An overview of Digital Signal Processors

Study plan

- **Part 1 : Become familiar with DSPs : Lec. 4H (19/03) + presentation 2H (23/04)**
  - Lecture "**DSP Basics**"
  - **Presentations "What's inside a DSP component ?"** (2H) : 15 min by pair
- **Part 2 : How to program DSPs : Lab 6H (02/04 + 23/04) + Project 6H (30/04 au 30/05)**
  - Application
  - Target
  - Demo

Study plan

- **Part 1 : Become familiar with DSPs : Lec. 4H (19/03) + presentation 2H (23/04)**
  - Lecture "**DSP Basics**"
  - **Presentations** (2H) : 15 min by pair
- **Part 2 : How to program DSPs : Lab 6H (02/04 + 23/04) + Project 6H (30/04 au 30/05)**
  - Application
  - Target
  - Demo

Become familiar with DSP

DSP Basics

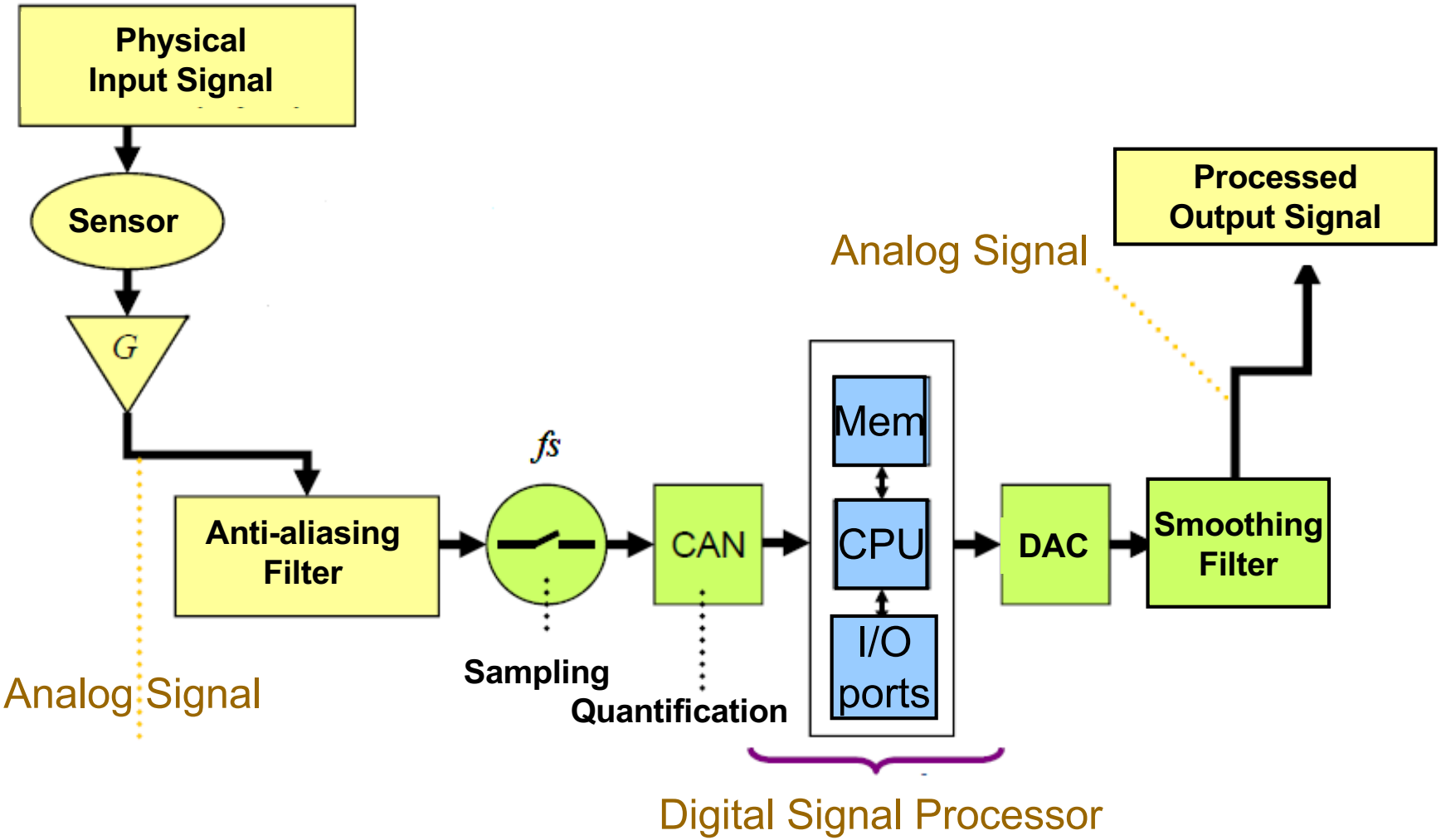
- Introduction to Digital Signal Processing (DSPg)
- Solutions to reduce processing time in Digital Signal Processors (DSP)
  - MAC
  - Circular buffers
  - Harward architecture
  - BSP, DMA
  - Pipeline
- Fixed- or Floating-Point precision ?
- Comparison between DSP and other solutions
  - Hardware solutions (GPP, µC, DSP, ASIC, ASSP, FPGA,GPU)

Become familiar with DSP

DSP Basics

- Introduction to Digital Signal Processing (DSPg)
- Solutions to reduce processing / sample time
  - MAC
  - Circular buffers
  - Harward architecture
  - BSP, DMA
  - Pipeline
- Fixed- or Floating-Point precision ?
- Comparison between DSP and other solutions
  - Hardware solutions (GPP, µC, DSP, ASIC, ASSP, FPGA,GPU)

