Opérateurs de Calcul en-ligne

Alain GUYOT

Concurrent Integrated Systems
TIMA

(33) 04 76 57 46 16

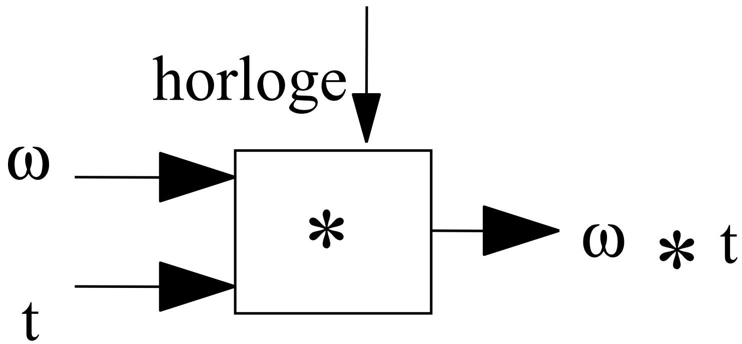
Alain.Guyot@imag.fr

http://tima-cmp.imag.fr/~guyot/Cours/Arithmetique

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

Opérateurs en ligne

entrées
reçues
en série
chiffre à
chiffre



sortie
transmise
en série
chiffre à
chiffre

Exemples d'opérations en ligne

Addition

$$
\begin{array}{r}
4\,3\,7\,2 \\
5\,3\,9\,1 \\
\hline
+ \quad 9\,7\,6\,3
\end{array}
$$

de droite à gauche ←

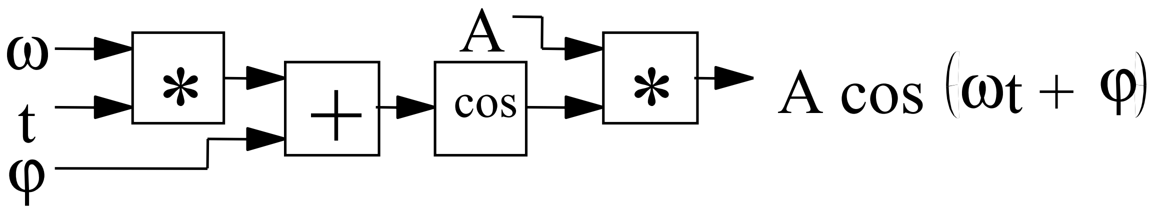
Maximum

$$
\begin{array}{r}
1\,7\,8\,9 \\
1\,7\,9\,0 \\
\hline
\geq \quad 1\,7\,9\,0
\end{array}
$$

de gauche à droite →

Avantages des opérateurs en ligne

Avantage 1: parallélisme à grain fin (niveau du chiffre)



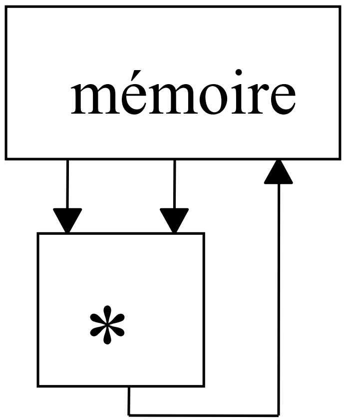
chaque opérateur travaille simultanément
(2 multiplieurs, 1 additionneur, 1 cosinus)

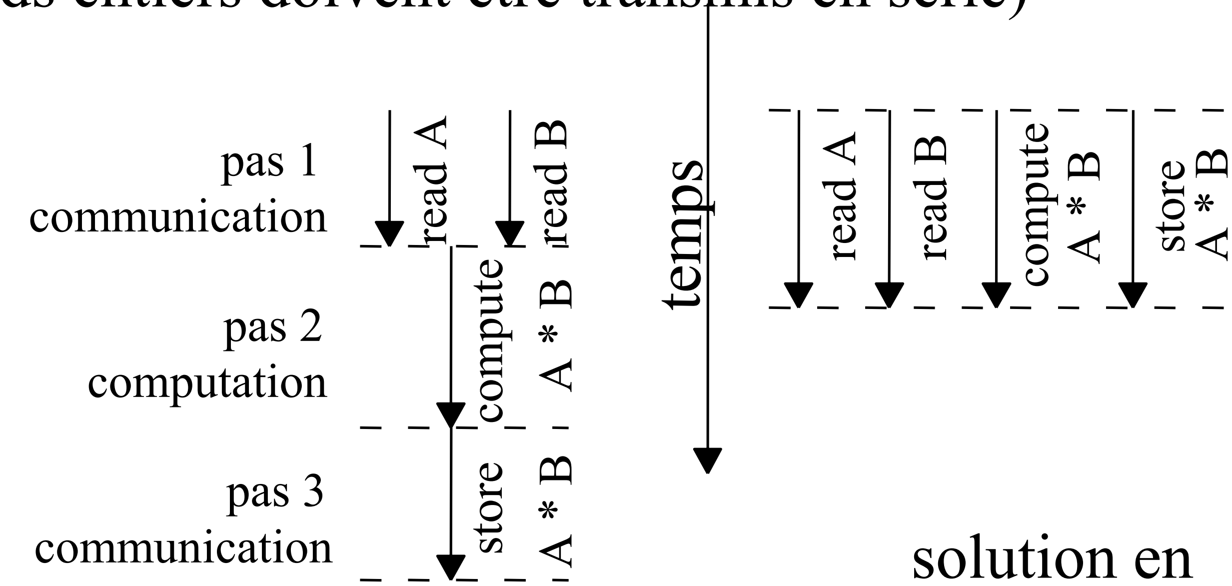
Avantages (2)

Avantage 2: la transmission recouvre l'exécution

exemple: multiplication de deux grands entiers A et B

(Les grands entiers doivent être transmis en série)



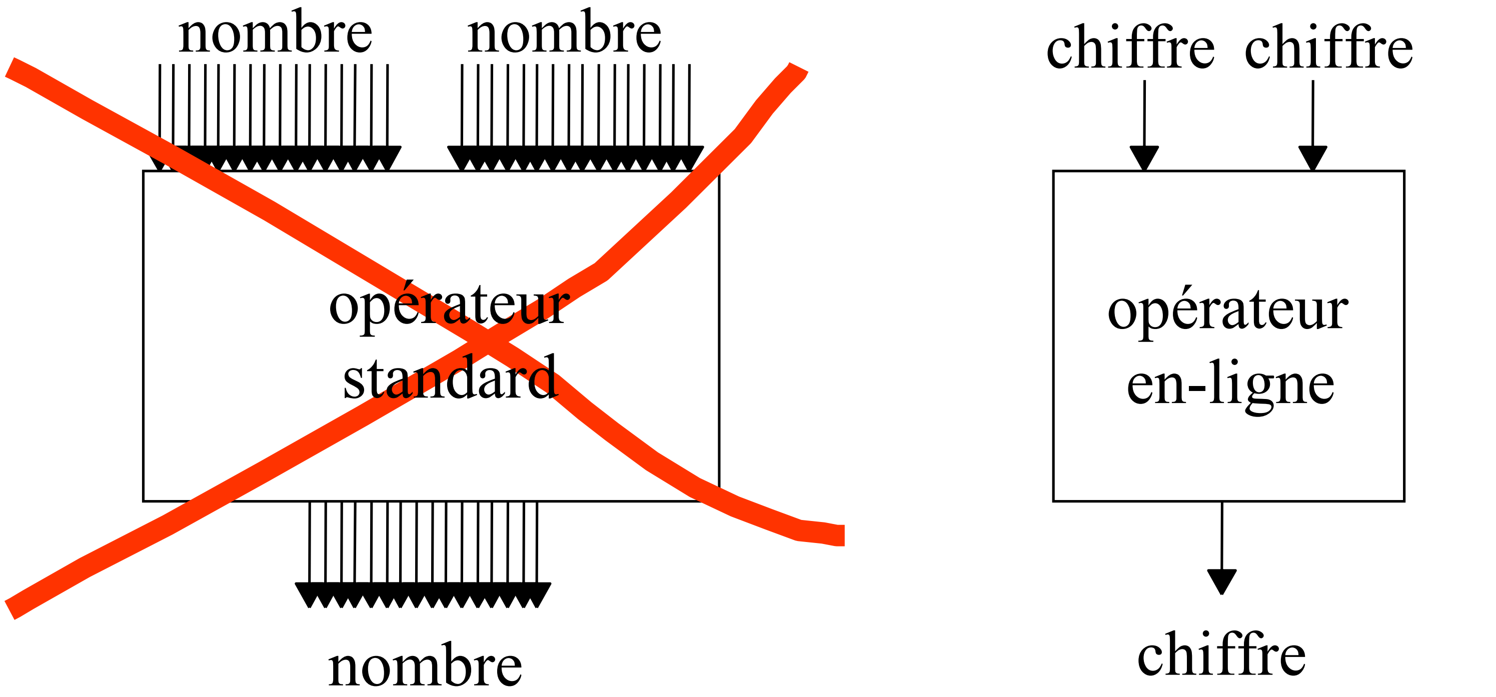


solution classique
séquentielle

solution en ligne
parallèle

Avantages (3)

Avantage 3: câblage



solution standard

solution en-ligne

Bonnes Nouvelles

Presque toutes les opérations habituelles peuvent être calculées en ligne

Addition	Maximum	Sinus/cosinus
Multiplication	Valeur Absolue	Tangente
Division	Saturation	Logarithme
Racine carrée	Tri	Exponentielle
Distance Euclidienne	Scaling	Polynômes

Opérations non calculables en ligne

Reste

Opérations modulaires

PGCD

Fonctions non continues

Arc sinus

Opérations en série

NOTATION En - tête	Standard Poids faibles	Standard Poids forts	Redondante Poids forts
addition	X		X (2)
multiplication	X (1)		X (2)
division			X (2)
maximum		X	X
racine carrée			X (2)

(1) Latence si on attend les poids forts
(2) Latence systématique

Comment mesurer les opérateurs en-ligne

période : inverse de la fréquence d'horloge

latence: nombre d'étages de registres entre entrée et sortie

(ou combinaison des poids des entrées moins le poids de la sortie)

La période peut être échangée contre de la latence

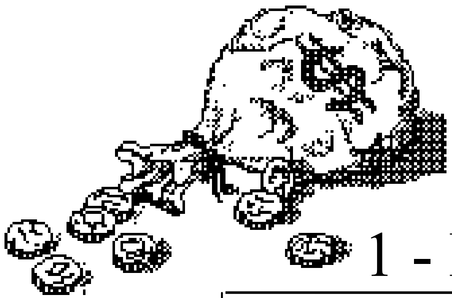
Quand des opérateurs sont mis en série:

période = max (périodes)

latence = Σ (latences) sur le chemin critique

(la latence peut devenir prédominante)

Question: Quel est le rapport coût/performance



1 - Délai

opération	construction habituelle naïve	construction habituelle usuelle	construction habituelle optimisée	en-ligne
addition	n	$\log_2 n$	1*	n
multiplication	n^2	n**	$\log_2 n$**	n
division	n^2	$n \log_2 n$	n***	n
racine carrée	n^2	$n \log_2 n$	n***	n

* Additionneur à retenue sauvegardée

** Arbre de Wallace

*** Division SRT (non Newton)

Question: Quel est le rapport coût/performance (2)

1 - Surface

opération	standard	en-ligne
addition	$n \log_2 n$	1
multiplication	n^2	n
division	n^2	n
racine carrée	n^2	n

a) add,sub,max,abs,tri,gain	coût fixe	1
b) mult,div,carré, racine	coût linéaire	θ (n)
c) fonctions élémentaires	coût quadratique	$\alpha n + \beta n^2$

Clefs

Toutes les opérations en ligne reposent sur

1- l'addition sans propagation de retenus
(c'est à dire la notation redondante)

2- l'addition en série poids forts en tête dérivée de la précédente

3- en registre d'"erreur" interne

Quelque opérations en lignes reposent également sur
un estimation de <u>faible précision</u> de certaine variable
(reste partiel, angle, etc ...)

Notation binaire redondante BS

$$A = \sum_{i=0}^{n-1} a_i\, 2^i \qquad a_i \in \{-1, 0, 1\}$$

In the Borrow-Save (B.S.) notation, each digit is coded by 2 bits

a_i coded on 2 bits

a_i^{+} a_i^{-}

	+ -		+ -
+1	1 0		
0	0 0	or	1 1
-1	0 1		

A number in redundant B.S. notation can be seen as the difference of two positive numbers

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

Autres notations binaires redondantes

$$A = \sum_{i=0}^{n-1} a_i \, 2^i$$

Every digit is coded on 2 bits

- unsigned
  - Carry save notation: $a_i \in \{0, 1, 2, 3\}$ coded as $[1, 2]$
- signed
  - sign+abs. value: $a_i \in \{-1, 0, 1\}$ coded as [sign, |value|]
  - Borrow save notation: $a_i \in \{-1, 0, 1\}$ coded as $[+, -]$

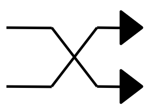
Borrow Save is symmetrical

truncation $\equiv$ rounding

easy to change sign (no $-B = \overline{B} + 1$)

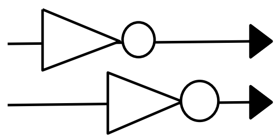
$a_i \Leftrightarrow - a_i$

a_i + -



+ - $- a_i$

a_i + -



+ - $- a_i$

Cas particulier du (Borrow Save)

$$A = \sum_{i=0}^{n-1} a_i \, 2^i \qquad a_i \in \{-1, 0, 1\}$$

1- $a_{n-1} \in \{-1, 0\}$ $\quad a_{0..n-1} \in \{0, 1\}$ standard 2's complement

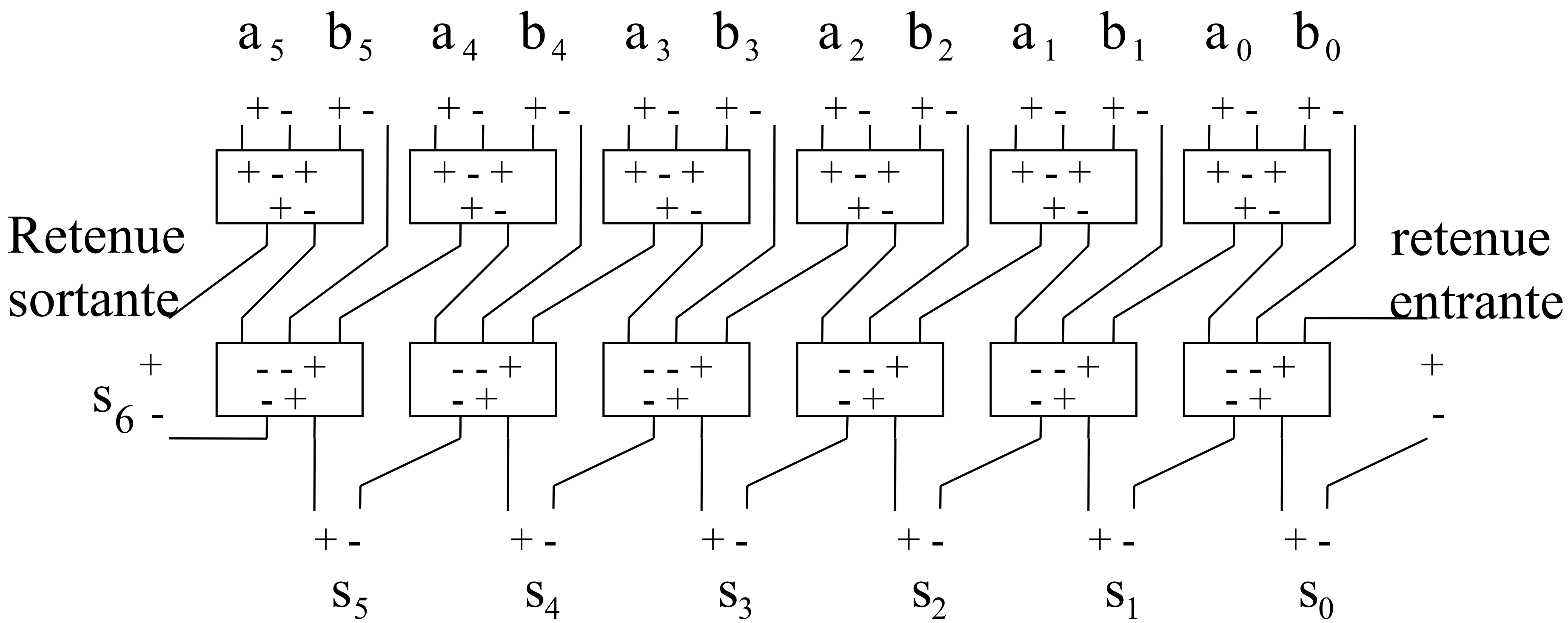
2- $a_i \in \{-1, 1\}$ on-line functions

3- $a_i \in \{-1, 0, 1\}$ with the maximum number of zero (average 2/3 zero 1/6 one 1/6 minus one)

