Master STIC P – Année 2008/2009

Traitement du Signal et des Images Temps Réel à base de FPGA et DSP

Michel PAINDAVOINE
Université de Bourgogne
Laboratoire LE2I – UMRCNRS 5158
Aile des Sciences de l 'Ingenieur
BP 400 - 21011 DIJON Cedex
email:paindav@u-bourgogne.fr

SOMMAIRE

I-Objectifs du Traitement du Signal et des Images Temps Réel

II- Présentation de la méthodologie « Adéquation Algorithme Architecture (AAA) »

III- Problématique d'implantation de filtres numériques

IV- AAA avec des approches SIMD

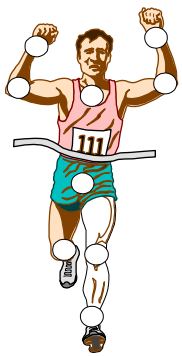
V- AAA avec une approche MISD

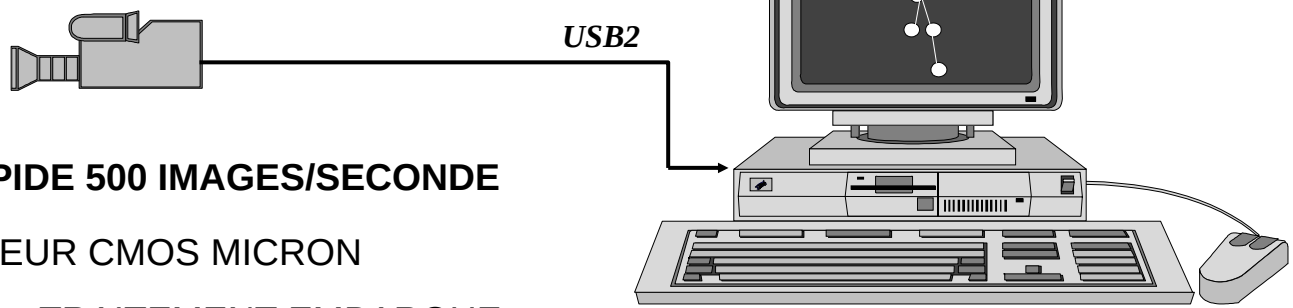
VI- AAA par optimisation des opérateurs arithmétiques

VII- AAA sous contraintes de dimensionnement

VIII- Exemples d'Application

Exemple 1: Analyse du Mouvement Humain en Temps Réel





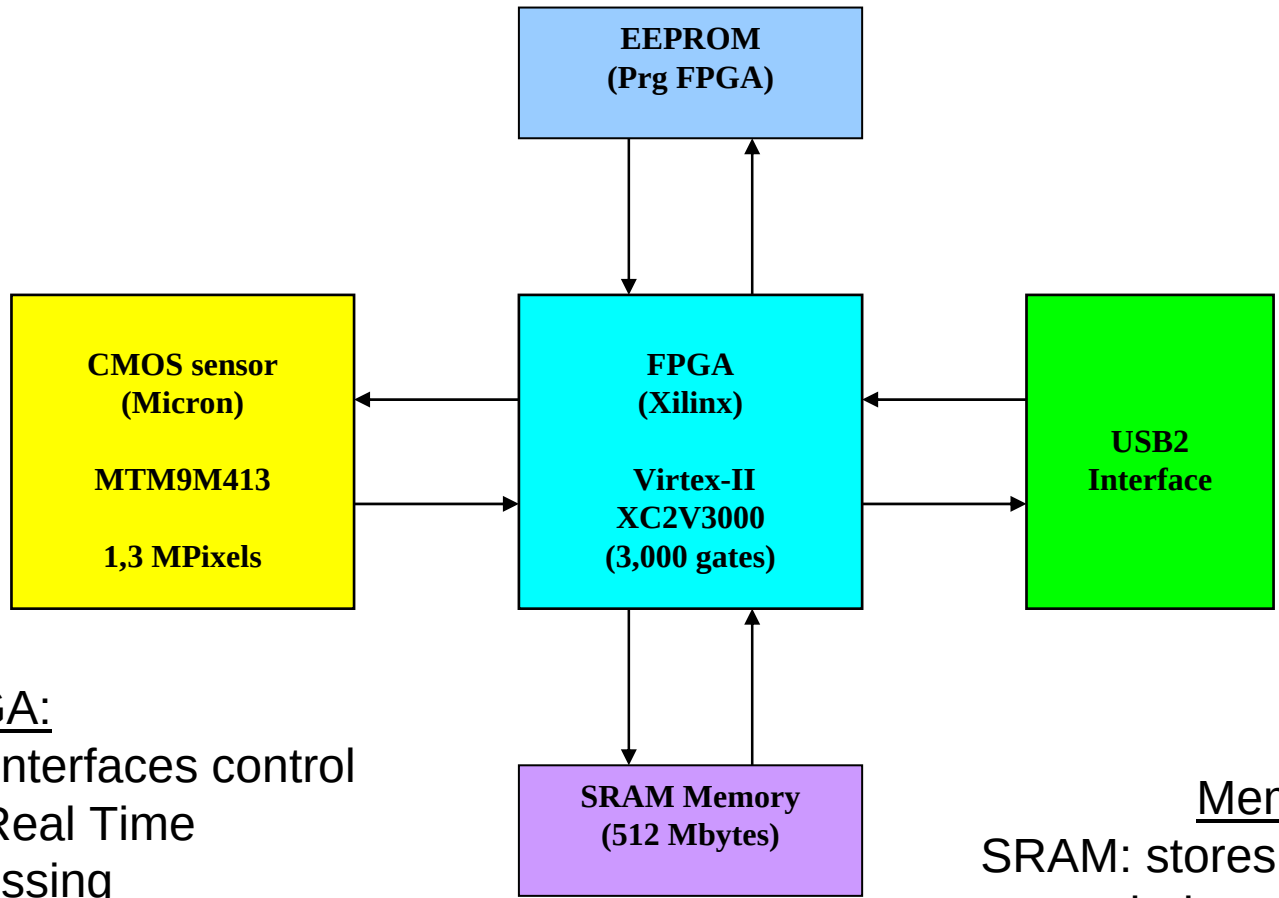
CAMERA RAPIDE 500 IMAGES/SECONDE

CAPTEUR CMOS MICRON

COMMANDE et TRAITEMENT EMBARQUE
PAR FPGA XILINX Virtex II

FORMAT IMAGE 1280x1024

Description de la caméra rapide « intelligente »



FPGA:

-Sensor and interfaces control

-Embedded Real Time Image Processing

Memory:

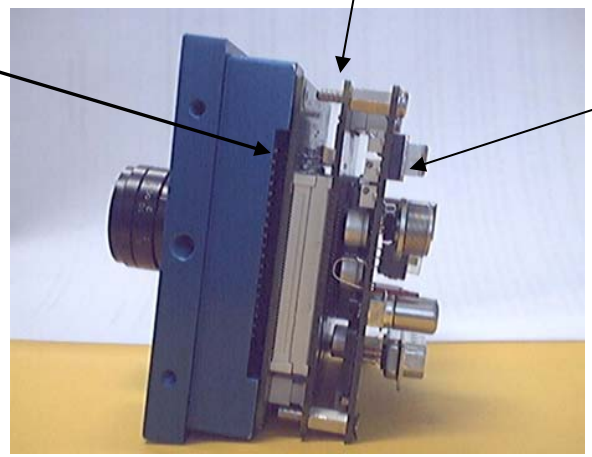
SRAM: stores temporary data during processing

Internal Camera view:

CMOS sensor board

FPGA board

Interface board



Camera back-face:

Connectors:

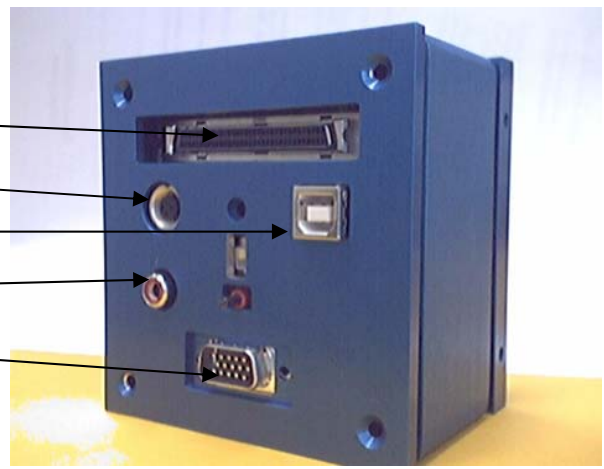
SCSI

Power supply

USB2

Video

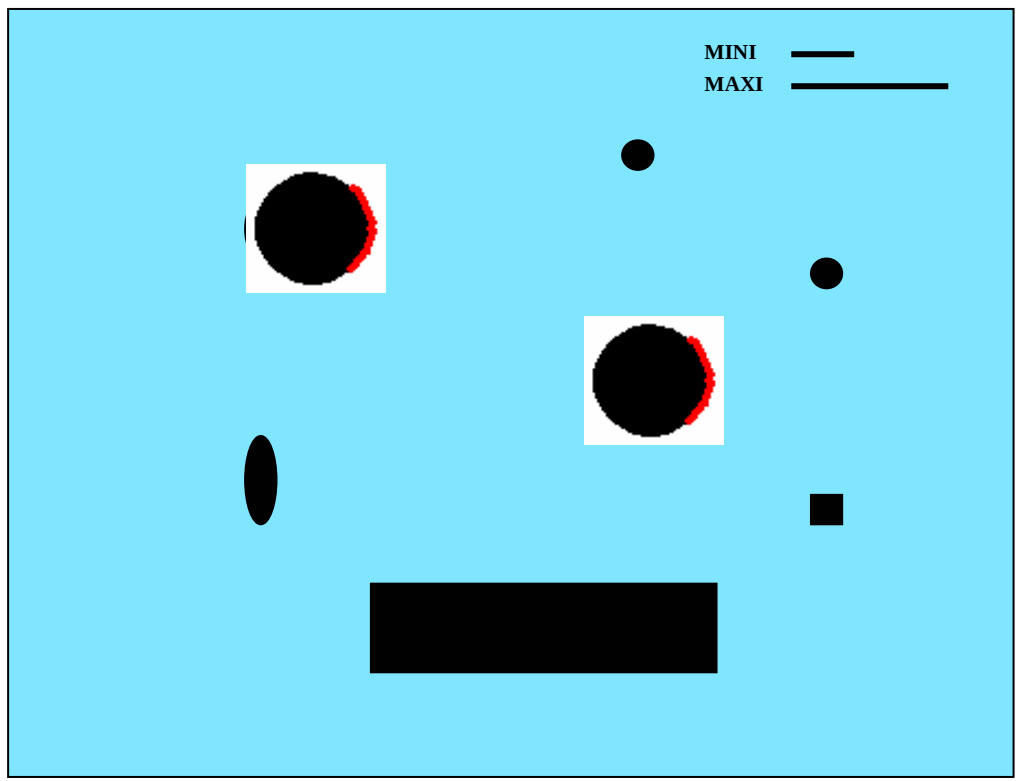
JTAG and Camera control



Détections des Marqueurs

PARAMETRES DE DETECTION:

- SEUIL NIVEAU GRIS
- TAILLE MINIMUM MARQUEUR
- TAILLE MAXIMUM MARQUEUR



AFFICHAGE RESULTATS DETECTION

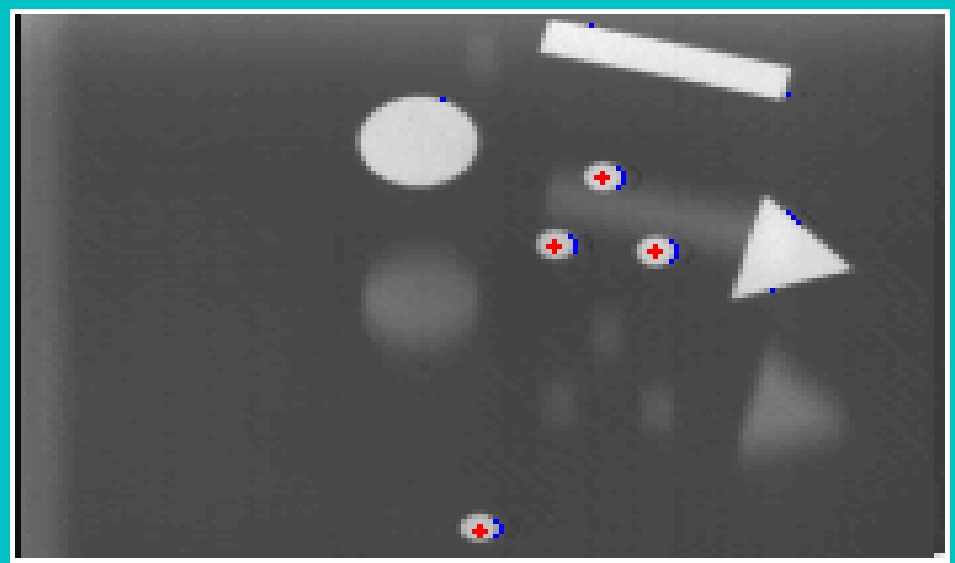
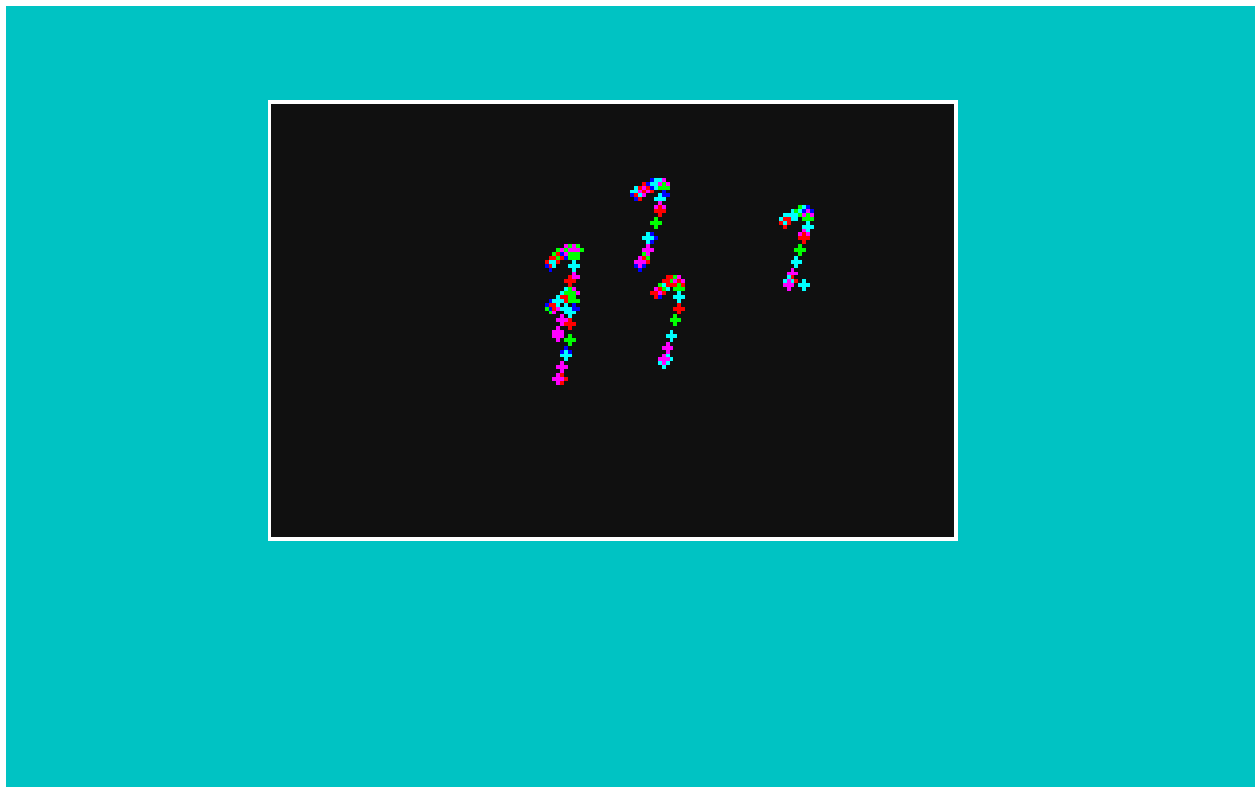


IMAGE N° 2452

ANALYSE DU MOUVEMENT HUMAIN EN TEMPS REEL

Résultat: **Suivi Marqueurs en Temps Réel (500 Images/sec)**

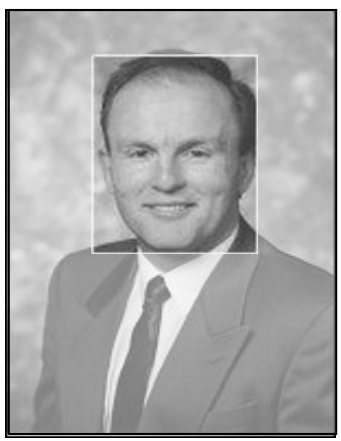


Exemple 2: Détection et Identification de visages par réseaux de neurones

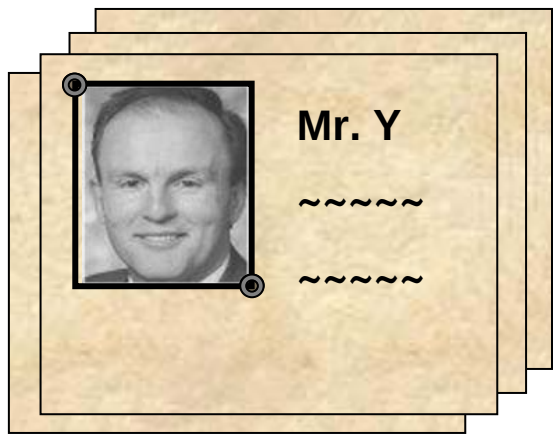
Détection des visages (ou localisation) :

Où ?

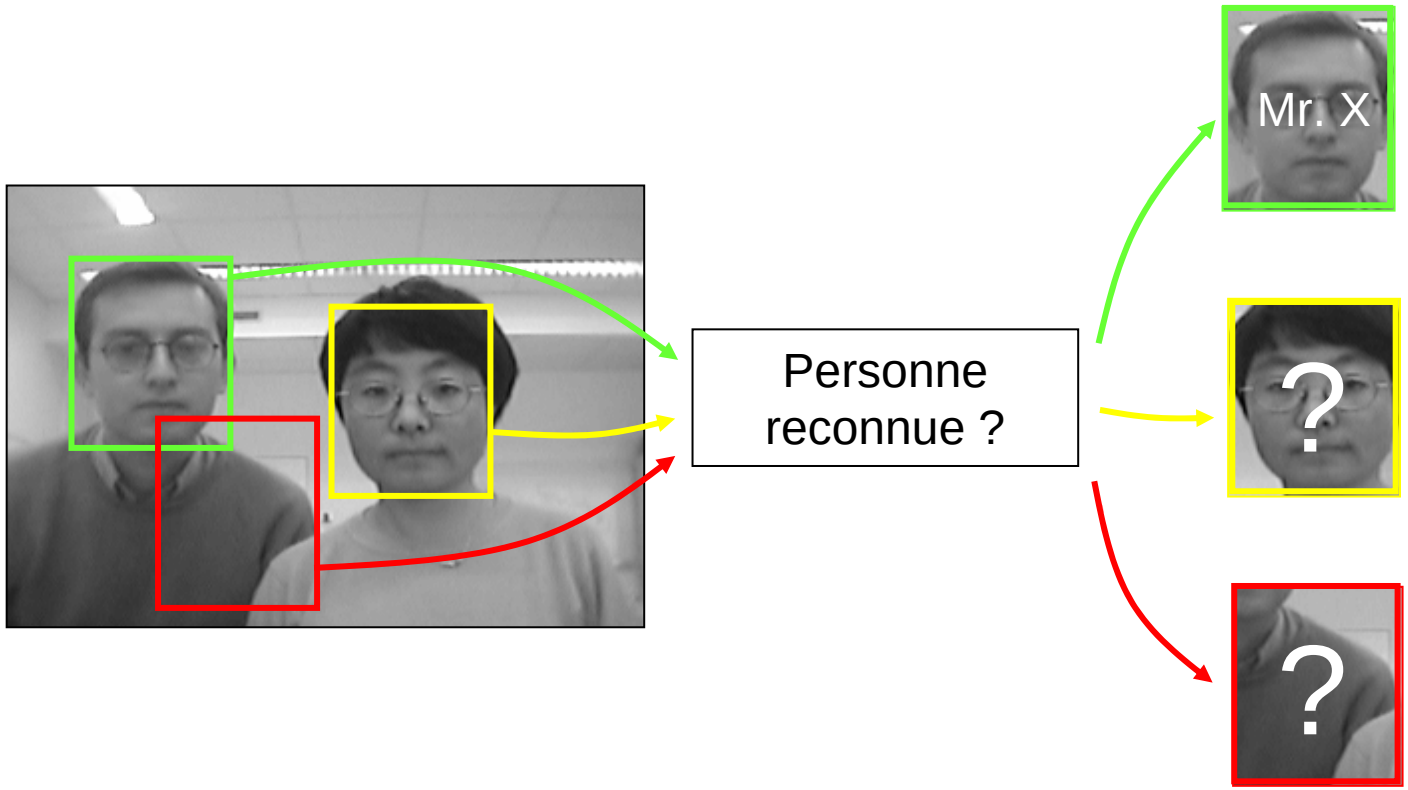
Identification des visages



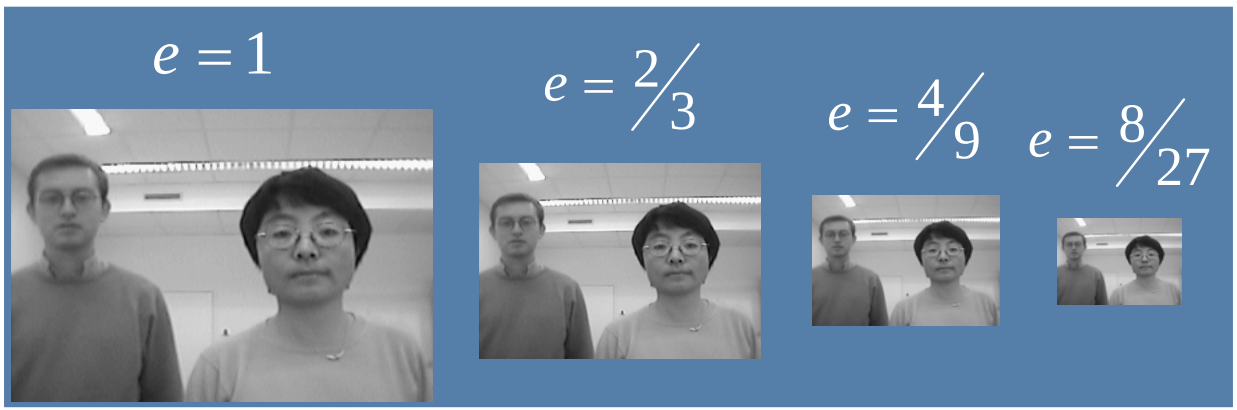
Qui ?!