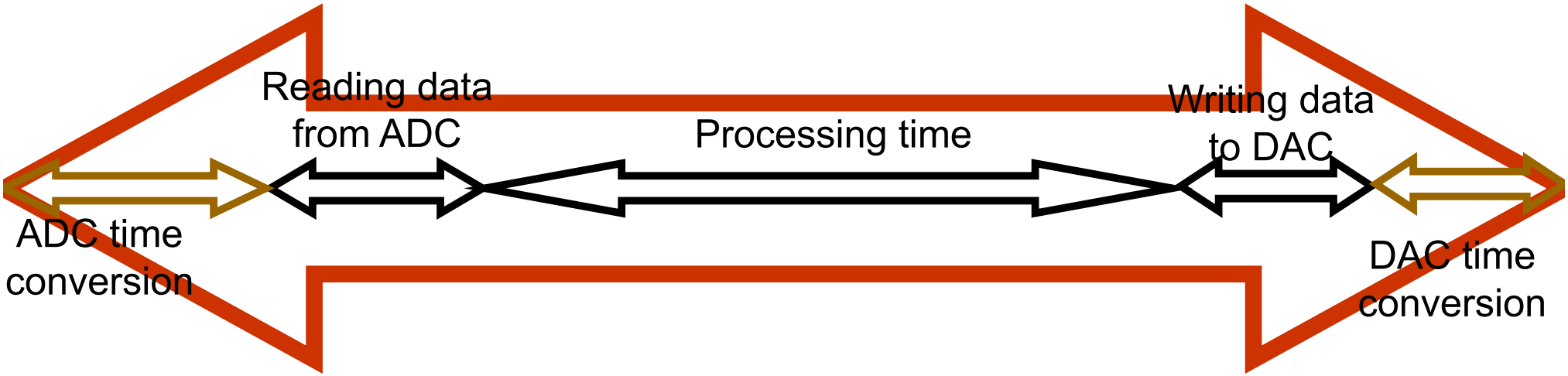
Signal acquisition and processing chain



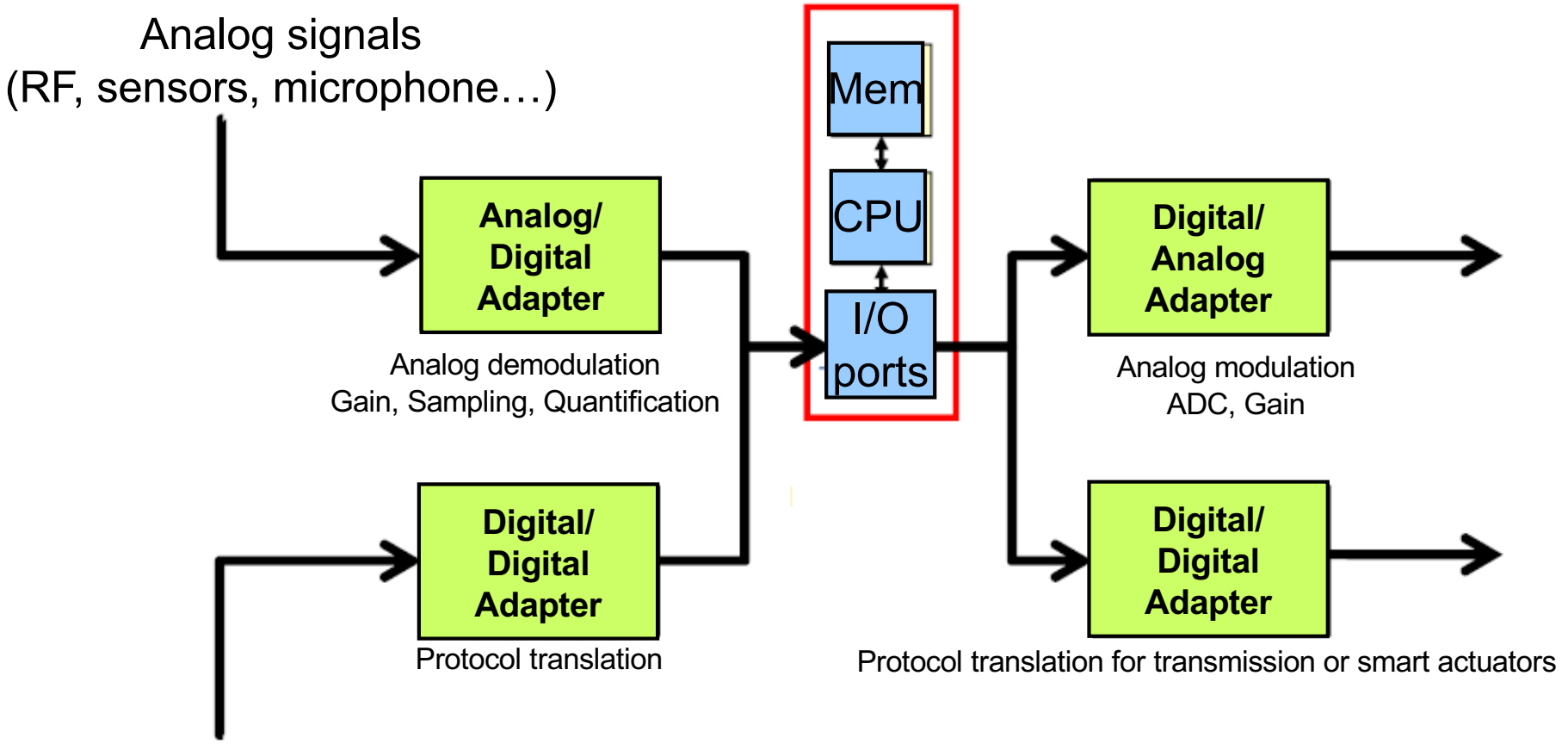
First DSPg constraint : Bandwidth

BWmax= Fs / 2 ; Nyquist frequency
How to increment BW ?

How to reduce sample time Ts ?



Multi Input, Multi Output processing chain



Other DSPg constraints

- **Real Time Processing (often) :** A defined time window is allowed between data occurrences and update of results.
- **High flow rate of data :** Sequential reception and transmission of a large amount of data.
- **Heavy processing load :**
  - Complex algorithms (FFT, Convolution, Matrix calculation…)
  - Definition of processing load = number of single operation per data * flow of data (amount of data per second)
- **Precision (often) :** Accuracy of the computation result

Classic operations in DSPg

- **MAC** (Multiply and Accumulate)
  Applications : Filters (convolution), Controlers, Correlation, Scalar Product, Matrix computation,…

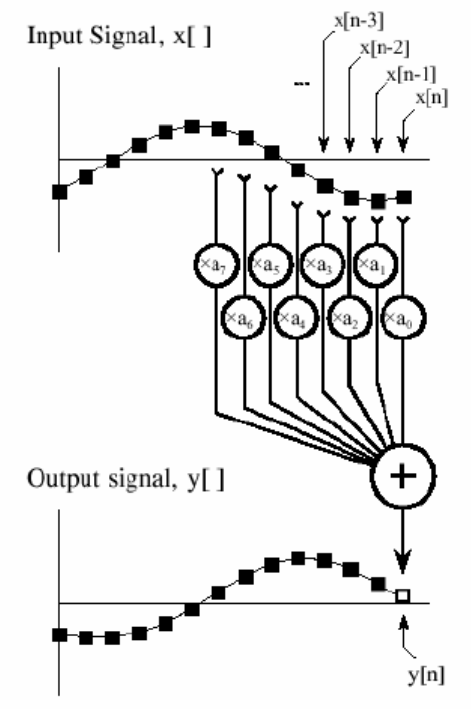
- **FFT** (Fast Fourrier Transform)
  Applications : Spectral analysis, Modulations, Sound processing and compression, etc…

- **DCT** (Discrete Cosinus Transform)
  Application : Image compression.

- **CSS** (Compare Select and Store)
  Application : Viterbi Algorithm (error correction codes)

Focus on MAC operation uses

MAC : Multipy - Accumulate



– $acc \leftarrow acc + b_i \,.\, x_i$

Finite Impulse-Response (FIR) filters

$$y(n) = \sum_{i=0}^{P-1} b(i)x(n-i)$$

Infinite Impulse-Response (IIR) filters and controlers

$$y(n) = \sum_{i=0} b(i)x(n-i) - \sum_{j=1} a(j)y(n-j)$$

Examples of MAC operation use

MAC : Multipy - Accumulate

- Correlation de deux signaux de longueur N

  On peut calculer la corrélation pour $n \in [0, 2N-1]$

$$y(n) = \sum_{k=0}^{N-1} x_1(k).x_2(k-n)$$

- Produit scalaire deux signaux de longueur N

$$y(n) = \sum_{k=0}^{N-1} x_1(k).x_2(k)$$

Examples of MAC operation use

MAC : Multipy - Accumulate

- Multiplication d'une matrice (a_{ij}) de type (m,n) et d'une matrice de type (b_{ij}) (n, p), donnant une matrice (c_{ij}) (m, p)

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj}$$

Become familiar with DSP

DSP Basics

- Introduction to Digital Signal Processing
- Solutions to reduce processing / sample time
  - MAC
  - Circular buffers
  - Harward architecture
  - BSP, DMA
  - Pipeline
- Fixed- or Floating-Point precision ?
- Comparison between DSP and other solutions
  - Hardware solutions (GPP, µC, DSP, ASIC, ASSP, FPGA,GPU)

Real Time Processors

A **real-time processor** goes beyond traditional processors maintaining the critical properly deterministic operation i.e. **each instruction or library function have a precise number of clock cycles**.