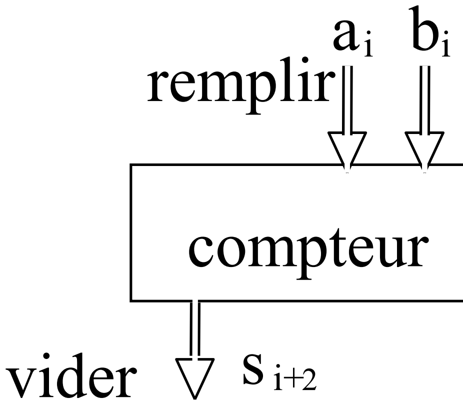
forms 1 and 3 may require an extra digit
only odd numbers representable in form 2
form 3 is the canonic signed binary digit form

Addition parallèle sans propagation de la retenue

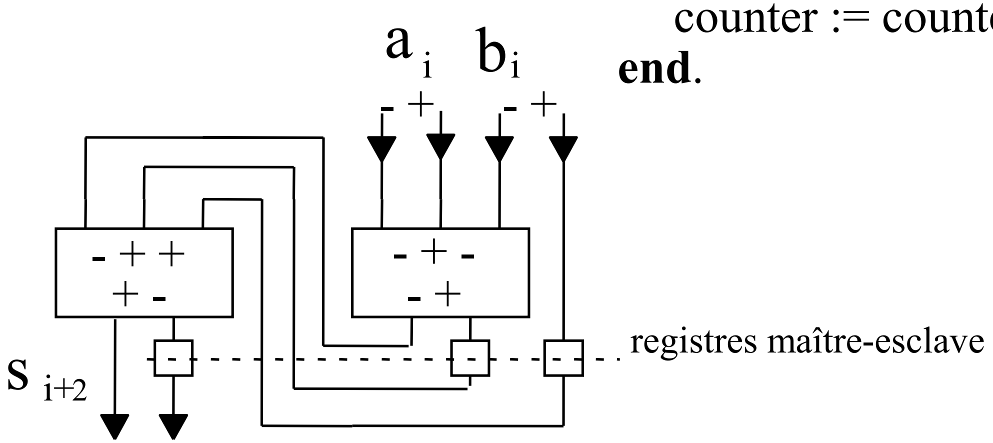
$$A = \sum_{i=0}^{n-1} a_i\, 2^i \quad B = \sum_{i=0}^{n-1} b_i\, 2^i \quad S = \sum_{i=0}^{n} a_i\, 2^i \qquad a_i, b_i, s_i \in \{-1, 0, 1\}$$



Addition en-ligne (addition de chiffres en série)



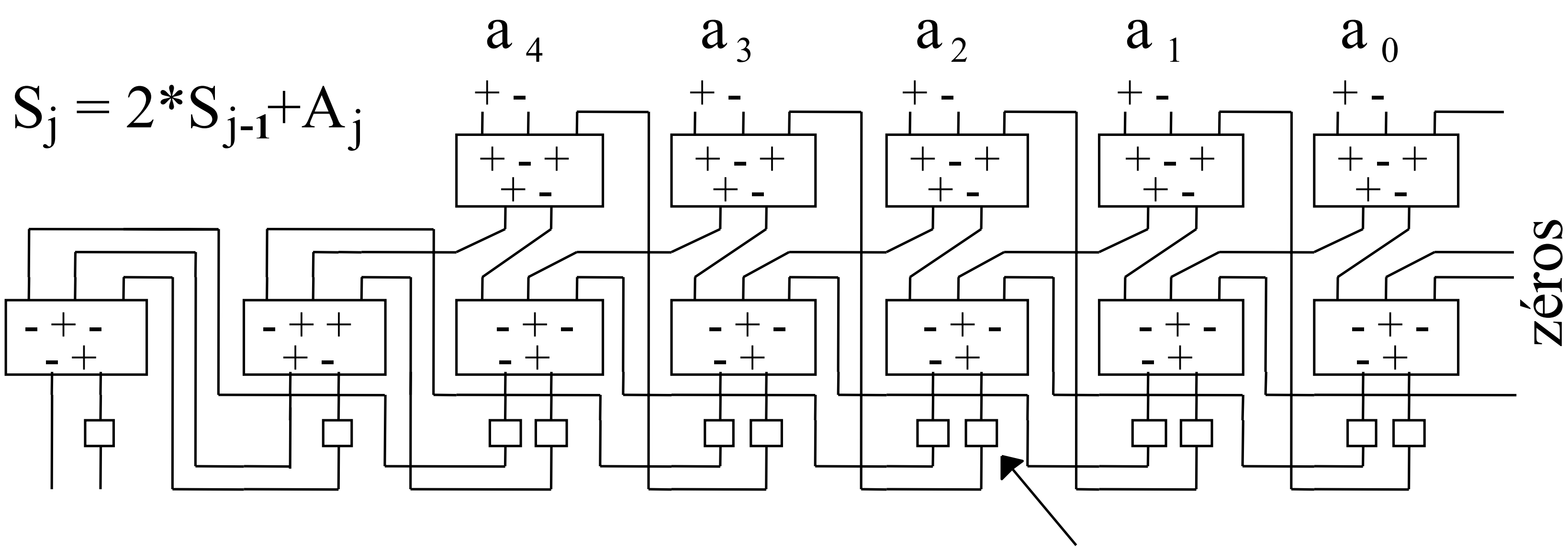
Procedure Addserie(a_i, b_i, s_i : SBD);
var counter: **integer**;
begin
counter := 2*counter + a_i + b_i ;
if counter > 2 **then** s_i := 1 {overflow}
else if counter < -2 **then** s_i := -1 {underflow}
else s_i := 0 ;
counter := counter - 4*s_i ;
end.



Addition d'une série de nombres avec résultat en-ligne

$$S = \sum_{j=m-1}^{0} A_j \, 2^j \qquad A_j = \sum_{i=0}^{n-1} a_{ij} \, 2^i \qquad a_{ij} \in \{-1, 0, 1\}$$

$$S_j = 2 * S_{j-1} + A_j$$

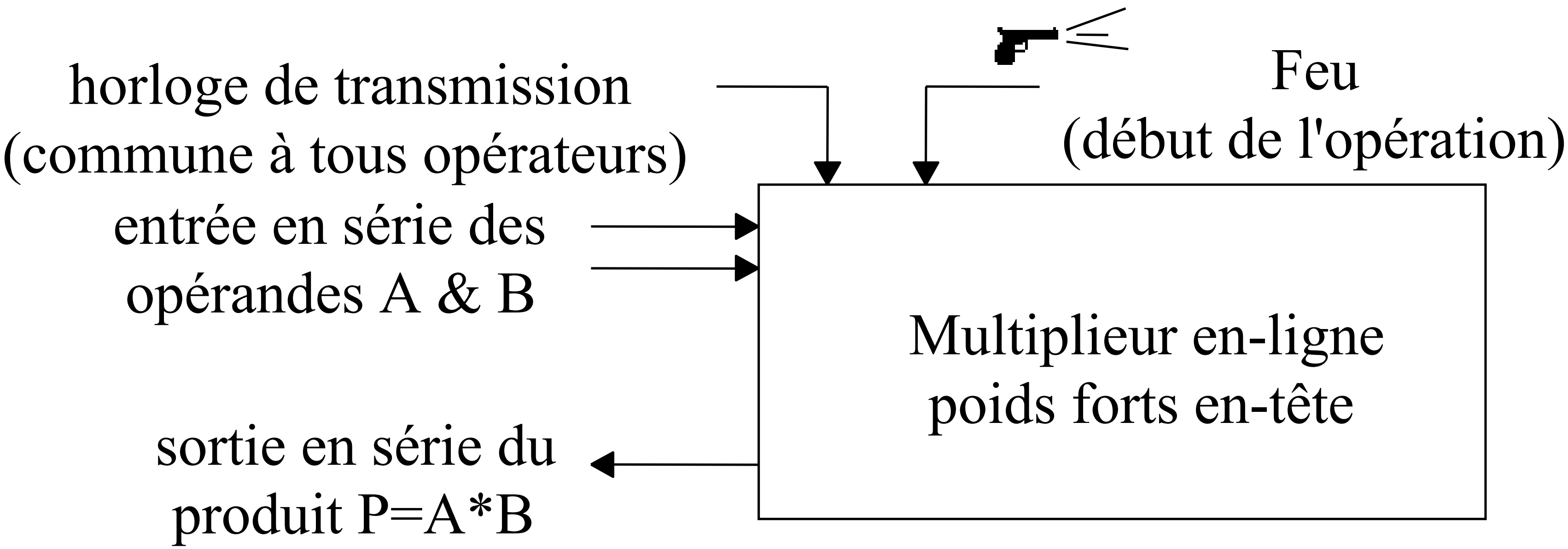


somme en-ligne
poids forte en-tête

registre d'erreur ou de récursion

Multiplieur en ligne poids forts en-tête

$$A = \sum_{i=0}^{n-1} a_i \, 2^i \qquad B = \sum_{i=0}^{n-1} b_i \, 2^i \qquad P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i \, b_j \, 2^{i+j}$$



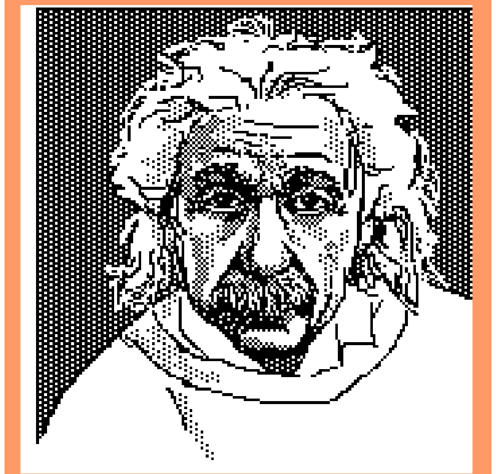
Multiplication en ligne poids forts en tête (2)

$$A = \sum_{i=0}^{n-1} a_i \, 2^i \qquad B = \sum_{i=0}^{n-1} b_i \, 2^i \qquad P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i \, b_j \, 2^{i+j}$$

$a_0 \rightarrow$	$a_0 b_3$	$a_0 b_2$	$a_0 b_1$	$a_0 b_0$
$a_1 \rightarrow$	$a_1 b_3$	$a_1 b_2$	$a_1 b_1$	$a_1 b_0$
$a_2 \rightarrow$	$a_2 b_3$	$a_2 b_2$	$a_2 b_1$	$a_2 b_0$
$a_3 \rightarrow$	$a_3 b_3$	$a_3 b_2$	$a_3 b_1$	$a_3 b_0$
	$\uparrow$ b_3	$\uparrow$ b_2	$\uparrow$ b_1	$\uparrow$ b_0

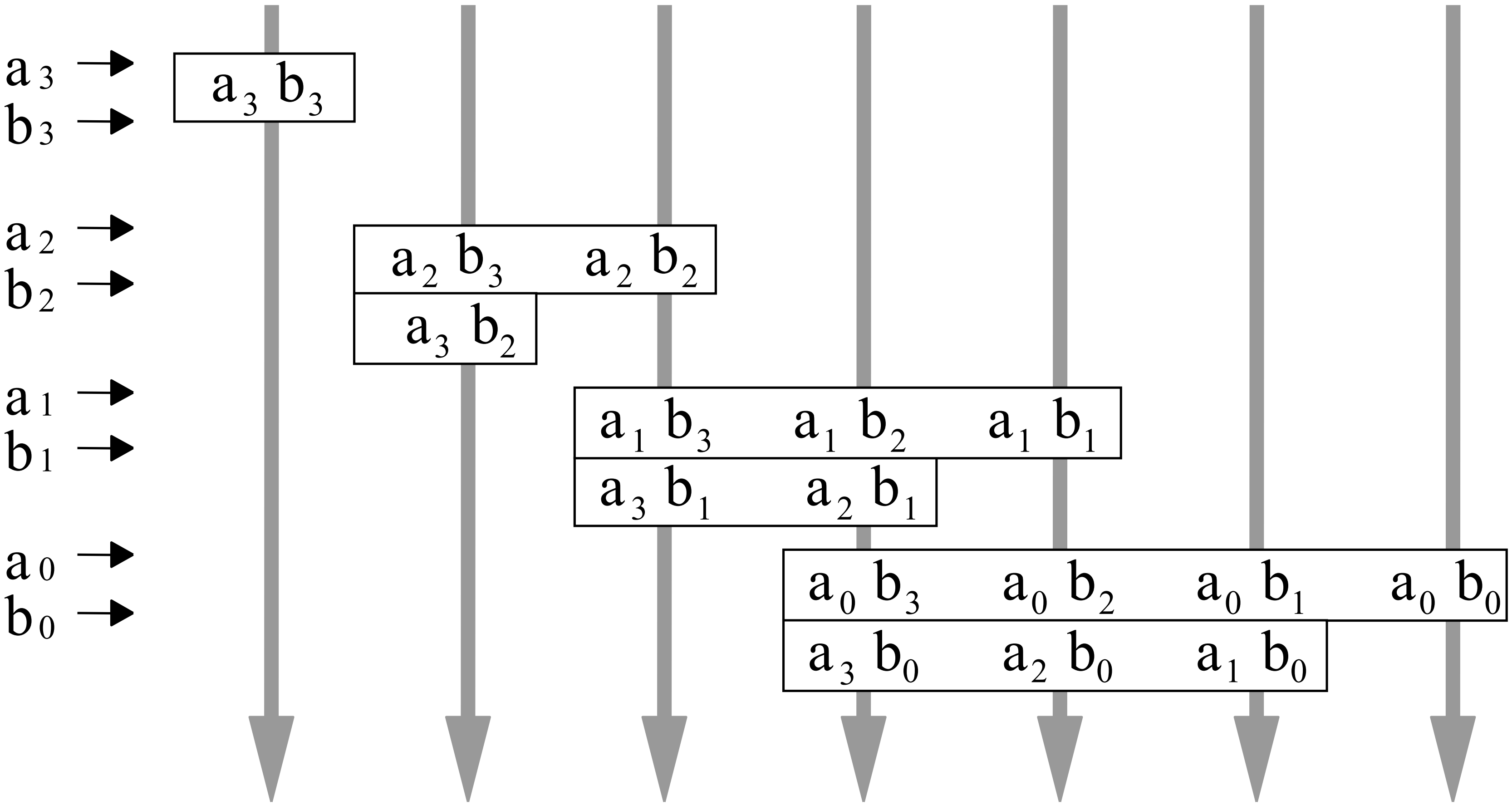
futur

présent



passé

Multiplication en ligne poids forts en tête (3)



sequence of numbers summed up to give a sequence of digits

Opérations à faire simultanément pour la multiplication en-ligne (4)

1- Ranger les chiffres reçus dans les registres A & B

$$A^j \Leftarrow A^{j-1} \,\&\, a_j \quad B^j \Leftarrow B^{j-1} \,\&\, b_j$$

2- Multiplier A et B par les chiffres reçus $\otimes$

$$A^j * b_j \;,\; B^{j-1} * a_j$$

3- Décaler et sortir le résultat partiel

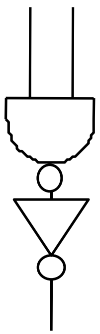
$$P^j \Leftarrow 2 * P^{j-1}$$

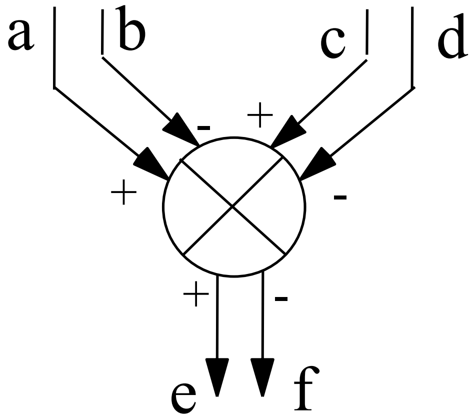
4- Ajouter les produits partiels au résultat partiel

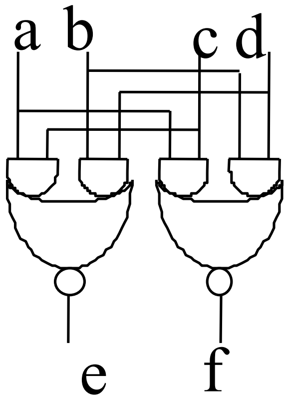
$$P^j \Leftarrow P^j + A^j * b_j + B^{j-1} * a_j$$

Multiplication de 2 chiffres

As for bits, the product of two signed digits is one digit.