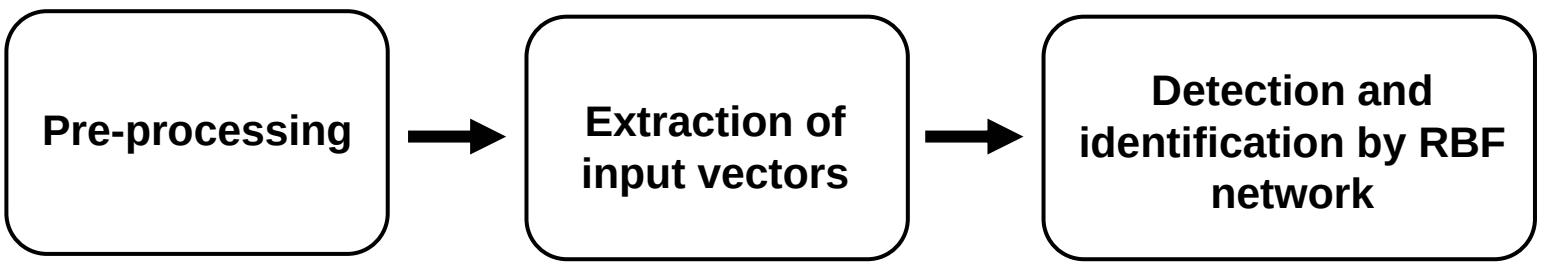


Approche conjointe de détection et d'identification de visages :

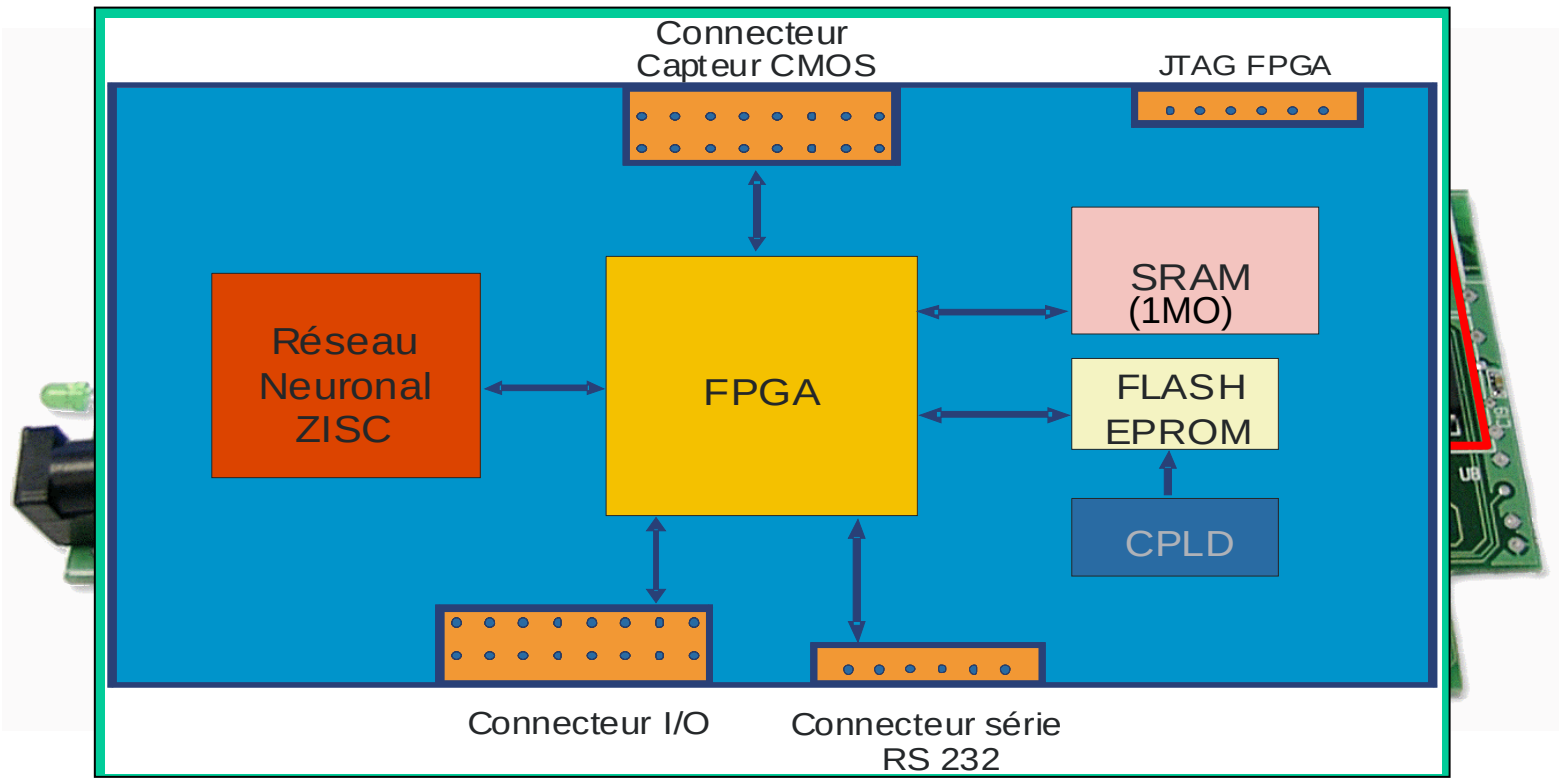


Détection et Identification de visages à plusieurs échelles:





Hardware resources : *Neurosight* board

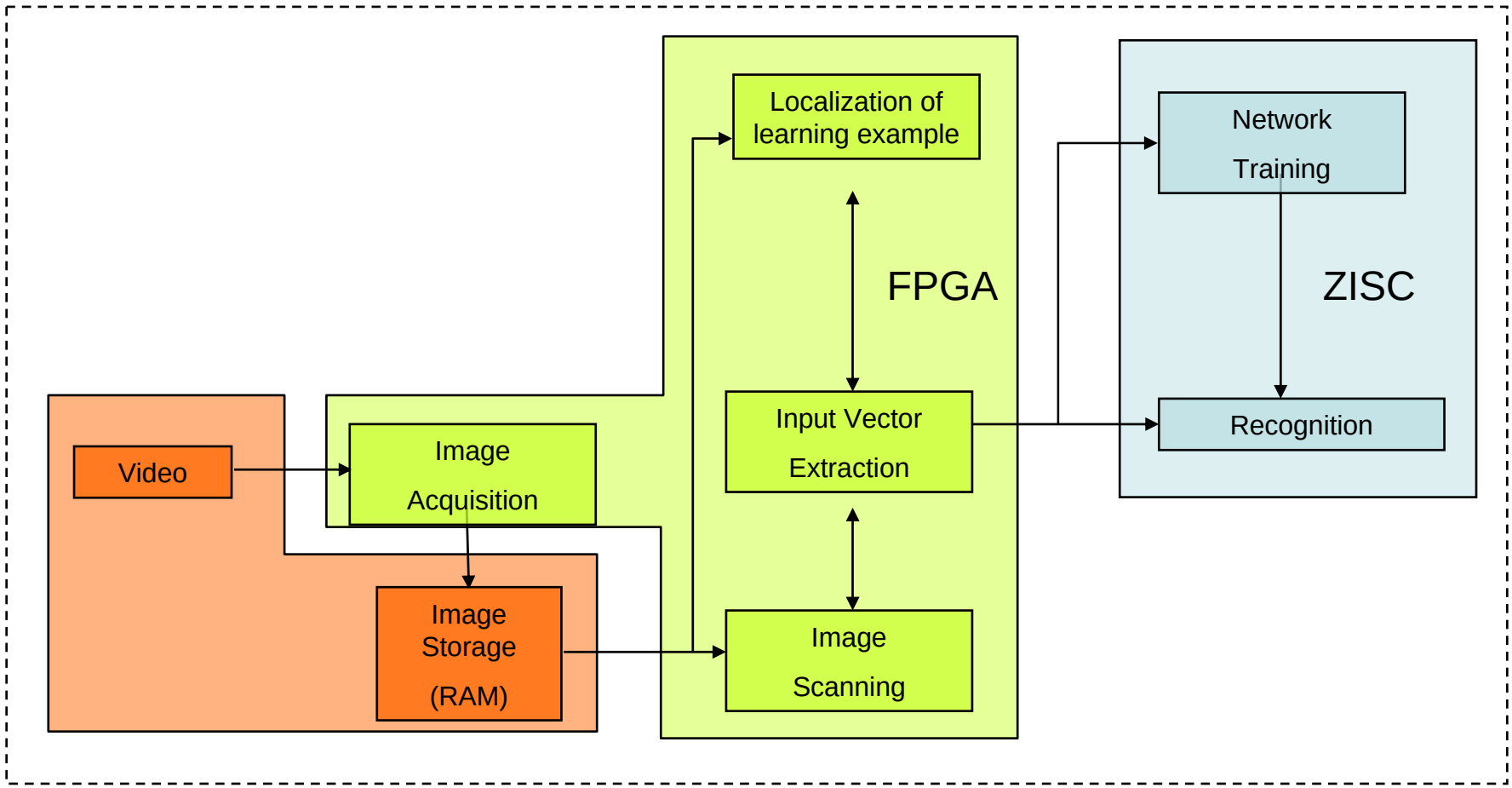


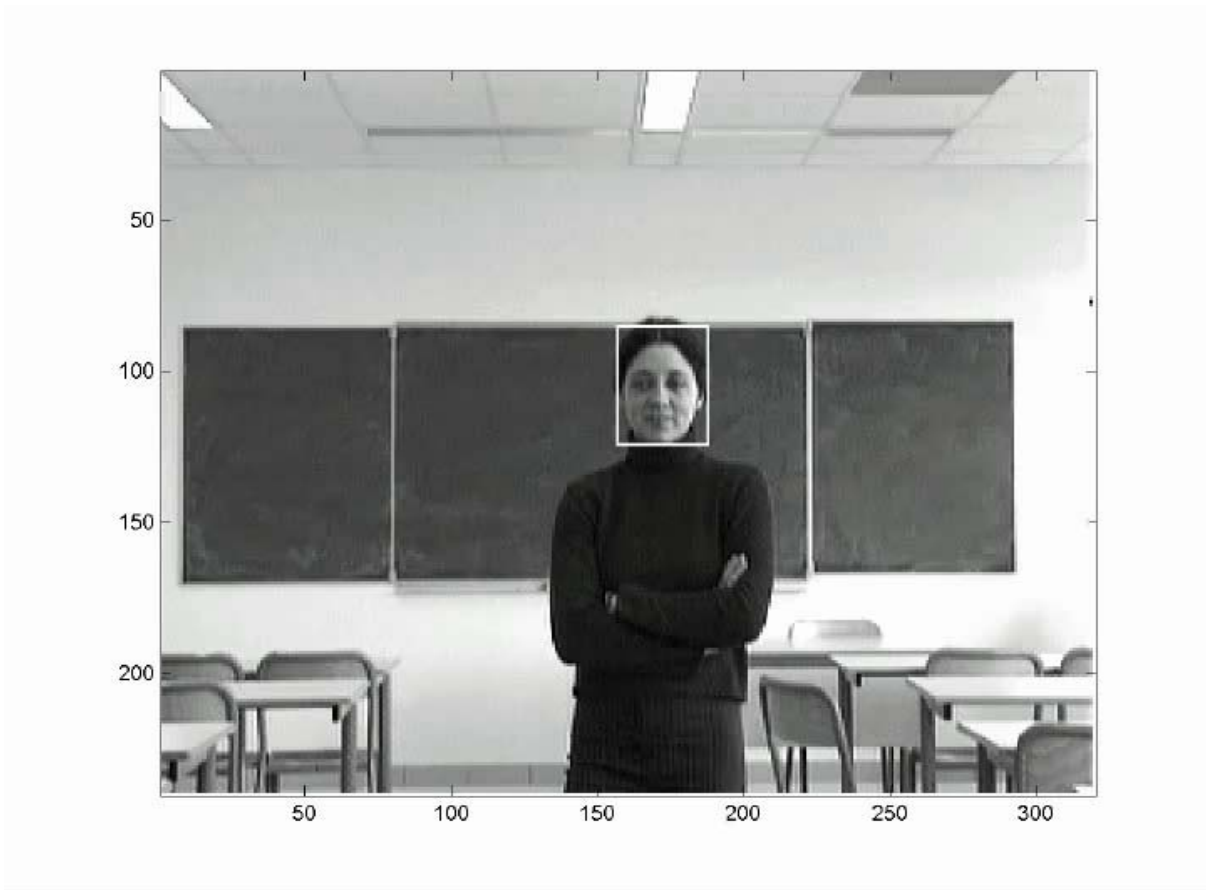
Zero Instruction set to compute (ZISC) chip

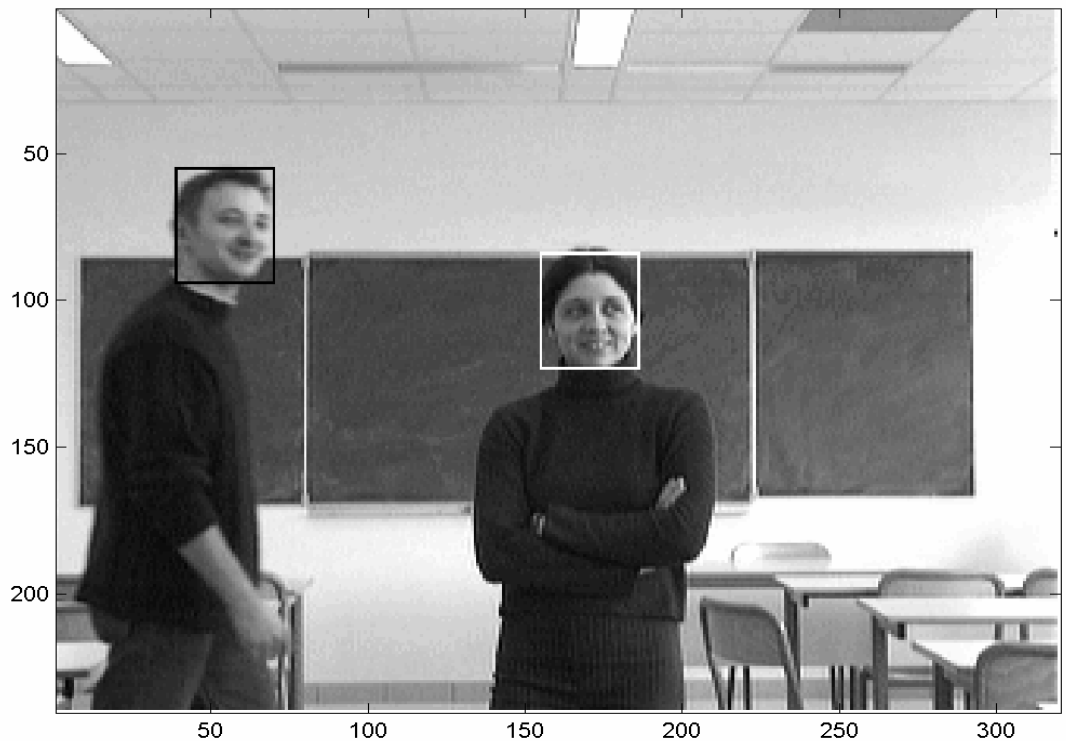
ZISC chip :

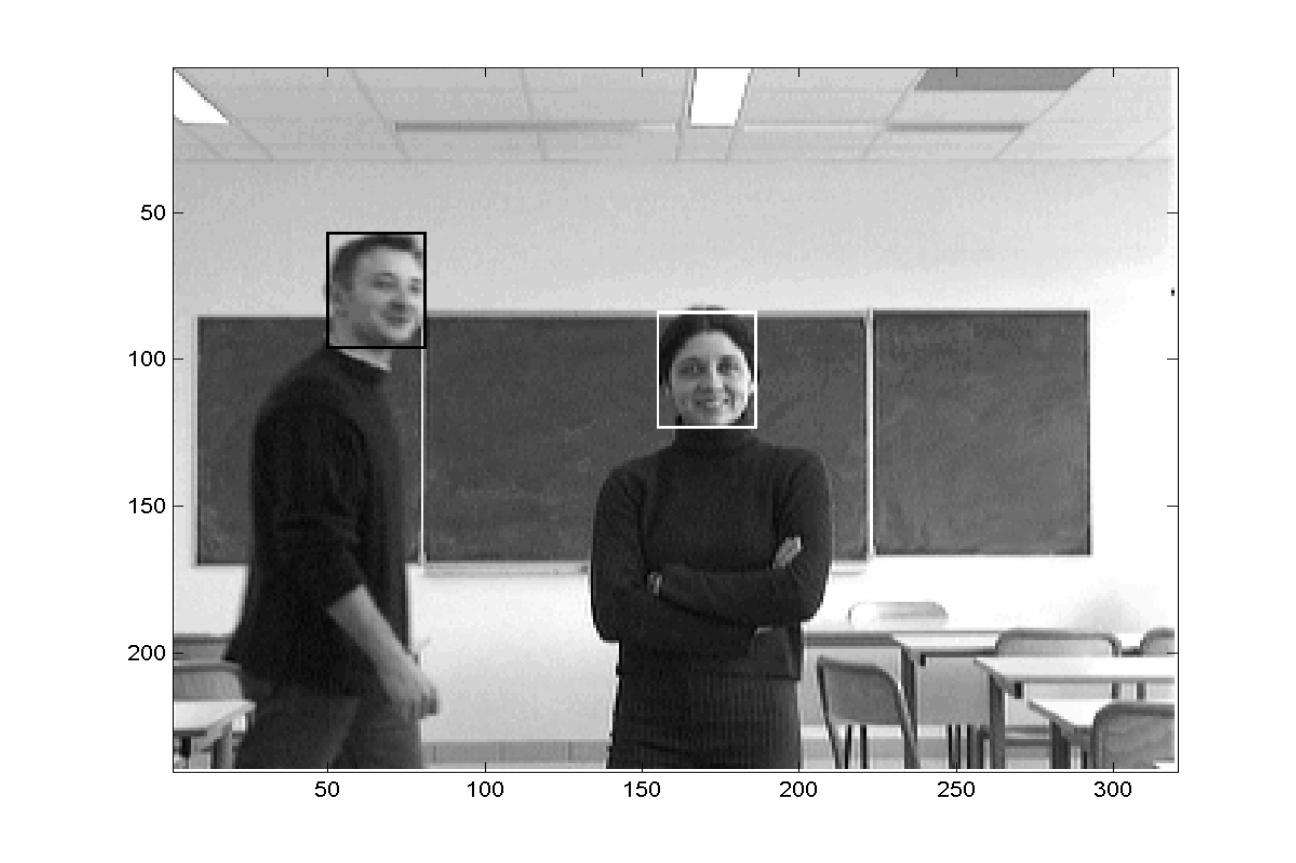
- maximal length of input vectors = 64 components,
- mesured distance = d1(x),
- kernel fonction = heaviside function.