Example :

Table 3-1. TMU Supported Instructions Summary

Operation	C Equivalent Operation	C28x Pipeline Cycles
Multiply by 2*pi	a = b * 2pi	2 cycles + Sine/Cosine function
Divide by 2*pi	a = b / 2pi	2 cycles + Sine/Cosine function
Divide	a = b / c	5 cycles
Square Root	a = sqrt(b)	5 cycles
Sin Per Unit	a = sin(b*2pi)	4 cycles
Cos Per Unit	a = cos(b*2pi)	4 cycles
Arc Tangent Per Unit	a = atan(b)/2pi	4 cycles
Arc Tangent 2 and Quadrant Operation	Operation to assist in calculating ATANPU2	5 cycles

Hardware multiplier

perform high speed multiplication-addition
= **in one clock step MAC (Float MAC or Integer MAC)**
("MAC" instruction = Multiply and Accumulate)
>> Hardware multiplier implemented in silicon

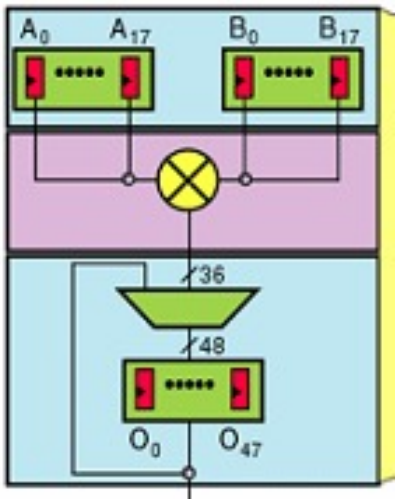


Software multiplyer

```
    1 0 1 1
  x 1 1 1 0
  ---------
    0 0 0 0     Cycle 1
  1 0 1 1 0     Cycle 2
1 0 1 1 0 0     Cycle 3
1 0 1 1 0 0 0   Cycle 4
  ---------
1 0 0 1 1 0 1 0 Cycle 5
```

Hardware multiplyer

```
    1 0 1 1
  x 1 1 1 0
  ---------
1 0 0 1 1 0 1 0
```



Hardware accelerators

Accelerators are intended to help the CPU perform certain tasks more efficiently by extending the capabilities of a CPU through registers and instructions.

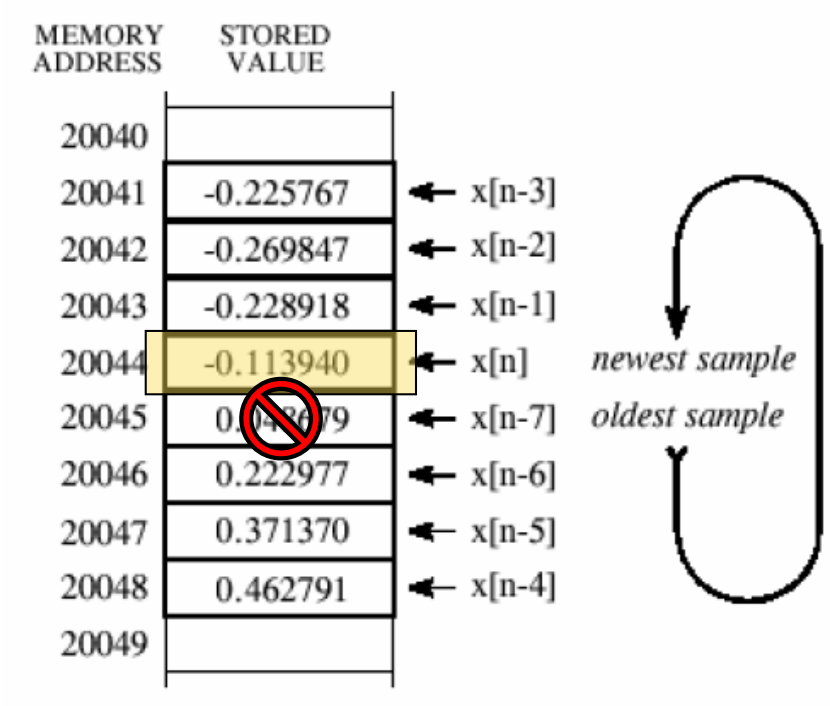
- **Trigonometric Math Unit:** Accelerates several specific trigonometric math operations like sine, cosine, arc tangent, divide, and square root.
- **Cryptographic Unit:** Accelerates encryption algorithms such as the data encryption standard (DES) symmetric encryption algorithm and the advanced encryption system (AES) symmetric encryption algorithm.
- **Floating-Point Unit:** Provides floating-point math support which alleviates scaling and saturation concerns.
- **Complex Math Unit:** Accelerates math operations like add, subtract, and multiply.

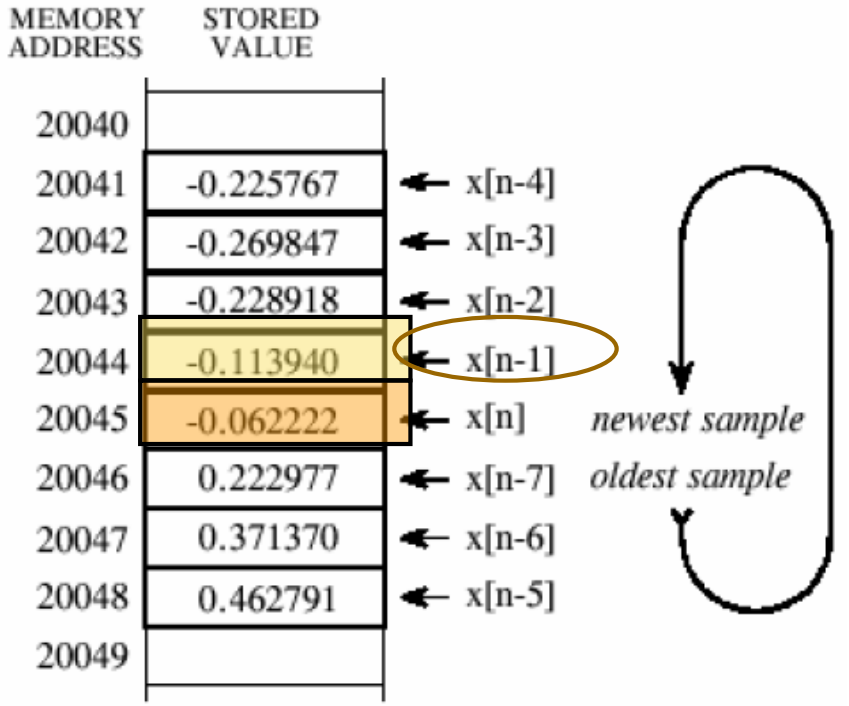
Real Time Processors : other caracteristics

Caractéristic	Description
Fast at trigonometric operations	SIN, COS, ATAN, DIV
Fast at saturation operations	Saturation operations check for out of bounds conditions and if detected the value is clamped or saturated
Fast Interrupt Response	The processor needs to respond to periodic interrupt events with low latency to satisfy real time deadlines
Cache memory	Cache is a faster memory which stores copies of the content from frequently used main memory locations. There can be unpredictable delays when content from cache is refreshed.

Circular buffering in cache memory

Circular Buffering vs Shift buffering

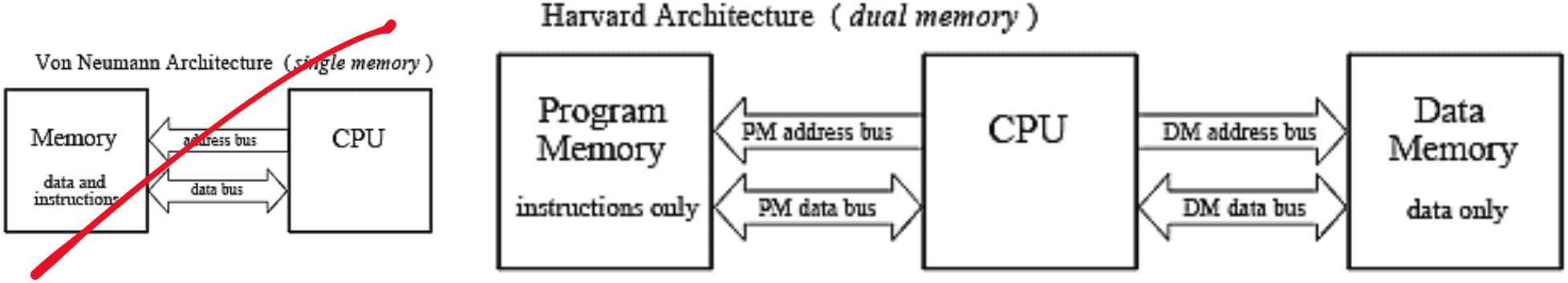




Harvard architecture : parallelized architecture

Fast access to data in write/read modes.