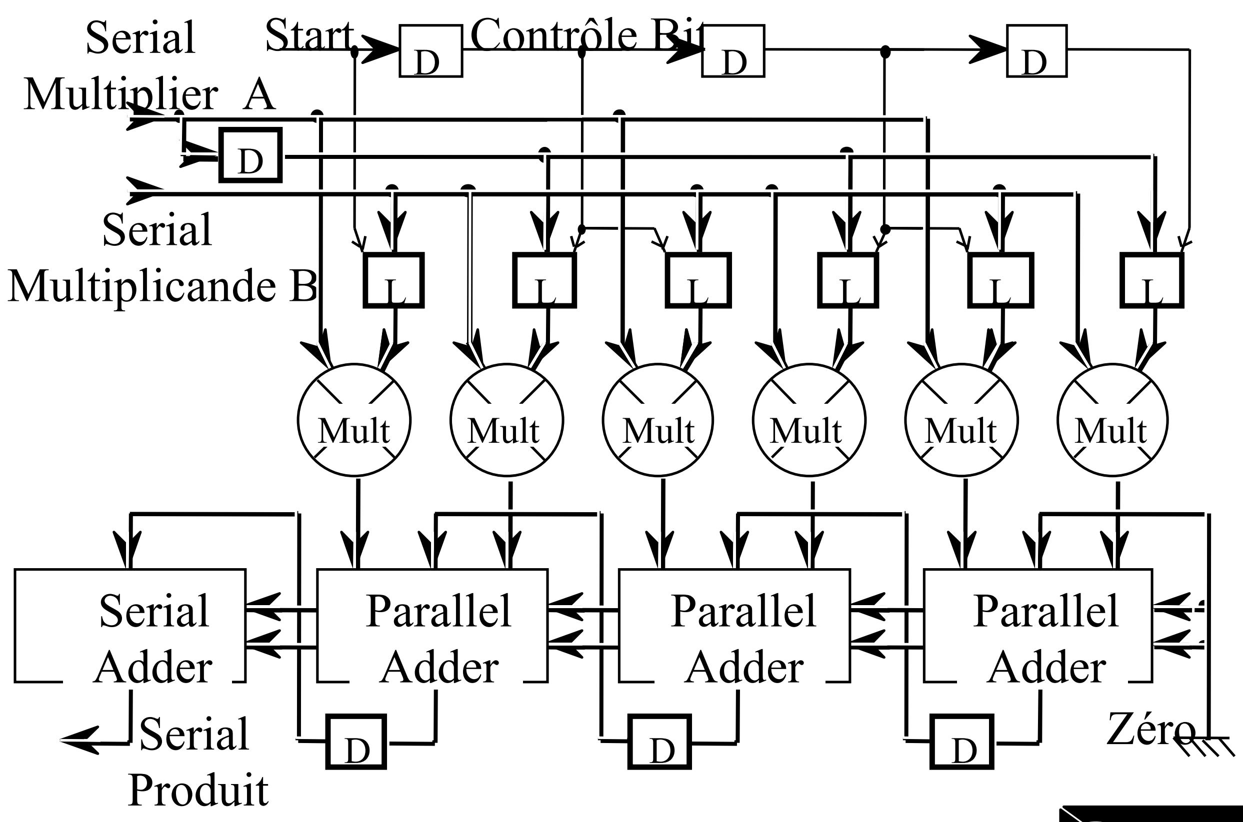




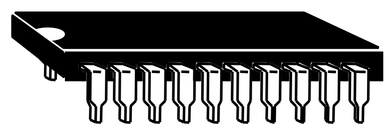
$$e = \overline{a\,c + b\,d}$$

$$f = \overline{a\,d + b\,c}$$

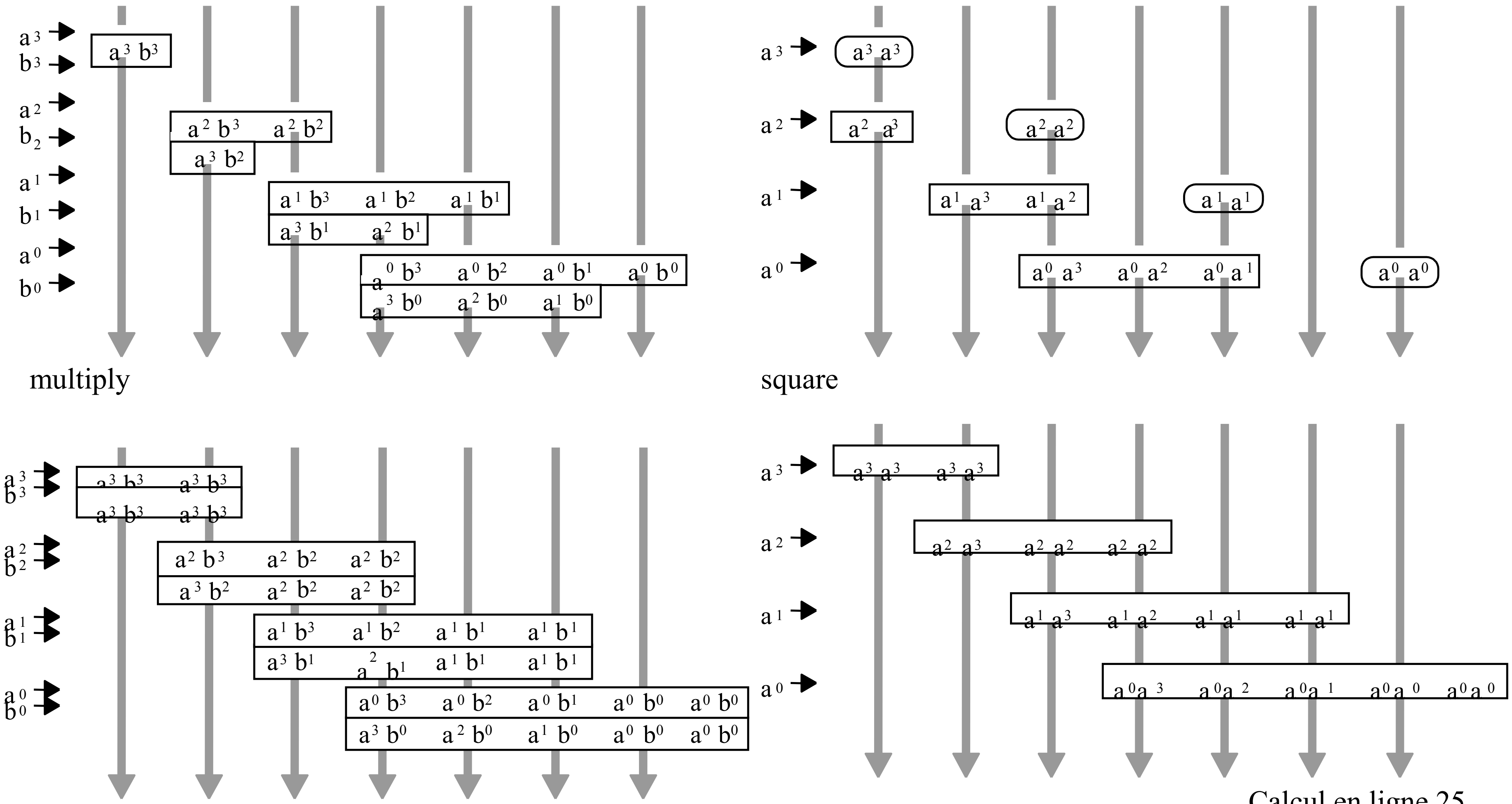
Multiplieur en-ligne poids forts en tête (5)



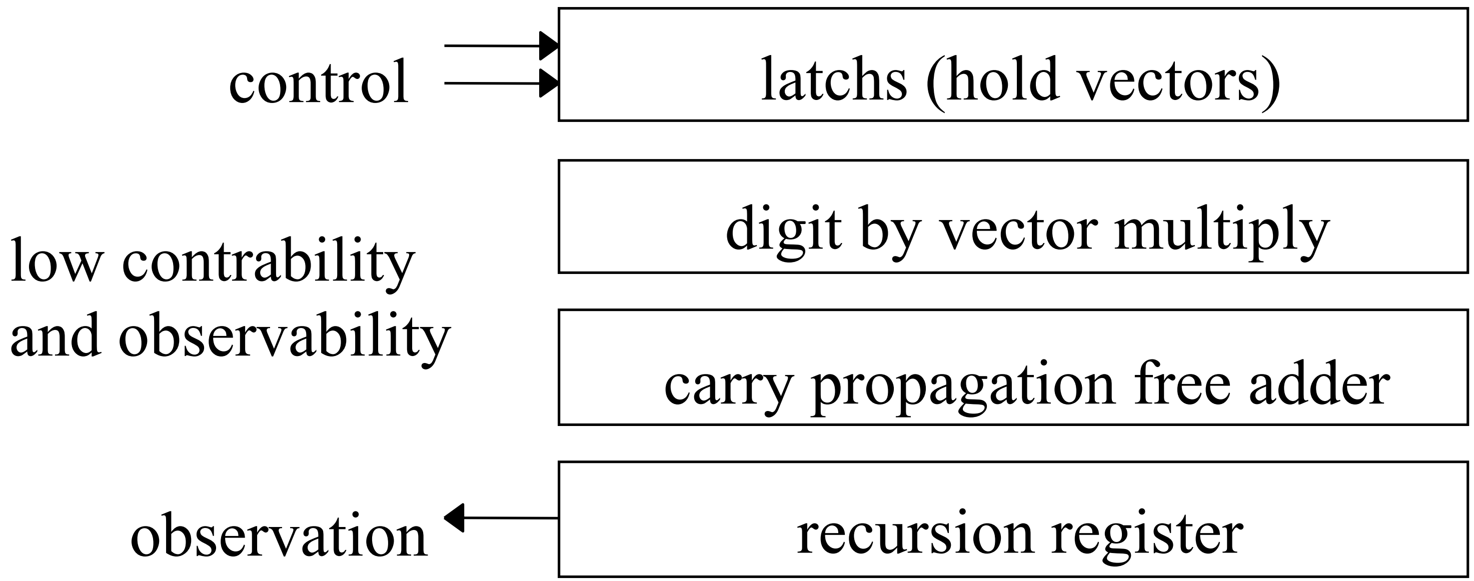
La latence vient du sommateur en-ligne



Multiplication et carré

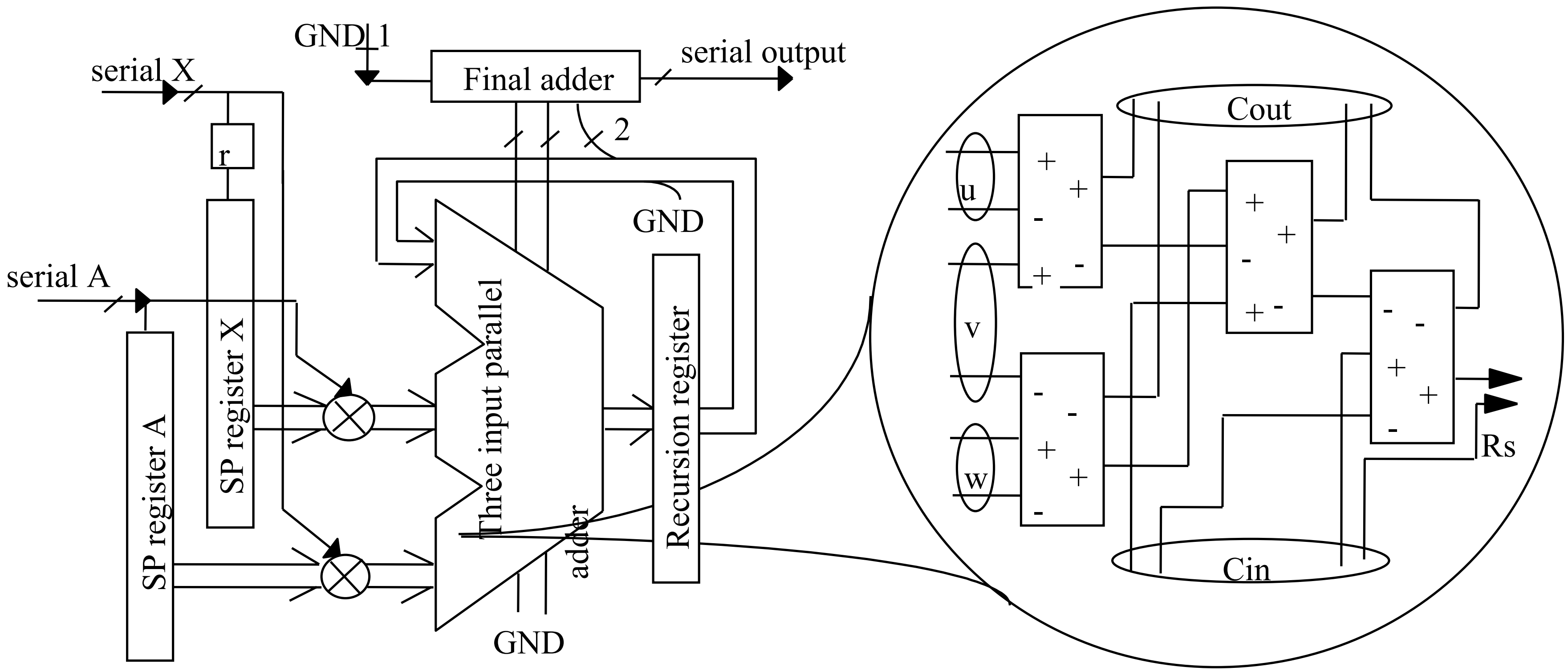


Test en ligne de multiplieur en ligne poids forts en tête



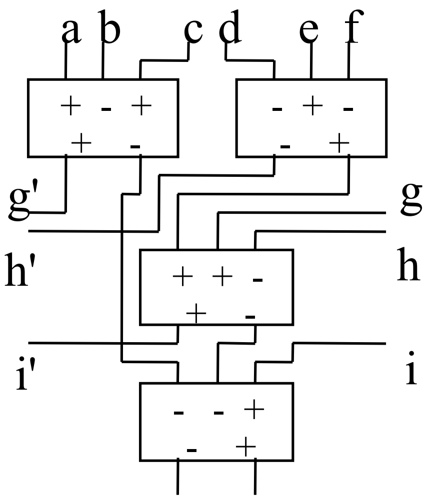
principle: exhaustive testing: no fault model
single fault : no masking
adder and recursion register to analyse the results

Test en ligne de multiplieur en ligne poids forts en tête (2)



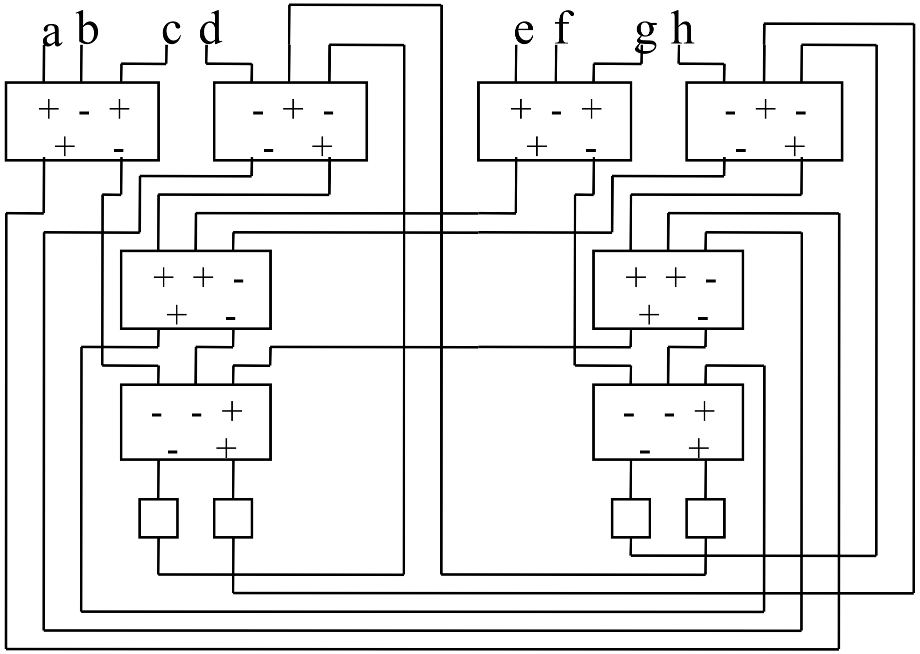
Test en ligne de multiplieur en ligne poids forts en tête (3)

The 3-input carry-free adder is C-testable



input vectors						carries		
a	b	c	d	e	f	g	h	i
0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	1	1	0
0	0	1	1	1	0	1	0	1
0	1	0	0	1	0	0	0	1
0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	1	0
1	0	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1
1	0	1	0	1	1	1	0	1
1	1	0	0	0	1	0	1	0
1	1	0	1	1	1	0	1	0
1	1	1	1	0	0	1	1	1

Test en ligne de multiplieur en ligne poids forts en tête (4)