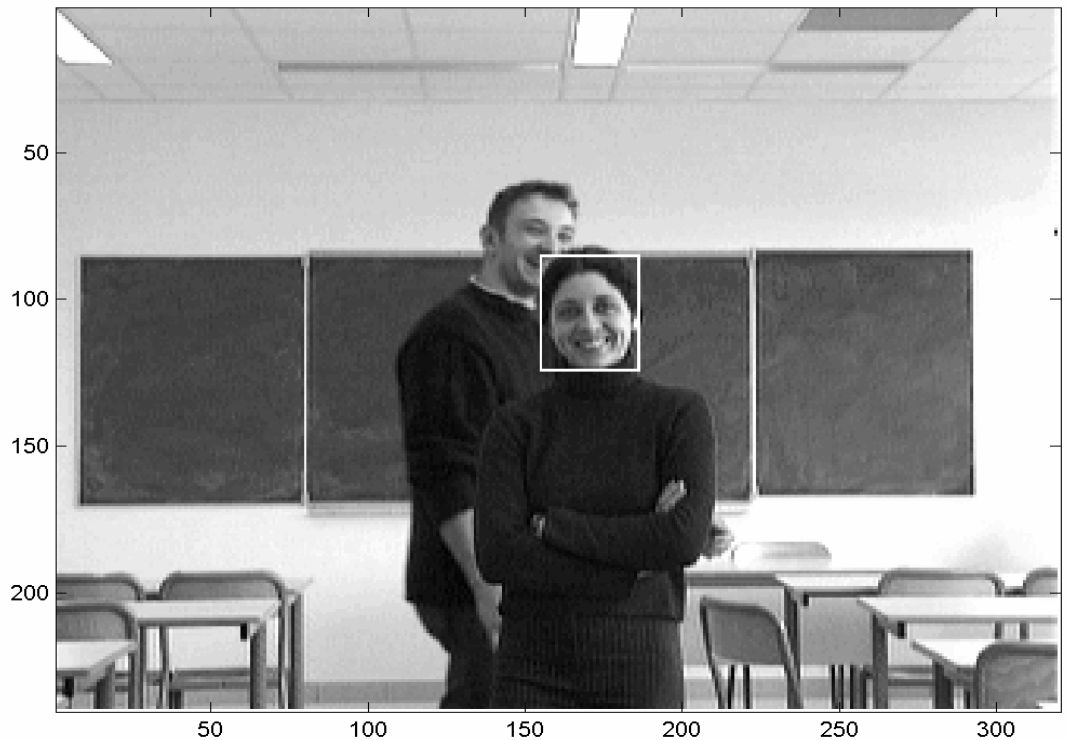
Distribution des calculs sur la carte Neurosight

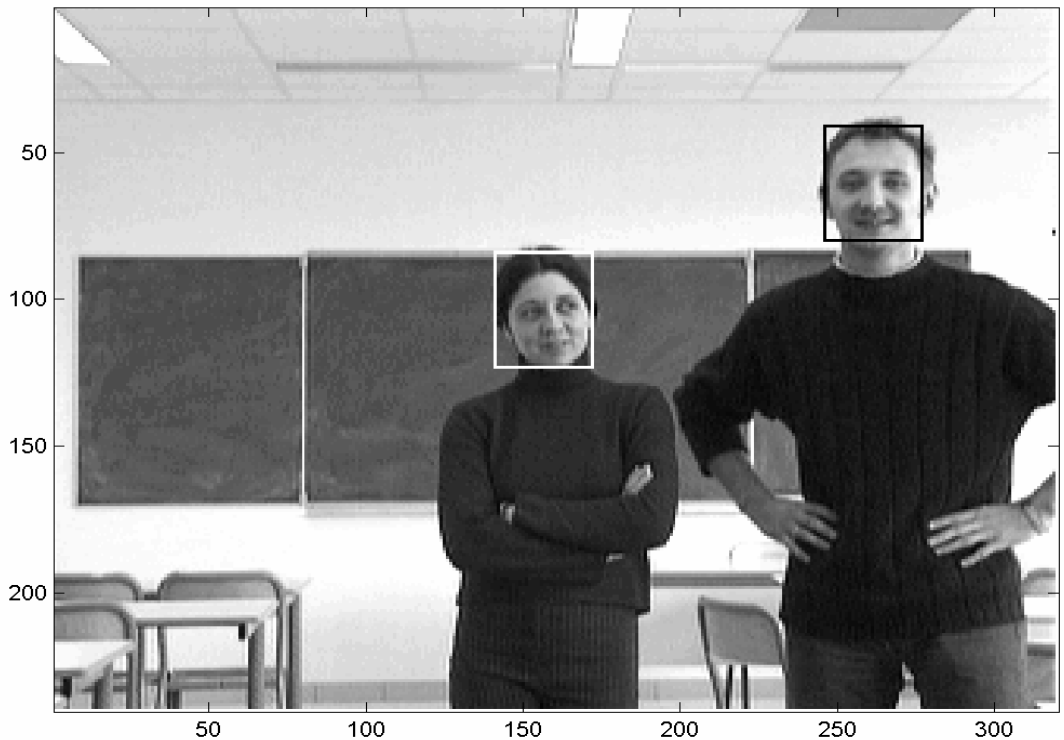


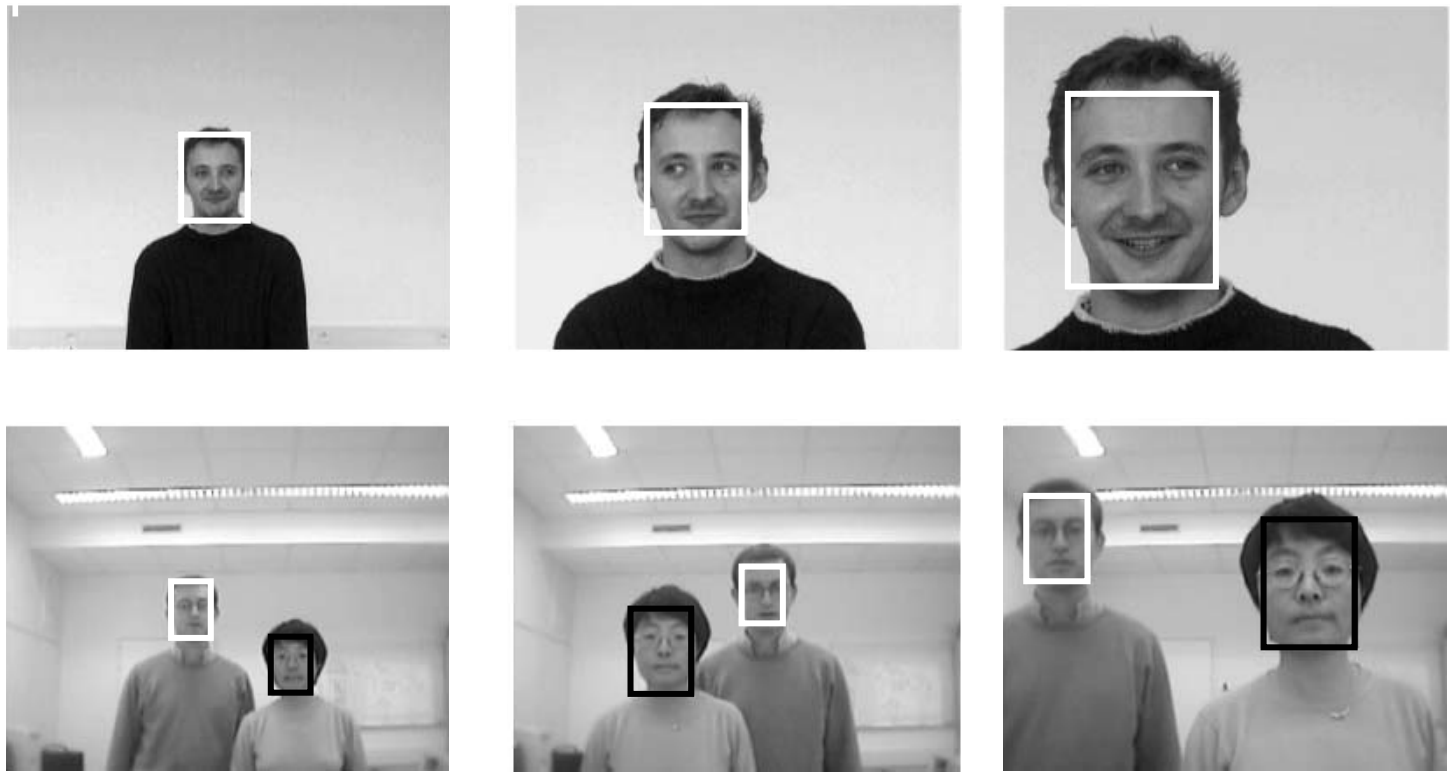

50
100
150
200
50
100
150
200
250
300


50
100
150
200
50
100
150
200
250
300


50
100
150
200
50
100
150
200
250
300


50
100
150
200
50
100
150
200
250
300


50
100
150
200
50
100
150
200
250
300



OBJECTIFS du TRAITEMENT d ’IMAGES TEMPS REEL

* Limitation du temps de calcul :

Exemples:

- Détection de Contours avec l ’algorithme de Canny-Deriche: correspond à 12 additions et 12 multiplications par pixel en moins de 100ns (fréquence Pixel=10MHz)

- Reconnaissance de visages par réseaux de neurones :

Nb Total opérations	Additions	Soustractions	Divisions	Comparaisons
$20{,}34.\ 10^6$	$10{,}16.10^6$	$10.05.10^6$	92736	146510

* Limitation Taille Calculateurs :

Exemple:

- Images Rapides:

1000 images par seconde, chaque image=512x512 pixels

correspond à 250 Mpixels par seconde et nécessite espaces mémoires importants.

→ **Traitement d ’Images Temps Réel nécessite Méthodologie Adéquation Algorithme Architecture (AAA)**

OBJECTIFS du TRAITEMENT d ’IMAGES TEMPS REEL

*** Limitation du temps de calcul :**

Exemples:

- Détection de Contours avec l ’algorithme de Canny-Deriche: correspond à 12 additions et 12 multiplications par pixel en moins de 100ns (fréquence Pixel=10MHz)

- Reconnaissance de visages par réseaux de neurones :
Image de 200x200 pixels correspond à 40K entrées pour le réseau de neurones et nécessite de traiter des matrices de 40Kx40K en moins de 40ms (25 images par seconde)

II- Description Méthodologie Adéquation Algorithme Architecture (AAA)

(Résultats Travaux Recherche GDR-CNRS ISIS) [http://gdr-isis.org]

◊ Choix et Test de l ’algorithme en utilisant un calculateur standard

◊ Comparaison de différentes architectures spécifiques (approche par simulation à l ’aide outils de CAO)

◊ Modification Algorithme

◊ Validation Architecture

◊ Implantation réelle de l ’algorithme sur l ’architecture

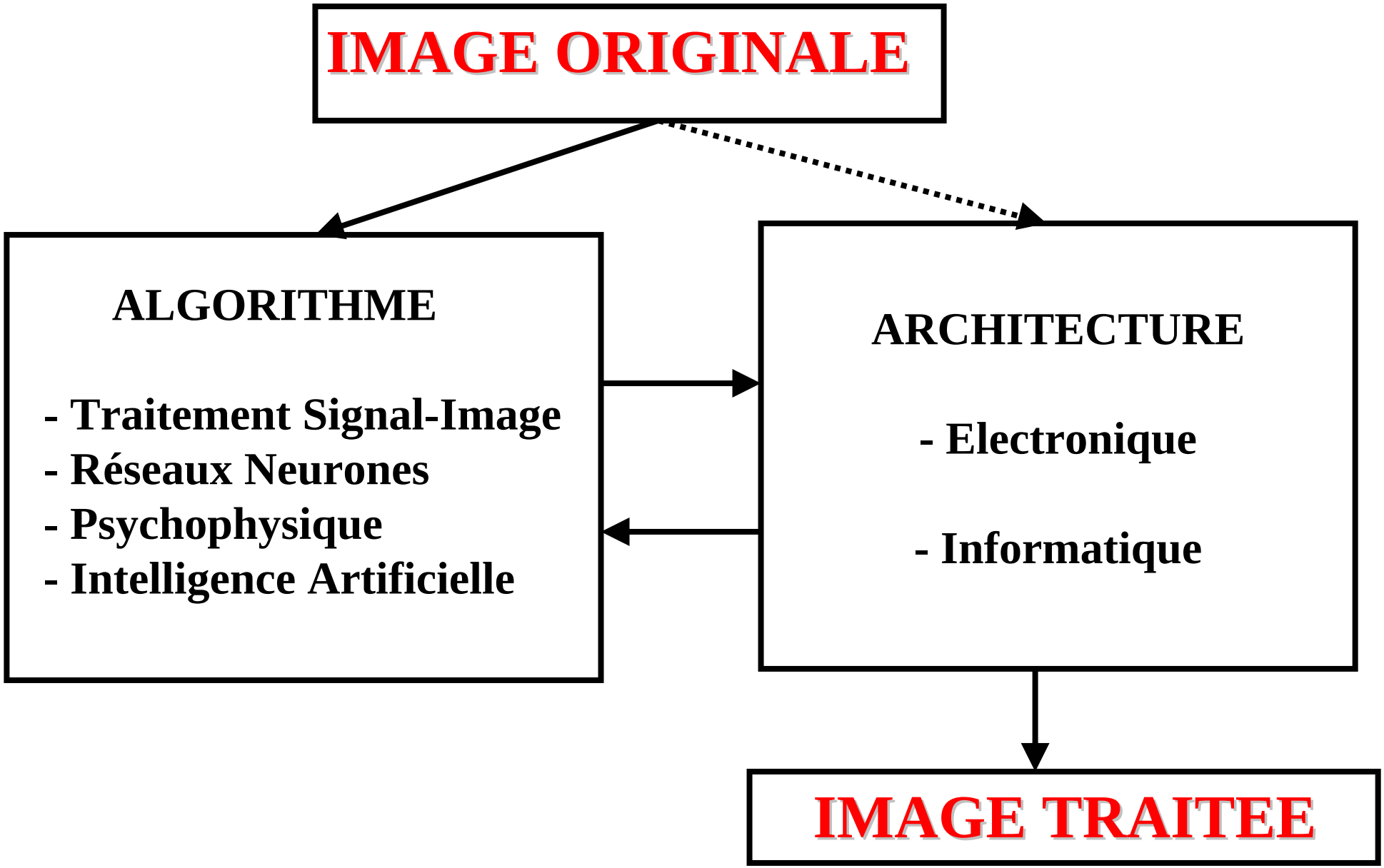
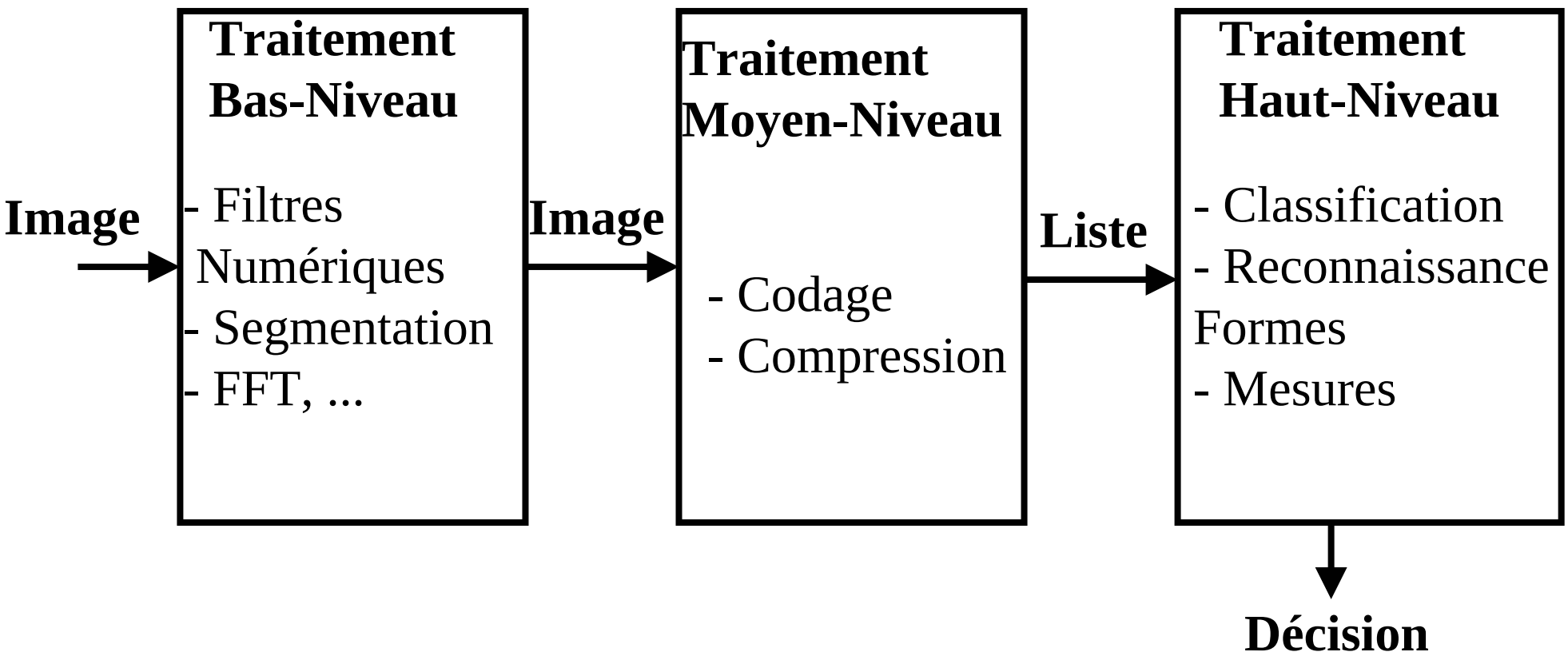


IMAGE ORIGINALE
ALGORITHME
- Traitement Signal-Image
- Réseaux Neurones
- Psychophysique
- Intelligence Artificielle
ARCHITECTURE
- Electronique
- Informatique
IMAGE TRAITEE

Classification des Algorithmes de Traitement d ’Images



Classification des Architectures

LOGICIELS

- MATLAB
- C, C++
- Langages Synchrones (SynDEx)
-VHDL, CADENCE, MENTOR GRAPHICS

MATERIELS

- **SISD** (ASICs, FPGAs,DSPs)
- **SIMD**: Systolic (ASIC)
- **MISD**: Pipeline (ASICs, FPGAs, DSPs)
- **MIMD** (Multi DSPs)

Exemple FPGA: Xilinx Spartan, Virtex
Exemple DSP: Texas TMS 320C60

SOMMAIRE

I-Objectifs du Traitement du Signal et des Images Temps Réel

II- Présentation de la méthodologie « Adéquation Algorithme Architecture (AAA) »

III- Problématique d'implantation de filtres numériques

IV- AAA avec des approches SIMD

V- AAA avec une approche MISD

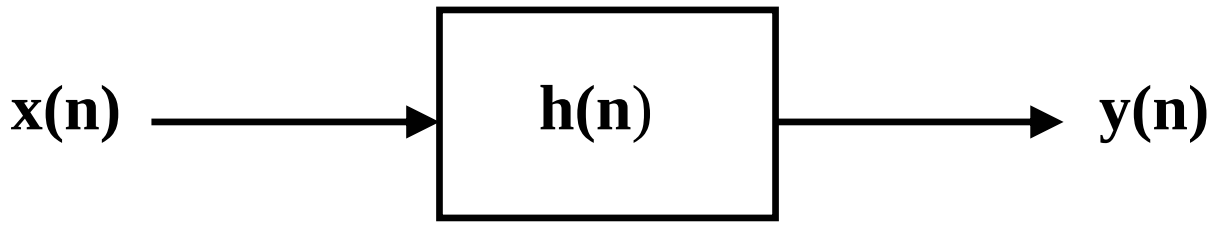
VI- AAA par optimisation des opérateurs arithmétiques