>> Simultaneous access for data and program instructions (**Harvard architecture**)



Howard Aiken – Harvard University - 1940

More parallelized architectures

The kind of instruction set implemented on the DSP can improve speed.

Flynn's taxonomy (1966)

- Single Instruction, Single Data stream (SISD)
  - A sequential computer which exploits no parallelism in either the instruction or data streams.
- Single Instruction, Multiple Data streams (SIMD)
  - A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized.
- Multiple Instruction, Single Data stream (MISD)
  - Multiple instructions operate on a single data stream. Uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result. Examples include the Space Shuttle flight control computer.
- Multiple Instruction, Multiple Data streams (MIMD)
  - Multiple autonomous processors of the simultaneously executing different instructions on different data.

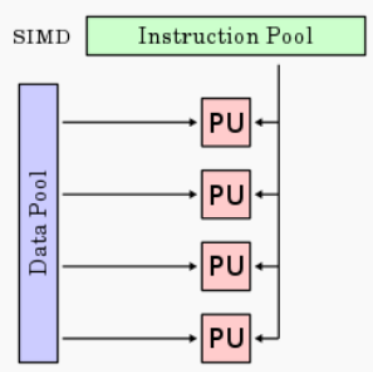
>> **SIMD** Single Instruction on Multiple Data

>> **MIMD** Multiple Instruction on Multiple Data

Multiple access to data

The same instruction is applied simultaneously to several data

>> **SIMD** Single Instruction on Multiple Data



Example: multiplication

MPYDP — *Double-Precision Floating-Point Multiply*

Syntax

MPYDP (.unit) *src1*, *src2*, *dst*

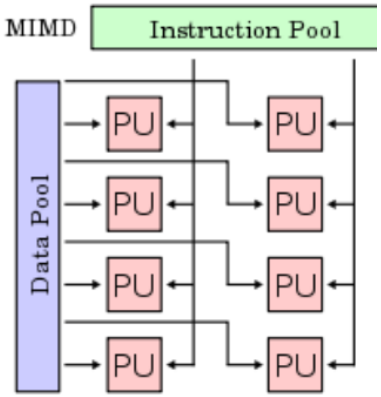
.unit = .M1 or .M2

Opcode map field used...	For operand type...	Unit
src1	dp	.M1, .M2
src2	dp	
dst	dp	

Multiple access to instructions

Multiple instructions are applied simultaneously to several data

>> **MIMD** Multiple Instruction on Multiple Data



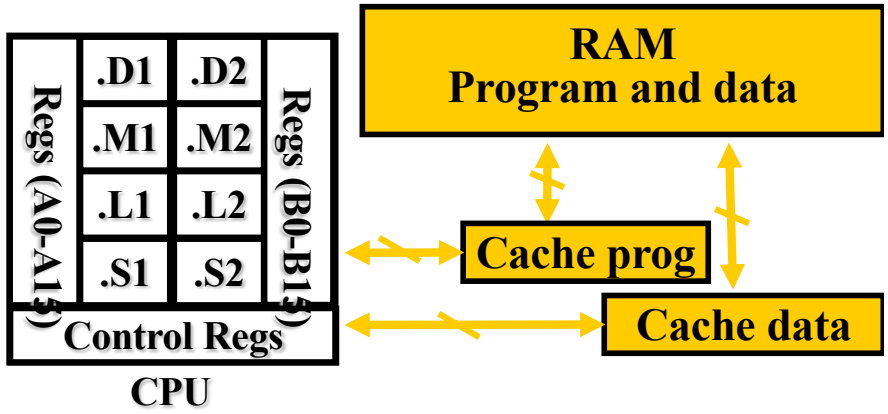
Example: VLIM

Very long instruction word :

• **Pipeline** architecture = parallel use of ALU on multiple instructions

• May be used on **multipath** or **multicore** processors

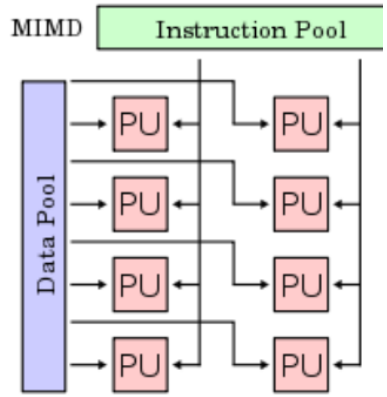
• Optimization of the parallelism by the **compiler** using an **out of order** execution of the instructions



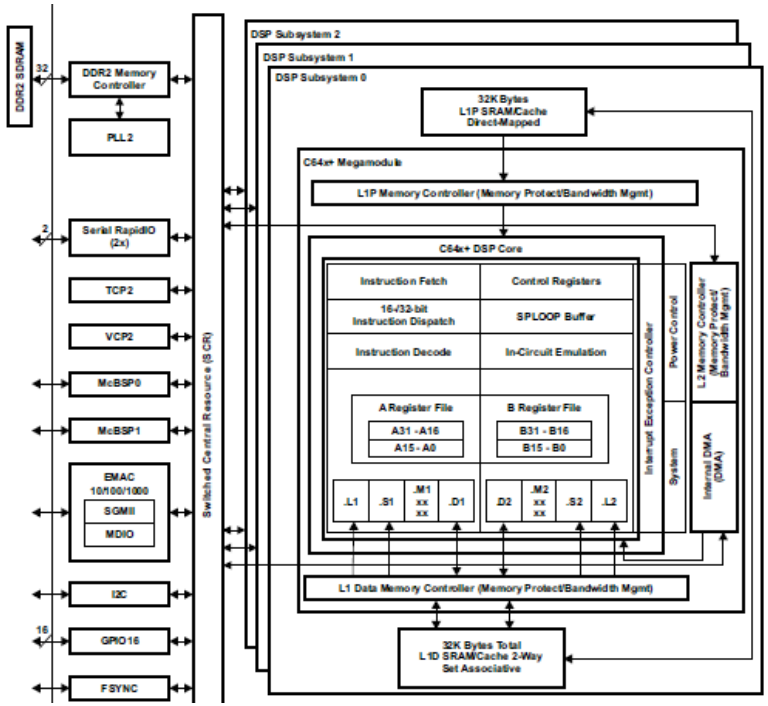
Multiple access to instructions

Multiple instructions are applied simultaneously to several data

>> **MIMD** Multiple Instruction on Multiple Data



TMS 320C6474



Example: VLIM

Very long instruction word

• **Pipeline** architecture = parallel use of ALU on multiple instructions

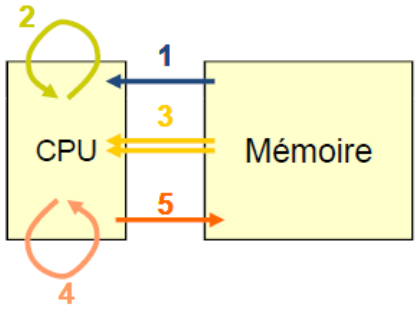
• May be used on **multipath** or **multicore** processors

• Optimization of the parallelism by the **compiler** using an **out of order** execution of the instructions

Parallelism inside the CPU

Pipeline: parallel execution of instructions

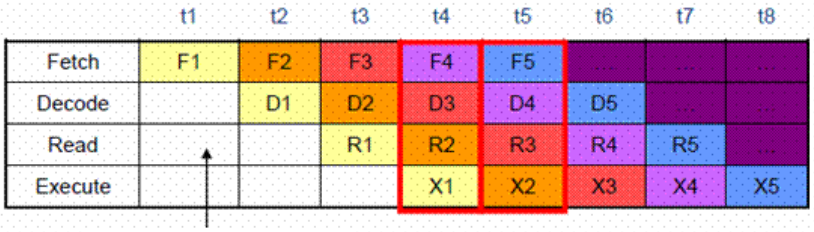
1. Reading the instruction code in program memory (***Fetch***) ;
2. Decoding the instruction code and the data addresses (***Decode***) ;
3. Reading data in datas memory (***Read***) ;
4. Processing the instruction (***Execute***);
5. writing the result (***Write***).



