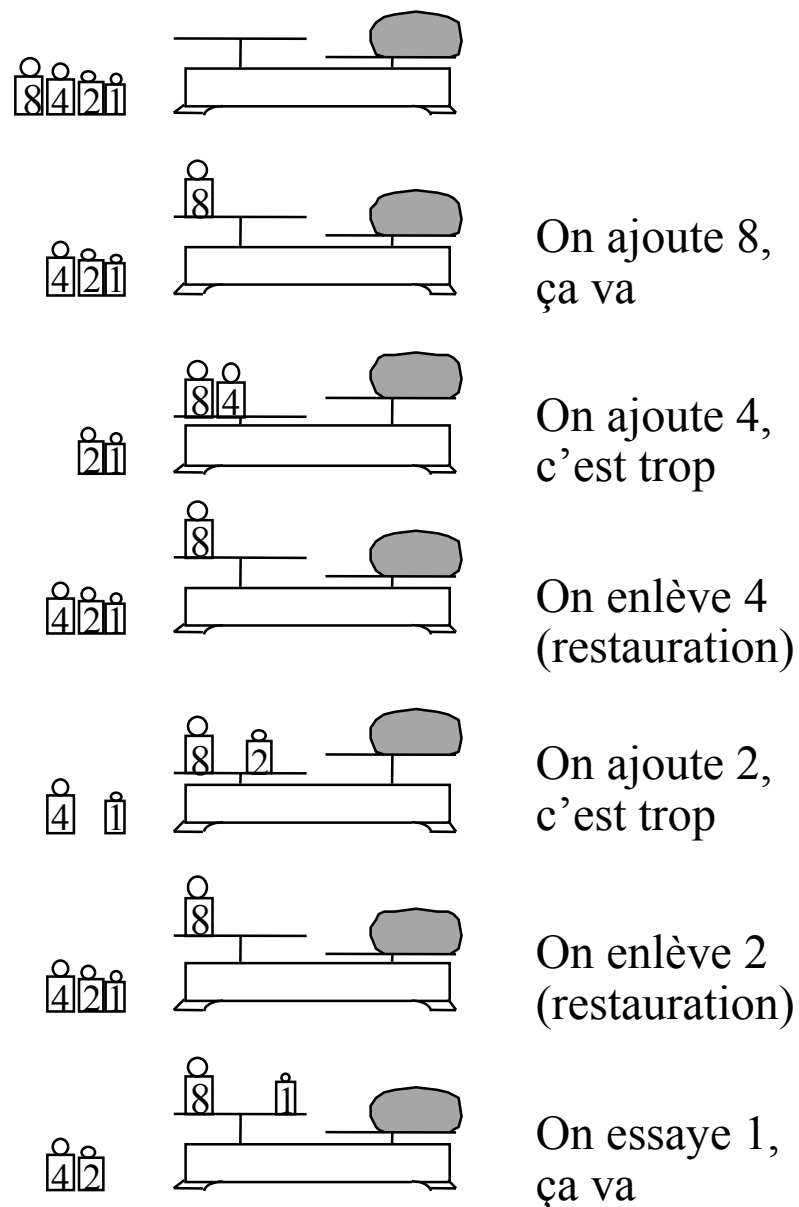
Ceci est la notation en complément à 2 $p_i \in \{0,1\}$



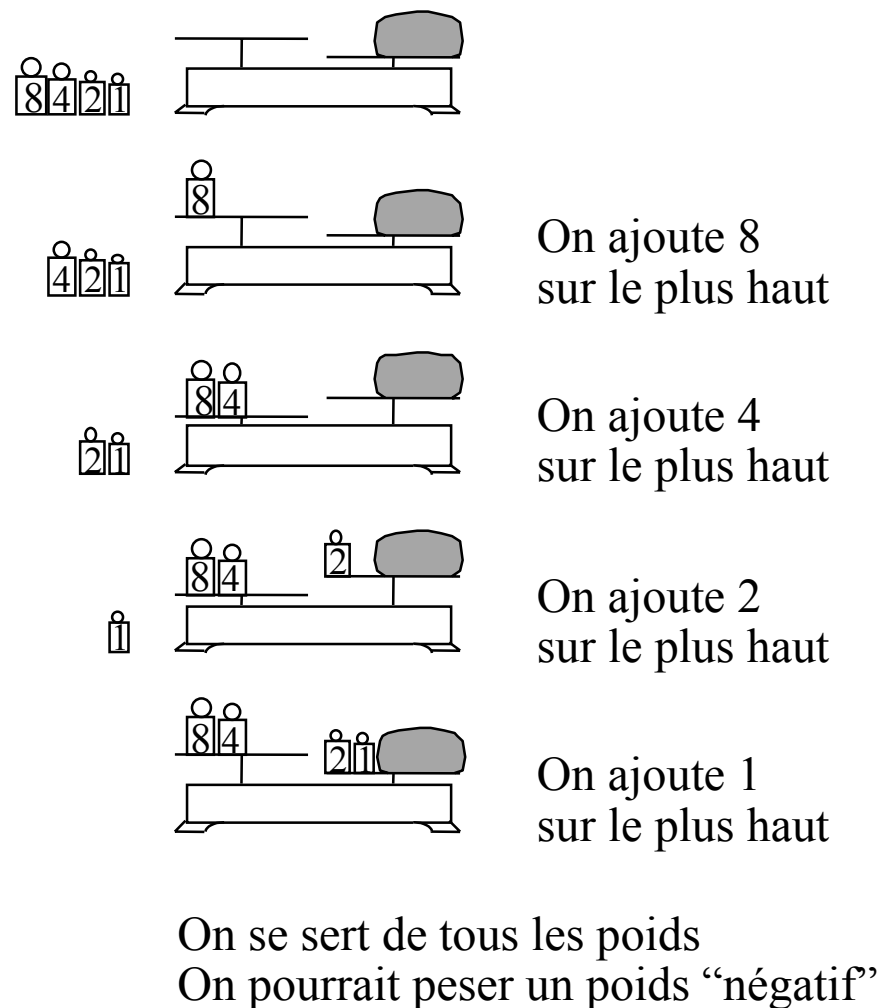
La conversion ne change pas la valeur de Q mais seulement sa représentation sous forme de chaîne de bits

Remarque: si on connaît a priori le signe du résultat, l'inverseur n'est même pas nécessaire

Pesée à restauration



Pesée sans restauration

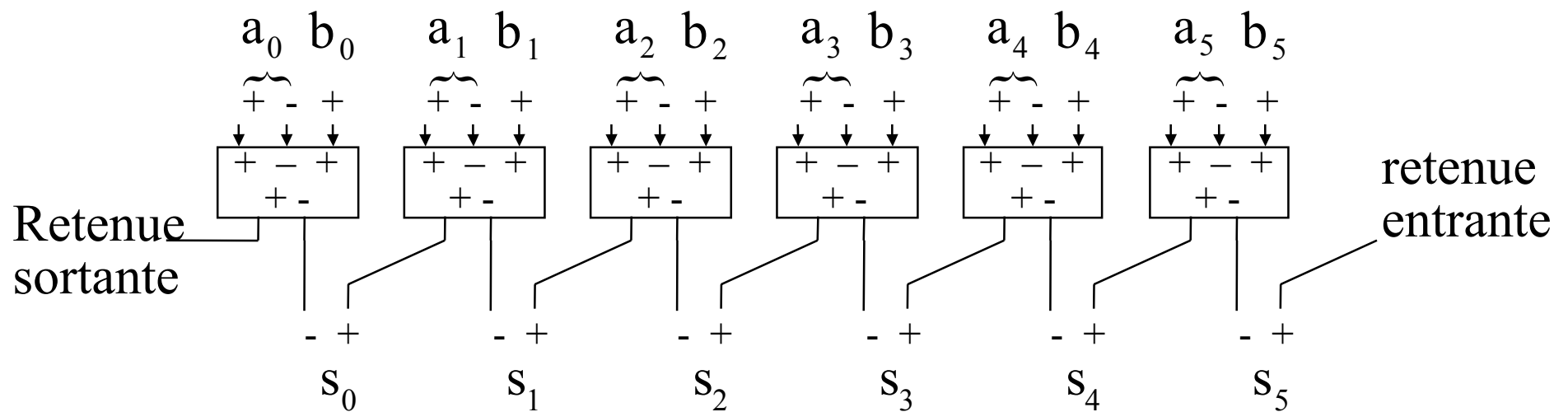


Le "BS", mais c'est très simple

$$A = A^+ - A^- = \sum_{i=0}^{n-1} a_i^+ 2^{-i} - \sum_{i=0}^{n-1} a_i^- 2^{-i} = \sum_{i=0}^{n-1} (a_i^+ - a_i^-) 2^{-i} = \sum_{i=0}^{n-1} a_i 2^{-i}$$