VII- AAA sous contraintes de dimensionnement

VIII- Exemples d'Application

III- Problématique d'implantation de filtres numériques

III-1 Rappels sur les filtres numériques



$$y(n) = x(n) * h(n) \xrightarrow{\ Z\ } Y(z) = X(z).H(z)$$

avec $$H(z) = \frac{a_0 + a_1 z^{-1} + \ldots + a_I z^{-I}}{b_0 + b_1 z^{-1} + \ldots + b_J z^{-J}}$$

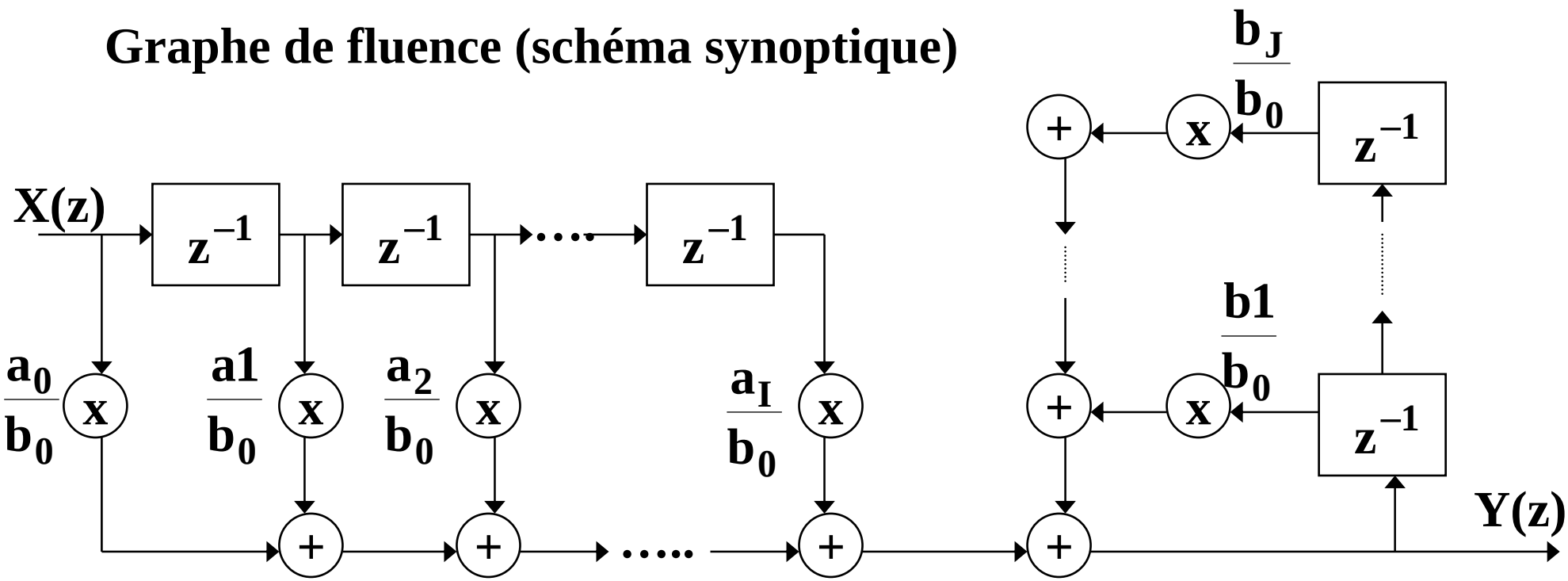
$$Y(z) = \frac{a_0}{b_0} X(z) + \frac{a_1}{b_0} z^{-1} X(z) + \ldots + \frac{a_I}{b_0} z^{-I} X(z) - \frac{b_1}{b_0} z^{-1} Y(z) - \ldots - \frac{b_J}{b_0} z^{-J} Y(z)$$

$$y(n) = \frac{a_0}{b_0} x(n) + \frac{a_1}{b_0} x(n-1) + \ldots + \frac{a_I}{b_0} x(n-I) - \frac{b_1}{b_0} y(n-1) - \ldots - \frac{b_J}{b_0} y(n-J)$$

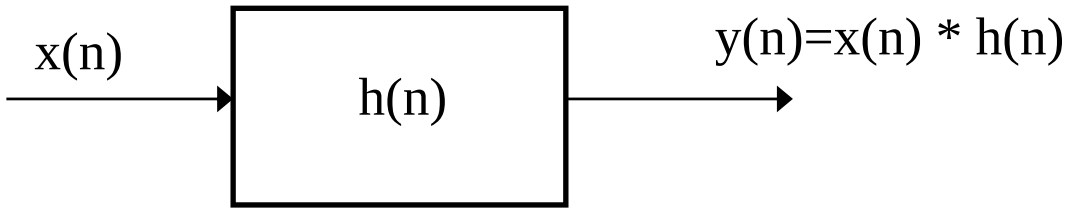
Implantation sous Matlab:

```
>> A = [a0 a1 .... aI];
>> B = [b0 b1 .... bJ];
>> y = filter(A,B,x);
```

Graphe de fluence (schéma synoptique)

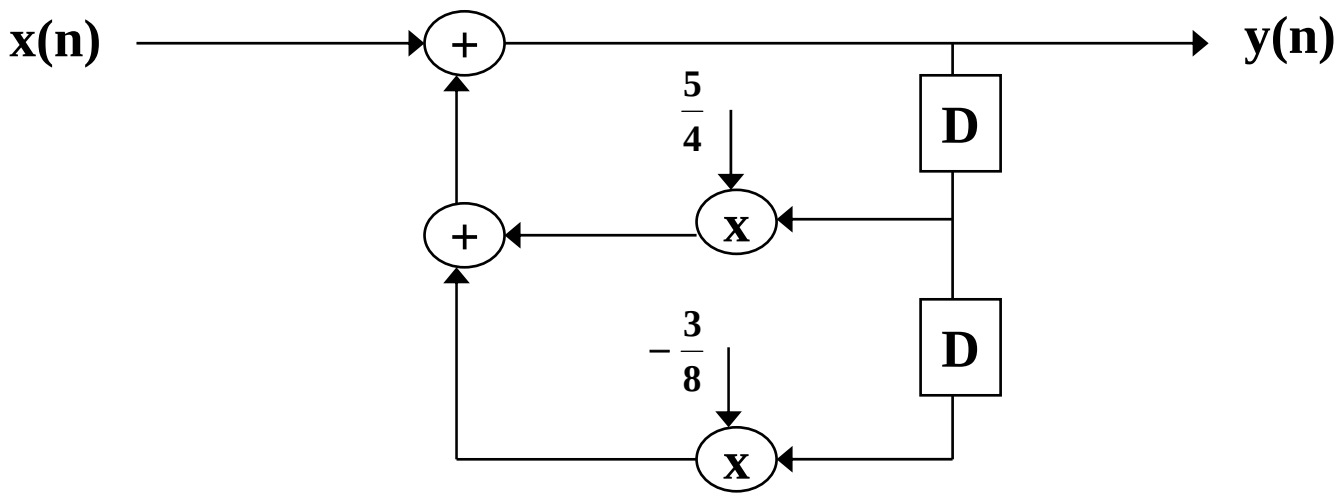


III-2 Implantation d'un filtre récursif d'ordre 2 en VHDL:



Exemple: **y(n)=x(n) +(5/4).y(n-1) – (3/8).y(n-2)** avec x(n) codé sur 8 bits
et y(n) codé sur 16 bits

a) Description RTL



b) Description VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity filter is
    Port ( clk : in std_logic;
           x: in unsigned(7 downto 0);
           y : out unsigned(15 downto 0));
end filter;
```