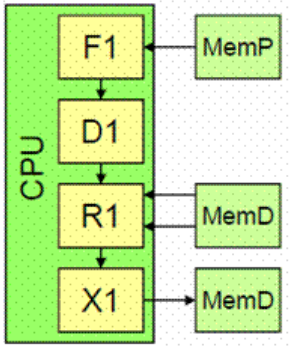
In this example steps 4 and 5 are grouped

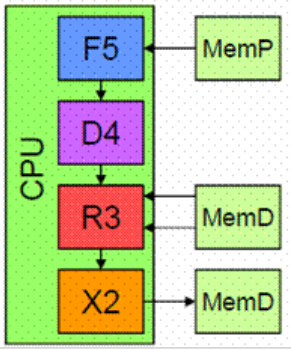
	t1	t2	t3	t4	t5	t6	t7	t8
Instruction	Instruction 1				Instruction 2			
Fetch	F1				F2			
Decode		D1				D2		
Read			R1				R2	
Execute				X1				X2

Sequential execution



Execution with pipeline : instruction interlacing

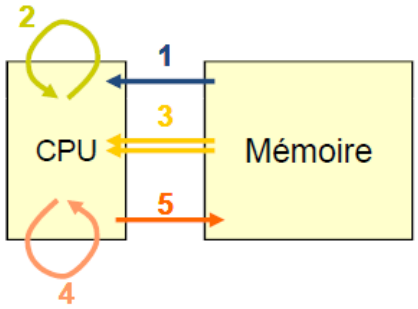




Parallelism inside the CPU

Pipeline: parallel execution of instructions

1. Reading the instruction code in program memory (***Fetch***) ;
2. Decoding the instruction code and the data addresses (***Decode***) ;
3. Reading data in datas memory (***Read***) ;
4. Processing the instruction (***Execute***);
5. writing the result (***Write***).



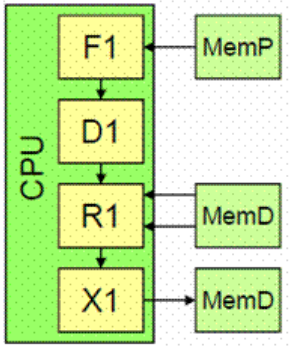
In this example steps 4 and 5 are grouped

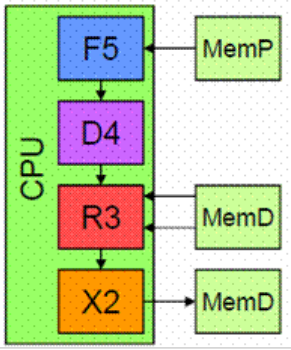
	t1	t2	t3	t4	t5	t6	t7	t8
Instruction	Instruction 1				Instruction 2			
Fetch	F1				F2			
Decode		D1				D2		
Read			R1				R2	
Execute				X1				X2

Sequential execution

	t1	t2	t3	t4	t5	t6	t7	t8
Fetch	F1	F2	F3	F4	F5			
Decode		D1	D2	D3	D4	D5		
Read			R1	R2	R3	R4	R5	
Execute				X1	X2	X3	X4	X5

Execution with pipeline : instruction interlacing





Paralelised communication with I/O and A/D

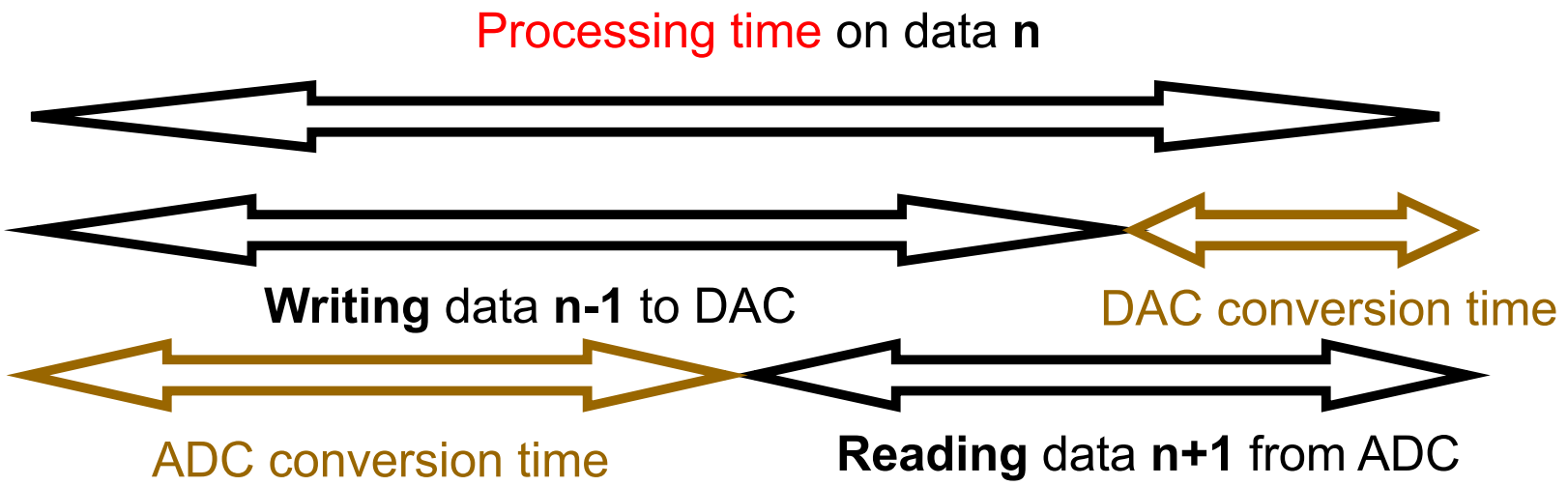
Parallel access to inputs/outputs during processing time