$$a_i^+, a_i^- \in \{0, 1\} \qquad a_i \in \{-1, 0, 1\}$$

Addition hybride $S = A + B$ $a_i, s_i \in \{-1, 0, 1\}$ $b_i \in \{0, 1\}$



Avec ou Sans propagation

Type d'opération

Propagation de Retenue

	Avec	Sans
Additionner de deux nombres en BS (A + B) ⇒ S	<input type="checkbox"/>	<input type="checkbox"/>
Déterminer le signe d'un nombre en BS (0 0 0 $\bar{1}$ ≥ 0 ?)	<input type="checkbox"/>	<input type="checkbox"/>
Forcer à 0 les poids forts d'un nombre petit 1 $\bar{1}$ $\bar{1}$ 0 ⇒ 0 0 1 0	<input type="checkbox"/>	<input type="checkbox"/>
Convertir de notation BS à Standard 0 1 0 $\bar{1}$ ⇒ 0 0 1 1	<input type="checkbox"/>	<input type="checkbox"/>

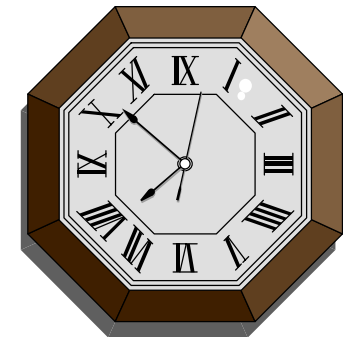
?? Utilité du BS ??

??? Comment utiliser cette *et c* *ensur* *é* de notation ???

- Sans comparaison
- Sans élimination de chiffres non significatifs
- Avec un coût de conversion exorbitant

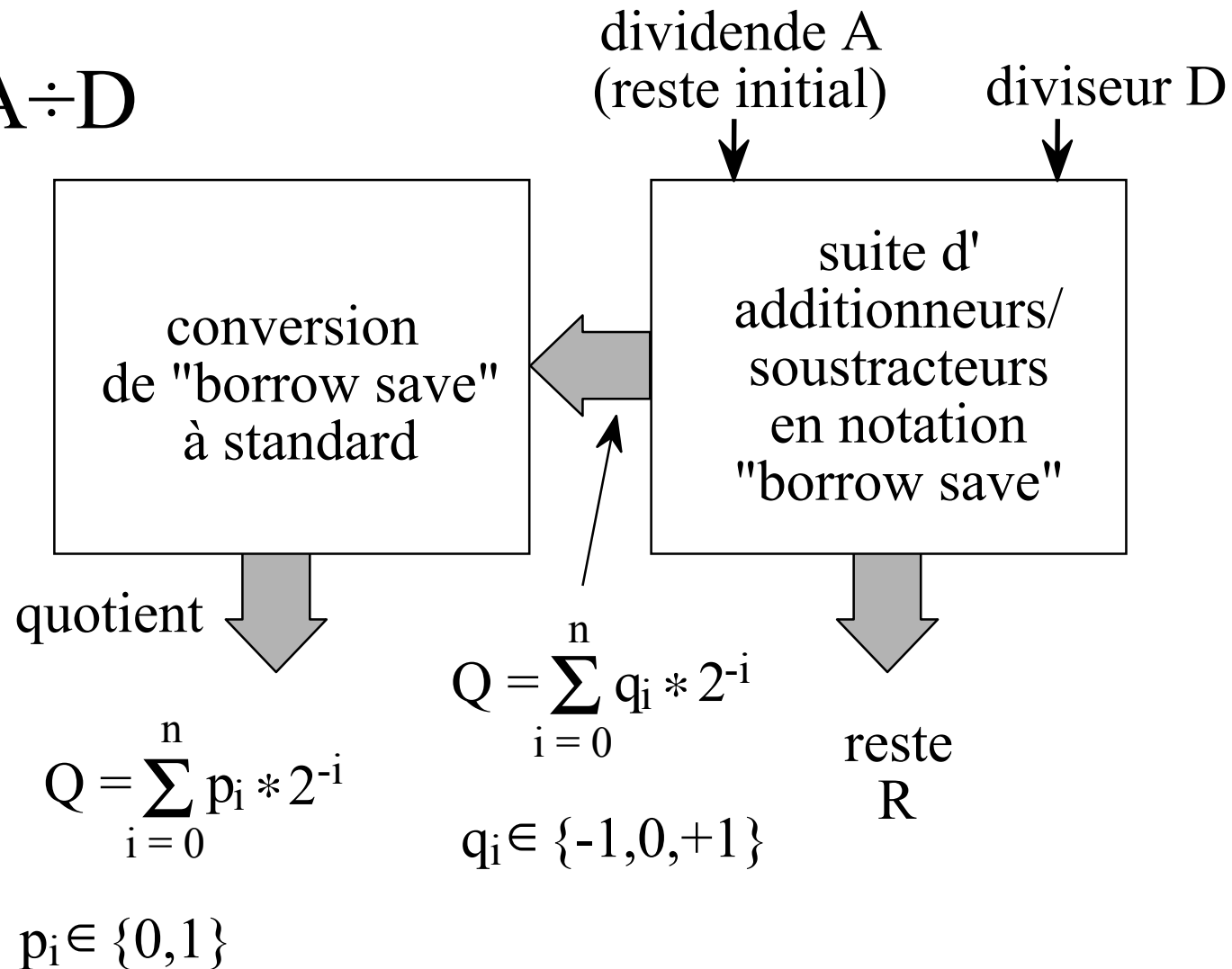
Dans un algorithme comme la DIVISION
RAPIDE