```
architecture filtre_arch of filtre is
signal y1, y2, ytemp: unsigned (y'range);
begin
process(clk)
 begin
  if rising_edge(clk) then
            ytemp <= x+ shr((y1+y1+y1+y1+y1),"100") - shr((y2+y2+y2),"1000");
            y2<=y1;
            y1<=ytemp;
 end if;
end process;
y <= ytemp;
end filtre_arch;
```

III-3 Déconvolution Image:

Observed Image*

Propagation

wavefront coding

Detected
blurred Image

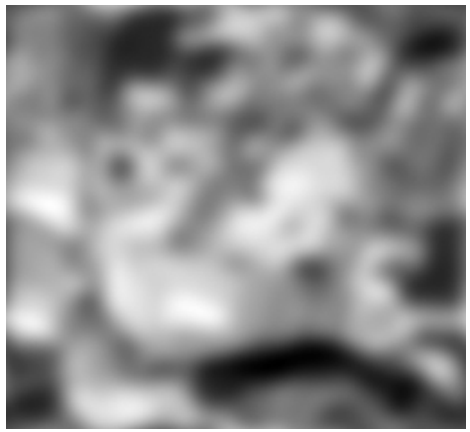


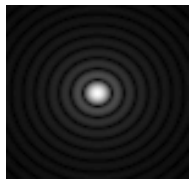
Image reconstruction = deconvolution

Reconstructed
Image

* : Image provided by the GDR ISIS

Image Deconvolution

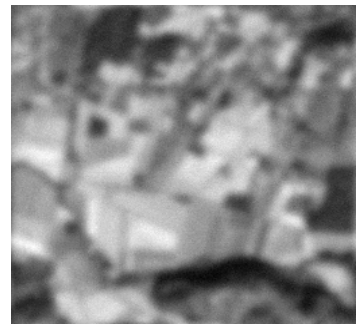
Unblurred Image



convolution

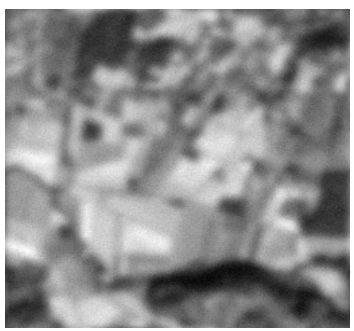
Additive noise

Blurred Image

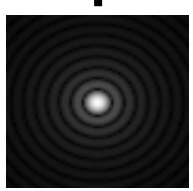


Goal : to recover the initial image

Blurred Image



Déconvolution



Restored Image

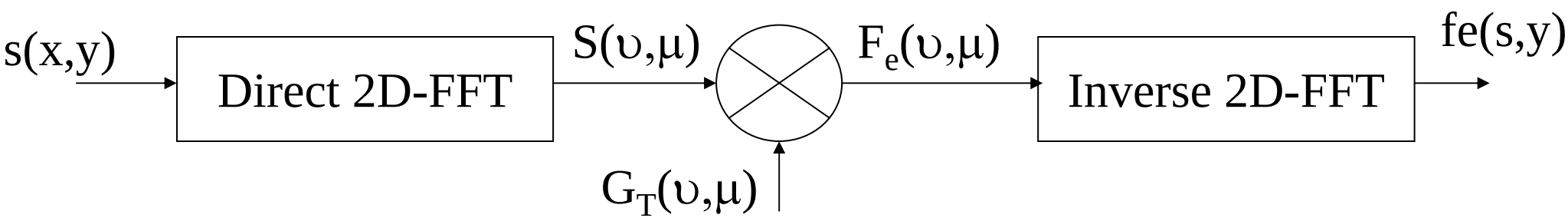
1- Calculations for Deconvolution

-> Noisy and Blurred Image: $s(x,y) = h(x,y) \otimes f(x,y) + b(x,y)$

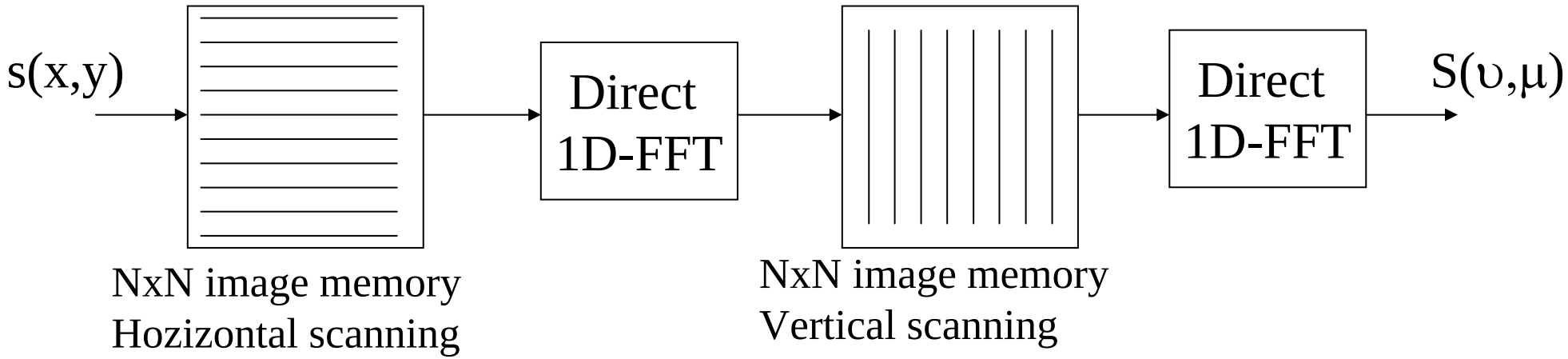
Thikonov deconvolution: $$G_T(\nu,\mu) = \frac{H^*(\nu,\mu)}{|H(\nu,\mu)|^2 + K(\nu,\mu)}$$

> Deconvoluted Image: $f_e(x,y) = g_T(x,y) \otimes s(x,y)$

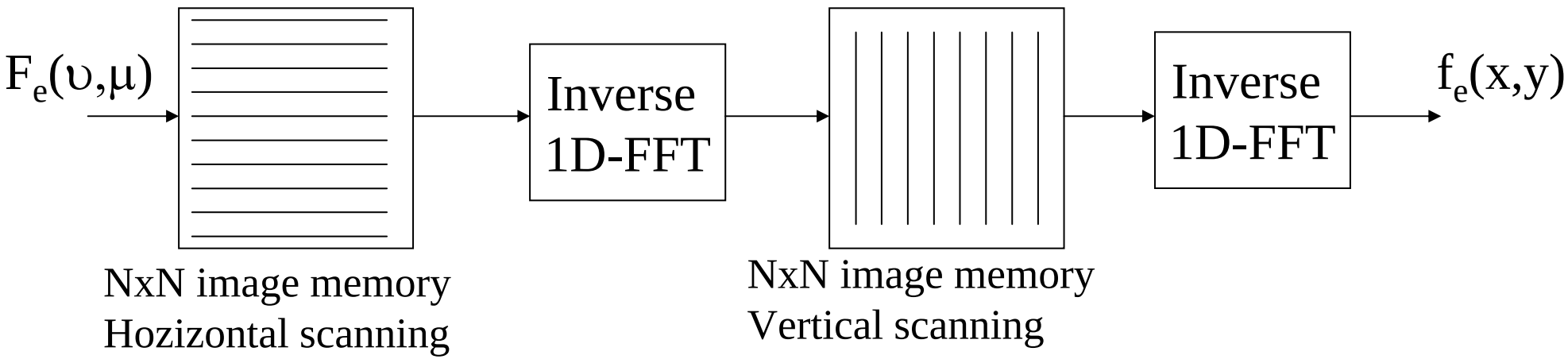
In Fourier domain: $F_e(\upsilon,\mu)=G_T(\upsilon,\mu).S(\upsilon,\mu)$



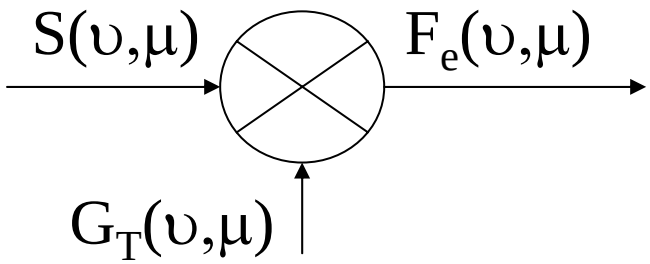
Direct 2D FFT:

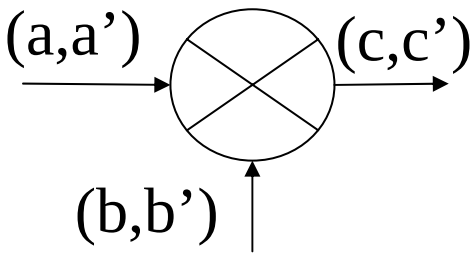


Inverse 2D FFT:

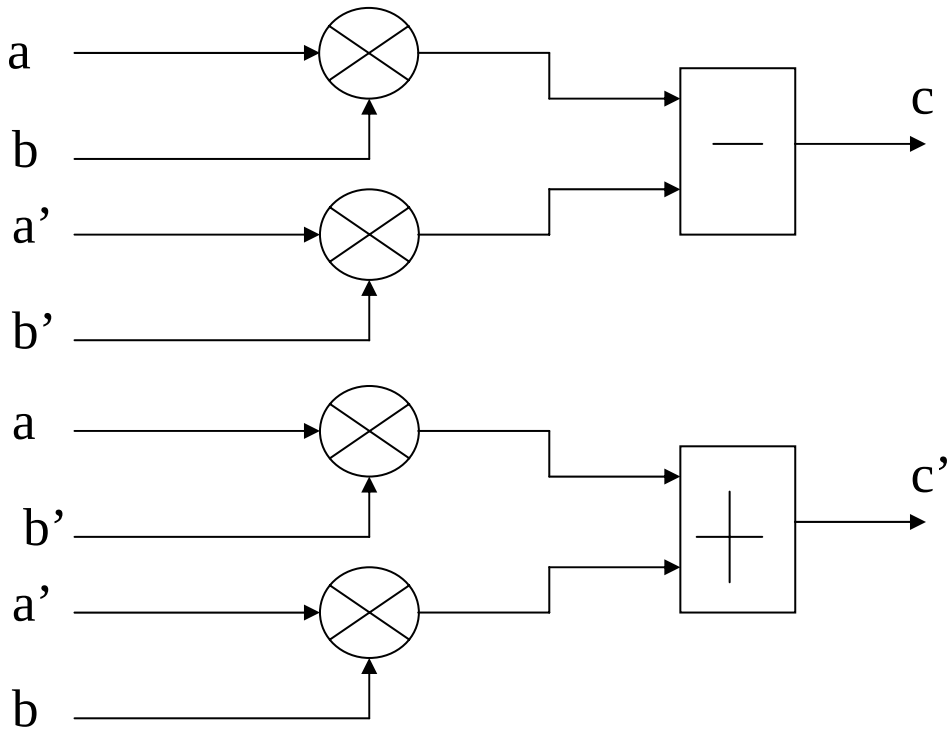


Complex Multiplication:





x:real component
x':imaginary component



2- Electronic components for real-time DSP and arithmetic calculation

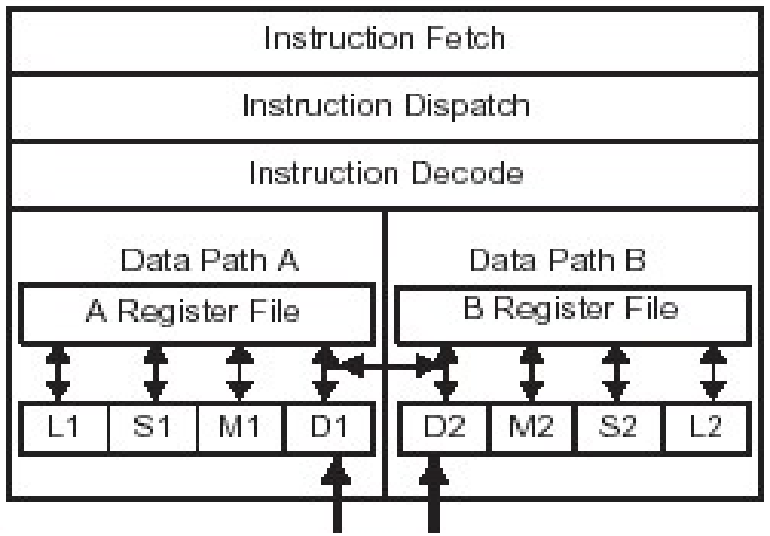
1- Digital Signal Processors (DSP)

- DSPs are optimized for signal processing applications compared to general purpose processors (GPP).
- The programmable flexibility of DSPs enables developers to implement complex algorithms in software (for example in C).
- DSPs offer many architectural features that actually reduce the number of instructions necessary for efficient signal processing.
- In other words, comparing performance is much more than counting instructions. For example, the VLIW architecture of TI's C6x generation of DSPs can initiate up to 8 operations each cycle clock.
- Integrated specialized compute engines increase performance by executing Complex functions in hardware.
-DSPs are also optimized for specific applications by providing a balanced mix of performance, integrated peripherals, and on-chip memory.

TMS320 C6X DSP CPU description

(see ti.com)

- VLIW processor architecture,
- different pipeline levels,
- 8 functional units in parallel,
- 2 datapaths A and B,
- 32 general purpose registers.



VLIW : Very Long Instruction Word

An instruction VLIW for C6201 == 32 bytes

FFT performance with a DSP C67 @167MHZ
- 124μs for a 1D-1024 FFT

For example: 1024x1024 image
-124μs for 1D-1024 FFT
- first scan: 1024x124μs=127ms
- second scan: 127ms
- total= 254ms for 2D-FFT-> **4 fps for a 1Kx1K image**

2- Field Programmable Gate Arrays (FPGA)

Main FPGA advantages for real time image processing:

-Large two-dimensionnal array of logic blocks (several millions of equivalent logic gates actually)

-Dedicated resources for multiplication and memory storage

-Well adapted for low-level processing (Filtering, FFT, Wavelets Transforms, …)

- Fast data processing (>100 Mhz)

- Easy to program (VHDL language)

FPGA example: SPARTAN-3E from Xilinx (see xilinx.com)

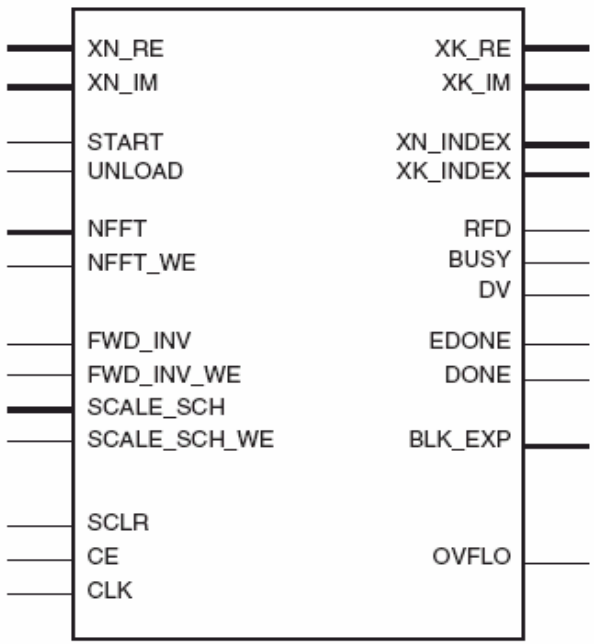
- 100K to 1.6M system gates

-From 66 to 376 IOs

-Up to 648 Kbits of block RAMs

-Up to 36 18x18 multipliers

FFT core which can be downloaded (free using):



FFT performances with Spartan-3E:

Table 11: Spartan-3E Family Pipelined Streaming I/O: Performance and Resource Utilization

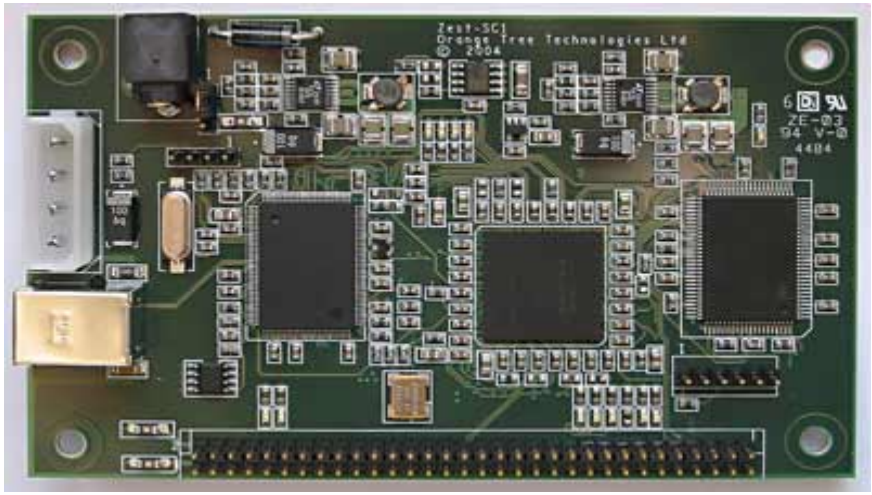
Transform Length	Slices	Block RAMs	18x18 Mults	Max Clock Frequency (MHz)	Transform Time		Device
					Clock Cycles	Time (µs)	
				-4		-4	
256	2203	4	12	137	256	1.86	xc3s500e
1024	2774	7	16	132	1024	7.76	xc3s1200e
8192	3687	24	24	132	8192	62.01	xc3s1600e

For example: 1024x1024 image
-7.76µs for 1D-1024 FFT
- first scan: 1024x7.76µs=8ms
- second scan: 8ms
- total= 16ms for 2D-FFT-> **62.5 fps for a 1Kx1K image**

-> FPGA 15 times faster than DSP for FFT with the same clock frequency!!