>> automation of the data access (read/write) using specific peripherals

On chip communication protocols

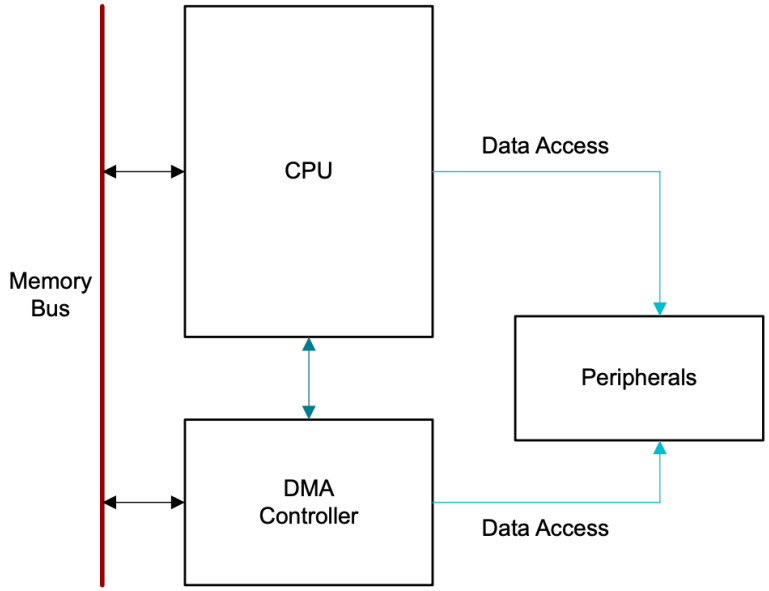
DMA : Direct Memory Acces

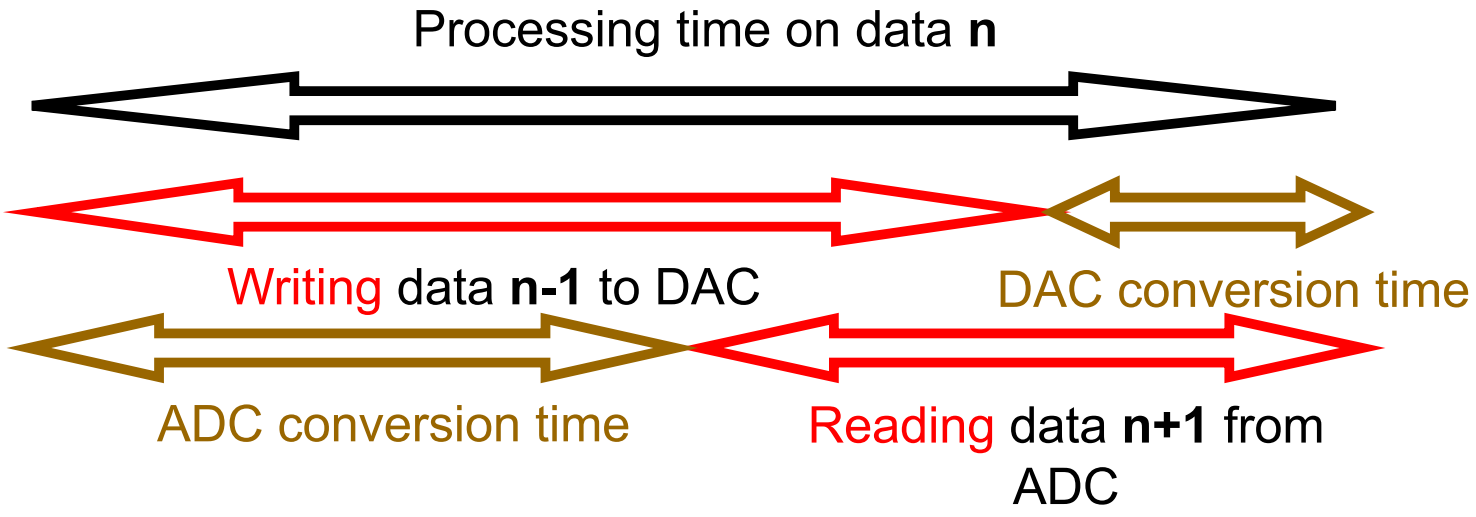
>> **Think new algorithms: calculation on data blocks**



Direct Memory Access (DMA)

DMA is a feature that allows subsystems or peripherals to access RAM without intervention of the CPU. This is a helpful feature for real-time control applications since it allows the CPU to work on other tasks of the control algorithm, while the DMA transfers data between address locations (both memory and register).

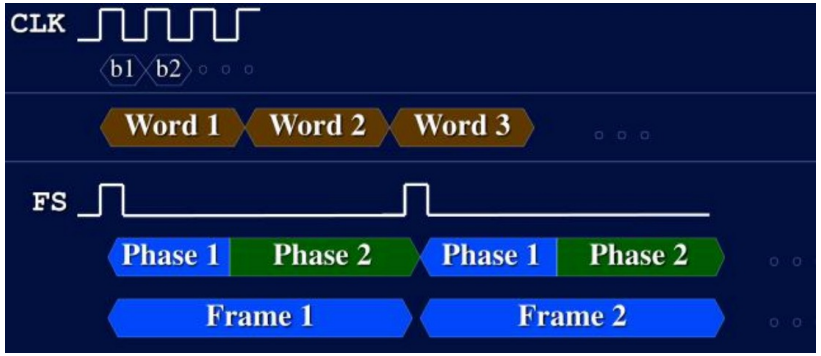




On chip interfaces

Dedicated On-Chip interfaces and accelerators

-Process control : PWM, Timers, ADCs, DACs, BSP, SPI, I2C, LAN

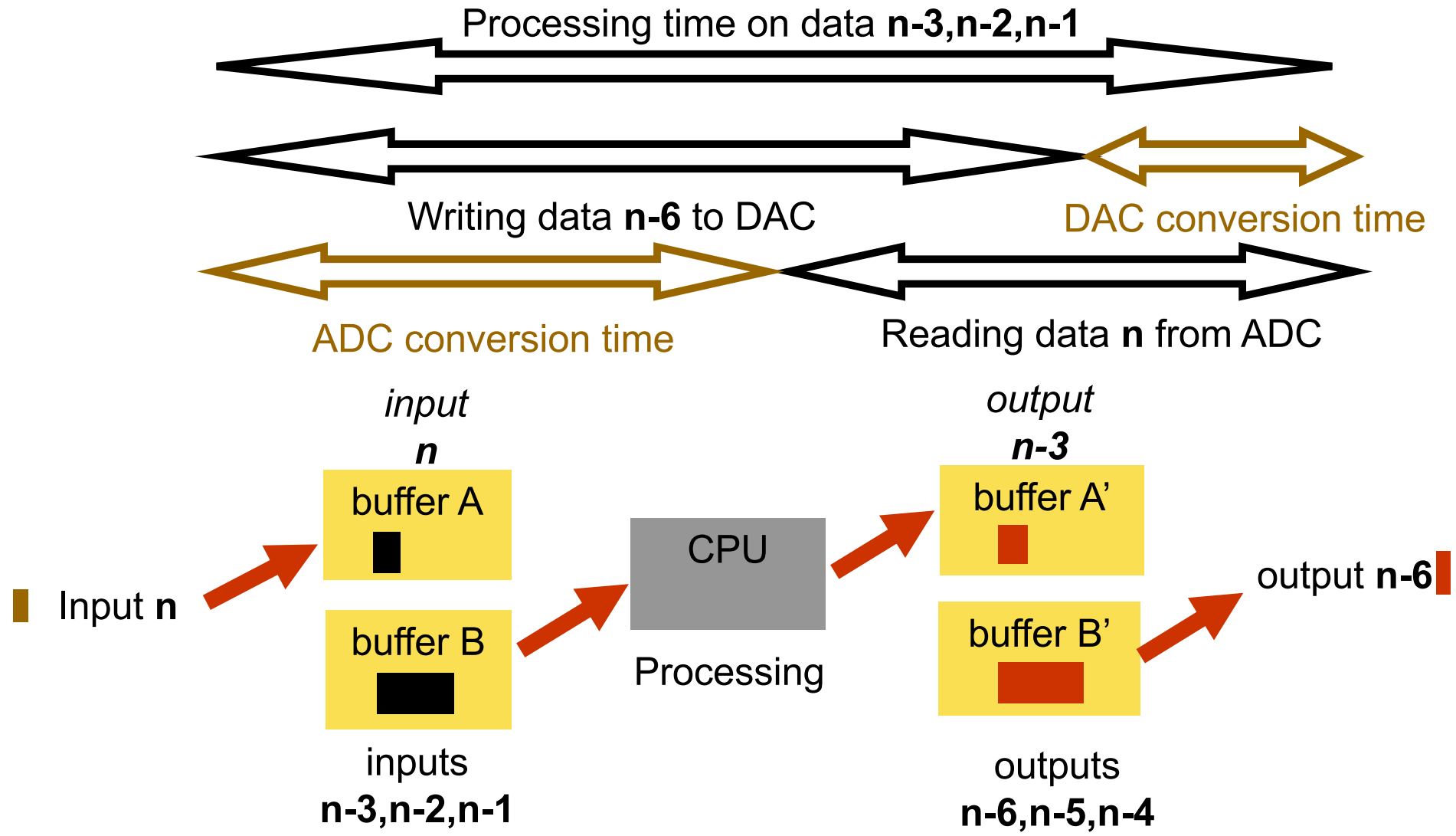


-Communications : Protocole Encoder/Decoder, MAC layers, Error detector/corrector…