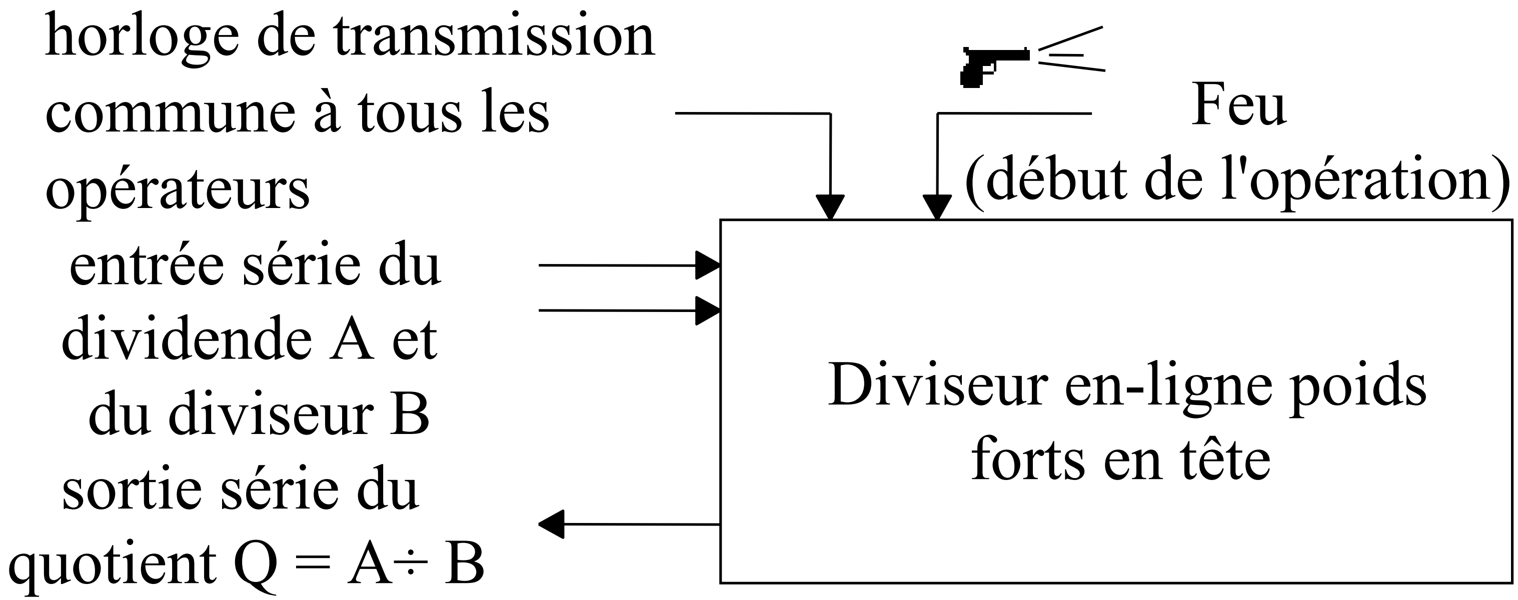


With a local feed-back loop, more vectors are required

a	b	c	d	e	f	g	h
0	1	0	0	1	1	0	0
0	0	0	0	0	0	1	1
1	1	1	0	0	0	0	1
0	0	1	1	0	0	0	0
0	0	0	1	1	1	1	0
1	0	0	1	0	1	1	0
0	1	1	0	1	0	0	1
1	0	1	0	0	1	0	0
0	1	0	0	1	0	1	0
1	1	1	0	1	0	1	0
1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	0	1	0	0	0	0	1
1	1	1	0	1	1	1	0

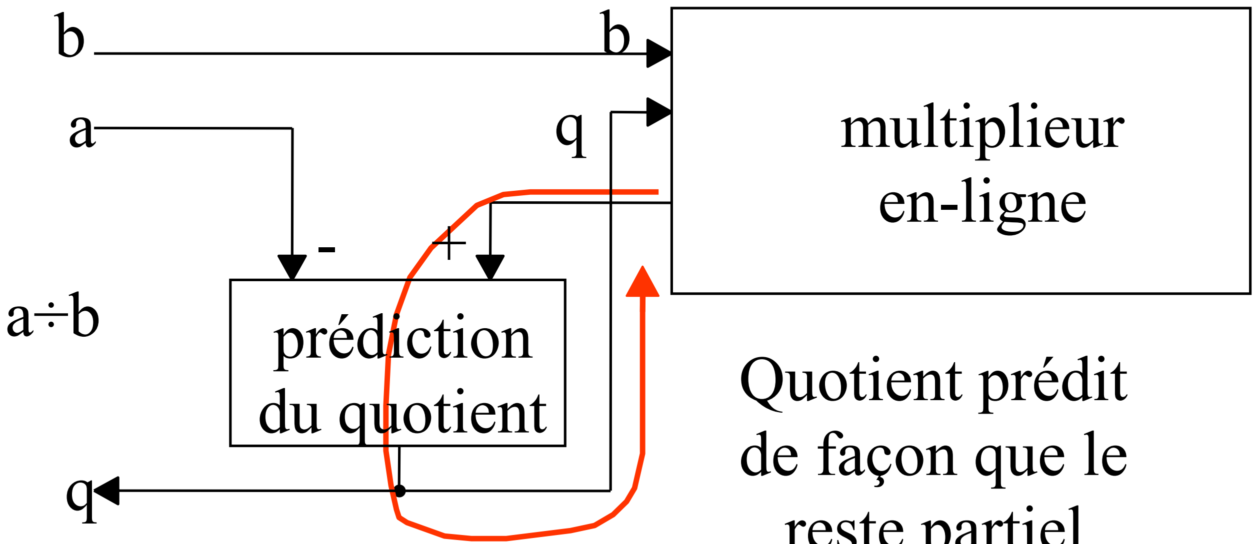
Diviseur en-ligne
(poids forts en tête)

La division s'appuie sur la multiplication en-ligne poids forts en tête



Division en-ligne (2)

Principe



boucle de rétroaction

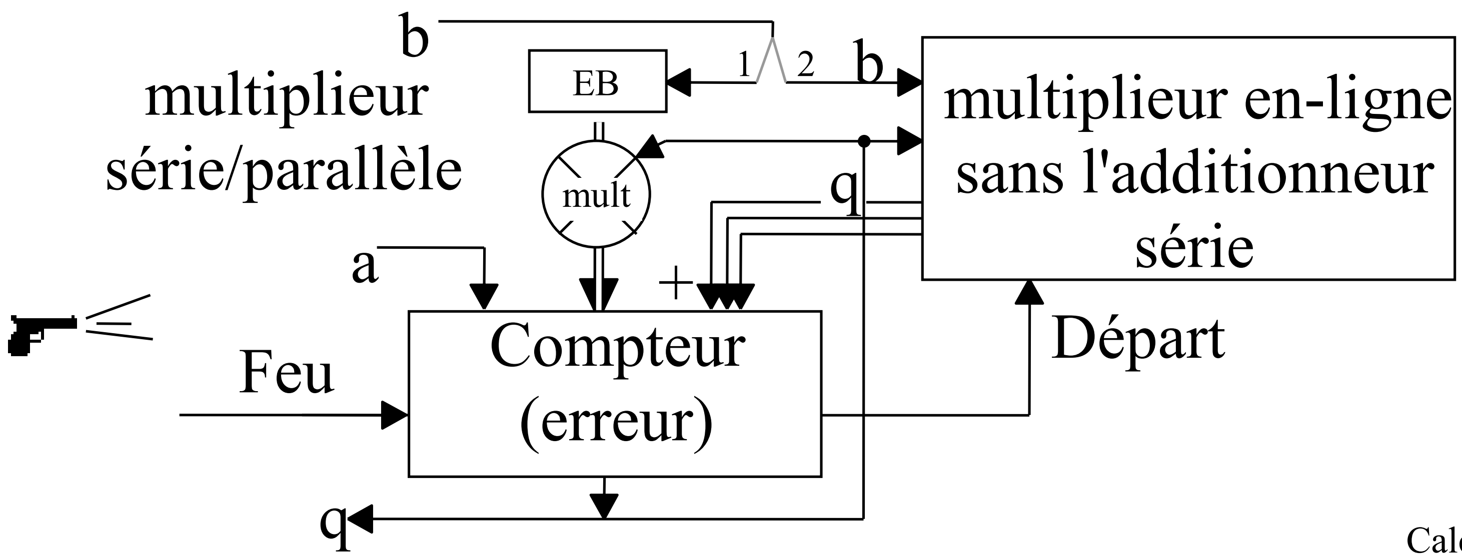
Quotient prédit de façon que le reste partiel

$Q^J * B^J - A^j \rightarrow 0$

Division en-ligne (3)

Pour assurer la stabilité (ou convergence) de l'algorithme, on doit
  1- raccourcir la boucle de rétroaction
  2- utiliser une partie du diviseur pour prédire le quotient

La multiplication est partiellement série/parallèle (4 chiffres)
partiellement série/série (en-ligne)



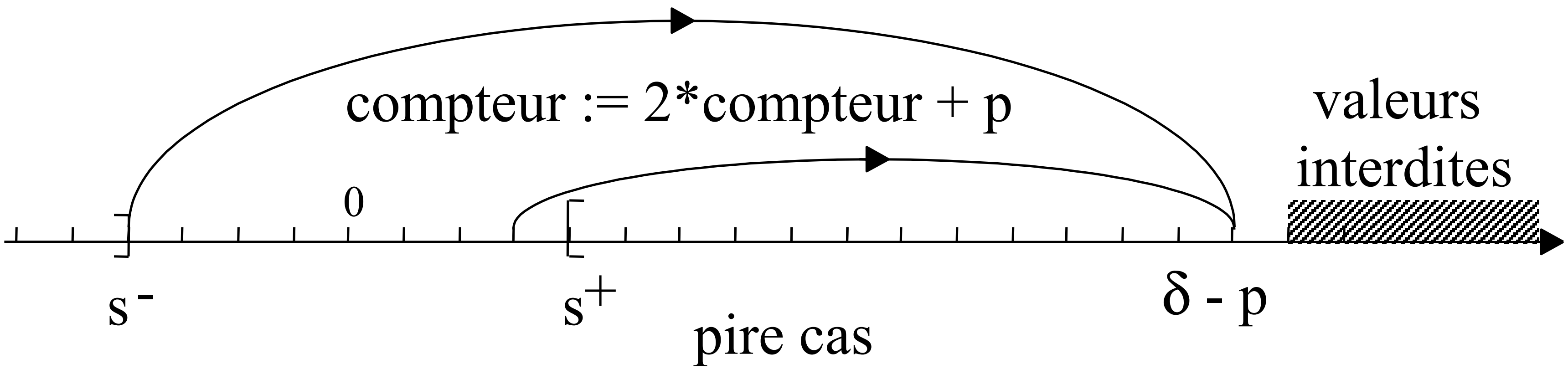
Division en-ligne (4)

A chaque cycle le compteur de prédiction du quotient reçoit
- deux fois son ancienne valeur (connue)
- plus une valeur inconnue p ($-4 \leq p \leq +4$)

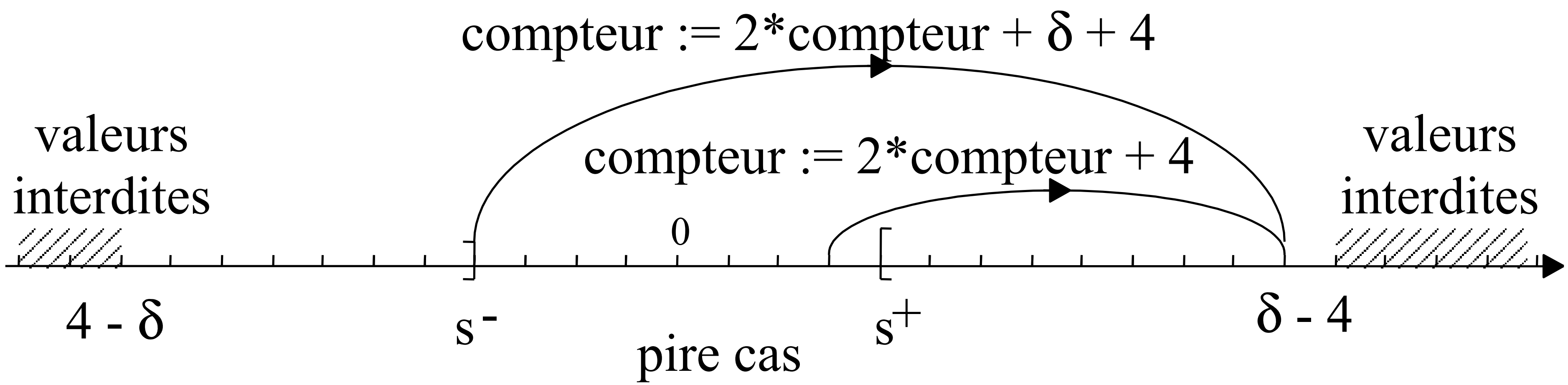
On rend le compteur petit en lui ajoutant ou soustrayant une valeur inconnue EB ($\delta \leq | EB | \leq 2*\delta -1$) en comparant son ancienne valeur à 2 seuils s+ et s-

Les seuils s+ et s- et δ doivent être faciles à comparer

compteur := 2*compteur - δ + p

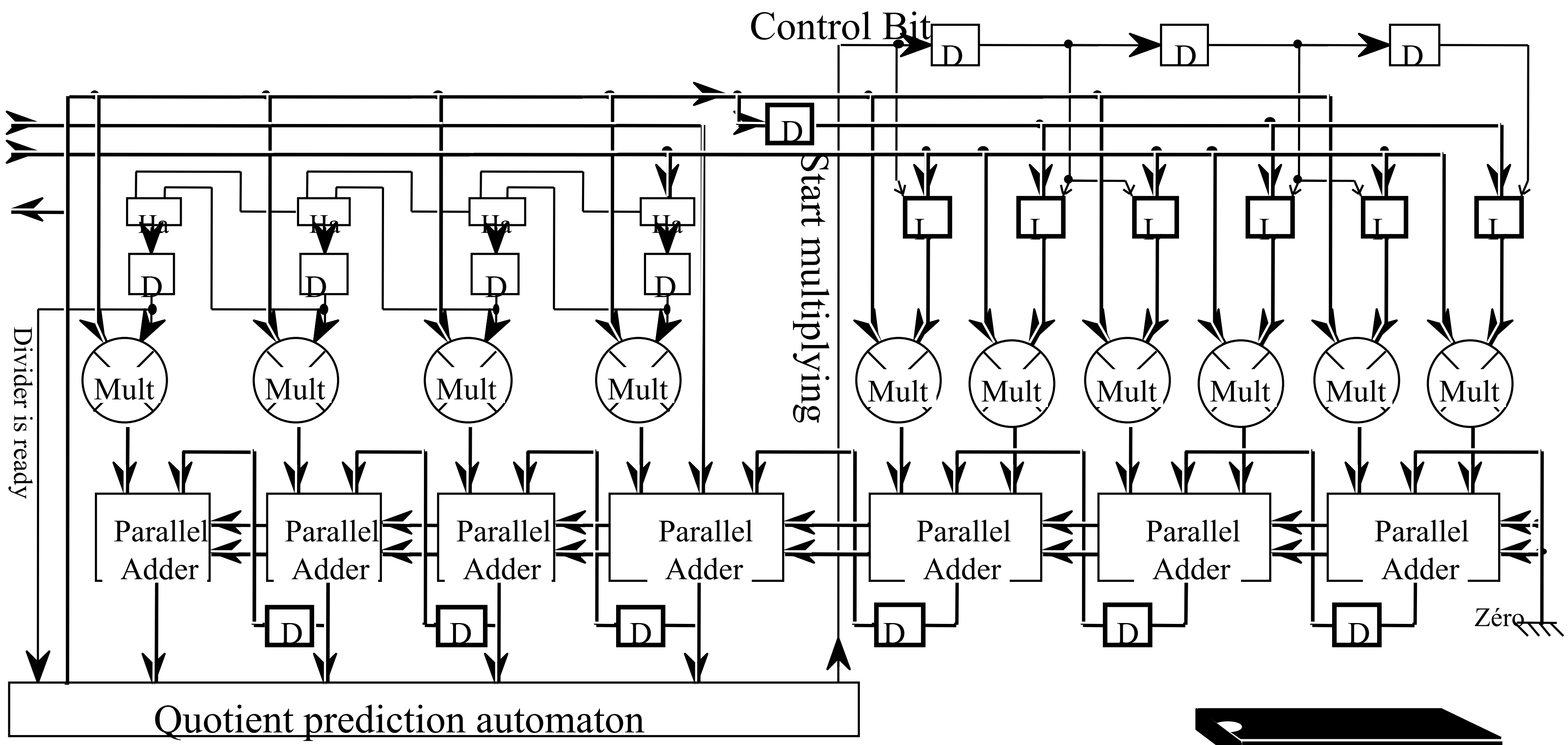


Division en-ligne (5)

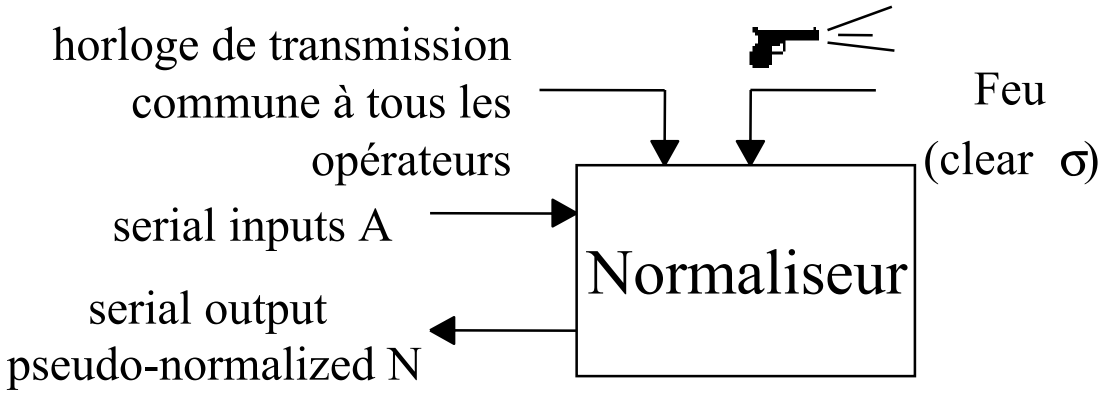


$$2*(s^{+}-1) + 4 \leq \delta - 4$$
$$2*s^{-} + \delta + 4 \leq \delta - 4$$
$$2*(s^{-}+1) - 4 \geq 4 - \delta$$
$$2*s^{+} - \delta - 4 \geq 4 - \delta$$