3- Implementation proposal: using FPGA boards

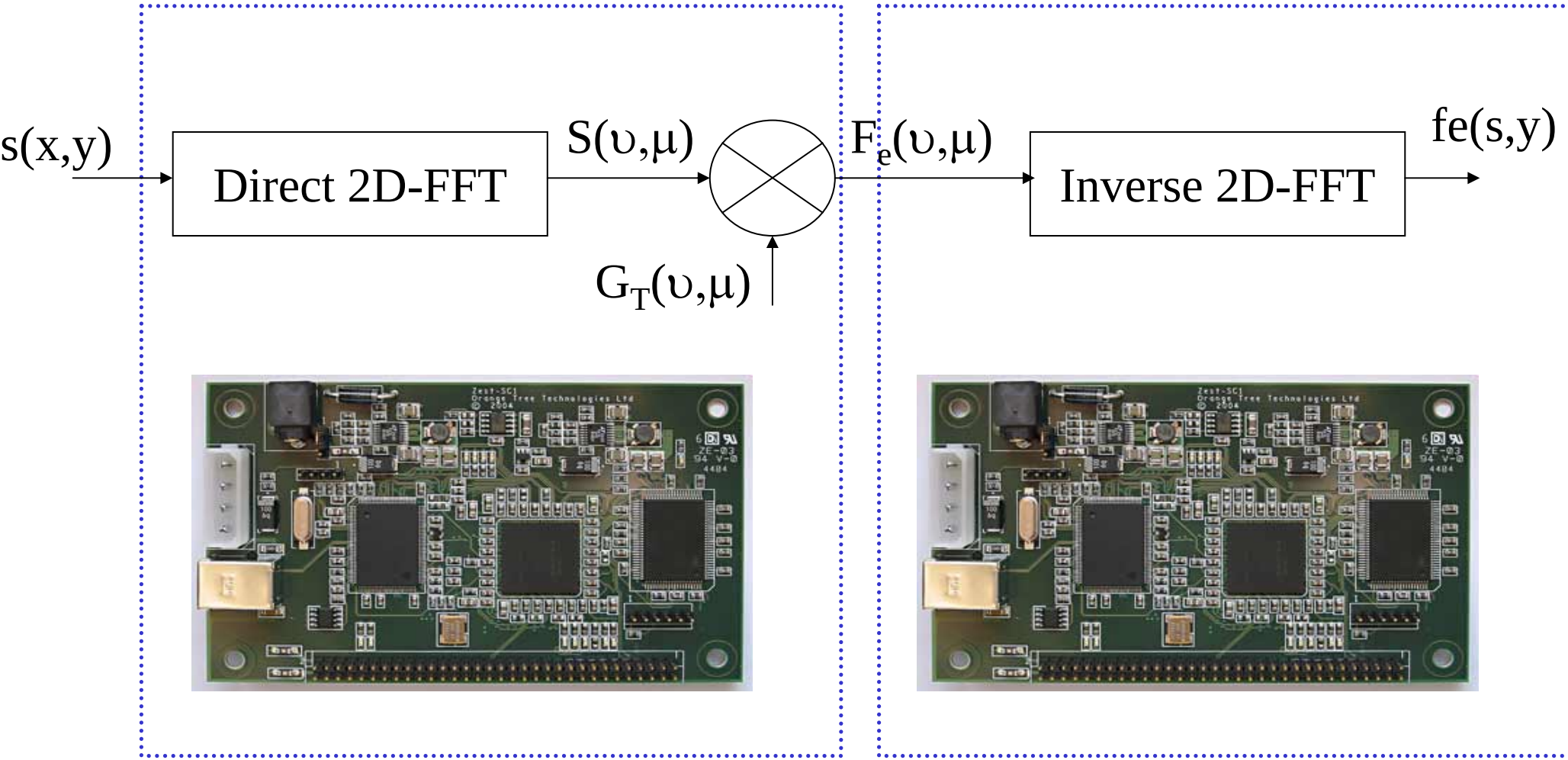
ZestSC1 Board example



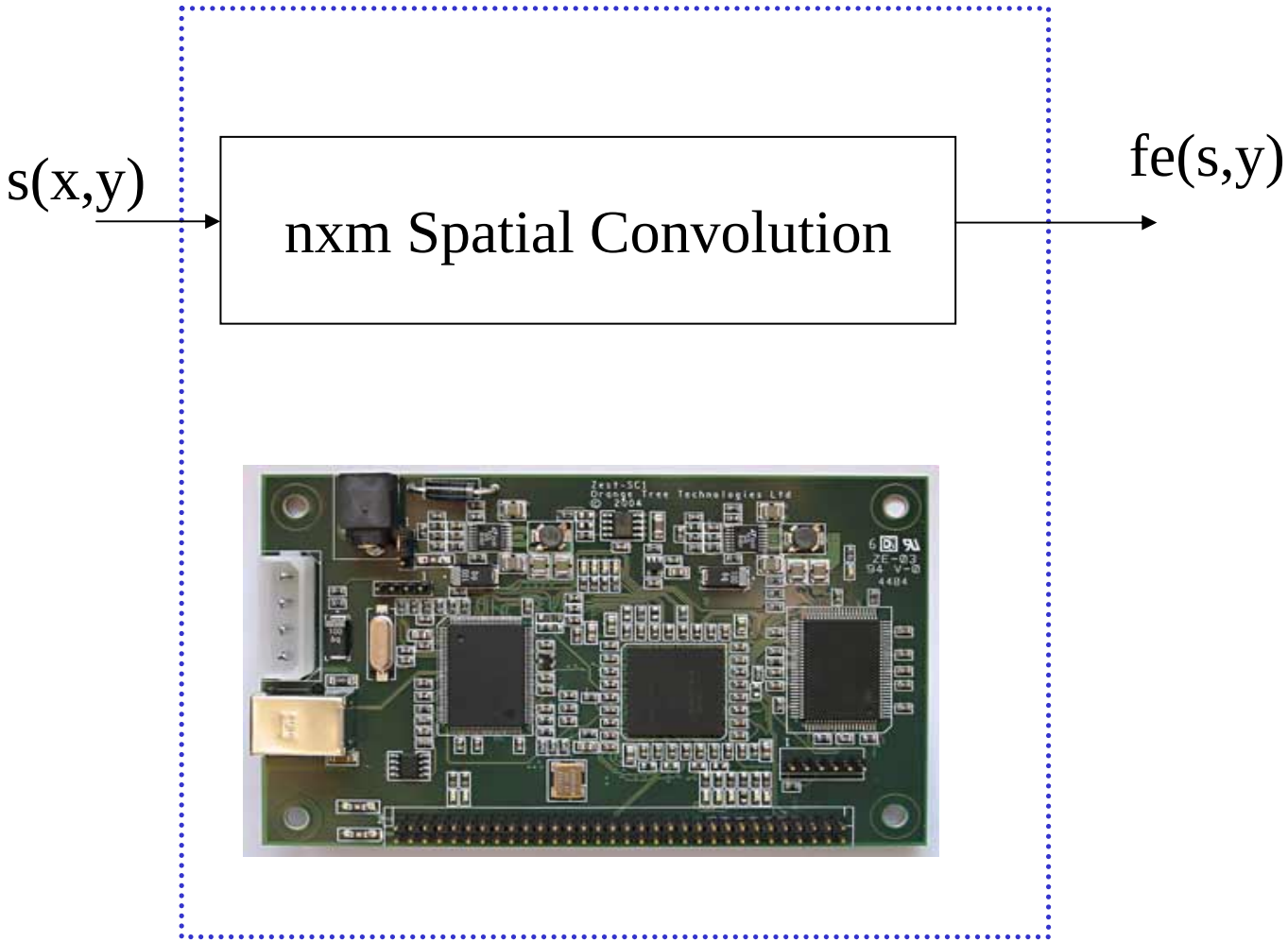
The ZestSC1 FPGA USB board is a low-cost high performance desktop FPGA board. With a 480 Mbits/sec USB interface, FPGA up to 1 million gates, memory and user I/O pins, it is ideal for data acquisition and processing, and FPGA development work.

400,000 gates Spartan-3 with 1MByte SRAM
1,000,000 gates Spartan-3 with 1MByte SRAM
1,000,000 gates Spartan-3 with 8MByte SRAM

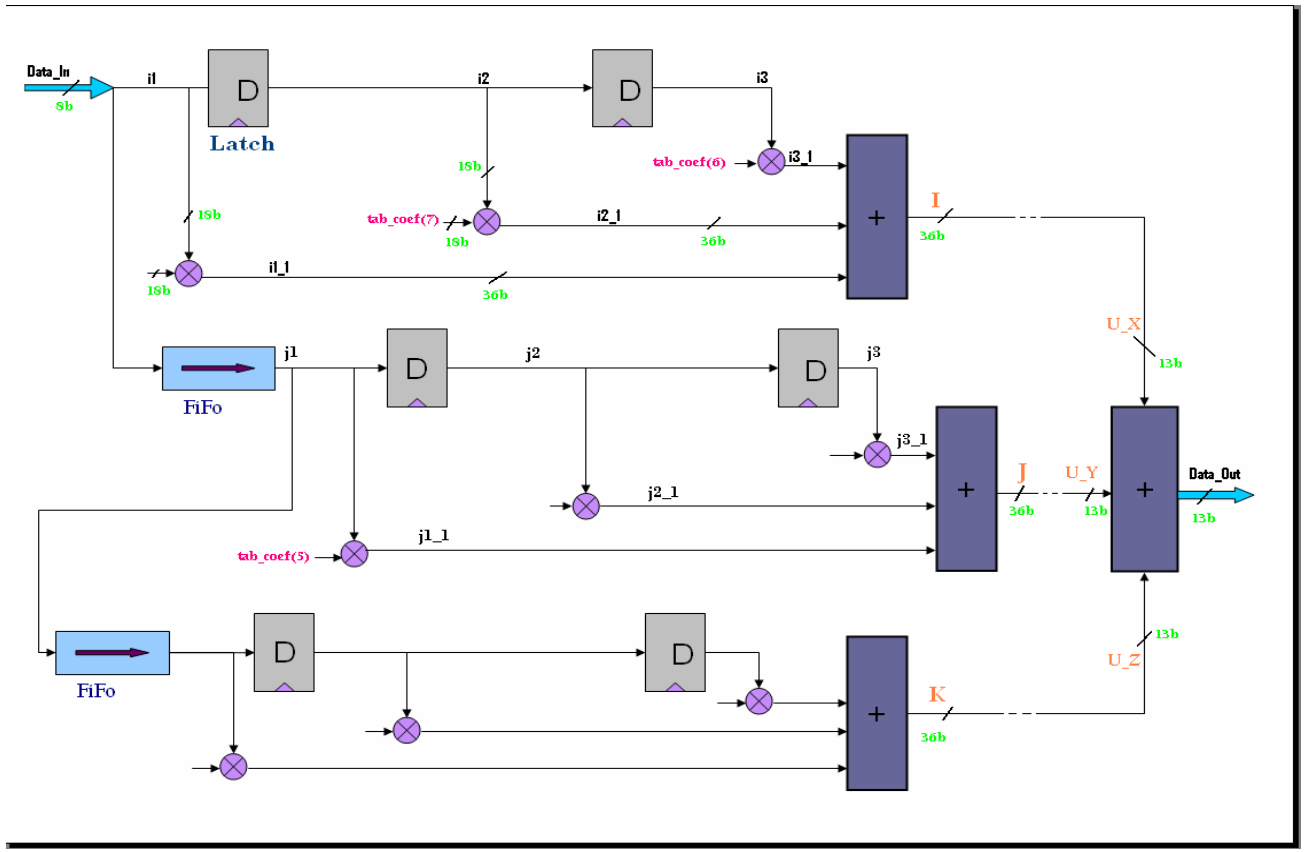
- Needs 2 Spartan-3E FPGA boards:



2D Spatial Convolution



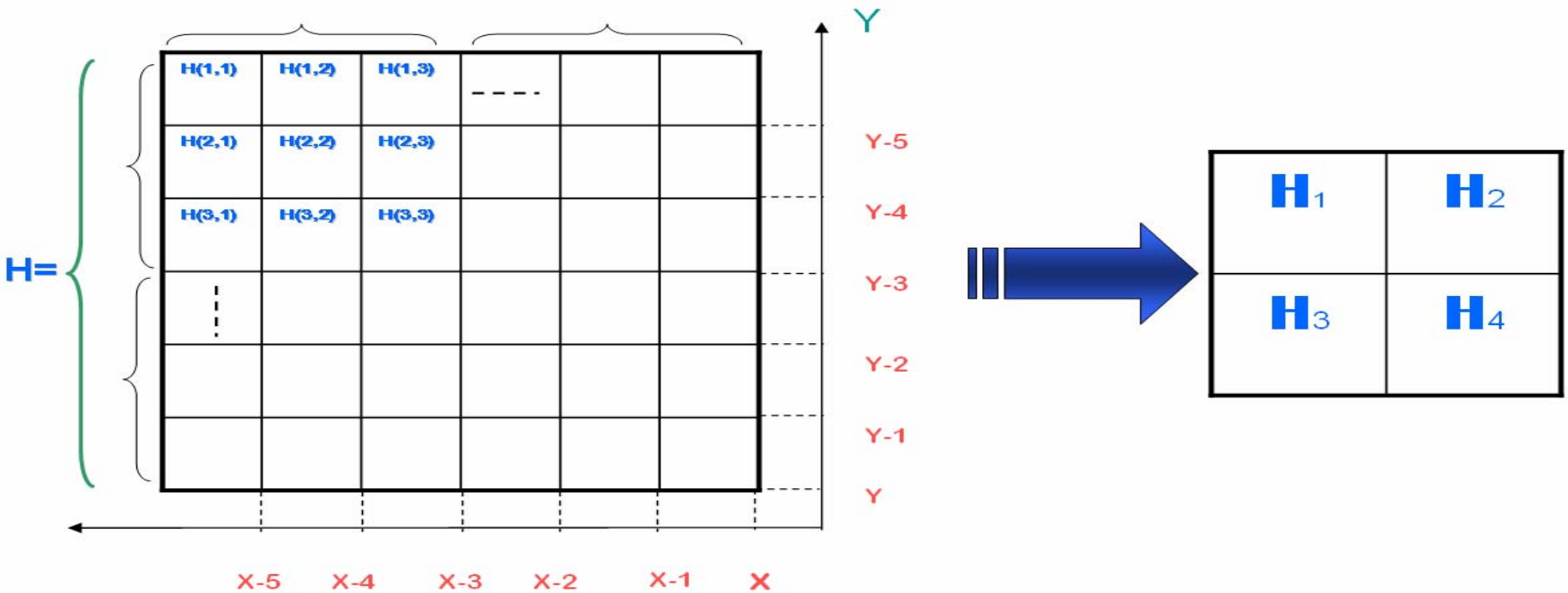
3 - Convolution 3x3 architecture



Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	287	7680	3%
Number of Slice Flip Flops	358	15360	2%
Number of 4 input LUTs	347	15360	2%
Number of bonded IOBs	46	221	20%
Number of BRAMs	2	24	8%
Number of MULT18X18s	9	24	37%
Number of GCLKs	1	8	12%

Results of Synthesis for FPGA Spartan 3 - 1000

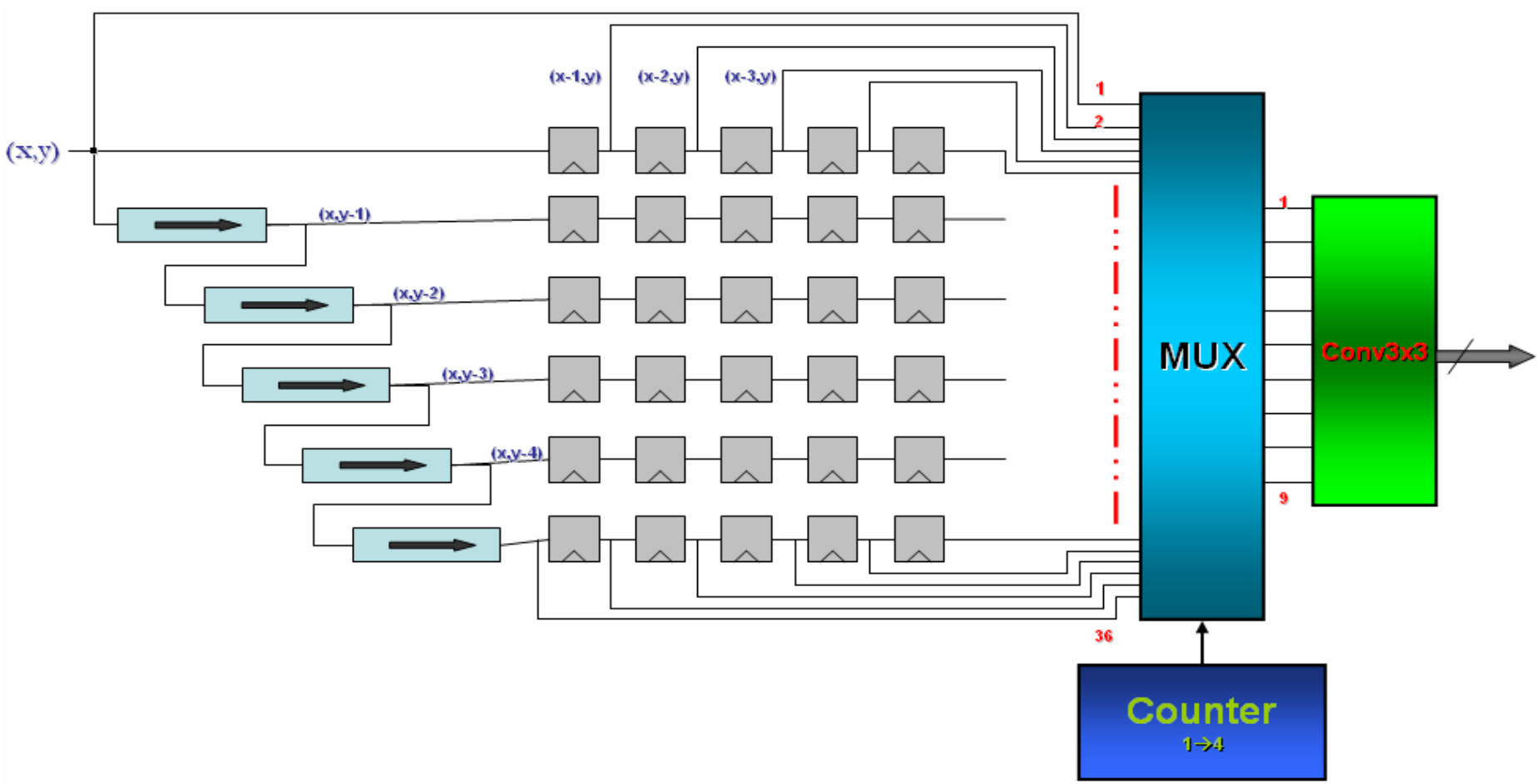
4 - Convolution 6x6 architecture



$$H_1 = \sum_{i=1}^{3} \sum_{j=1}^{3} H(i,j) * I(x-i, y-j)$$

$$H = \sum_{k=1}^{4} H_k$$

Convolution 6x6: four 3x3 convolution blocks



Convolution 6x6 architecture

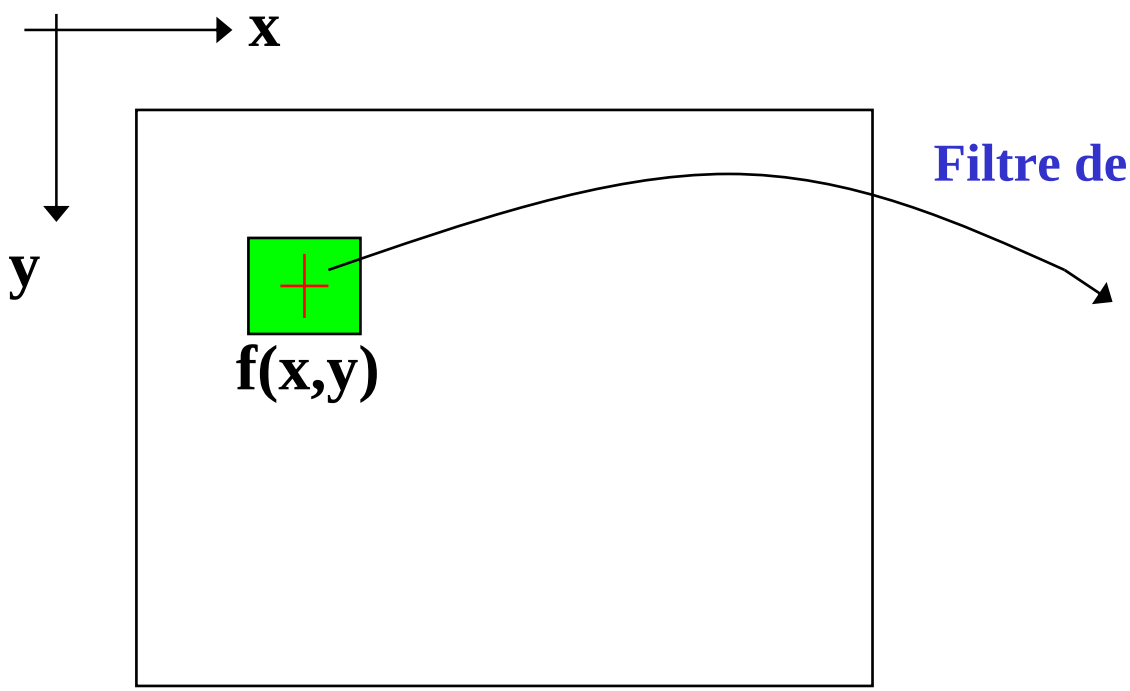
Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	287	7680	3%
Number of Slice Flip Flops	358	15360	2%
Number of 4 input LUTs	347	15360	2%
Number of bonded IOBs	46	221	20%
Number of BRAMs	2	24	8%
Number of MULT18X18s	9	24	37%
Number of GCLKs	1	8	12%

3x3 implementation

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	677	7680	8%
Number of Slice Flip Flops	774	15360	5%
Number of 4 input LUTs	865	15360	5%
Number of bonded IOBs	46	221	20%
Number of BRAMs	5	24	20%
Number of MULT18X18s	9	24	37%
Number of GCLKs	1	8	12%

6x6 implementation

III-4 Détection de Contours:



Filtre de Sobel

$$h1=\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad h2=\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$g(x,y) = |f(x,y) * h1| + |f(x,y) * h2|$$

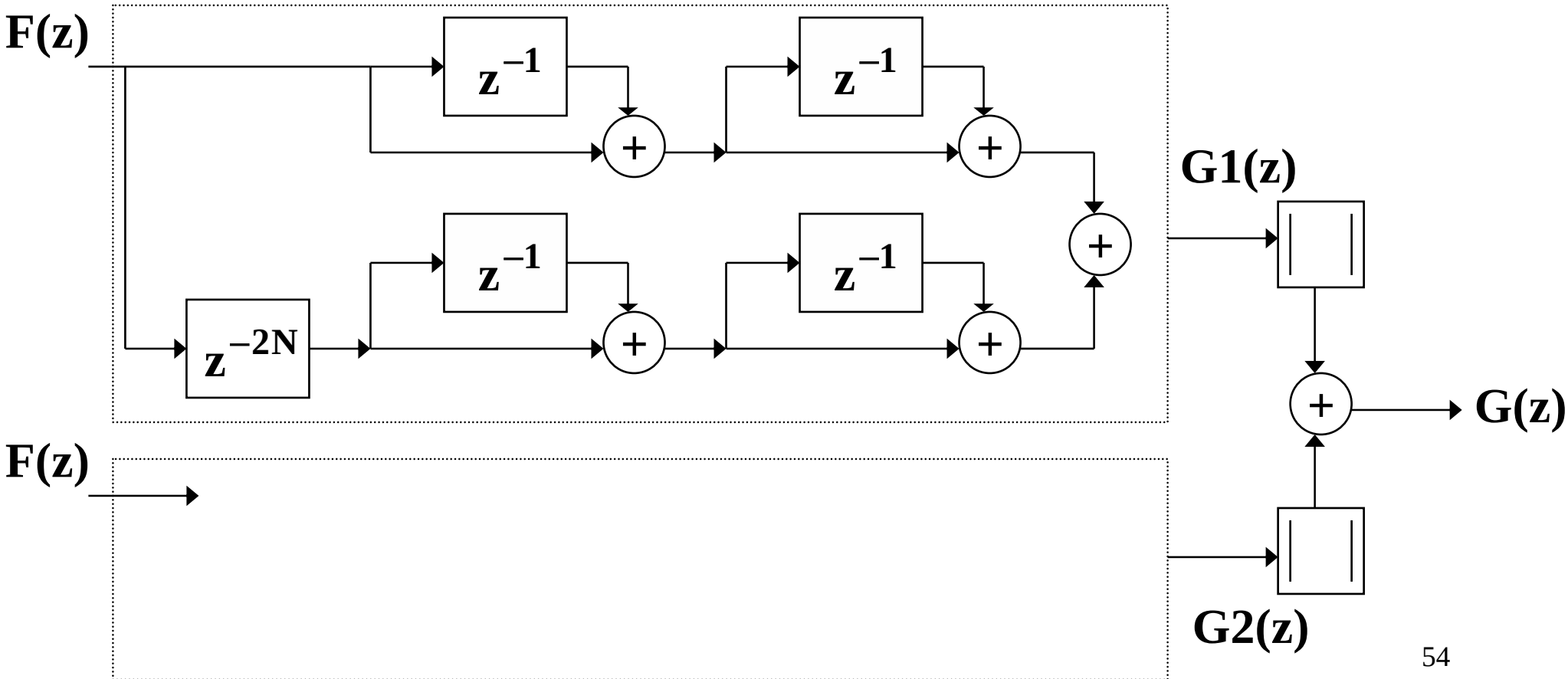
$$g1(x,y) = f(x,y) * h1 \qquad \text{et} \quad g2(x,y) = f(x,y) * h2$$

$$g1(x,y) = \{f(x,y) + 2f(x-1,y) + f(x-2,y)\} - \{f(x,y-2) + 2f(x-1,y-2) + f(x-2,y-2)\}$$

$$G1(z) = F(z) + 2z^{-1}F(z) + z^{-2}F(z) - z^{-2N}\left[F(z) + 2z^{-1}F(z) + z^{-2}F(z)\right]$$

$$G1(z) = F(z)\left[1 + z^{-1}\right].\left[1 + z^{-1}\right] \quad - \quad z^{-2N}F(z)\left[1 + z^{-1}\right].\left[1 + z^{-1}\right]$$

Avec N : Nombre Pixels par ligne



Exercice:

1) Retrouver l'équation et le schéma bloc permettant de générer G2(z)

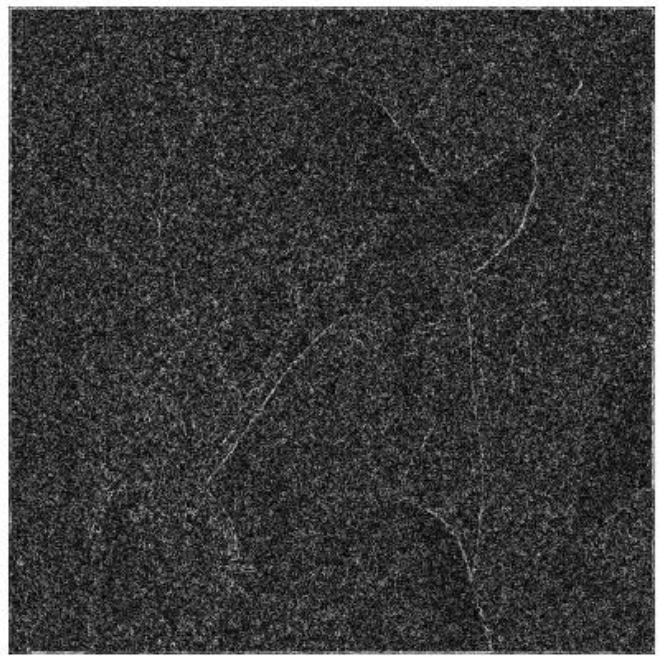
2) Proposer le programme VHDL réalisant le filtre de Sobel

Image originale avec un rapport signal sur bruit élevé

Image originale avec un rapport signal sur bruit faible

Image originale

Contours obtenus avec le filtre de Sobel



⇨ **Contours noyés dans le bruit**

⟶ Détection de Contours par Filtrage Optimal:

Filtres de CANNY-DERICHE, SHEN-CASTAN, …

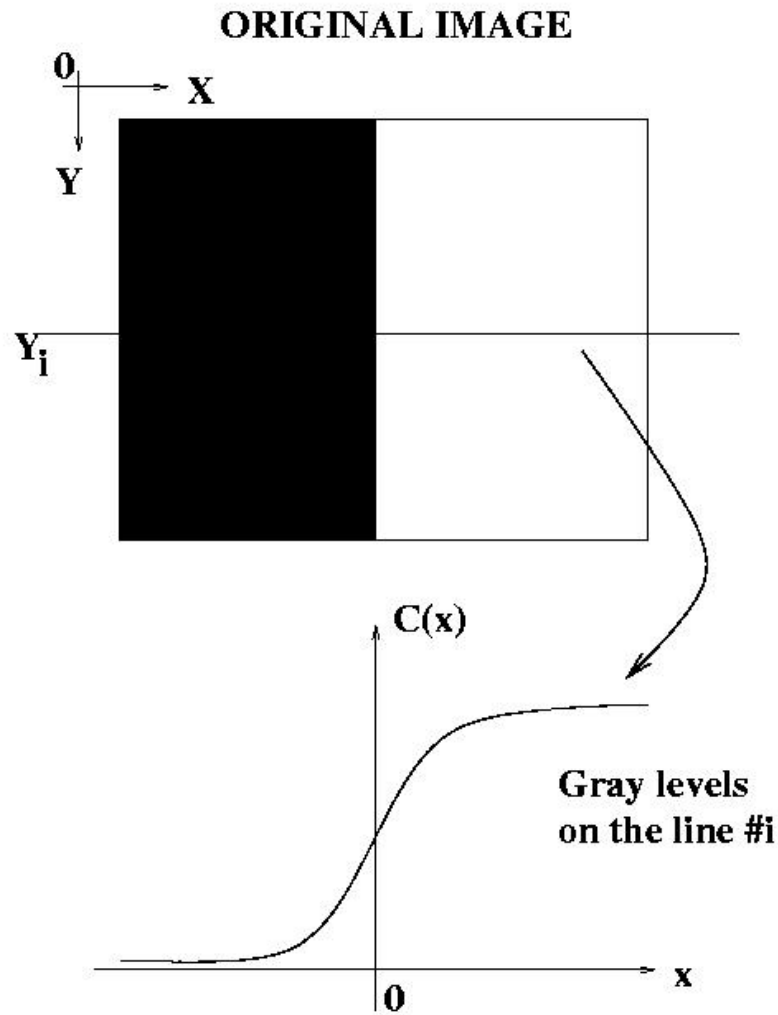
Filtre de CANNY-DERICHE basé sur 3 critères:

- **Augmentation Rapport Signal sur Bruit**

- **Meilleure localisation**

-**Elimination réponses multiples**

Description du filtre de Canny-Deriche:



Modèle d'Image:

$$C(x) = 1 - \frac{e^{-sx}}{2} \quad pour \; x \geq 0$$

et

$$C(x) = \frac{e^{sx}}{2} \quad pour \; x \prec 0$$

Utilisation de 2 Filtres: Filtre Dérivateur et Filtre Lisseur

En considérant les 3 critères de Canny-Deriche
On obtient les deux filtres récursifs du 2ème ordre:

Transformée en Z du filtre Dérivateur:

$$d^{+}(z) = \frac{a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \quad (x \geq 0)$$

$$d^{-}(z) = \frac{a_1 z + a_2 z^{2}}{1 + b_1 z + b_2 z^{2}} \quad (x \prec 0)$$

Transformée en Z du filtre Lisseur:

$$\boldsymbol{l^{+}(z) = \frac{c_1 z^{-1} + c_2 z^{-2}}{1 + d_1 z^{-1} + d_2 z^{-2}} \quad (x \geq 0)}$$

$$\boldsymbol{l^{-}(z) = \frac{c_1 z + c_2 z^{2}}{1 + d_1 z + d_2 z^{2}} \quad (x \prec 0)}$$

Passage de l ’espace « Z » au domaine spatial:

Exemple pour $d^+(z)$:

$$d^+(z) = \frac{g^+(z^{-1})}{i(z^{-1})} = \frac{a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

$$g^+(z^{-1}).(1 + b_1 z^{-1} + b_2 z^{-2}) = i(z^{-1}).(a_1 z^{-1} + a_2 z^{-2})$$