Divide ut regnes Machiavel

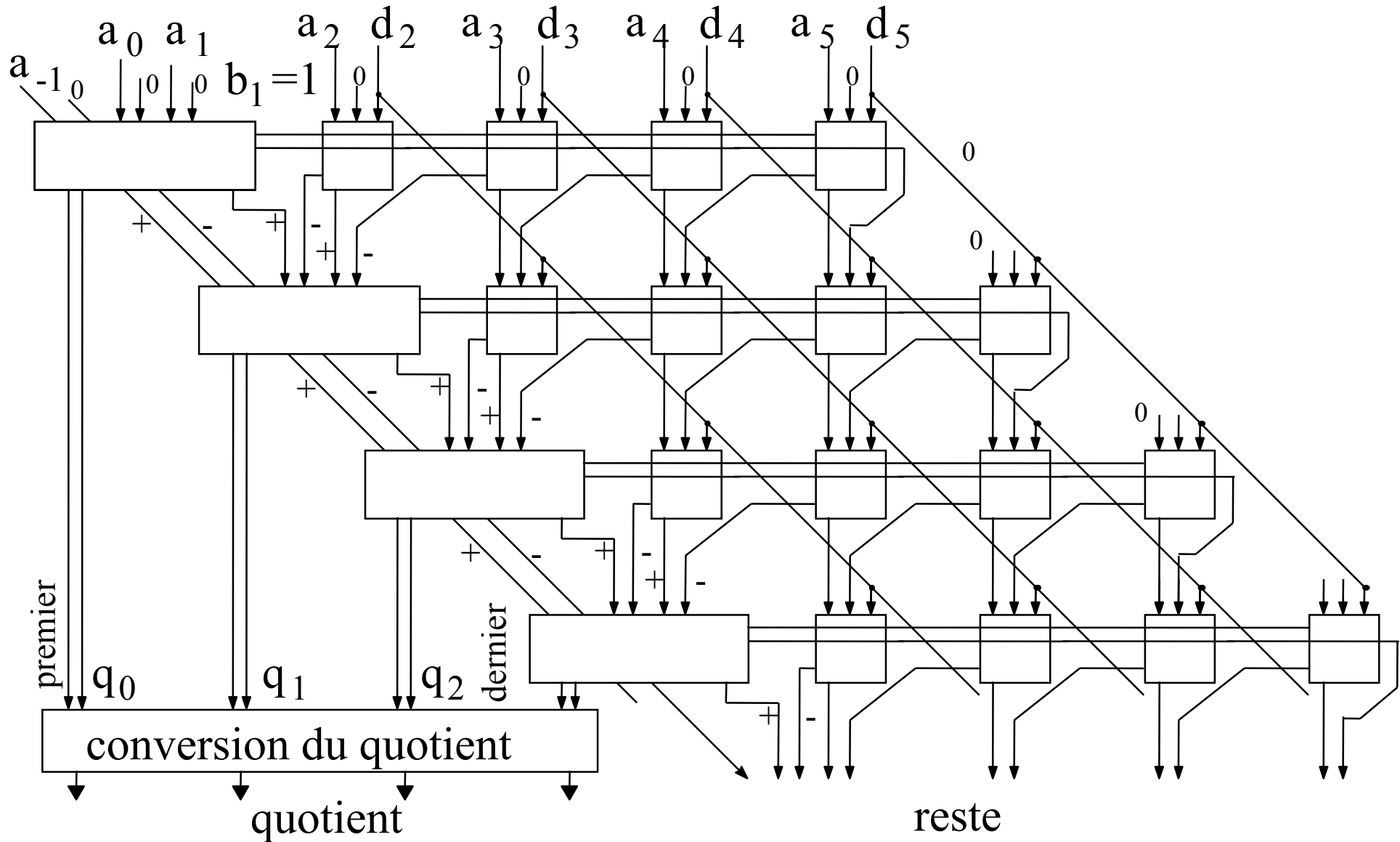


Diviseur en notation "BS"

$$Q := A \div D$$



Diviseur régulier en notation "borrow save"



Format du reste partiel R_j

On suppose D normalisé $1 \leq D < 2$: $D = \sum_{i=0}^n d_i 2^{-i}$ $d_0 = 1$

Donc $-4 < -2D \leq R_j * 2^j \leq +2D < 4$.

Pour être additionné ou soustrait rapidement, R_j est écrit en “BS”
Pour avoir une écriture bornée de R_j , les 2 premiers chiffres de R_j
ne peuvent pas être non nul de signe différent

$$R_j * 2^j = \sum_{i=-2}^n r_i 2^{-i} = r_{-2} r_{-1} r_0 , r_1 r_2 r_3 \dots r_n \quad r_{-2} * r_{-1} \geq 0$$

Choix de l'opération à exécuter

Itération à l'étape j : $R_{j+1} := R_j - q_j * D * 2^{-j}$ avec $q_j \in \{-1, 0, +1\}$

Pour déterminer l'opération q_j à exécuter à l'étape j ,
il suffit d'examiner les 3 premiers chiffres de R_j seulement.

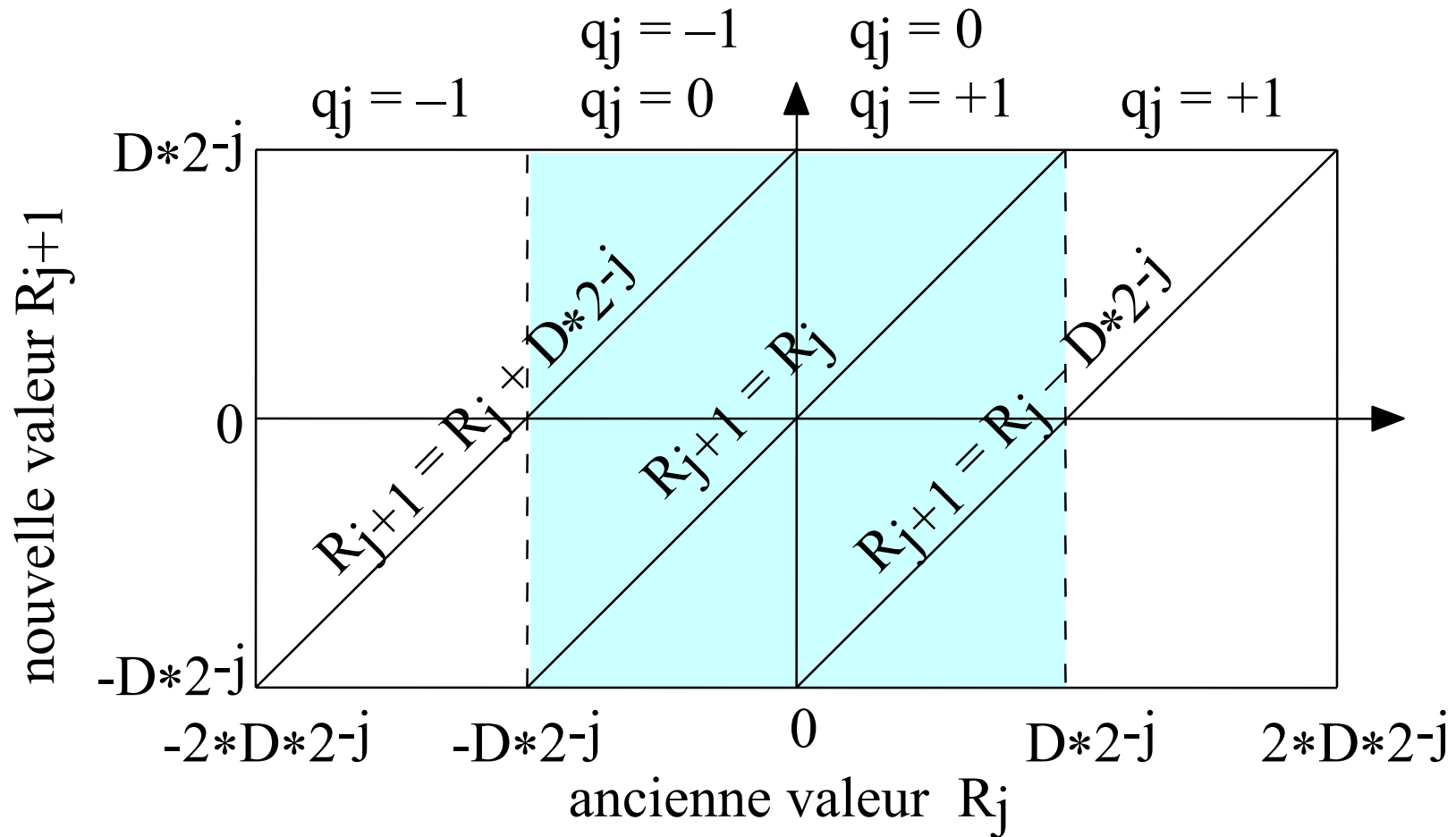
Condition de convergence: $|R_j| \leq 2 * D * 2^{-j} \implies |R_{j+1}| \leq 2 * D * 2^{-j-1}$

Examen de	r_{-2}	r_{-1}	r_0	
r_{-2} r_{-1} et r_0	-1	x	x	} $-2 * D * 2^{-j} \leq R_j < 0 \implies R_{j+1} = R_j + D * 2^{-j}$
	0	-1	x	
	0	0	-1	
	0	0	0	} $-2^{-j} < R_j < +2^{-j} \implies R_{j+1} = R_j$
	0	0	+1	} $0 < R_j \leq 2 * D * 2^{-j} \implies R_{j+1} = R_j - D * 2^{-j}$
	0	+1	x	
	+1	x	x	