Diviseur en-ligne (6)

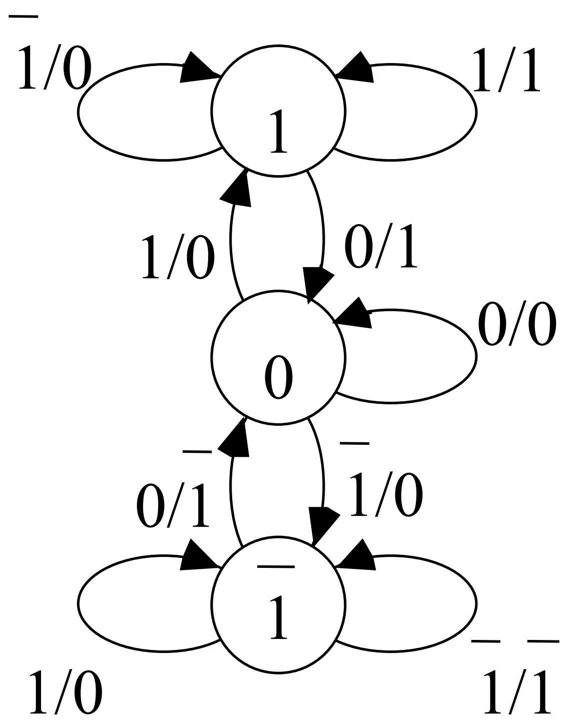


Normaliseur en ligne




```
procedure normalize (ai, ni : SBD) ;
var σ : SBD ;
begin
      σ:= 2 * σ+ai;
      if σ ≥ 2 then ni := 1
      else if σ ≤ 2 then ni := -1
      else ni := 0 ;
      σ:= σ - 2 * ni;
end .
```

values of σ

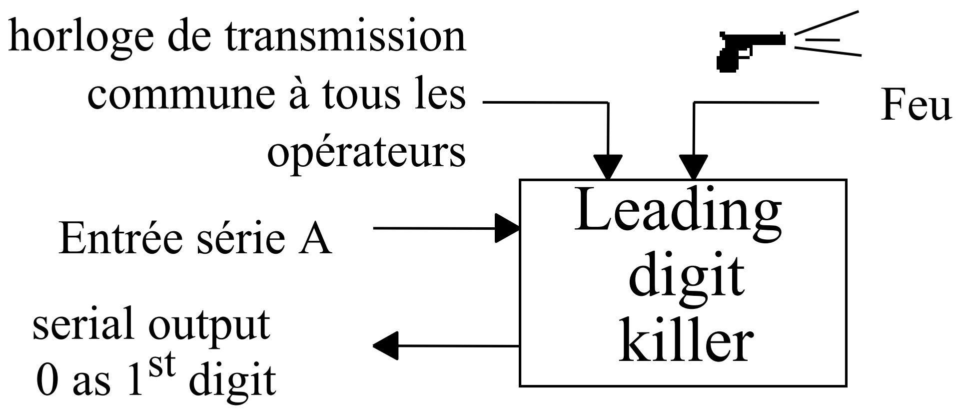


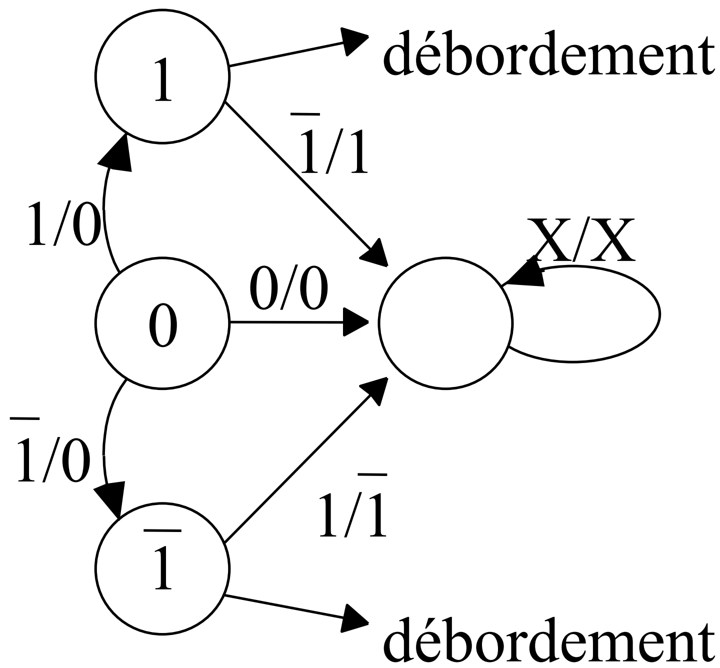
Détection de débordement

$$|\mathbf{A}| < \mathbf{2}^{\mathbf{n}}$$

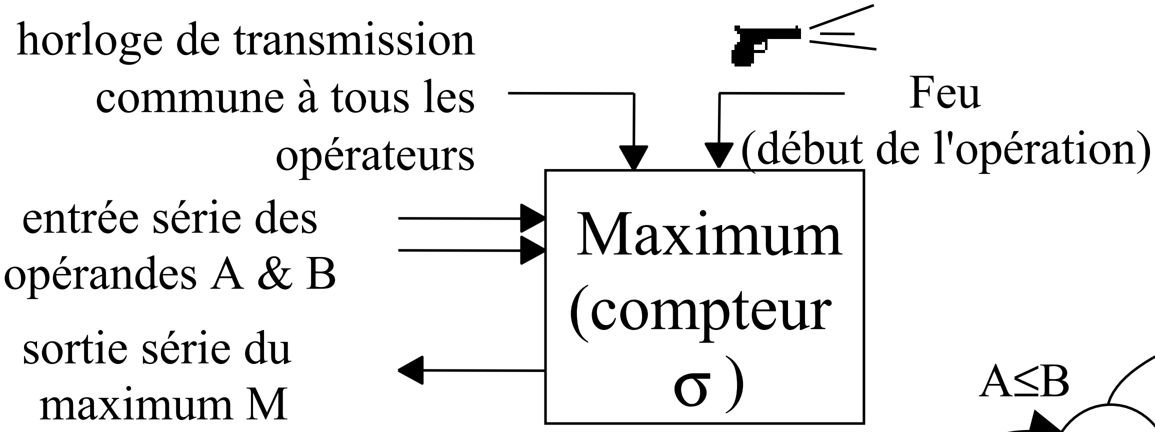
valeurs de σ

Force $a_{\mathbf{n}}$ à 0

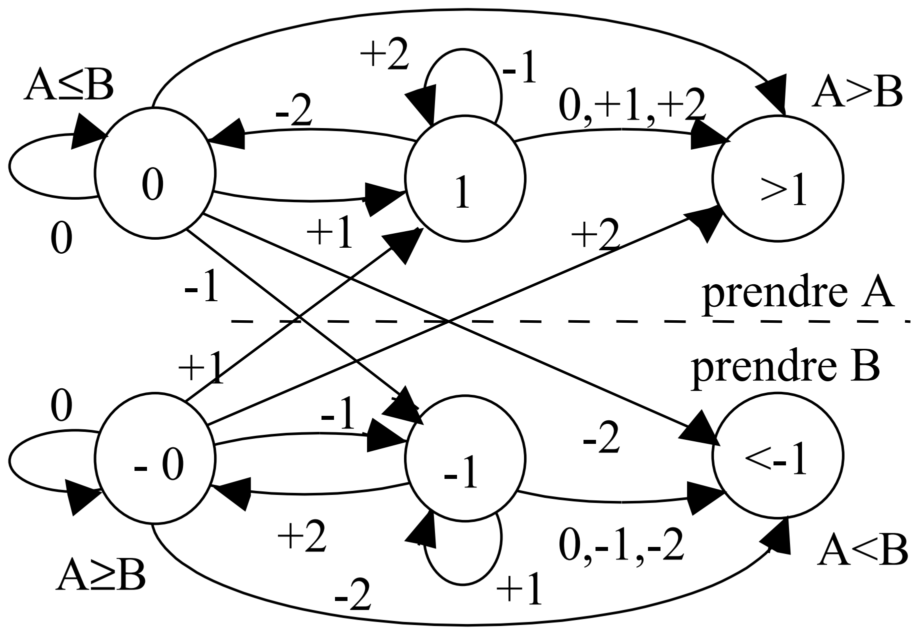




Maximum en-ligne de nombres redondants

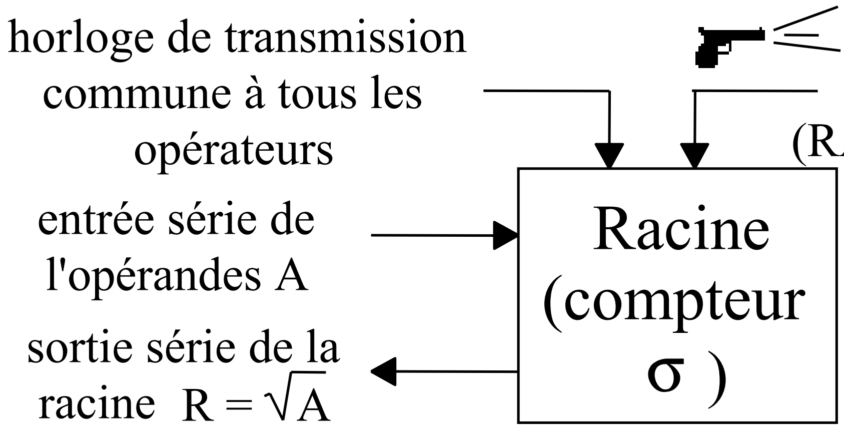


valeurs de σ

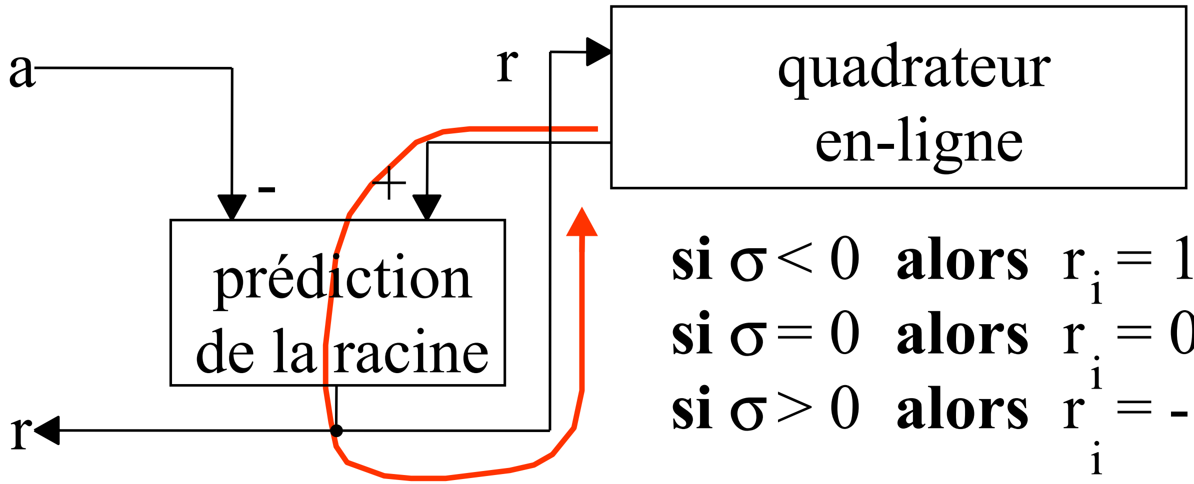


if σ_{j-1} **in** {-1,0,+1}
then $\sigma_j = 2*\sigma_{j-1}+(a_j - b_j)$ **else** $\sigma_j = \sigma_{j-1}$;
if $\sigma_j > 0$ **then** up = **true**;
if $\sigma_j < 0$ **then** up = **false** ;
if up **then** $m_j = a_j$ **else** $m_j = b_j$;

Racine carrée en-ligne de nombres redondants



La racine carrée est sensible à la parité du rang du chiffre le plus significatif du radicande

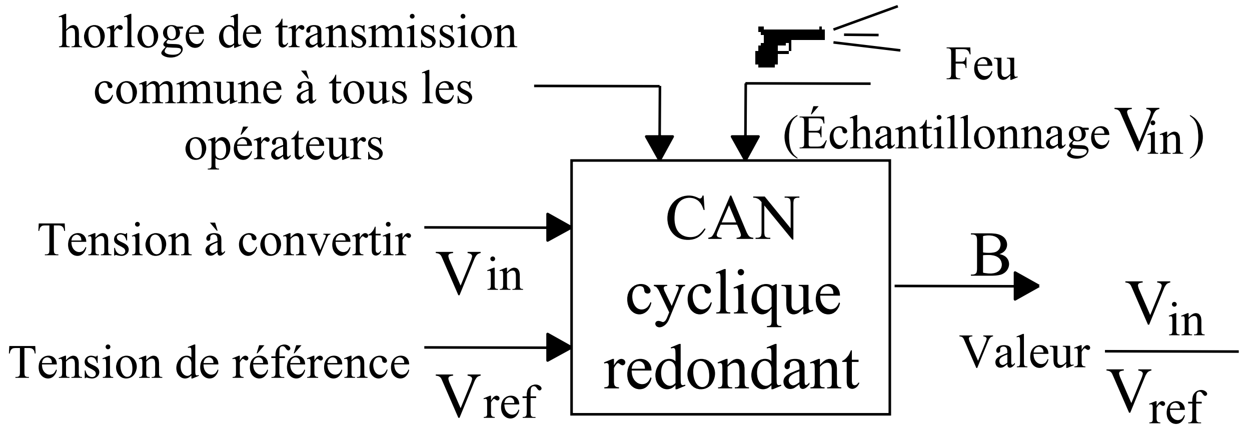


si $\sigma < 0$ **alors** $r_i = 1$

si $\sigma = 0$ **alors** $r_i = 0$

si $\sigma > 0$ **alors** $r_i = -1$

Convertisseur analogique-numérique redondant



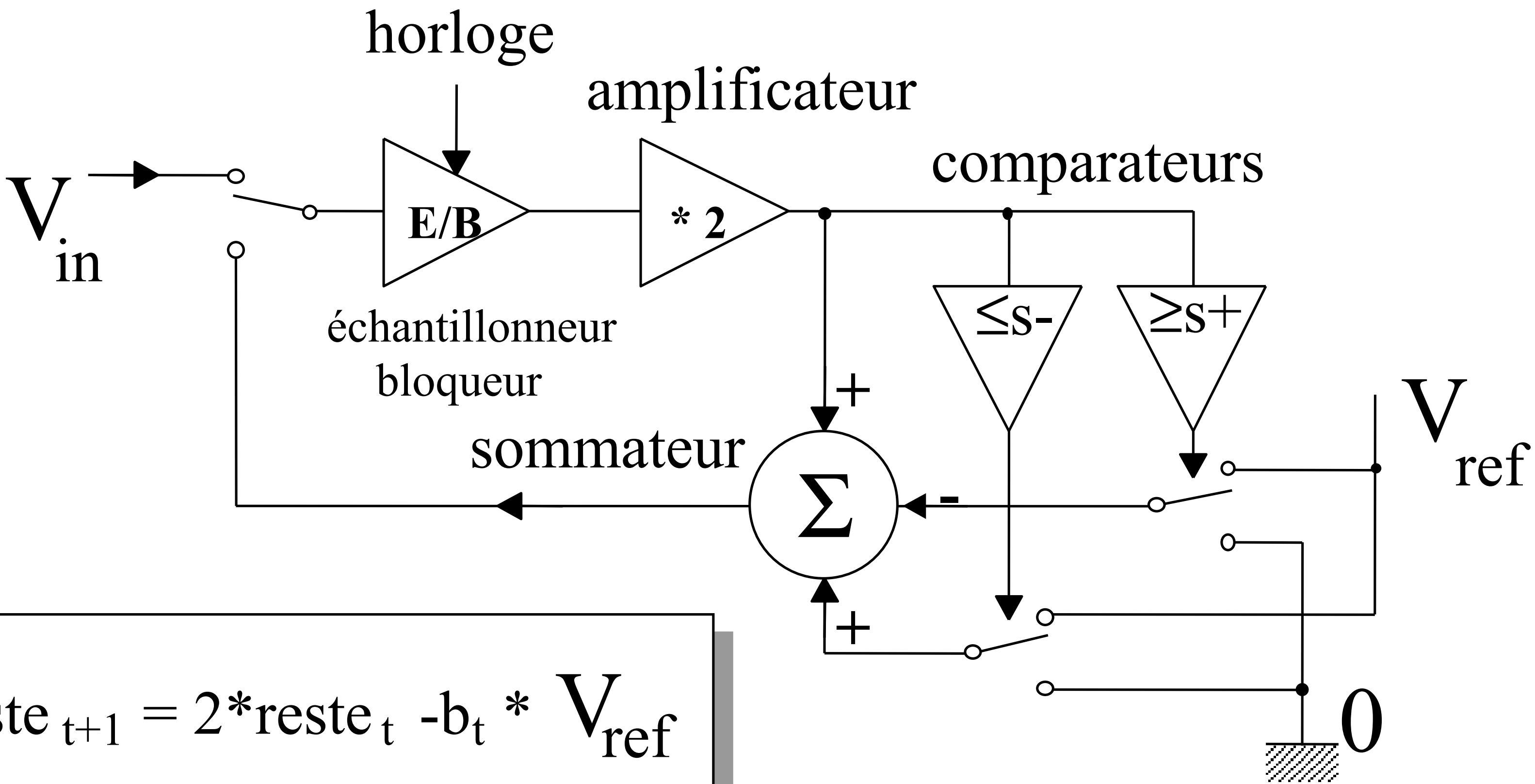
$reste_0 := V_{in}$;

si $reste_t \geq (V_{ref}/2 \pm V_{ref}/2)$ **alors** $reste_{t+1} := 2*(reste_t - V_{ref})$

sinon **si** $reste_t \leq -\ (V_{ref}/2 \pm V_{ref}/2)$ **alors** $reste_{t+1} := 2*(reste_t + V_{ref})$

sinon $reste_{t+1} := 2*(reste_t)$

Convertisseur analogique-numérique redondant (2)