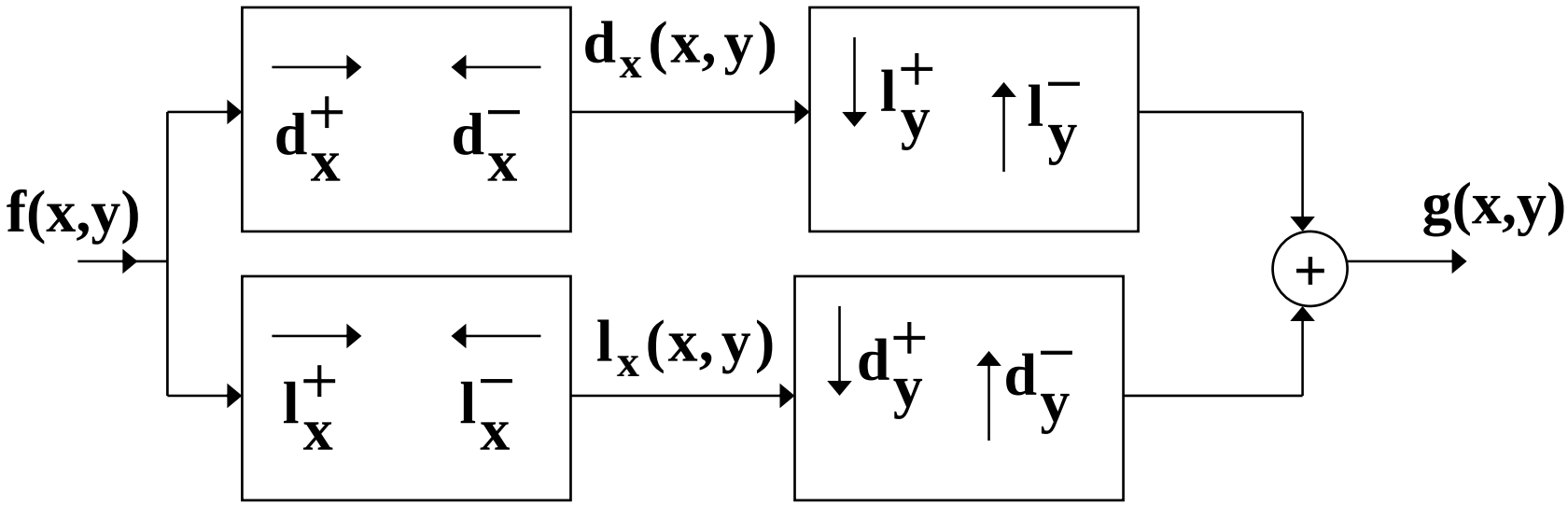
$$g^+(z^{-1}) = a_1 z^{-1}.i(z^{-1}) + a_2 z^{-2}.i(z^{-1}) - b_1 z^{-1}.g^+(z^{-1}) - b_2 z^{-2}.g^+(z^{-1})$$

comme $Z^{-1}\{i(z^{-1})\} = i(x)$ *et* $Z^{-1}\{az^{-k}.i(z^{-1})\} = a.i(x-k)$

$$g^+(x) = a_1.i(x-1) + a_2.i(x-2) - b_1.g^+(x-1) - b2.g^+(x-2)$$

Total des Calculs: 1 Addition, 2 Soustractions et 4 Muliplications pour un pixel et pour un filtre!

Détection de Contours par Filtrage Optimal (Canny-Deriche)



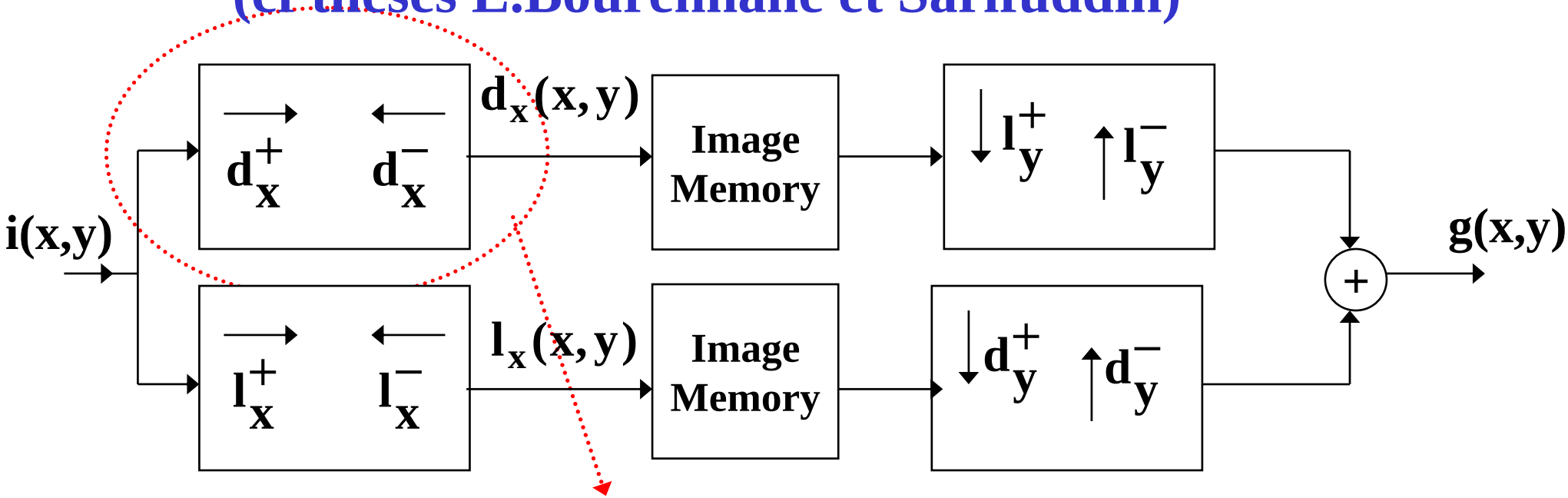
$$d_x^+(x,y) = a_1 f(x-1,y) + a_2 f(x-2,y) + b_1 d_x^+(x-1,y) + b_2 d_x^+(x-2,y)$$

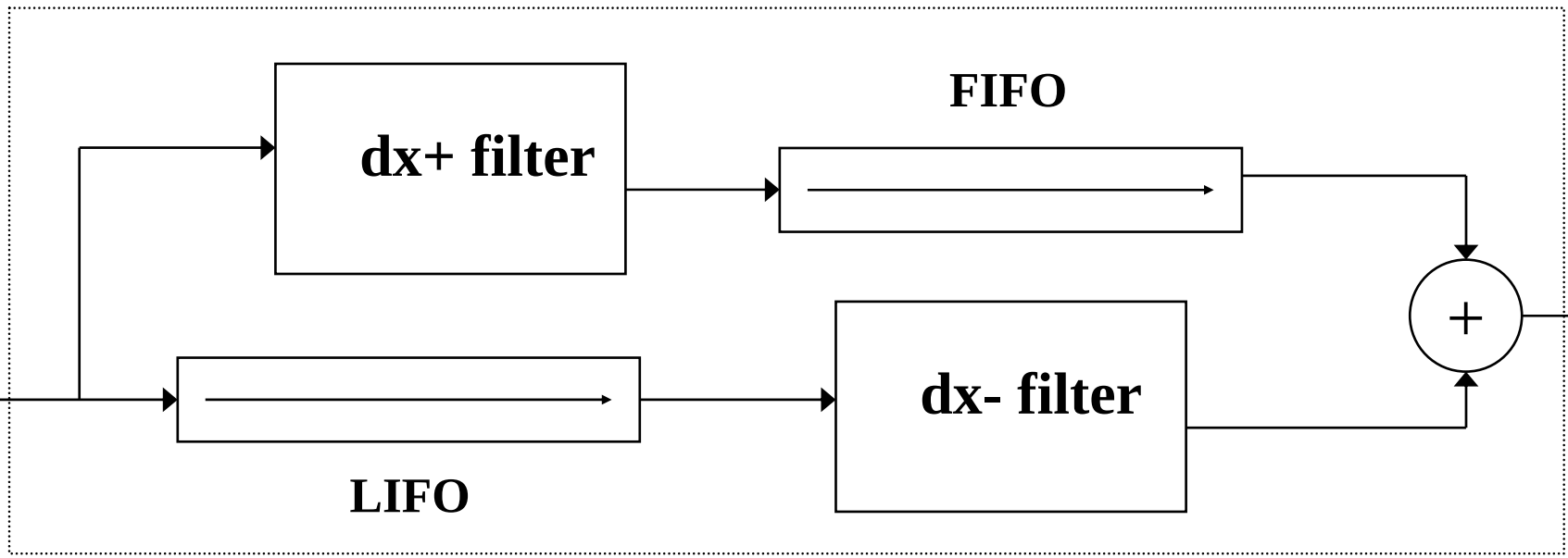
$$d_x^-(x,y) = a_1 f(x+1,y) + a_2 f(x+2,y) + b_1 d_x^-(x+1,y) + b_2 d_x^-(x+2,y)$$

$$d_x(x,y) = d_x^+(x,y) + d_x^-(x,y)$$

Equations semblables pour $l_x^+, l_x^-, d_y^+, d_y^-, l_y^+$ et l_y^+

Implementation du filtre de Canny-Deriche optimisé (cf thèses E.Bourennane et Sarifuddin)





Optimized Canny-Deriche filter implementation: Total electronics resources

Example for an image with 512x512 pixels coded on 8 bits each:

- 8 IIR filters:

-For each IIR filter: 1 additions, 2 substractions and 4 multiplications

so for the 8 IIR filters: 13 additions (8+4+1), 16 substractions and 32 multiplications

inside one FPGA Xilinx Virtex-II

-4 FIFOs and 4 LIFOs:

inside the same FPGA Xilinx Virtex-II used for 8 IIR filters

-Two Image Memories:

- each image memory stores results coded on 16 bits: 512x512x16bits (256Kx16 bits)
- needs 512Kx16bits

-outside FPGA, but with only one memory component

- For example using of the Virtex-II XC2V3000 from Xilinx:

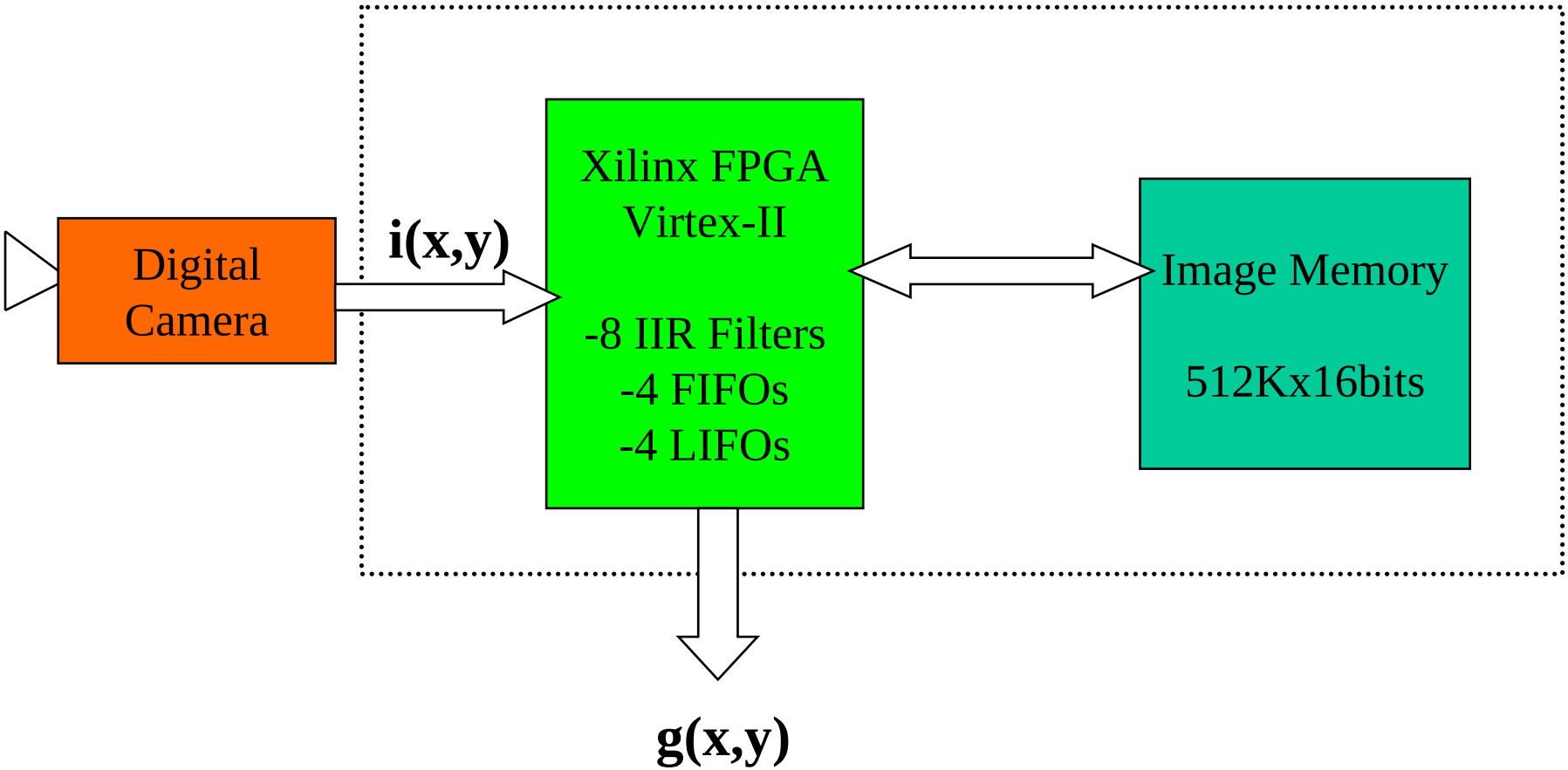
-3,000 system gates organised in 14,336 slices

-96 dedicated 18-bit x 18-bit multipliers blocks

-1,728 Kbits of dual-port RAM in 18 K-bit SelectRAM resources

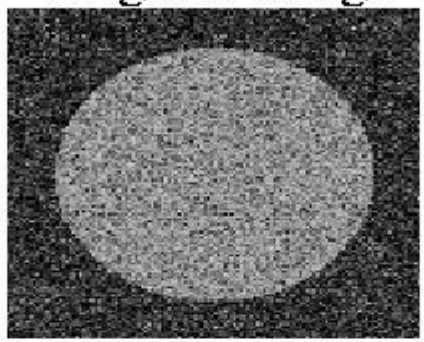
-720 I/O pads allow to manage input from camera, exchange data beteween FPGA et Memory, outpout result of edge detection.

Optimized Canny-Deriche filter implemented in a simple board

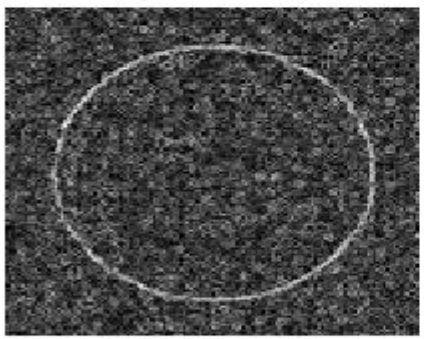


Results in SUN station

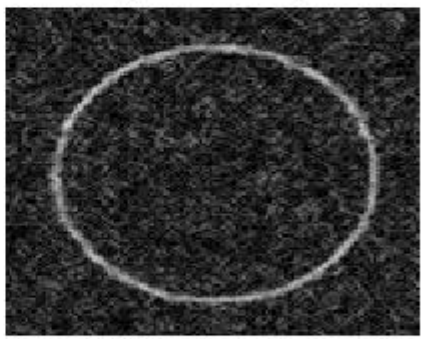
Original image



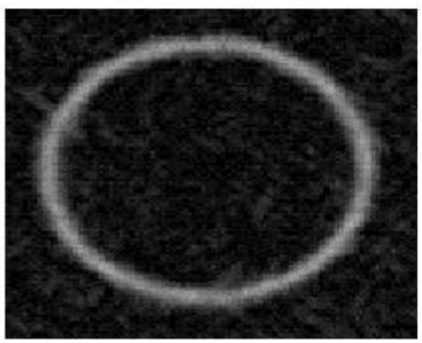
Edges for j=3



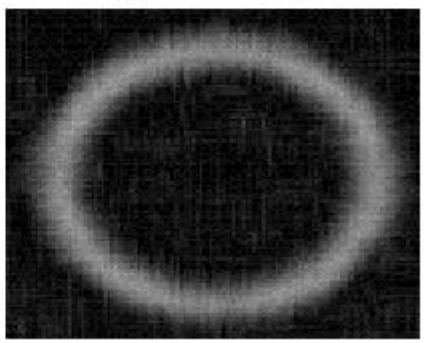
Edges for j=2



Edges for j=1



Edges for j=0



Modèle d'image:

$$C(x) = 1 - \frac{e^{-sx}}{2} \quad pour\ x \geq 0$$

et

$$C(x) = \frac{e^{sx}}{2} \quad pour\ x \prec 0$$

with $\boldsymbol{s = 2^j}$

SOMMAIRE

I-Objectifs du Traitement du Signal et des Images Temps Réel

II- Présentation de la méthodologie « Adéquation Algorithme Architecture (AAA) »

III- Problématique d'implantation de filtres numériques

IV- AAA avec des approches SIMD

V- AAA avec une approche MISD

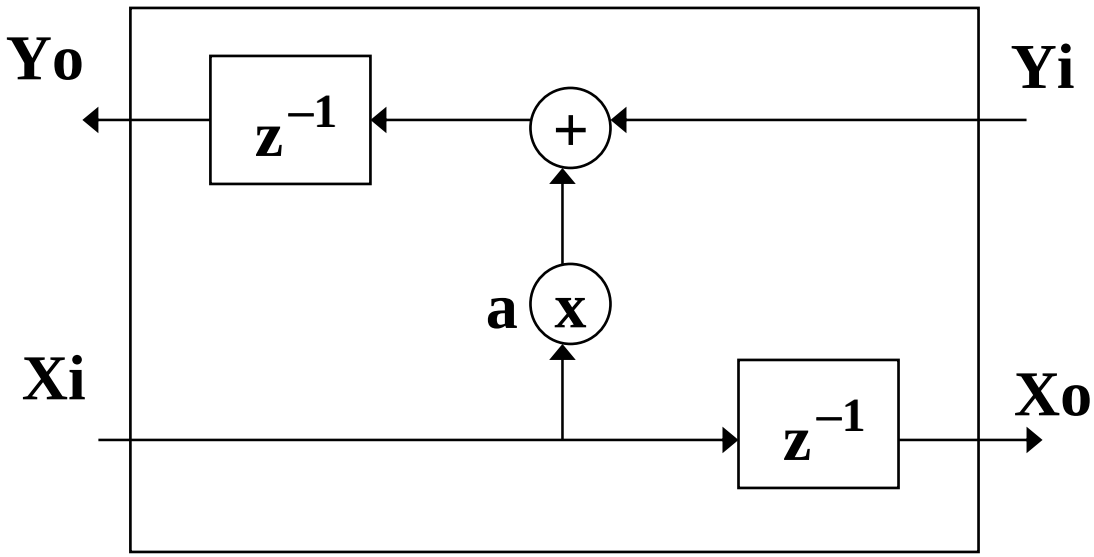
VI- AAA par optimisation des opérateurs arithmétiques

VII- AAA sous contraintes de dimensionnement

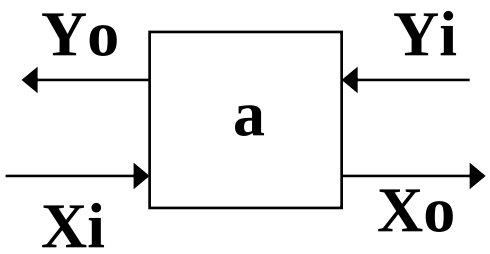
VIII- Exemples d'Application

AAA avec une approche SIMD

(Réseaux systoliques)



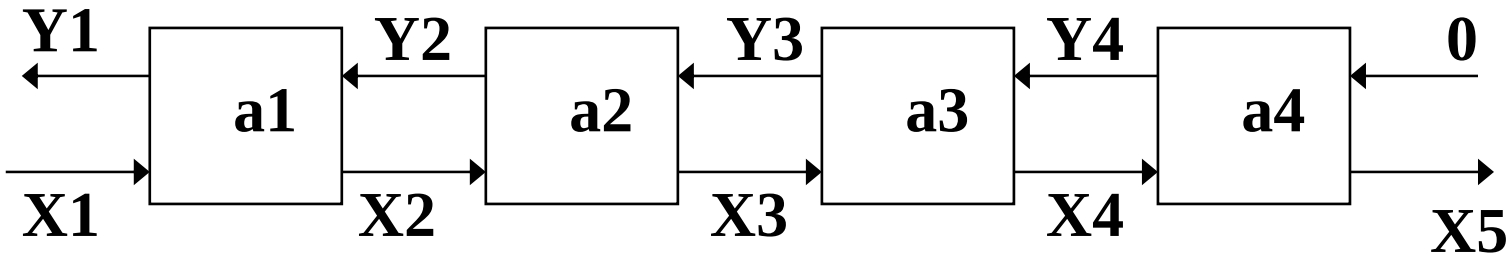
=



$$X_o = z^{-1} X_i$$

$$Y_o = (aX_i + Y_i) z^{-1}$$

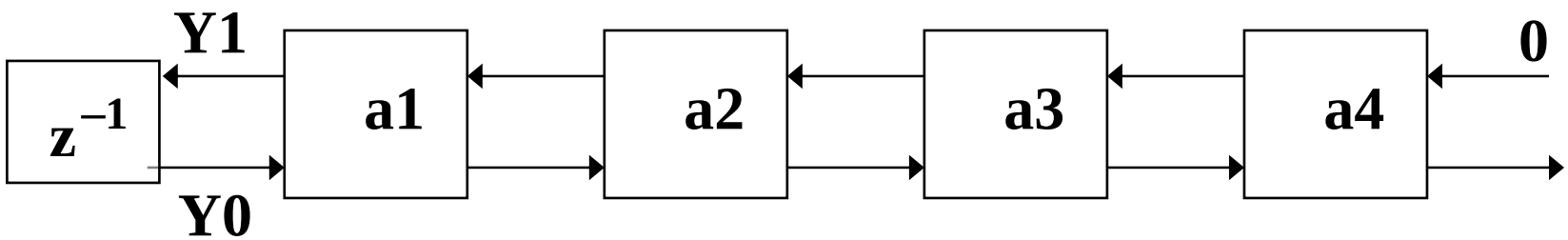
Exemple 1: Filtre non récursif d ’efficacité 2



$$Y_1(z) = a_1 z^{-1} X_1(z) + a_2 z^{-3} X_1(z) + a_3 z^{-5} X_1(z) + a_4 z^{-7} X_1(z)$$

Question: Filtre non récursif d’efficacité 1?

Exemple 2: Filtre récursif d ’efficacité 2



Questions: - Expression de Y1?
- Filtre récursif d ’efficacité 1?
- Filtre de Sobel?
- Filtre de Canny-Deriche?

SOMMAIRE

I-Objectifs du Traitement du Signal et des Images Temps Réel

II- Présentation de la méthodologie « Adéquation Algorithme Architecture (AAA) »

III- Problématique d'implantation de filtres numériques

IV- AAA avec des approches SIMD

V- AAA avec une approche MISD

VI- AAA par optimisation des opérateurs arithmétiques

VII- AAA sous contraintes de dimensionnement

VIII- Exemples d'Application

V- AAA avec une approche MISD: Méthode par anticipation de calcul

Références:

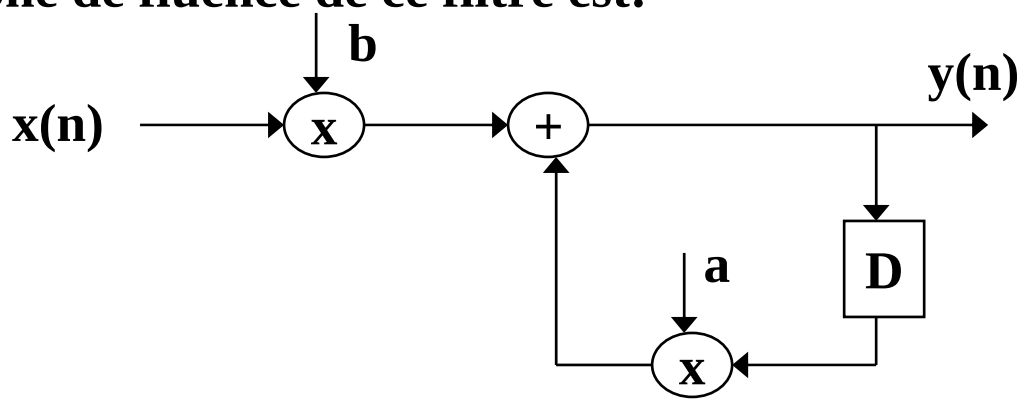
1- K.PARHI and al. : « Pipeline Interleaving and Parallelism in Recursive Digital Filters. » (IEEE Transactions on Acoustics Speech and Signal Processing, VOL 37, N°7, Ju1y1989)

2- E.BOURENNANE: « Conception et Implantation d 'un détecteur de contours optimisé sous forme d 'un circuit ASIC. » Thèse de l 'Université de Bourgogne, Février 1994.