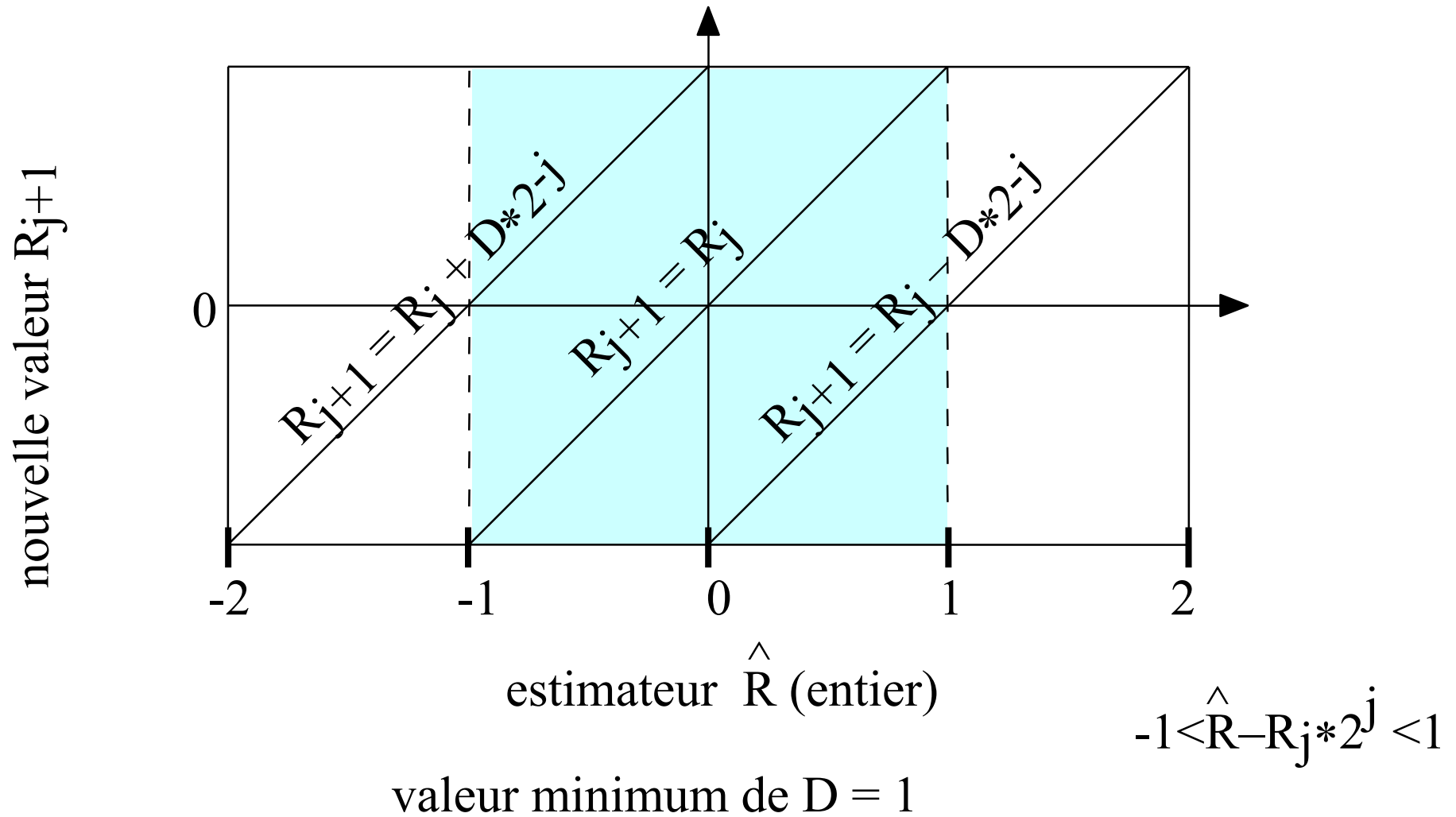
On introduit $\hat{R} = 4 * r_{-2} + 2 * r_{-1} + r_0$, "estimation" de $R_j * 2^j$ $-1 < \hat{R} - R_j * 2^j < 1$

Division SRT

Invariant: $-2 \cdot D \cdot 2^{-j} \leq R_j \leq 2 \cdot D \cdot 2^{-j}$

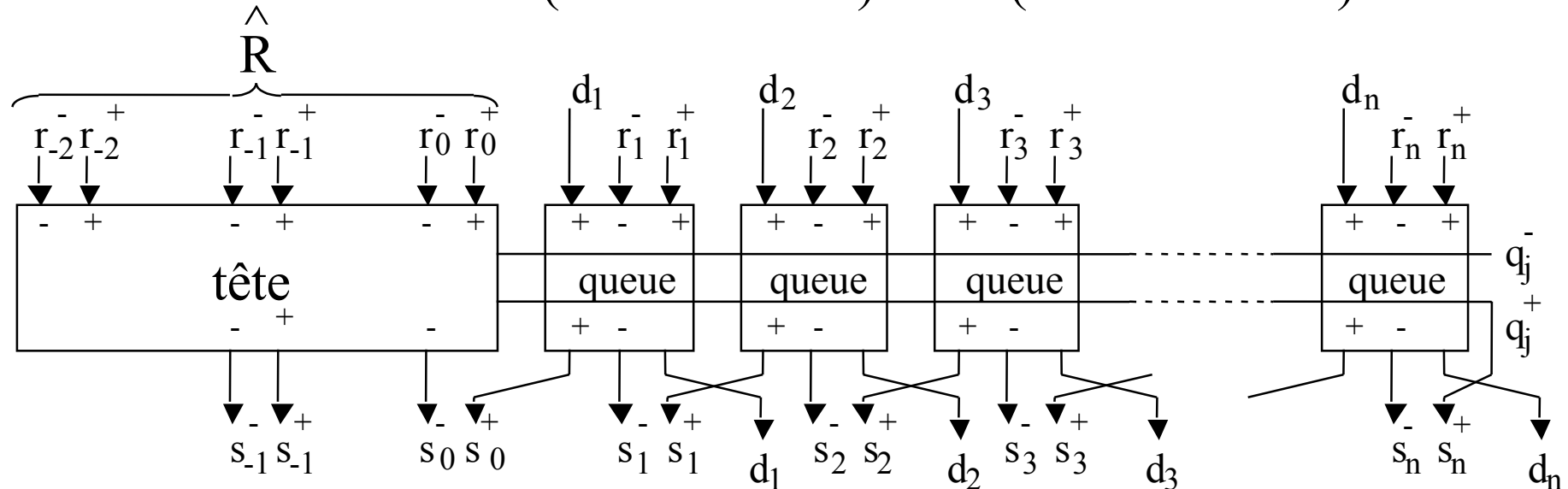


Division SRT (2)



Étage de diviseur

Entrées: \hat{R} (en redondant) et D (conventionnel)



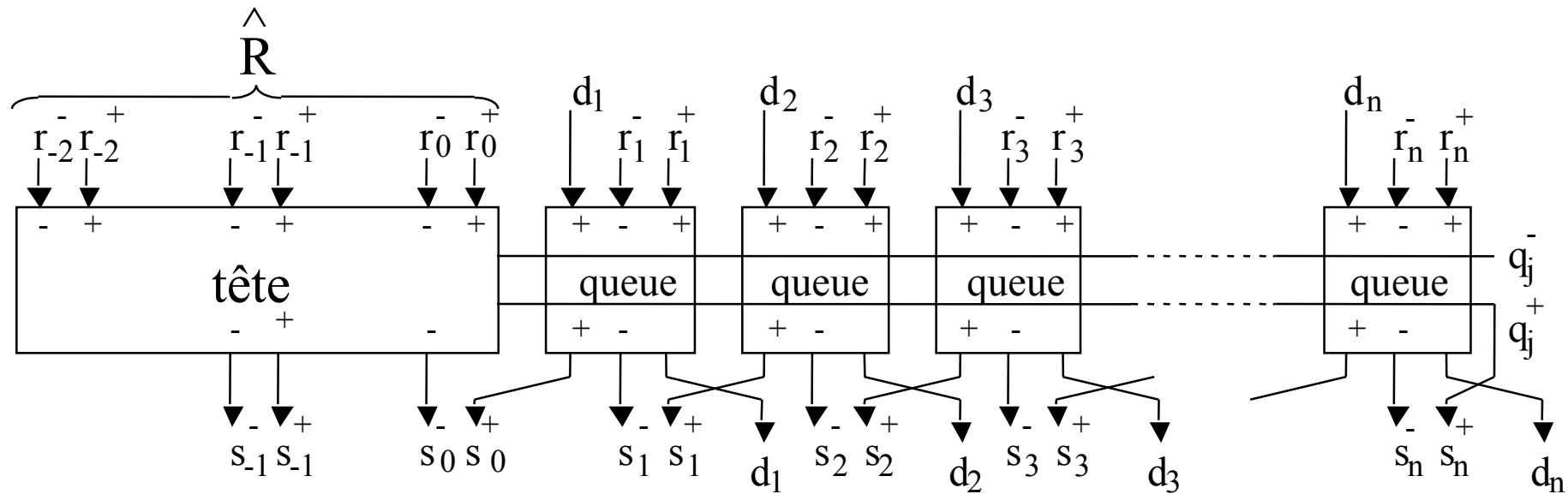
Sorties: S (en redondant) et D (conventionnel)