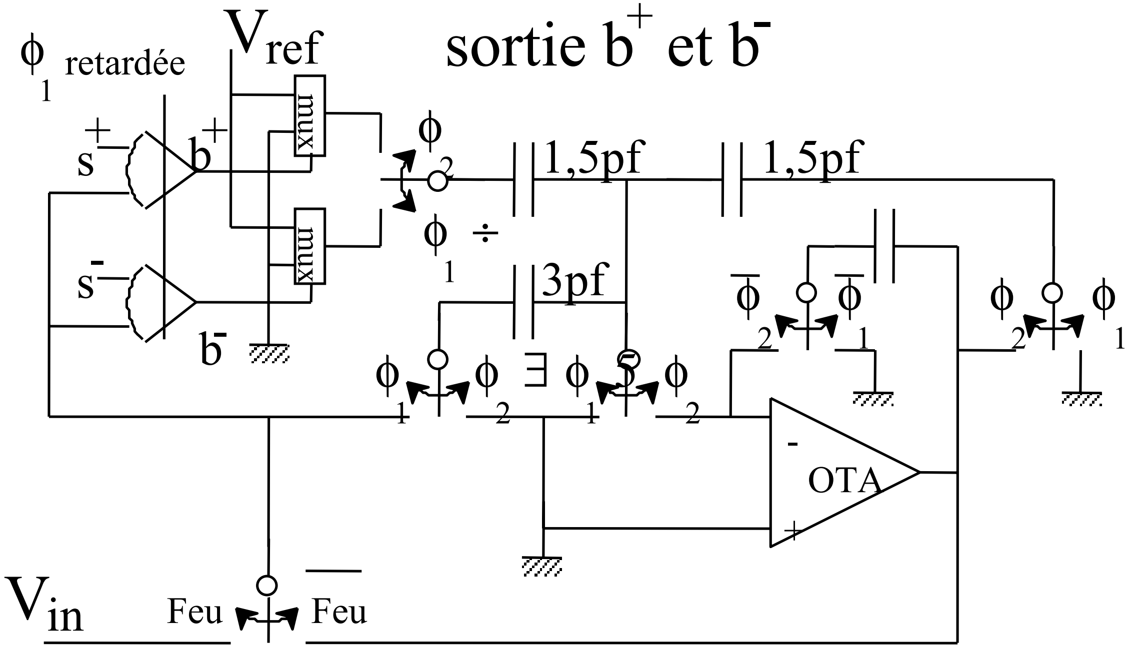


$$reste_{t+1} = 2*reste_t - b_t * V_{ref}$$

$$b_t \quad \{-1,0,+1\}$$

Convertisseur analogique-numérique redondant (3)

(4)



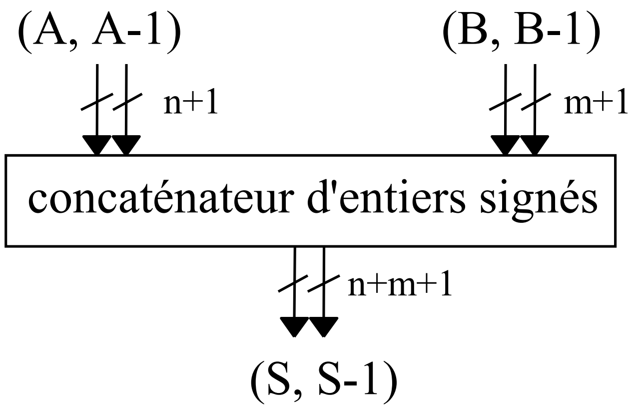
$$\phi_1 \wedge \phi_2 = 0$$

Les phases ϕ_1 et ϕ_2 ne se recouvrent pas

Calcul en ligne 42

Conversion "à la volée" du redondant

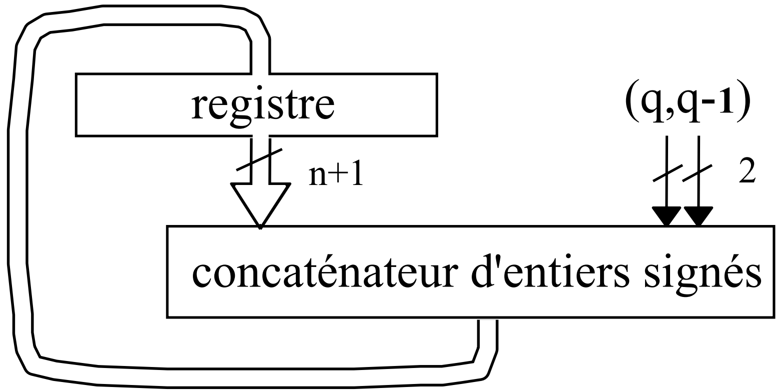
Concaténation de A et B
$S := A * 2^m + B$



si $B \geq 0$ **alors** $S := A \,\&\, B$
sinon $S := (A-1) \,\&\, B$;
si $(B-1) \geq 0$ **alors** $(S-1) := A \,\&\, (B-1)$
sinon $(S-1) := (A-1) \,\&\, (B-1)$;

(le bit de signe de B est omis dans la concaténation &)

q	q-1	bits
1		01
0	0	00
-1	-1	11
	-2	10

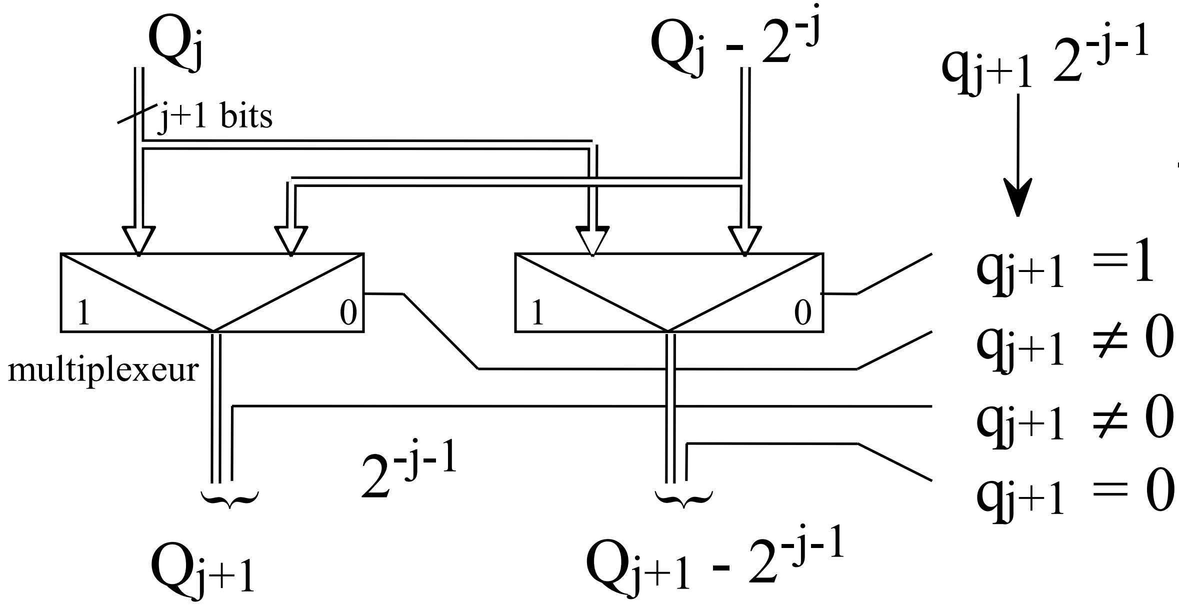


Conversion "à la volée" du redondant (2)

premier $\longrightarrow$ dernier

suite $q_0\, q_1\, q_2 \qquad q_n \qquad q_i \quad \{-1,0,+1\}$

$$Q_j = \sum_{i=0}^{j} p_i 2^{-i} \quad p_i \quad \{0,1\} \qquad Q_0 = q_0\,,\ Q_0 - 1 = \overline{q_0} \qquad q_0 \neq 0$$



$$- 2^{-j-1} = - 2^{-j} + 2^{-j-1}$$

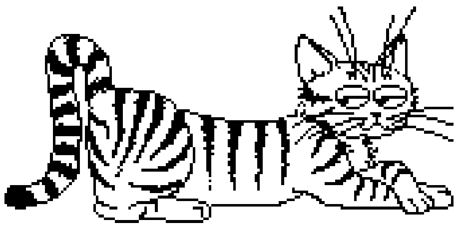
Combinaisons d'opérateurs en-ligne

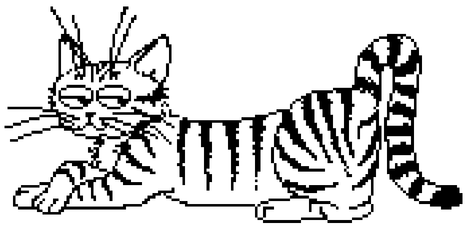
Certains opérateurs en-ligne parallélisent implicitement leur(s) opérande(s) ou leur résultat.

Cette propriété peut être exploitée pour construire des opérateurs complexes avec
- un coût moindre
- une latence moindre (mais une période supérieure)

Par exemple :

Opérateur	coût par chiffre	latence
$C = A^2$	76	3
$S = A + B$	0	2
$R = \sqrt{A}$	76	1
$D=\sqrt{A^2 + B^2}$	126	**1**

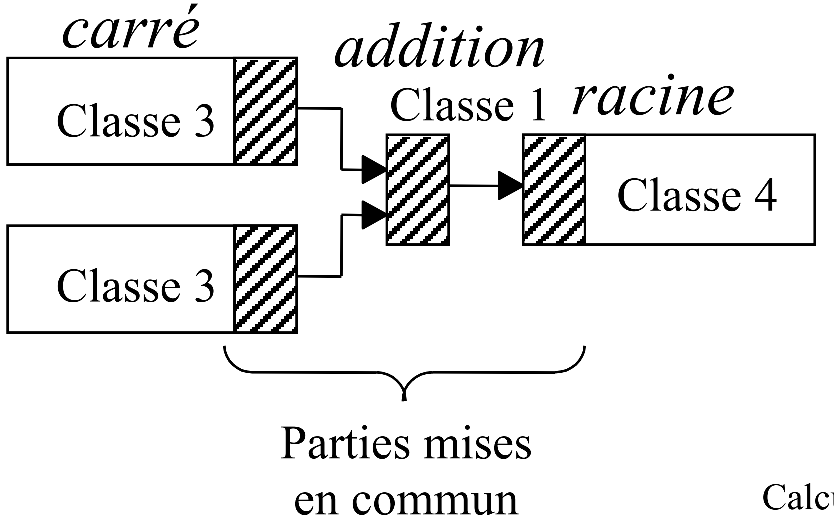




Classes d'opérateurs en-ligne

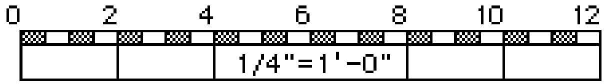
Opérateur	
Abs/max/min	1
Add/Sous $S = A \pm B$	2
Multiplication $P = A * B$	3
Carré $C = A^2$	
Division $Q = A \div B$	4
Racine carrée $R = \sqrt{A}$	

Classement des opérateurs en-ligne en fonction de leur affinité (possibilité de partage)



Opérateur	# classe	# variante	Opérandes A et B		résultat en série	résultat en parallèle	opérande en parallèle	coût matériel de l'opérateur en nombre de :				
			A	B				SP register	Parallel inpuy register	vector/digit multiply⊗	2-input adder	Recursion register
Abs/max/min	1	/	ser	ser	M	na	na	0	0	0	0	0
Add/Sous $S = A \pm B$	2	1	par	par	na	S	A,B	0	0	0	1	0
		2	ser	ser	S	na	na	0	0	0	0	0
Multiplication $P = A * B$	3	1	par	ser	P	na	/	0	1	1	1	1
		2	ser	ser	P	na	A,B	2	0	2	2	1
Carré $C = A^2$		1	ser		C	na	A	1	0	1	1	1
		2	par/ser		C	na	/	0	1	1	1	1
Division $Q = A \div B$	4	1	par	par	Q	na	/	0	0	1	1	1
		2	par	ser	Q	Q	B	2	0	2	2	1
		3	ser	par	Q	na	/	0	0	1	1	1
		4	ser	ser	Q	Q	B	2	0	2	2	1
Racine carrée $R = \sqrt{A}$		1	par		R	na	/	1	0	1	1	1
		2	ser		R	R	na	0	1	1	1	1

Règles de combinaison d'opérateurs



Règle 1 : Deux opérateurs de classe 3 partageant la même entrée peuvent partager le même SP-registre.

Règle 2 : Si la sortie d'un opérateur de classe 4 est reliée à l'entrée d'un opérateur de classe 3 alors ils peuvent partager un SP-registre.

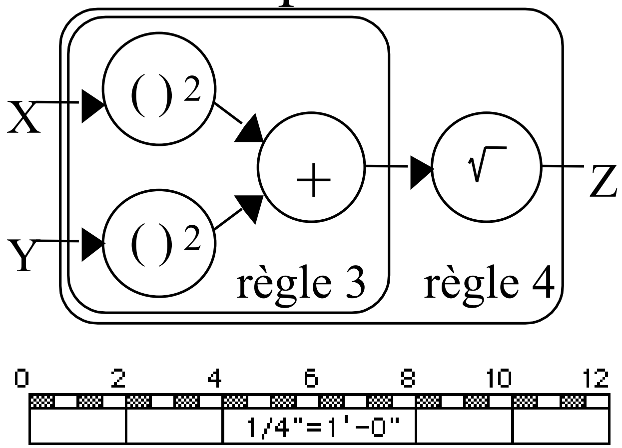
Règle 3 : Si les sorties d' opérateurs de classe 3 sont ajoutées (ou soustraites) alors ils peuvent partager leur registre de récursion.
Le résultat de cette fusion est de classe 3.

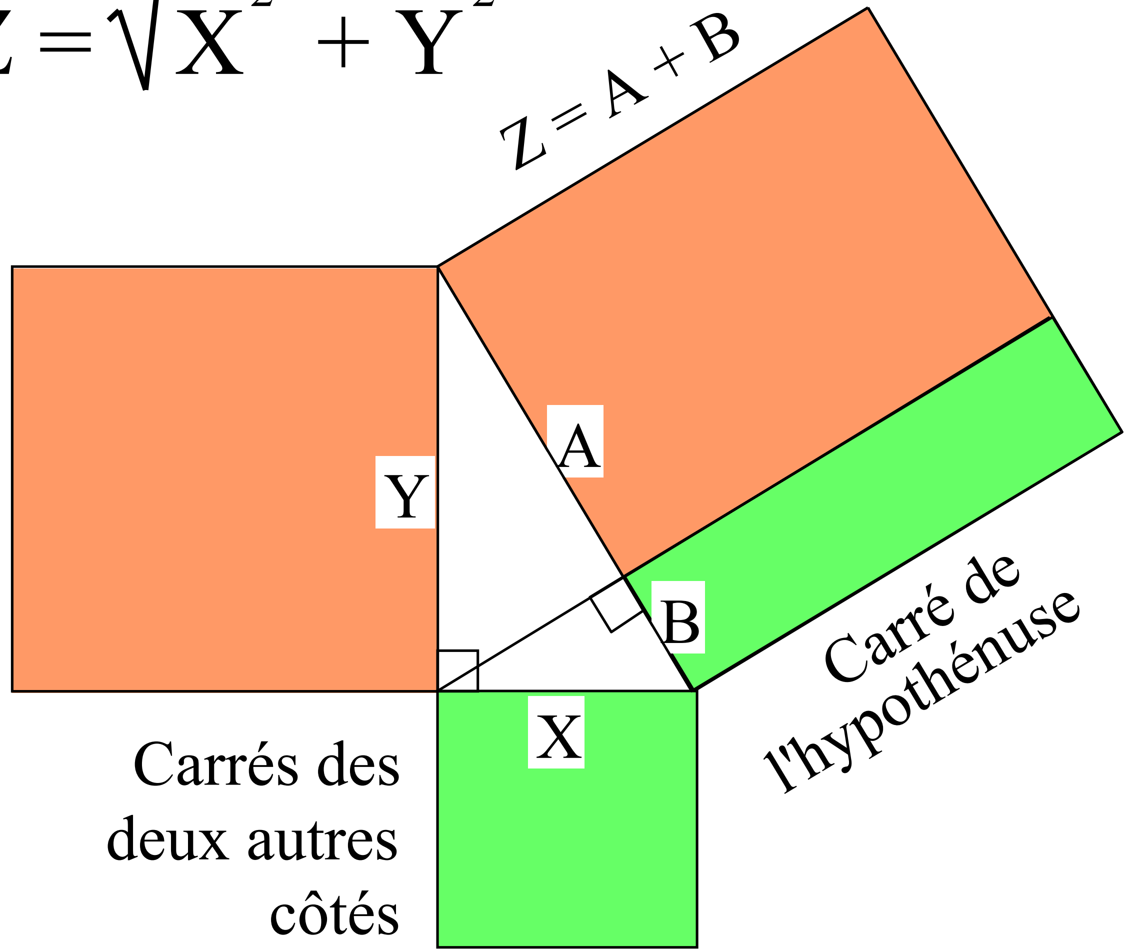
Règle 4 : Si la sortie d'un opérateur de classe 3 est reliée à l'entrée d'un opérateur de classe 4 ils peuvent partager leur registre de récursion.