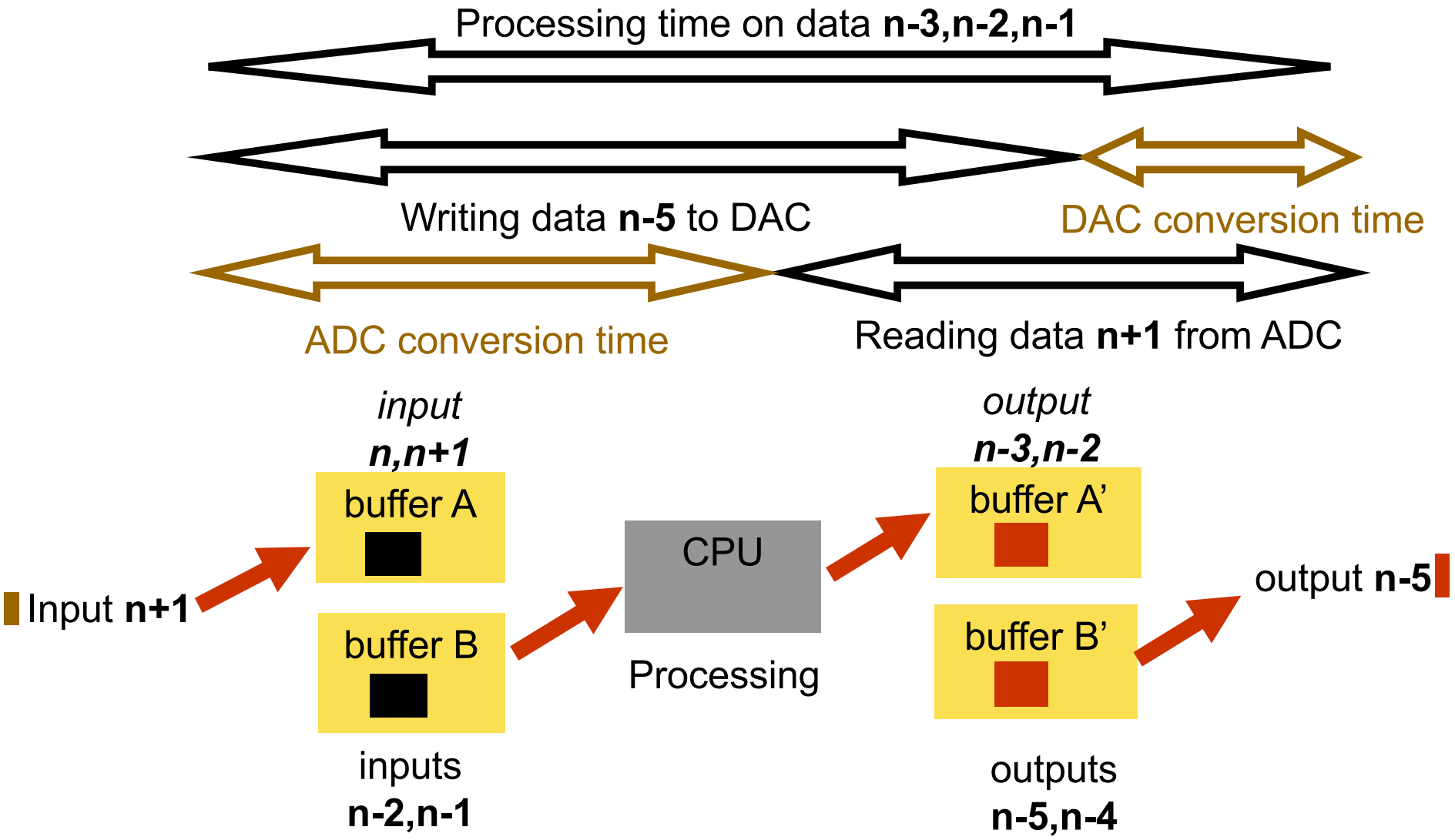
-Audio Processing : Audio serial port (I2S, SPDIF,…), Decoders (DTS, Dolby,…)

-Video/Image Processing : Hardware Accelerators (Resizer, OSD,…), Hardware Coprocessor (MPEG, H261,…), Video port input and output (YUV, RGB, HDTV output…)