V-1 INTRODUCTION

Soit le filtre récursif d'ordre 1 défini par son équation aux différences: y(n)=b.x(n)+a.y(n-1) avec x(n) l 'entrée du filtre et y(n) la sortie du filtre.

Le graphe de fluence de ce filtre est:



Représentons le filtre pour l ’échantillon n+1:

$$y(n+1)=b.x(n+1)+a.y(n)$$

soit en remplaçant y(n) par son expression précédente:

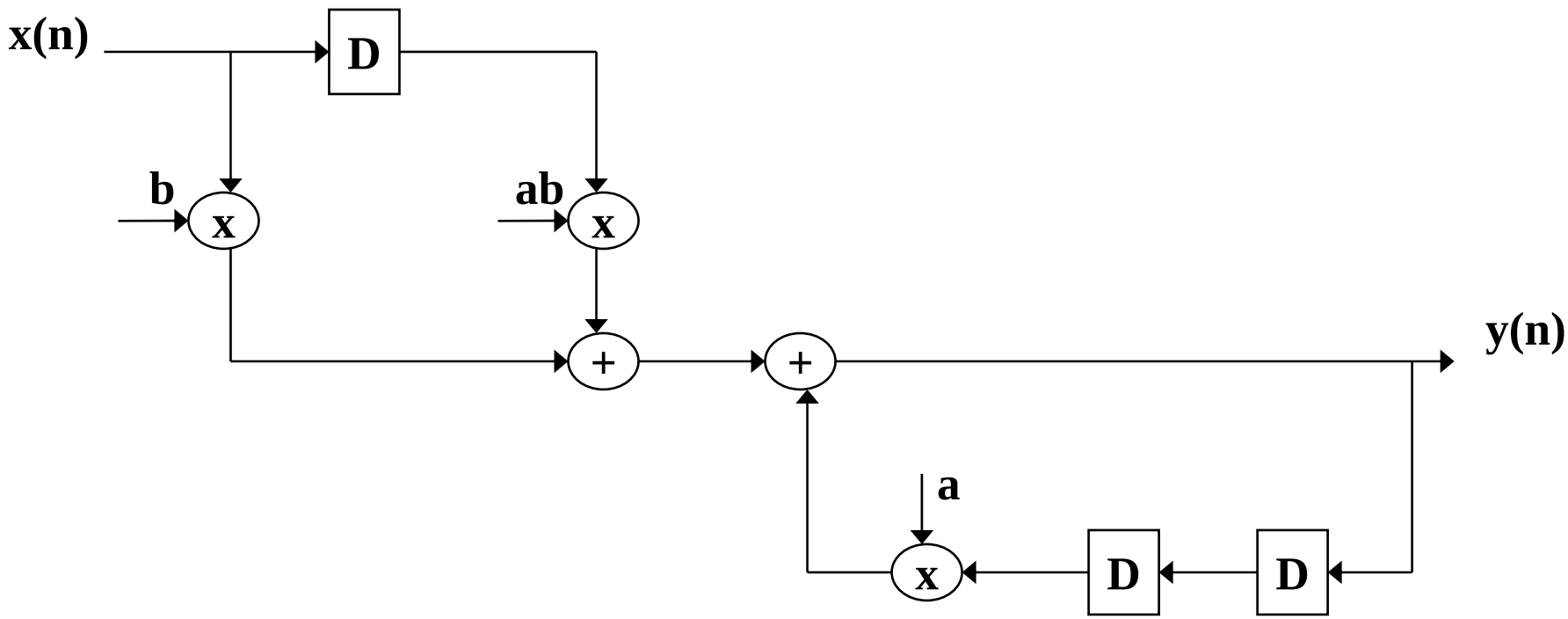
$$y(n+1)=b.x(n+1)+a.[b.x(n)+a.y(n-1)]$$

ou encore: $y(n+1)= b.x(n+1)+ab.x(n)+a.y(n-1)$

en exprimant à nouveau y(n) à partir de cette nouvelle expression:

$$y(n)= b.x(n)+ab.x(n-1)+a.y(n-2)$$

Le nouveau graphe de fluence de ce filtre est:



La nouvelle représentation fait apparaître un retard supplémentaire dans la partie récursive du filtre (boucle de retour):

→ **Le temps de calcul est divisé par deux**

En effet la période d ’échantillonnage est :

$$T = \frac{1}{L} \mathrm{Max}\left[\frac{D_l}{M_l}\right]$$

avec :

D_l : latence de calcul boucle l

M_l : nombre retards(bascules D) boucle l

L : temps échantillonnage chaque bascule

(L fois plus lent que temps échantillonnage T de x(n))

En général Dl est fixé donc pour diminuer T, augmenter L et/ou augmenter Ml

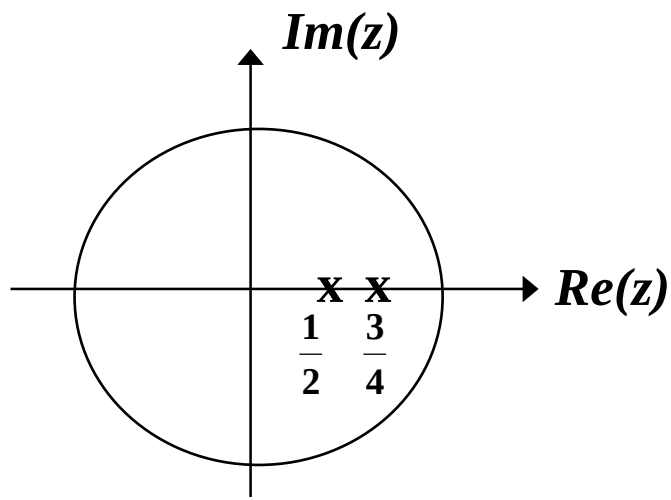
Dans l ’exemple précédent, L est fixé et Ml passe de 1 à 2 donc T est divisé par deux.

V-2 ANTICIPATION PAR AJOUT DE PÔLES REGROUPES (Méthode « CLUSTERED »)

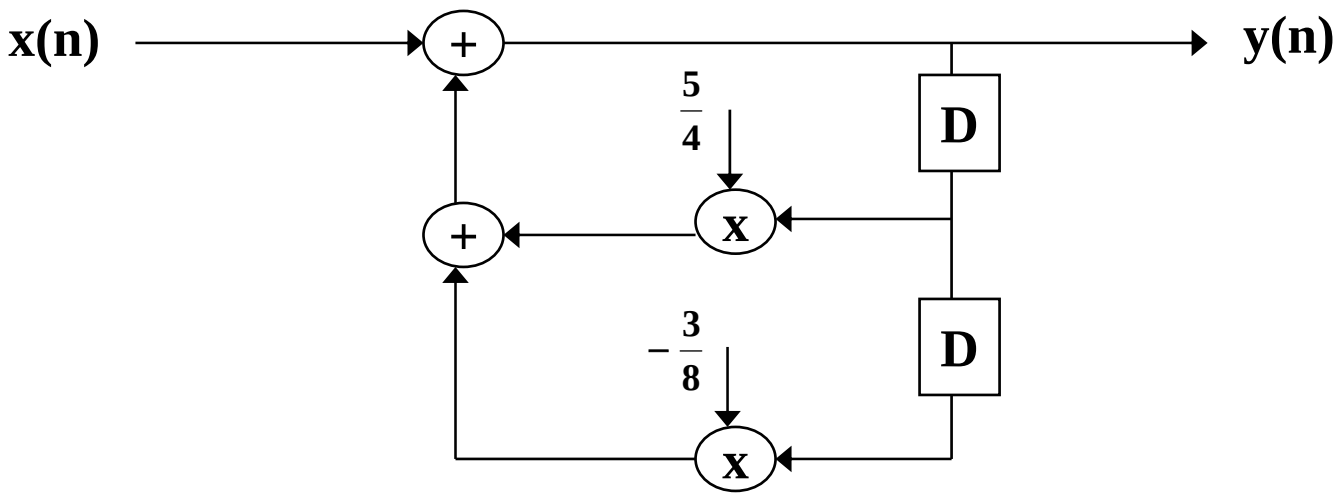
$$H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 - \sum_{i=1}^{N} a_i z^{-i}}$$

Exemple :

$$H(z) = \frac{1}{1 - \frac{5}{4}z^{-1} + \frac{3}{8}z^{-2}} \rightarrow \text{Pôles : } \frac{1}{2} \text{ et } \frac{3}{4}$$



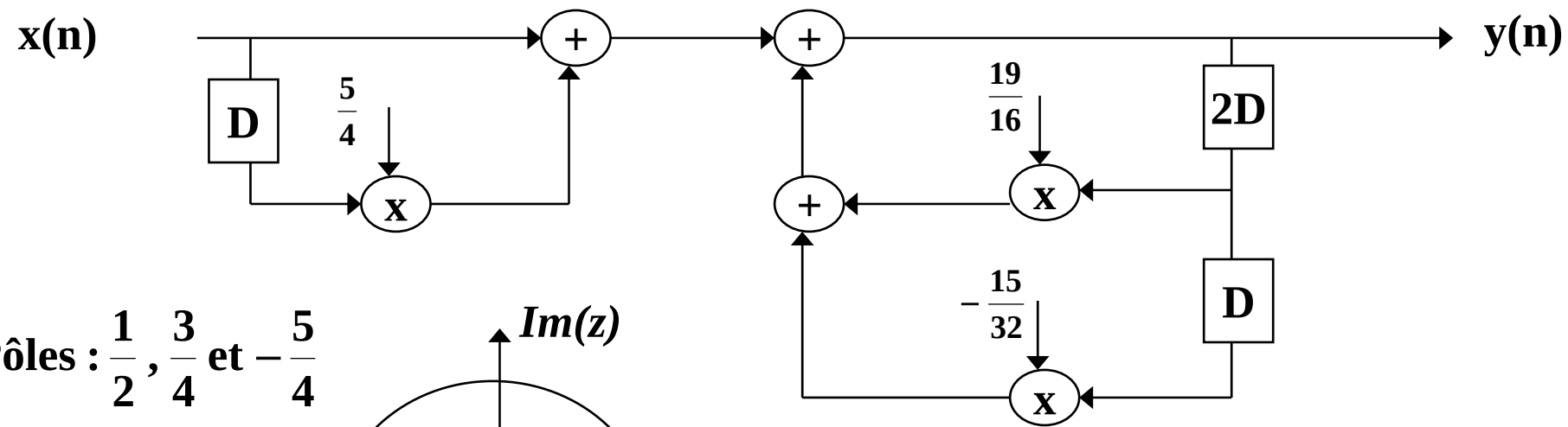
→ **Filtre récursif d 'ordre 2 stable**



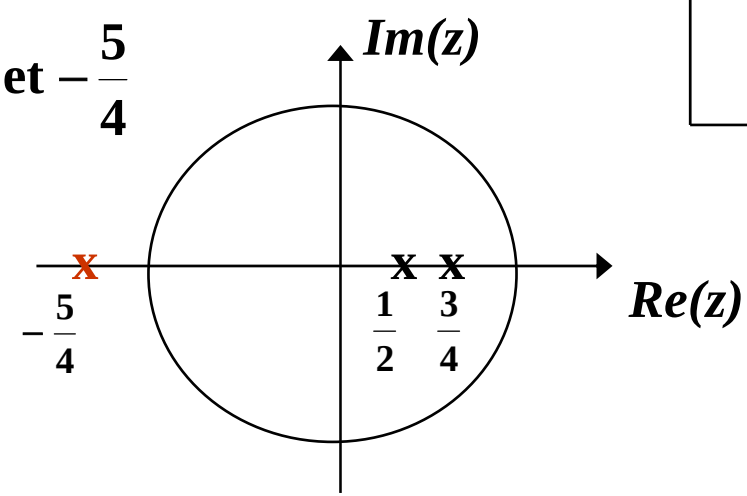
⟶ **Modification de H(z) dans le but d'éliminer le terme en z^{-1} :**

$$H(z) = \frac{1}{1-\frac{5}{4}z^{-1}+\frac{3}{8}z^{-2}} \times \frac{1+\frac{5}{4}z^{-1}}{1+\frac{5}{4}z^{-1}} = \frac{1+\frac{5}{4}z^{-1}}{1-\frac{19}{16}z^{-2}+\frac{15}{32}z^{-3}}$$

Obtention d'un nouveau filtre récursif d 'ordre 3: accélération facteur 2 (2 bascules D):



→ **Pôles : $\frac{1}{2}$, $\frac{3}{4}$ et $-\frac{5}{4}$**



⟶ **Filtre Instable**

V-3 ANTICIPATION DE CALCUL: Méthode « SCATTERED »

$$H(z)=\frac{N(z)}{D(z)}=\frac{N(z)}{1-\sum_{i=1}^{N} a_i z^{-i}}=\frac{N(z)\prod_{k=1}^{M-1} D(ze^{j2\Pi k/M})}{\prod_{k=0}^{M-1} D(ze^{j2\Pi k/M})}$$ **Avec M: nombre étages pipeline**

Exemple 1: Filtre récursif d'ordre 1

$$H(z)=\frac{N(z)}{D(z)}=\frac{1}{1-az^{-1}}$$ avec pôle $z=a$ stable si $-1 \prec a \prec 1$

Pour M = 3 Ajout de Pôles et de Zéros en $z=a.e^{j\frac{2\Pi}{3}}$ et en $z=a.e^{j\frac{4\Pi}{3}}=a.e^{-j\frac{2\Pi}{3}}$

$$H(z)=\frac{(z-a.e^{j\frac{2\Pi}{3}}).(z-a.e^{-j\frac{2\Pi}{3}})}{(1-az^{-1}).(z-a.e^{j\frac{2\Pi}{3}}).(z-a.e^{-j\frac{2\Pi}{3}})}=\frac{1+az^{-1}+a^2z^{-2}}{1-a^3z^{-3}}$$

Exemple 2: Filtre récursif d'ordre 2

$$H(z) = \frac{N(z)}{D(z)} = \frac{1}{1-(r_1+r_2)z^{-1}+r_1r_2z^{-2}} = \frac{1}{(1-r_1z^{-1})(1-r_2z^{-1})}$$

$\rightarrow$ 2 pôles r_1 et r_2 Filtre stable si $-1 \prec r_1 \prec 1$ et si $-1 \prec r_2 \prec 1$

Pour $M = 3$ Ajout de Pôles et de Zéros en :

$z = r_1 e^{\pm j\frac{2\Pi}{3}}$ et en $z = r_2 e^{\pm j\frac{2\Pi}{3}}$ ce qui garantit la stabilité du filtre.

$$H(z) = \frac{(1-r_1e^{j\frac{2\Pi}{3}}z^{-1})(1-r_1e^{-j\frac{2\Pi}{3}}z^{-1})(1-r2e^{j\frac{2\Pi}{3}}z^{-1})(1-r2e^{-j\frac{2\Pi}{3}}z^{-1})}{(1-r_1z^{-1})(1-r_2z^{-1})(1-r_1e^{j\frac{2\Pi}{3}}z^{-1})(1-r_1e^{-j\frac{2\Pi}{3}}z^{-1})(1-r2e^{j\frac{2\Pi}{3}}z^{-1})(1-r2e^{-j\frac{2\Pi}{3}}z^{-1})}$$

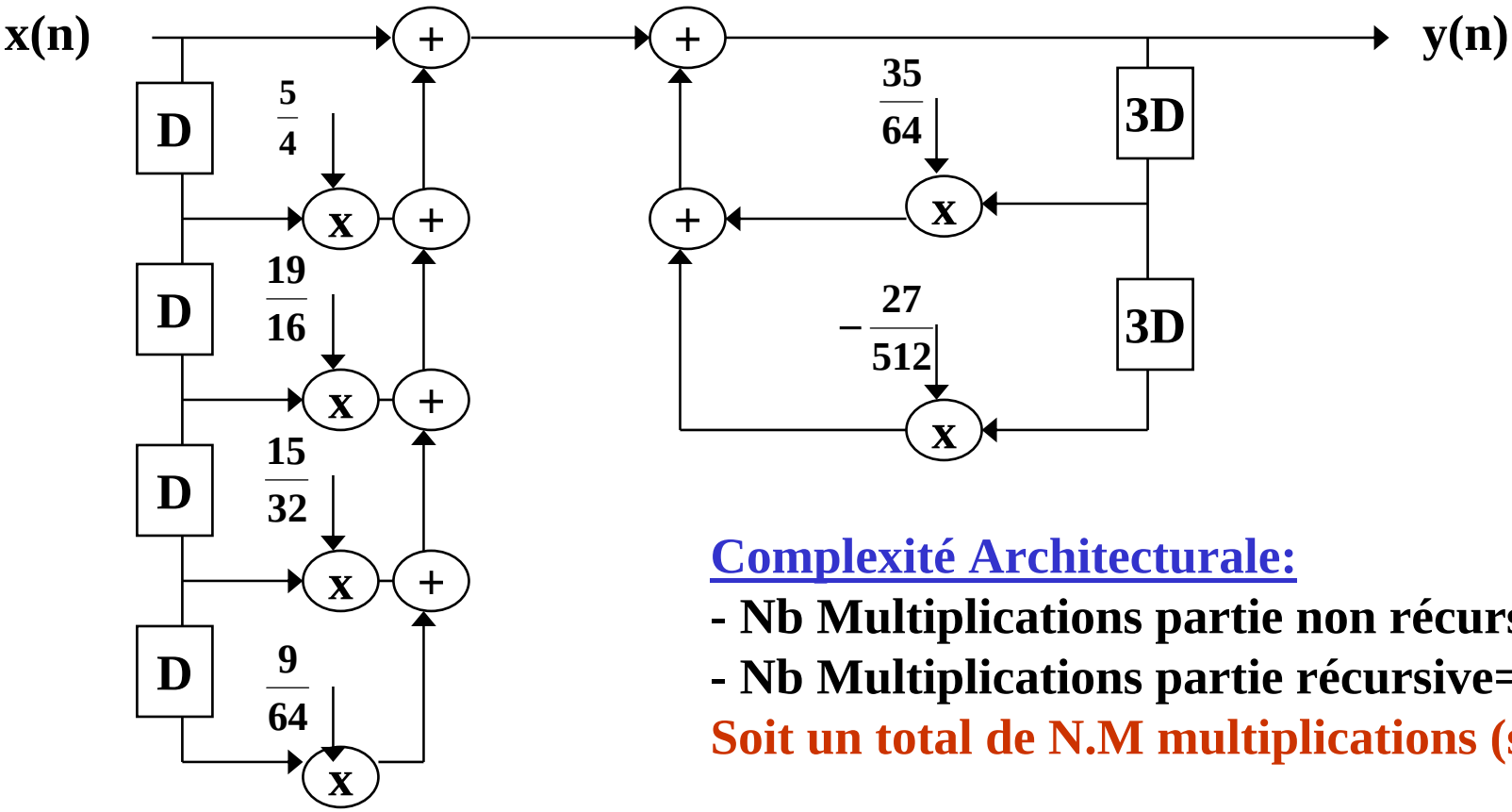
$$H(z) = \frac{1+(r_1+r_2)z^{-1}+(r_1^2+r_1r_2+r_2^2)z^{-2}+r_1r_2(r_1+r_2)z^{-3}+r_1^2r_2^2z^{-4}}{1-(r_1^3+r_2^3)z^{-3}+r_1^3r_2^3z^{-6}}$$

Application Numérique:

$$H(z) = \frac{1}{1 - \frac{5}{4}z^{-1} + \frac{3}{8}z^{-2}} \rightarrow \text{Pôles : } r_1 = \frac{1}{2} \text{ et } r_2 = \frac{3}{4}$$

(Ici N=2 et M=3)

$$H(z) = \frac{1 + \frac{5}{4}z^{-1} + \frac{19}{16}z^{-2} + \frac{15}{32}z^{-3} + \frac{9}{64}z^{-4}}{1 - \frac{35}{64}z^{-3} + \frac{27}{512}z^{-6}}$$



Complexité Architecturale:

- Nb Multiplications partie non récursive=N(M-1)
- Nb Multiplications partie récursive=N

Soit un total de N.M multiplications (soit N.M=6)