Exemple 1: processeur Euclidien

$$Z = \sqrt{X^2 + Y^2}$$

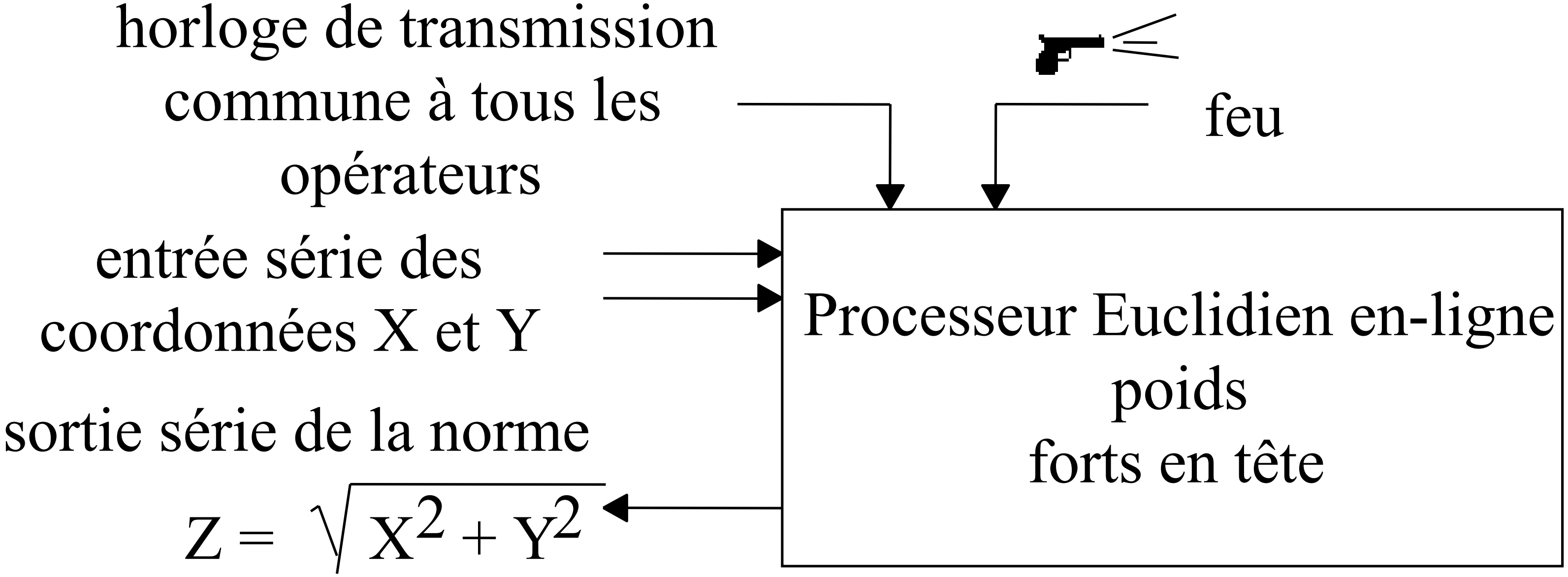
Règles d'assemblage des opérateurs





Norme Euclidienne en-ligne

$$Z = \sqrt{X^2 + Y^2}$$

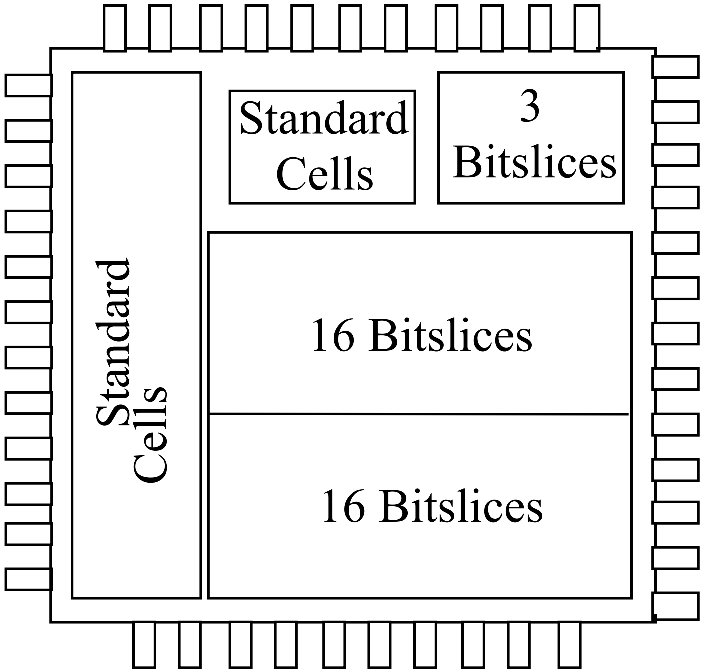


Norme Euclidienne en-ligne (2)

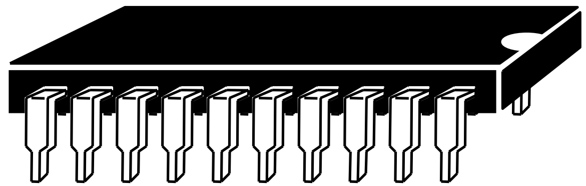
$$Z = \sqrt{X^2 + Y^2} \text{ en-ligne}$$

avec 32 chiffres de précision

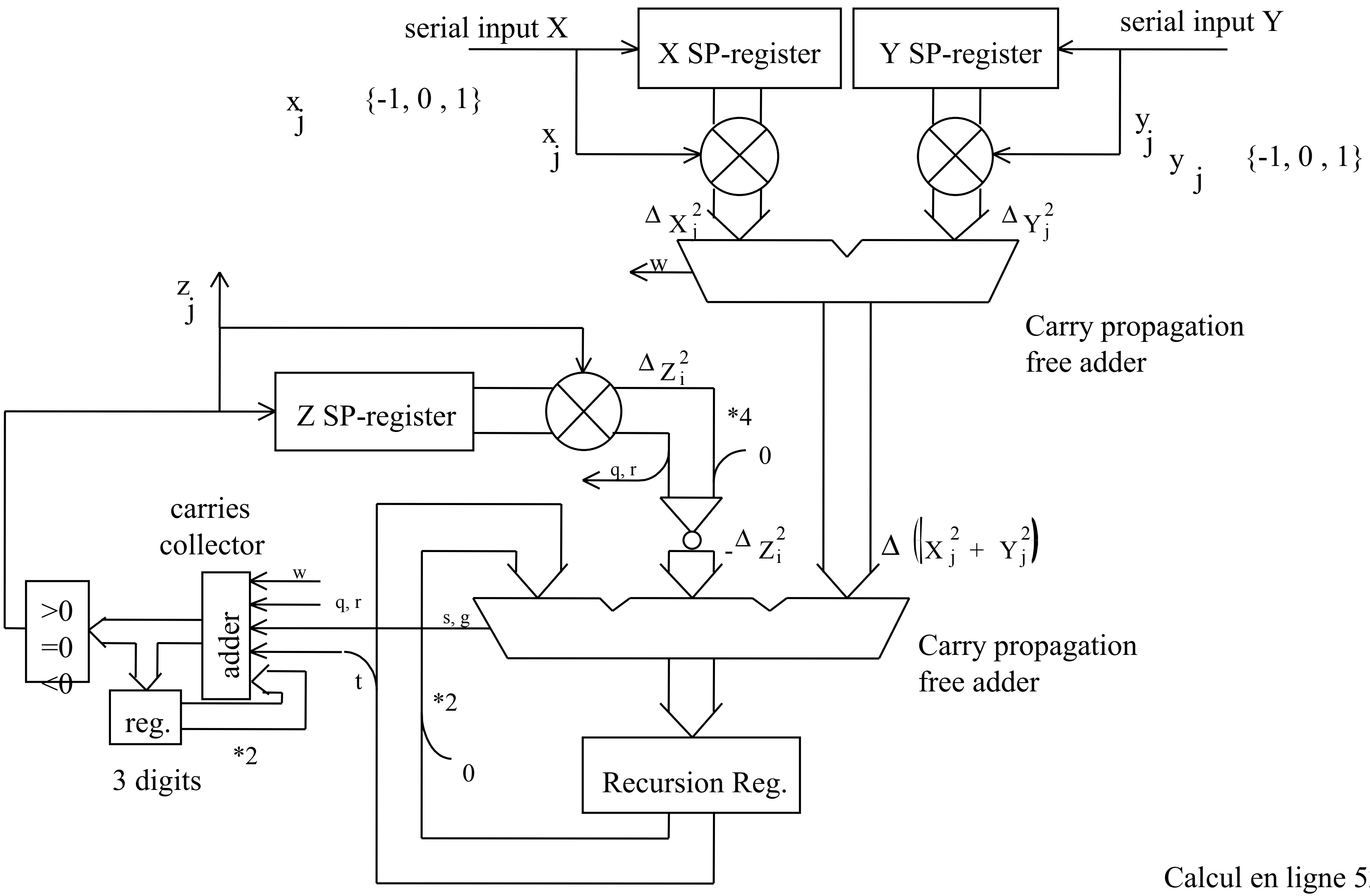
Application des règles : gain 19%



$\Rightarrow$



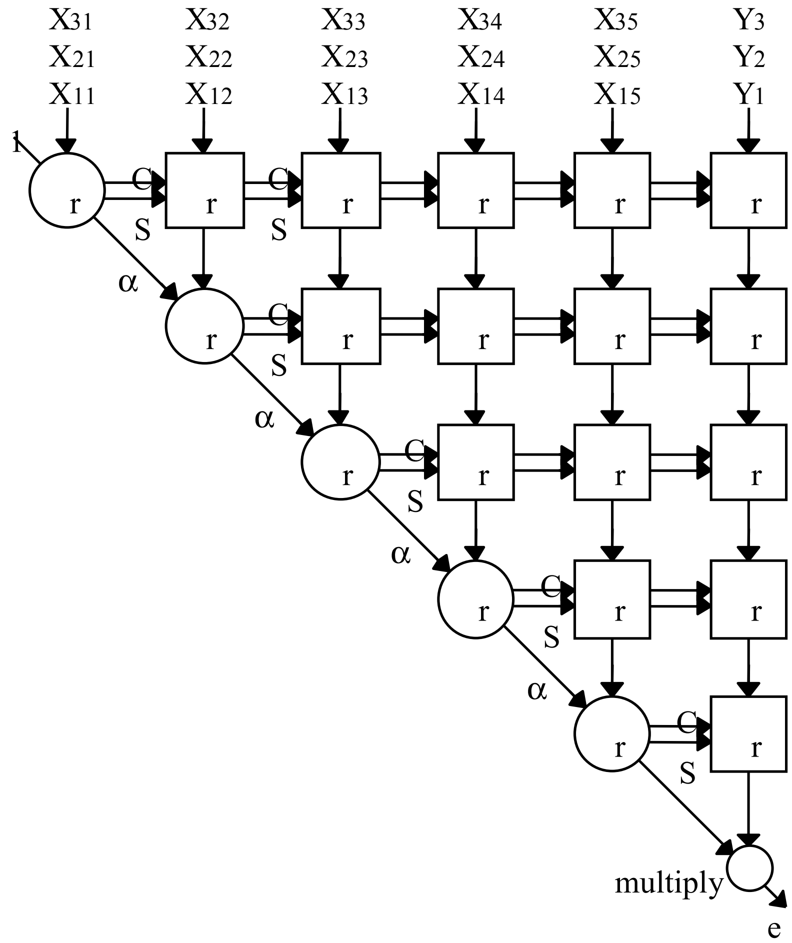
Norme Euclidienne en-ligne (3)



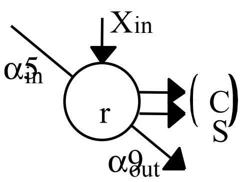
Exemple 2: filtre de Kalman

avec Jin Zian MOU   ENST-Paris

β est le carré du facteur d'oubli.
Le circuit ci-contre n'est pas systolique, il n'y a pas de retard horizontal. Il est "pipeliné" au niveau du chiffre. Toutes les entrées se font en série.

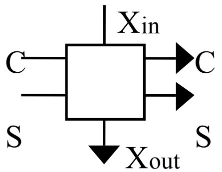


Description fonctionnelle de la cellule ronde



$$r' = \sqrt{r^2 + X_{in}^2}$$
$$C = r \div r'$$
$$S = X_{in} \div r'$$
$$\alpha_{out} = C \,.\, \alpha_{in}$$
$$r = \beta \,.\, r'$$

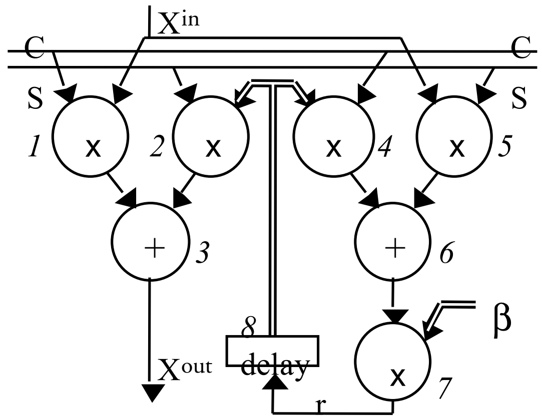
Description fonctionnelle de la cellule carrée



$$X_{out} = (C*X_{in} - S*r)$$
$$r = \beta*(S*X_{in} - C*r)$$

Filtre de Kalman (2)