si $\hat{R} < 0$ **alors** $S = R + D$ { add et sous sans propagation }

sinon si $\hat{R} > 0$ **alors** $S = R - D$

sinon $S = R$

Rôles de la tête et de la queue



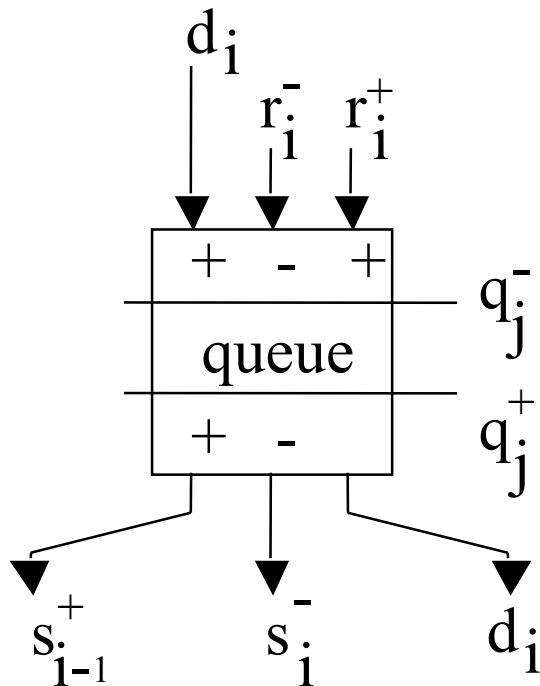
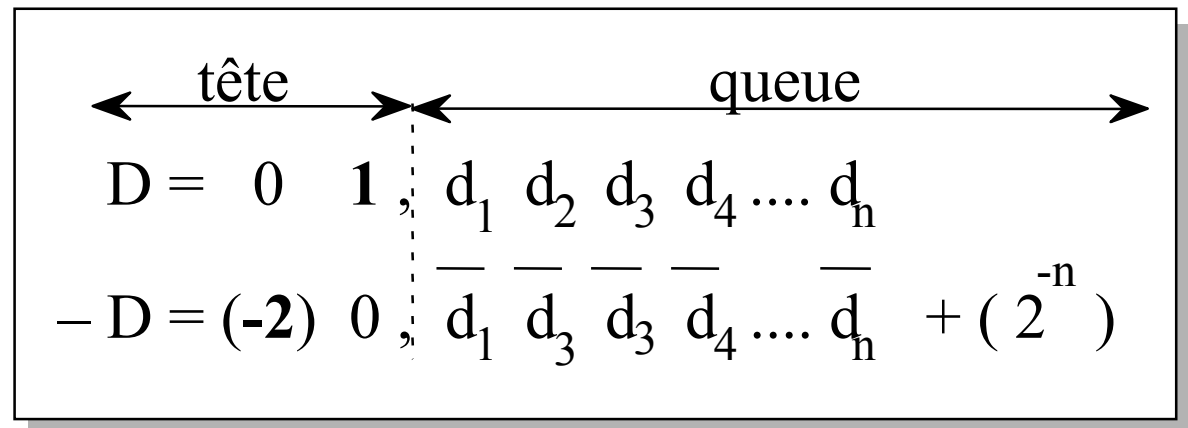
Tête:

- 1- Déterminer l'opération à exécuter (add, sous ou rien)
- 2- Exécuter cette opération sur les chiffres de tête
- 3- Recoder le résultat pour éliminer le chiffre poids fort s_{-2}

Queue:

- Exécuter l'opération (add, sous ou rien) sur les chiffres de queue sans retenue propagé
- Transmettre D décalé vers les poids faibles

Equations d'une cellule de queue



Equation arithmétique

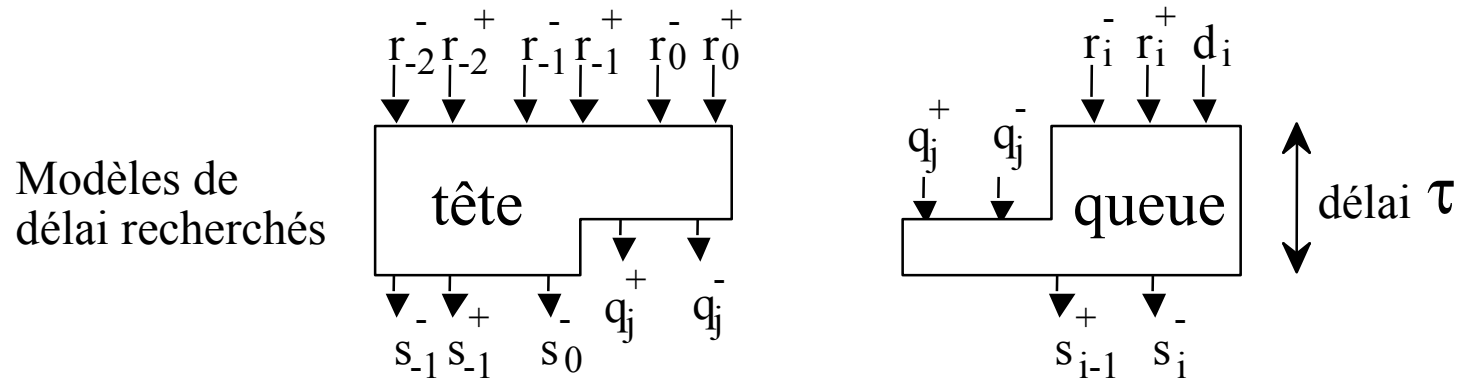
$$2 * s_{i-1}^+ - s_i^- = r_i^+ - r_i^- + (q_j^+ \overline{d_i} + q_j^- d_i)$$

$$2 * s_{i-1}^+ + \overline{s_i^-} = r_i^+ + \overline{r_i^-} + (q_j^+ \overline{d_i} + q_j^- d_i)$$

Les contrôles q_j^- et q_j^+ arrivent après les autres

Optimisation de la cellule de queue

Les contrôles q_j^- et q_j^+ arrivent après les autres dans la queue



queue	s_{i-1}^+	s_i^-
q_j^+	$(r_i^+ \wedge \overline{r_i^-}) \vee (r_i^+ \vee \overline{r_i^-}) \wedge \overline{d_i}$	$r_i^+ \oplus \overline{r_i^-} \oplus \overline{d_i}$
q_j^-	$(r_i^+ \wedge \overline{r_i^-}) \vee (r_i^+ \vee \overline{r_i^-}) \wedge d_i$	$r_i^+ \oplus \overline{r_i^-} \oplus d_i$
$\overline{q_j^+ \vee q_j^-}$	$(r_i^+ \wedge \overline{r_i^-})$	$r_i^+ \oplus \overline{r_i^-}$

Equations du bloc de tête

$$\hat{R} := 4 * (r_{-2}^+ - r_{-2}^-) + 2 * (r_{-1}^+ - r_{-1}^-) + (r_0^+ - r_0^-)$$

$$q_j^+ := (\hat{R} > 0); \quad q_j^- := (\hat{R} < 0);$$

$$\text{si } q_j^+ \text{ alors } \Sigma_{\text{out}} := \hat{R} - 2$$

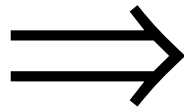
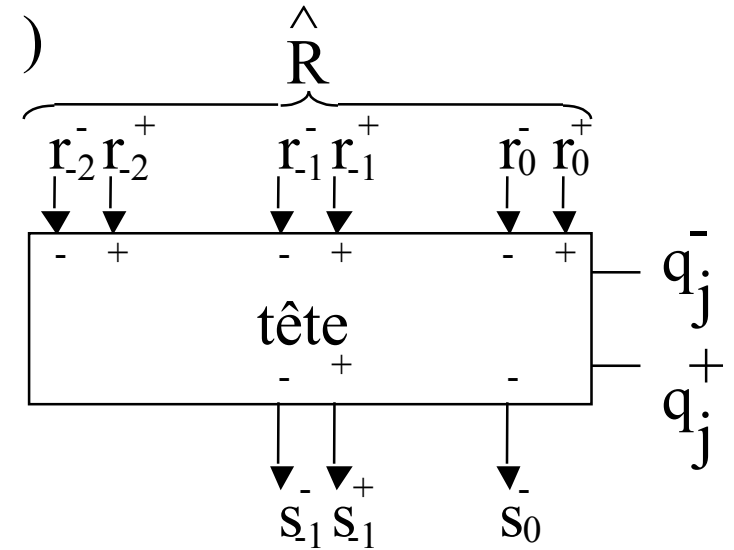
$$\text{sinon si } q_j^- \text{ alors } \Sigma_{\text{out}} := \hat{R} + 1$$

$$\text{sinon } \Sigma_{\text{out}} := 0;$$

$$s_{-1}^+ := (\Sigma_{\text{out}} > 0);$$

$$s_{-1}^- := (\Sigma_{\text{out}} \leq -2);$$

$$s_0^- := \Sigma_{\text{out}} - 2 * (s_{-1}^+ - s_{-1}^-);$$



$$q_j^+ := r_{-2}^+ + \overline{r_{-2}^-} (r_{-1}^+ \overline{r_{-1}^-} + \overline{r_{-1}^-} r_0^+ \overline{r_0^-} + r_{-1}^+ r_0^+ \overline{r_0^-});$$