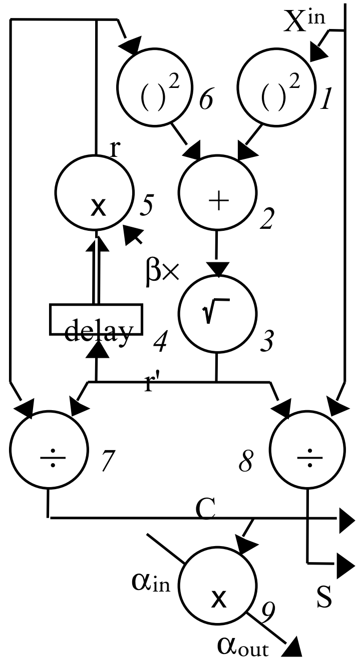
cellule carrée

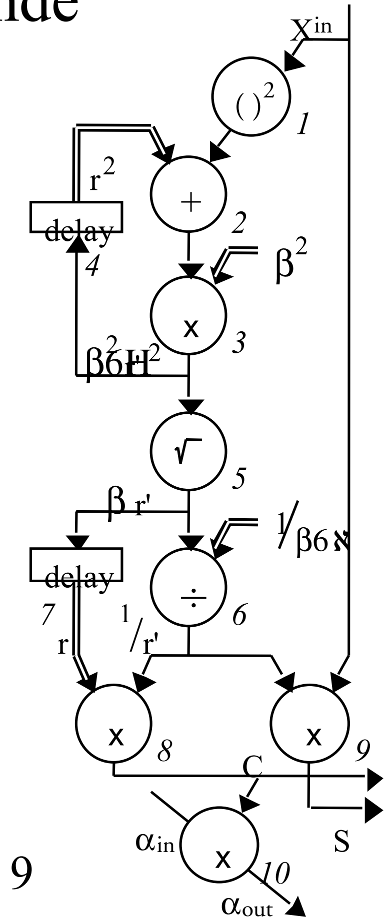


Règle 1 sur 1 & 4
règle 3 sur 1,2,3,4,5 &
6
gain de 19%

Règle 3 puis 4 sur 1,2,3 & 6
règle 2 sur 3,4,7 & 8
gain de 20%

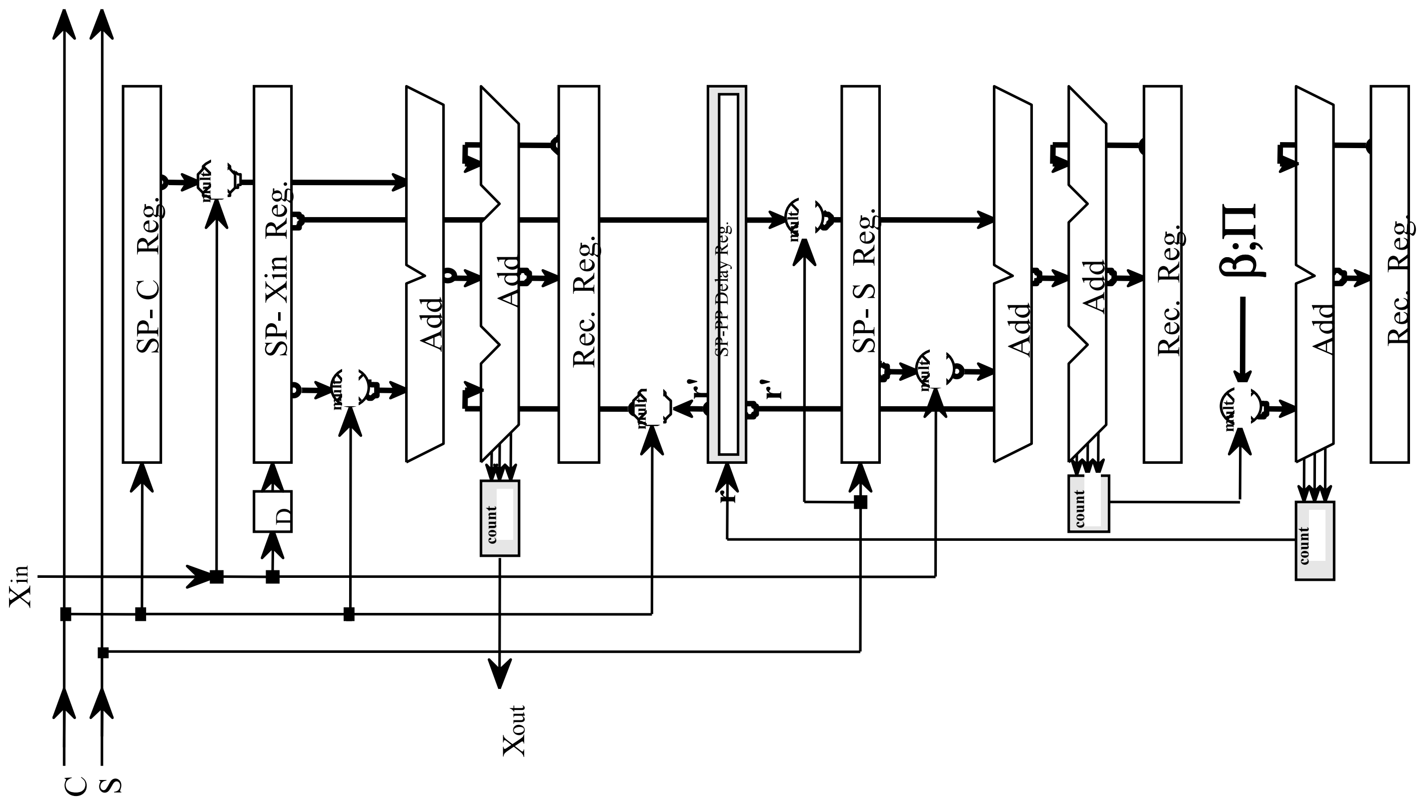
cellule ronde



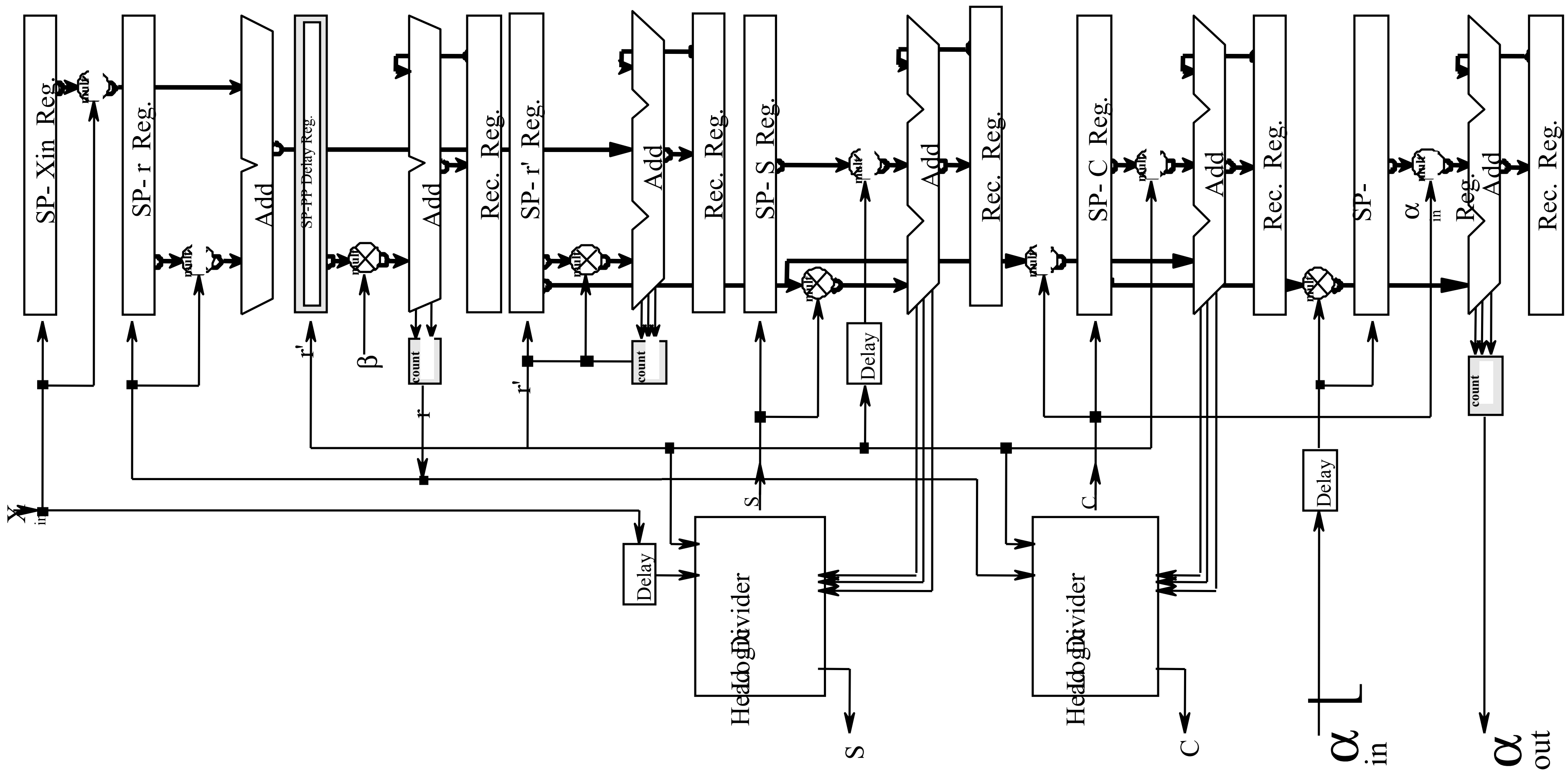


Règle 1 sur 1 & 2 puis 1 & 9
règle 2 sur 5,6 & 7 puis 6,8 & 9
gain de 21%

Cellule carrée



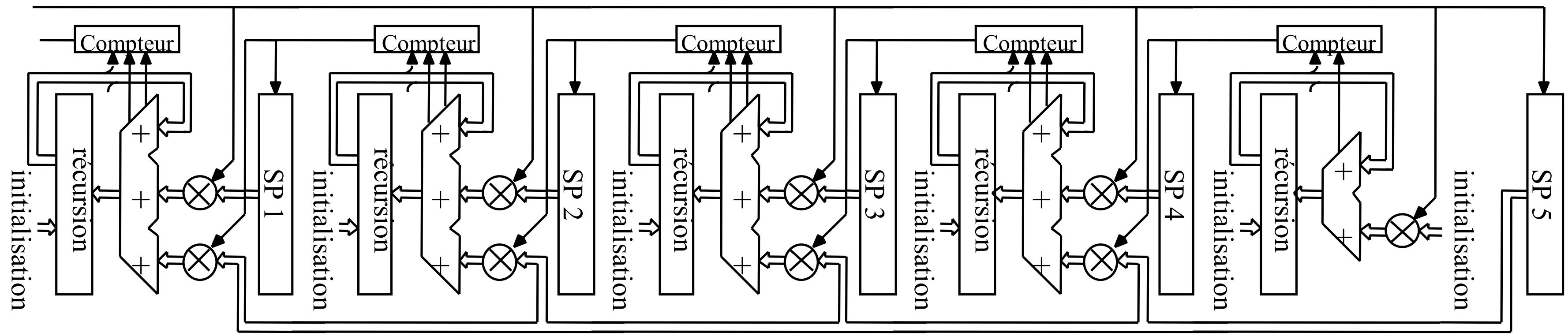
Cellule ronde



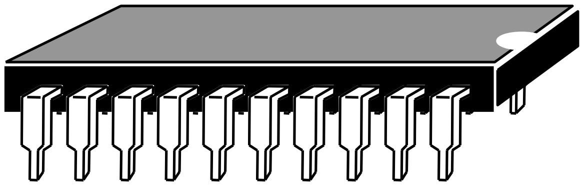
Polynômieur en ligne

$$e^{x} = \sum_{i=0}^{n} \frac{x^{i}}{i\,!} \quad \text{en-ligne}$$

application des règles: gain 19%



$$a_0 + x * (a_1 + x * (a_2 + x * (a_3 + x * (a_4 + x * a_5))))$$



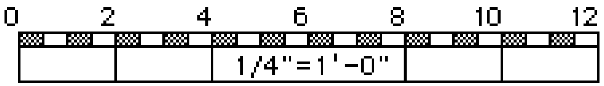
Polynômieur en ligne (2)

$$e^{x} = \sum_{i=0}^{n} \frac{x^{i}}{i\,!} \quad \text{en-ligne}$$

<u>Règle</u> : Si une constante est à ajouter à la sortie d'un opérateur de type 3 ou à l'entrée d'un opérateur de type 4 alors

son coût est nul (initialisation du registre de récursion)

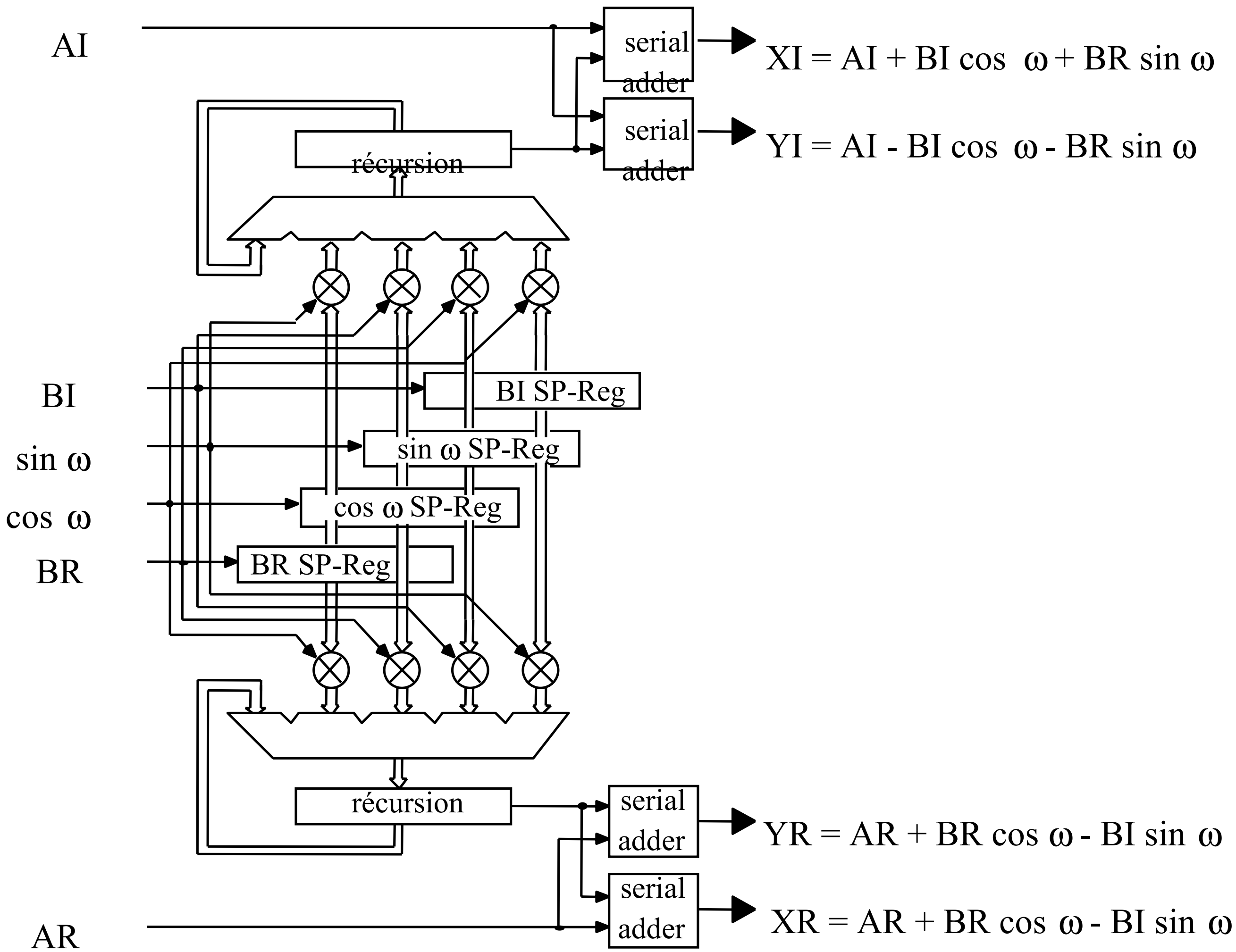
sa latence d'addition est nulle



$$a_0 \underbrace{+\, x \ast (a_1}_{\text{latence 3}} \underbrace{+\, x \ast (a_2}_{\text{latence 3}} \underbrace{+\, x \ast (a_3}_{\text{latence 3}} \underbrace{+\, x \ast (a_4}_{\text{latence 3}} \underbrace{+\, x \ast a_5}_{\text{latence 3}}))))$$

Latence indépendante du polynôme si $\exists$ k tq. $\forall\, n \geq k \quad \dfrac{a_{n+1}x}{a_n} \leq 2^{-3}$

Papillon FFT en ligne



Références

" *Design of an on-line Euclidean Processor* "
R. Bouraoui, A. Guyot and G. Walker
International Conference on Microelectronics 1992 (ICM'92), Monastir, Tunisia, Décembre 1992

" *On-line approximation of real functions using polynomials* "
A. Skaf, J.C. Bajard*, A. Guyot and J.M. Muller* (* LIP-ENS Lyon)
International Conference on Microelectronics 1992 (ICM'92), Monastir, Tunisia, Décembre 1992

" *Design of an on-line Euclidean processor* "
R. Bouraoui, A. Guyot and G. Walker
6 th International Conference on VLSI design, Bombay, India, Janvier 1993

" *On-line operator for Euclidean distance* "
R. Bouraoui, A. Guyot and G. Walker
EDAC-EUROASIC Conference, Paris, France, 22-25 Février 1993

" *A VLSI Circuit for on-line polynomial computing: application to exponential, trigonometric and hyperbolic functions* "
A. Skaf, J.C. Bajard*, A. Guyot and J.M. Muller* (* LIP-ENS Lyon)
VLSI 93, Grenoble, France, Septembre 1993

" *VLSI design of on-line add/multiply algorithms* "
A. Skaf and A. Guyot
ICCD 93, Cambridge, Massasusett, USA, Octobre 1993

" *A VLSI implementation of Parallel Fast Fourier Transform* "
A. Vacher, M. Benkhebbab, A. Guyot, T. Rousseau and A. Skaf
EDAC (European Design Automation Conference) , Paris, Février-Mars 1994

" *Error-Speed trade-off for FFT VLSI* "
A. Vacher and A. Guyot
26th IEEE Southeastern Symposium on System Theory, Athens, Ohio, Mars 1994

" *A VLSI implementation of Fast Fourier Transform for Large Sample Number* "
A. Vacher and A. Guyot
International Symposium on Signal Processing and Neural Network, Lille, Avril 1994

" *Design for testability of an on-line multiplier* "
H. Bederr*, M. Nicolaïdis* and A. Guyot (* Reliable Integrated System)
VLSI Test Symposium, Cherry Hill, New Jersey, Avril 1994

" *SAGA: The first general purpose on-line arithmetic co-processor* "
A. Skaf and A. Guyot
8 th International Conference on VLSI design (VLSI Design 95) New Delhi, India, Janvier 1995

" *On-line Hardware Implementation for Complex Exponential and Logarithm* "
A. Skaf, J.M. Muller* and A. Guyot (* LIP/ENSL Lyon)
20th European Solid-State CIRcuits Conference (ESSCIRC), Ulm, Germany, Septembre 1994