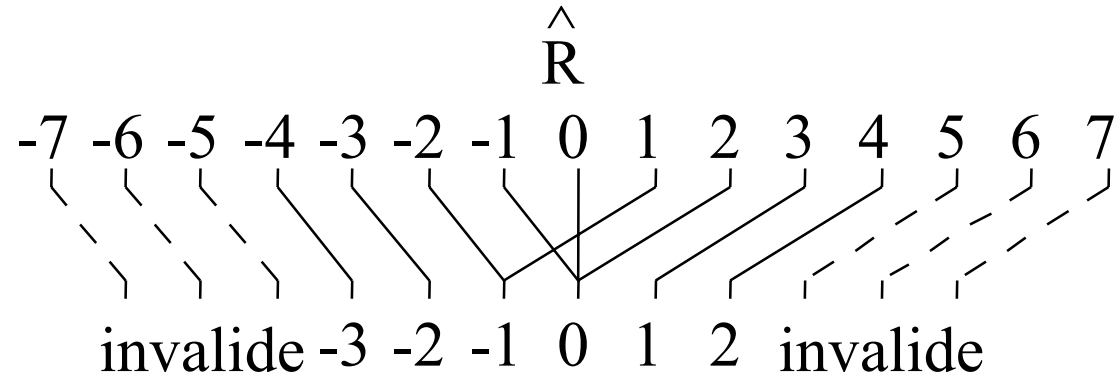
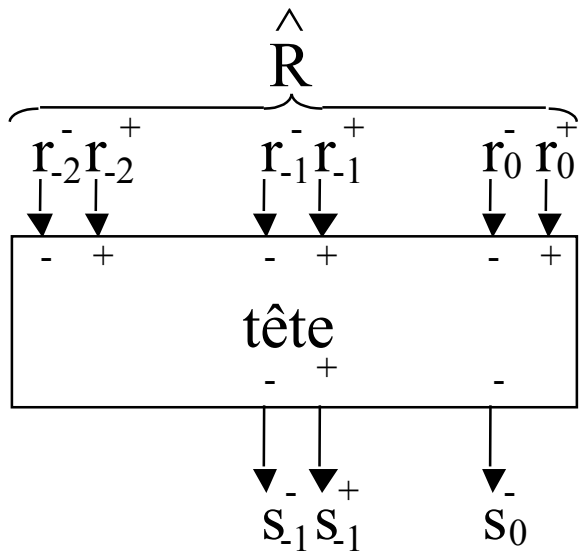
$$q_j^- := r_{-2}^- + \overline{r_{-2}^+} (\overline{r_{-1}^+} r_{-1}^- + \overline{r_{-1}^+} \overline{r_0^+} r_0^- + r_{-1}^- \overline{r_0^+} r_0^-);$$

$$s_{-1}^+ := r_{-2}^+ (r_{-1}^+ + \overline{r_{-1}^-} r_0^+ \overline{r_0^-}) + \overline{r_{-2}^-} r_{-1}^+ \overline{r_{-1}^-} r_0^+ \overline{r_0^-};$$

$$s_{-1}^- := r_{-2}^- (\overline{r_{-1}^+} r_{-1}^- + \overline{r_{-1}^+} \overline{r_0^+} r_0^-) + \overline{r_{-2}^+} \overline{r_{-1}^+} r_{-1}^- \overline{r_0^+} r_0^-;$$

$$s_0^- := q_j^- \oplus r_0^+ \oplus r_0^- \quad \{ \text{seuls chiffres de poids } 2^0 \};$$

Condition arithmétique



Il faut que $-4 \leq \hat{R} \leq +4$

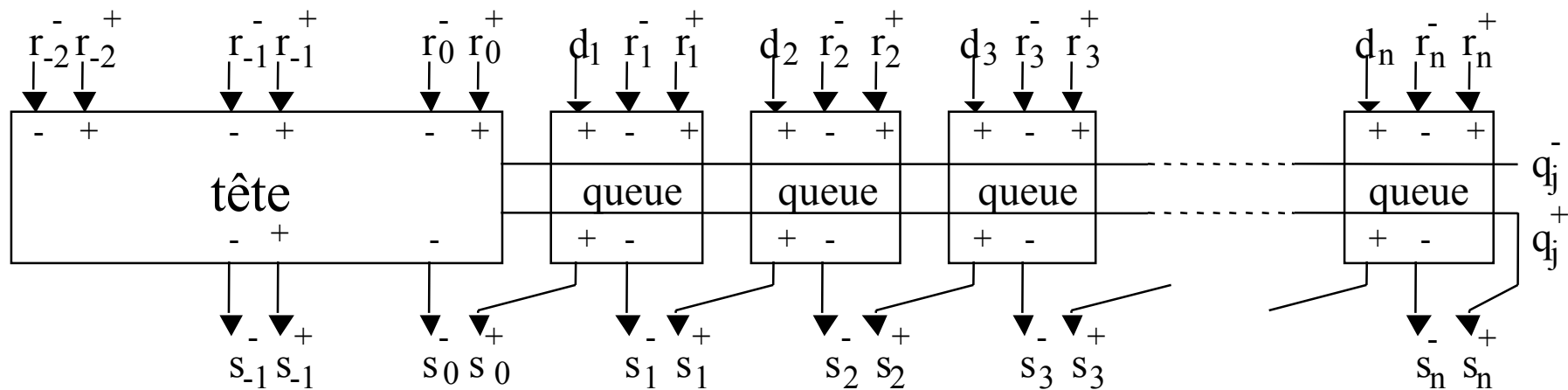
$$-4 < -2 * D \leq R_j * 2^j \leq 2 * D < 4$$

$$-1 < \hat{R} - R_j * 2^j < 1$$

$$\Rightarrow -5 < \hat{R} < 5$$

Le Quotient est en notation BS

⇒ il faut le convertir



La conversion ne change pas la valeur de Q mais seulement sa représentation sous forme de chaîne de bits.

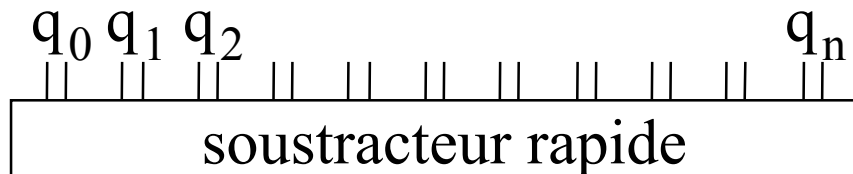
La somme pondérée des bits qui entrent dans le convertisseur est égale à la somme pondérée des bits qui en sortent.

$$q_j = q_j^+ - q_j^-$$
$$q_j \in \{-1, 0, +1\}$$

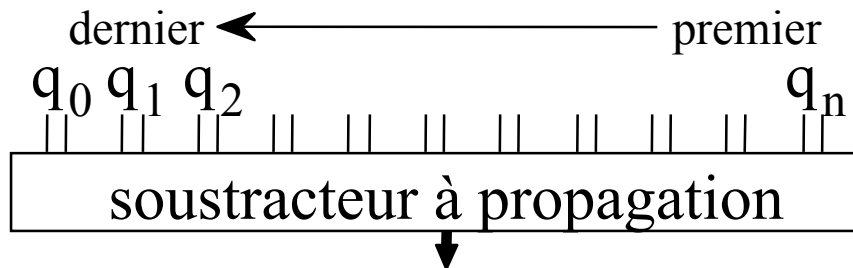
Conversion de "borrow save" à standard

$$Q_n = \sum_{i=0}^n q_i * 2^{-i} \quad q_i \in \{-1, 0, +1\} \implies Q_n = \sum_{i=0}^n \max(q_i, 0) * 2^{-i} - \sum_{i=0}^n \max(-q_i, 0) * 2^{-i}$$

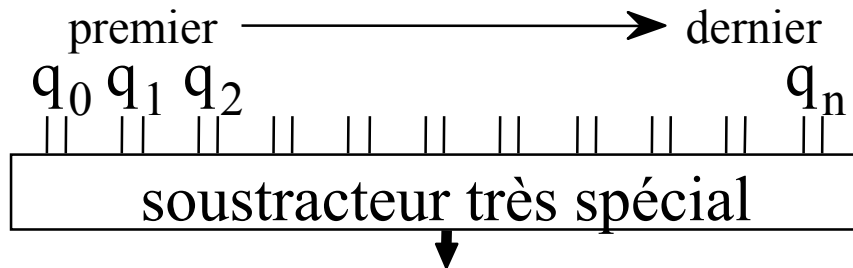
tous simultanément



L'additionneur doit être très rapide



Le délai de propagation de la retenue sur une position doit être inférieur au retard entre 2 q

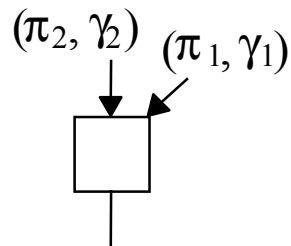


On veut le résultat dès que q_n est prêt

Conversion de "borrow save" à standard (2)

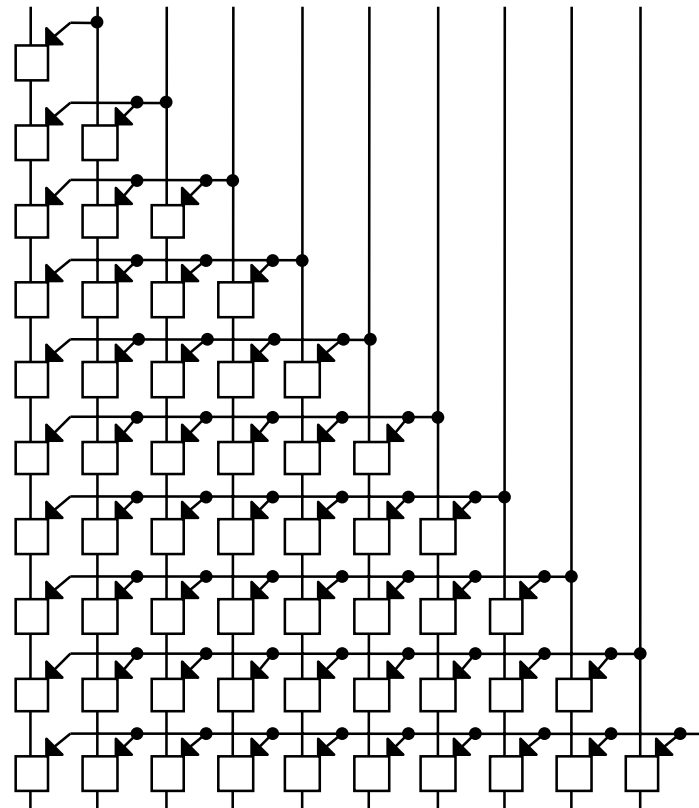
génération des p_i et des g_i

$$q_i \in \{-1, 0, +1\}$$



$$(\pi_2 \wedge \pi_1, \gamma_2 \vee \pi_2 \wedge \gamma_1)$$

- Associatif
- Non commutatif
- Idempotent
- Croissant (inverseurs)



q_i	p_i	g_i
-1	0	1
0	1	0
+1	0	0

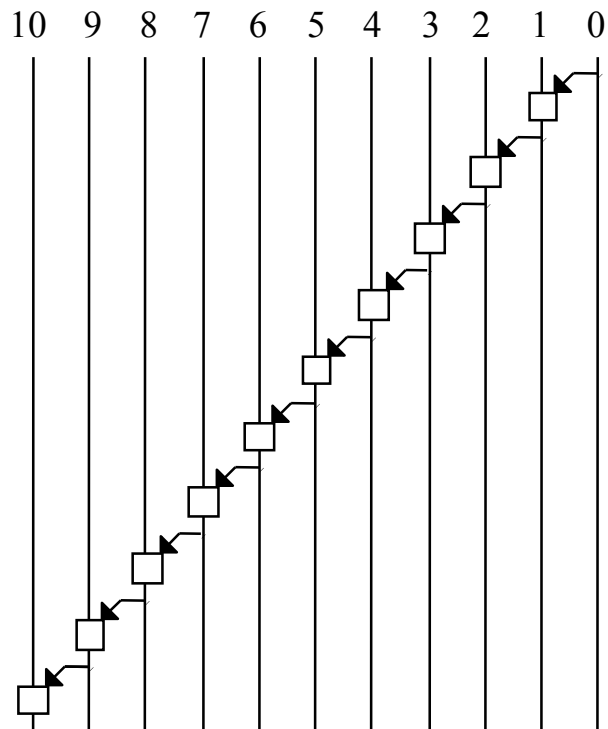
génération de la somme $s_i = p_i \oplus G_{i-1,0}$

$$s_i \in \{0, 1\}$$

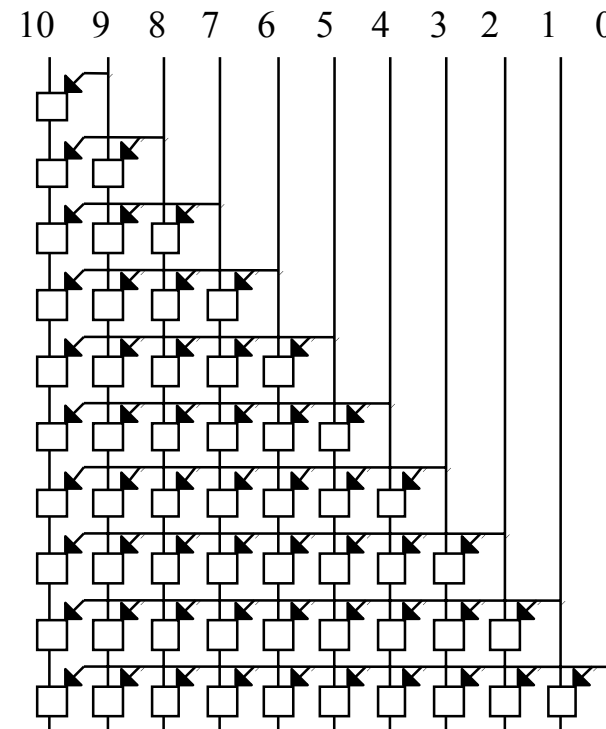
Conversion de "borrow save" à standard (3)

$$q_i \in \{-1, 0, +1\} \quad p_i \in \{0, 1\}$$

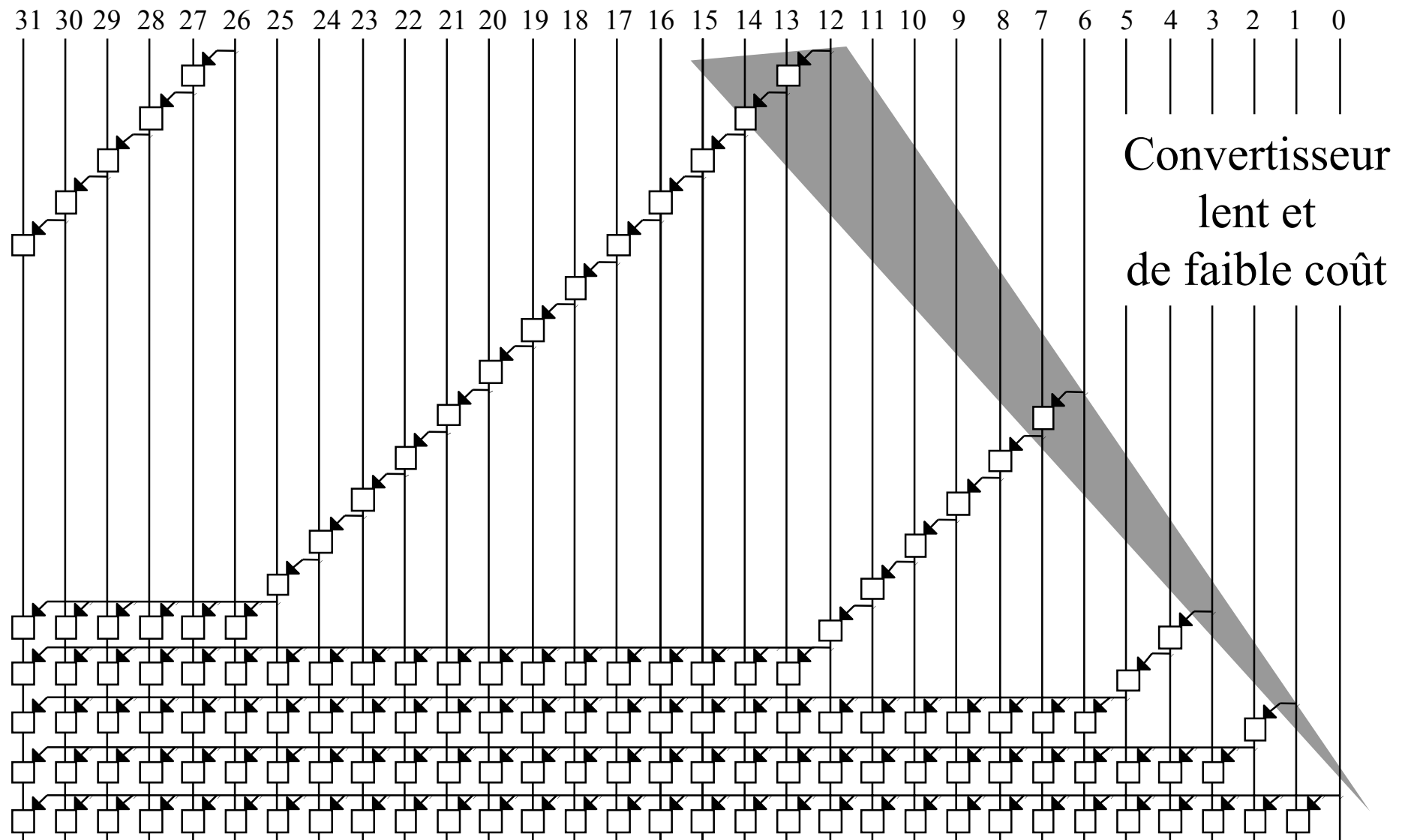
séquentiellement
poids faibles d'abord



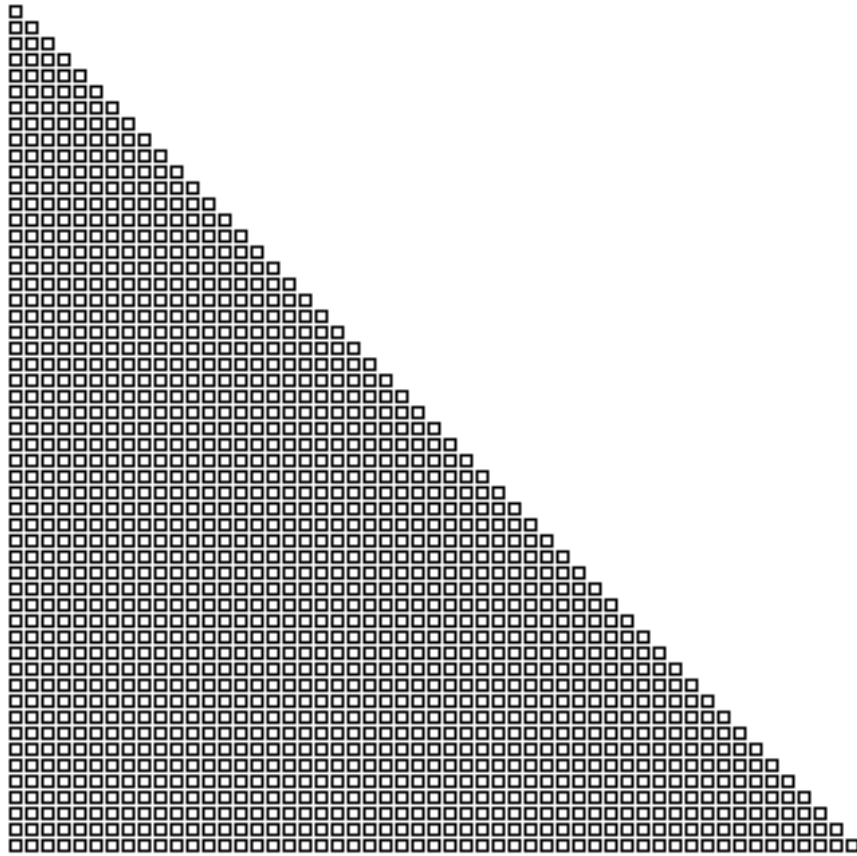
séquentiellement
poids forts d'abord



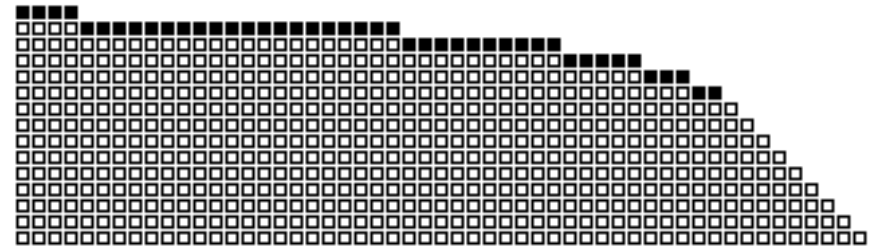
Conversion de "borrow save" à standard (4)



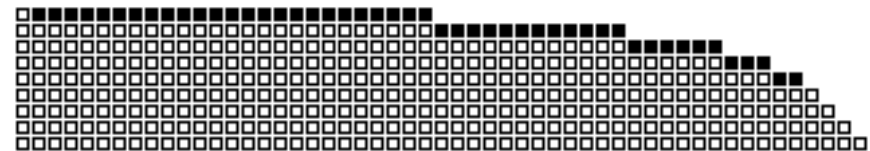
Formes du convertisseur



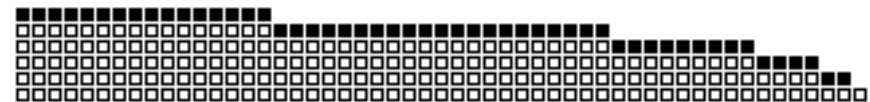
≤ 1



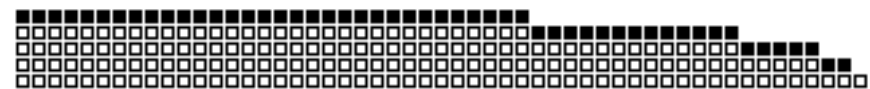
1,1



1,2



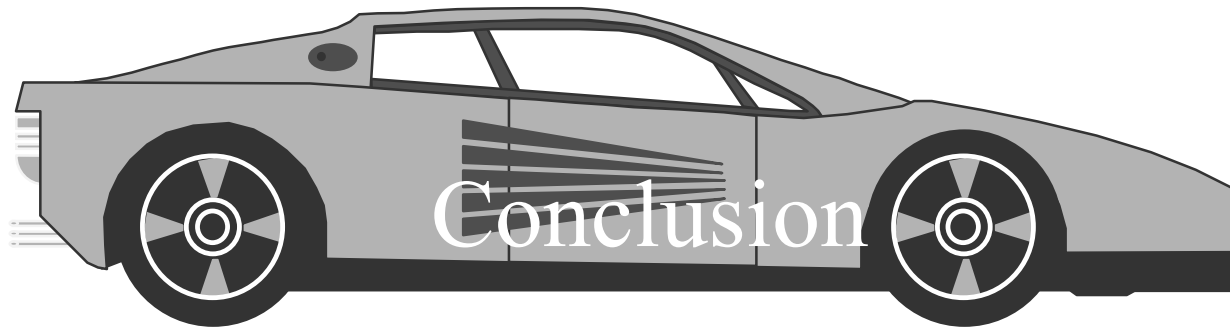
1,5



1,8



2,2



Un algorithme de division rapide a été proposé:



- Les Additions/Soustraction ne propagent pas la retenue



- Le calcul, le recodage et le test du reste partiel sont effectués simultanément (et non séquentiellement)



- Les lignes longues et leurs amplis sont éliminées du chemin critique



- Un nouvel algorithme de conversion du quotient, rapide et peu coûteux, a été introduit.