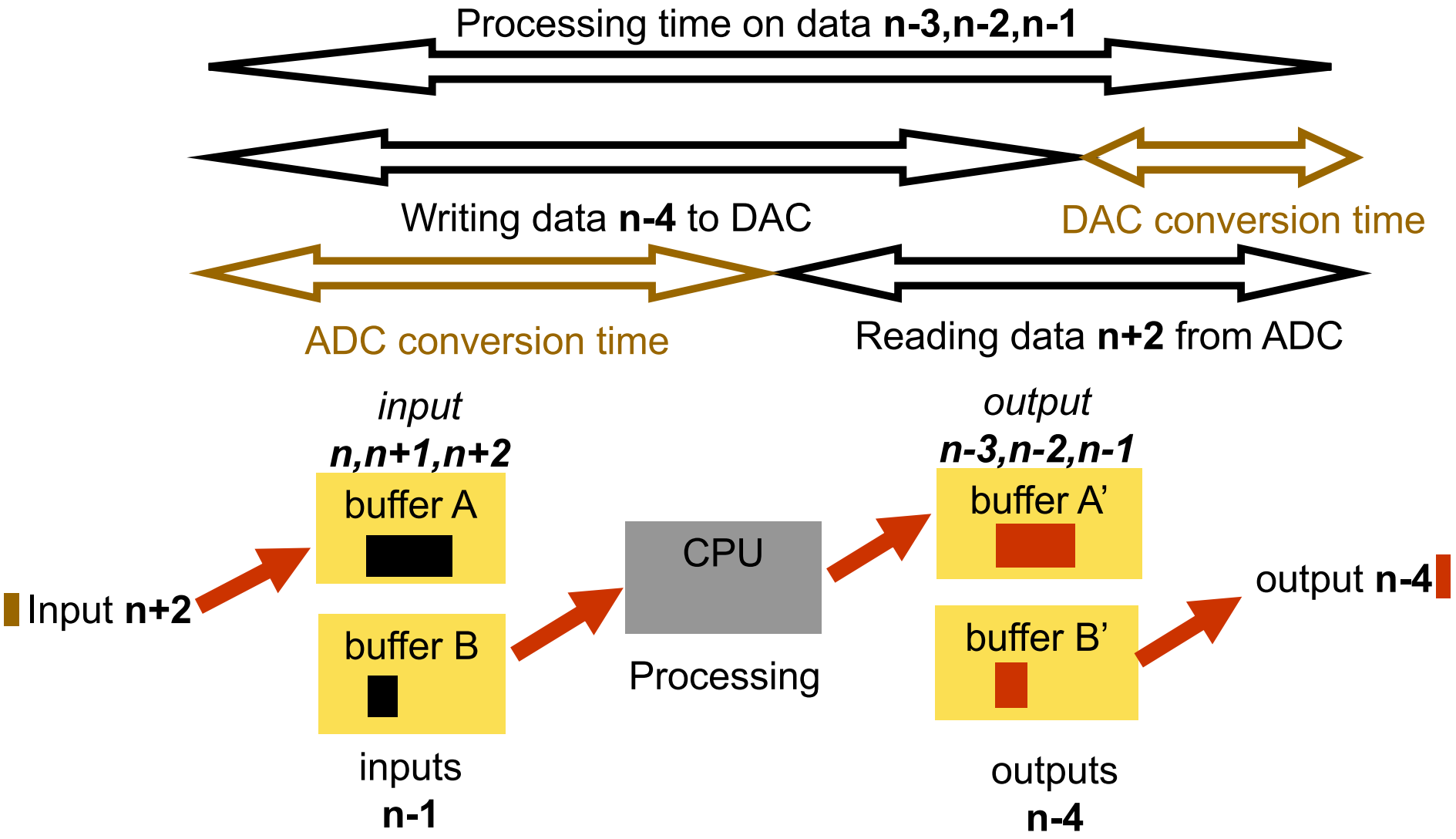
Ping-Pong buffering example



Ping-Pong buffering example



Ping-Pong buffering example

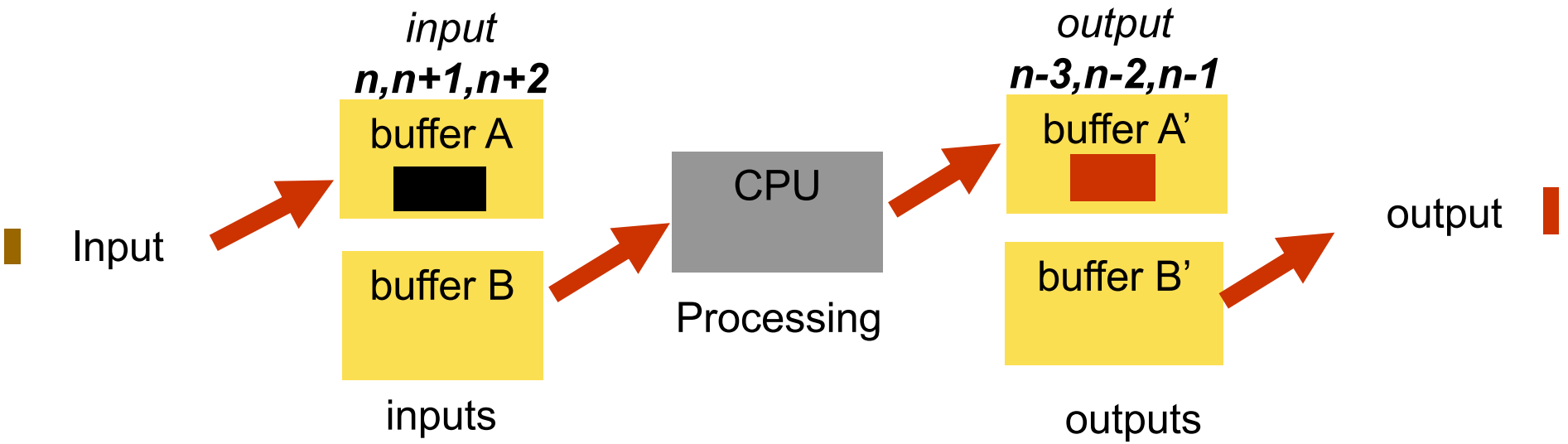


Ping-Pong buffering example

The **"ping-pong" structure** consists on switching
- between the inputs buffers (A and B)
- between the output buffers

in order to ensure a continuous data flow.

The switching is monitored by the **DMA**, allowing the CPU to use all time for processing

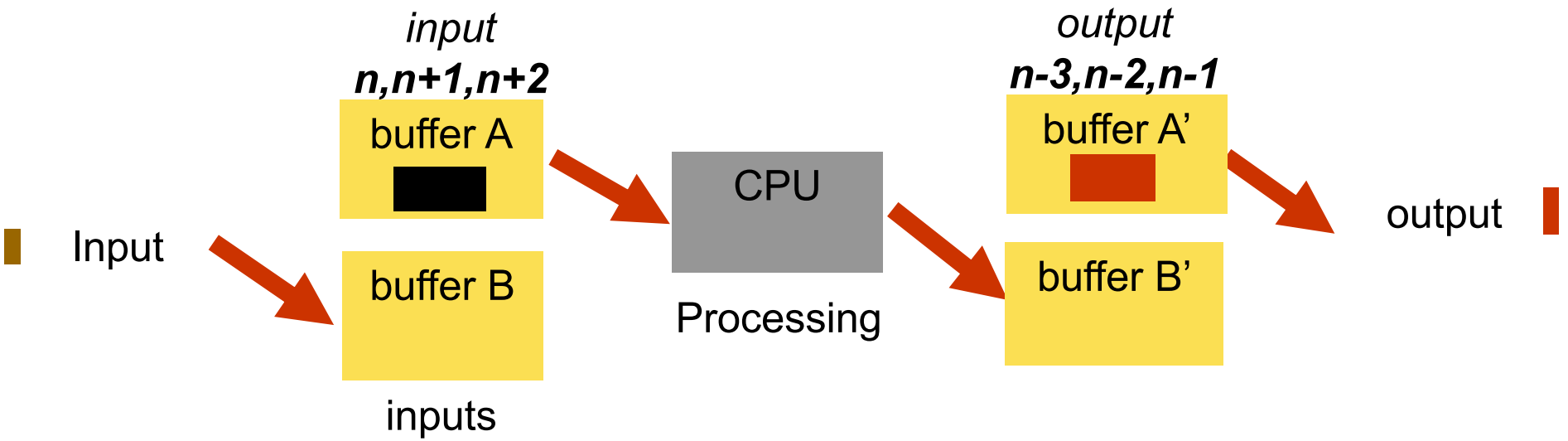


Ping-Pong buffering example

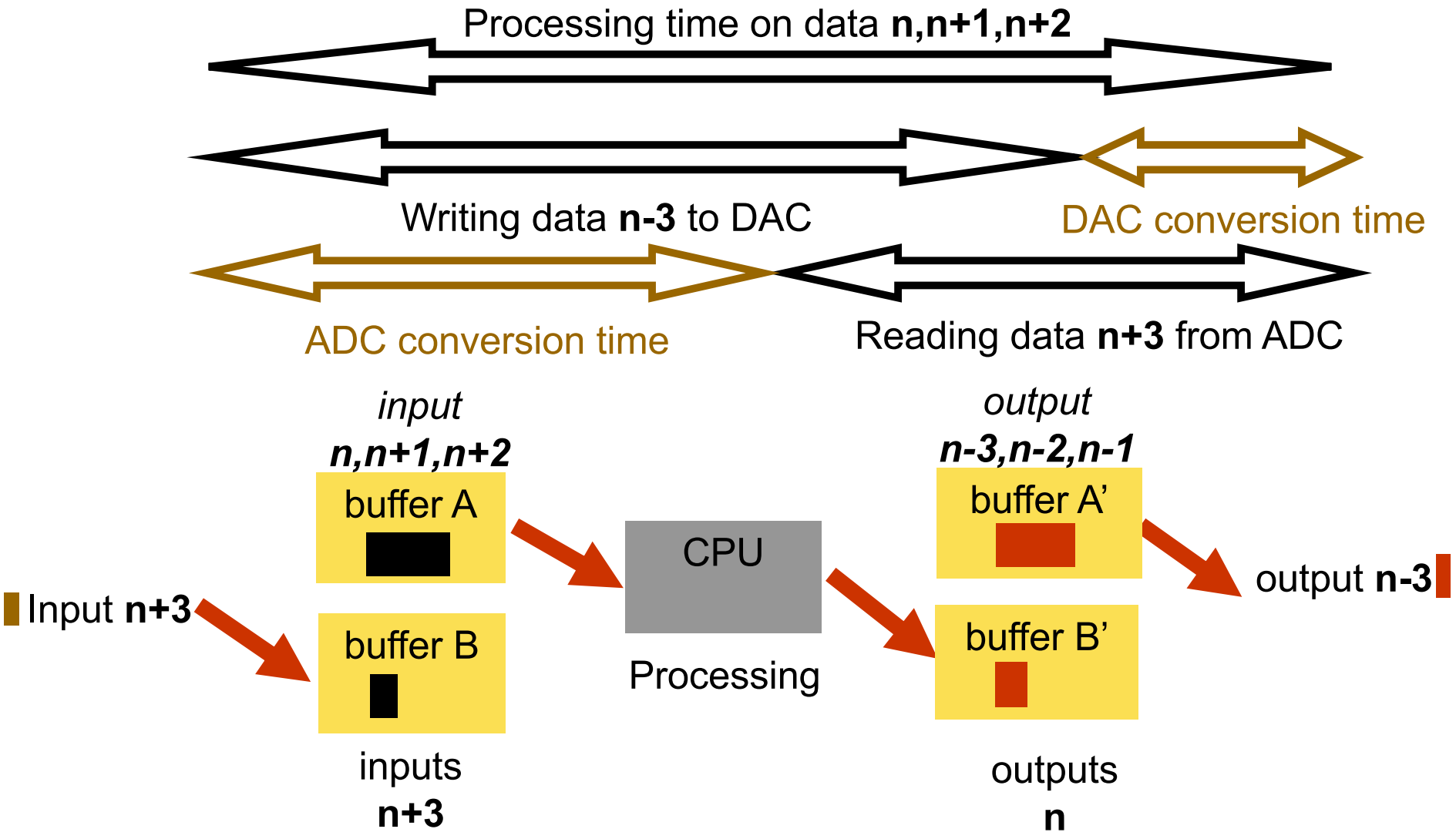
The **"ping-pong" structure** consists on switching
- between the inputs buffers (A and B)
- between the output buffers (A' and B')

in order to ensure a continuous data flow.

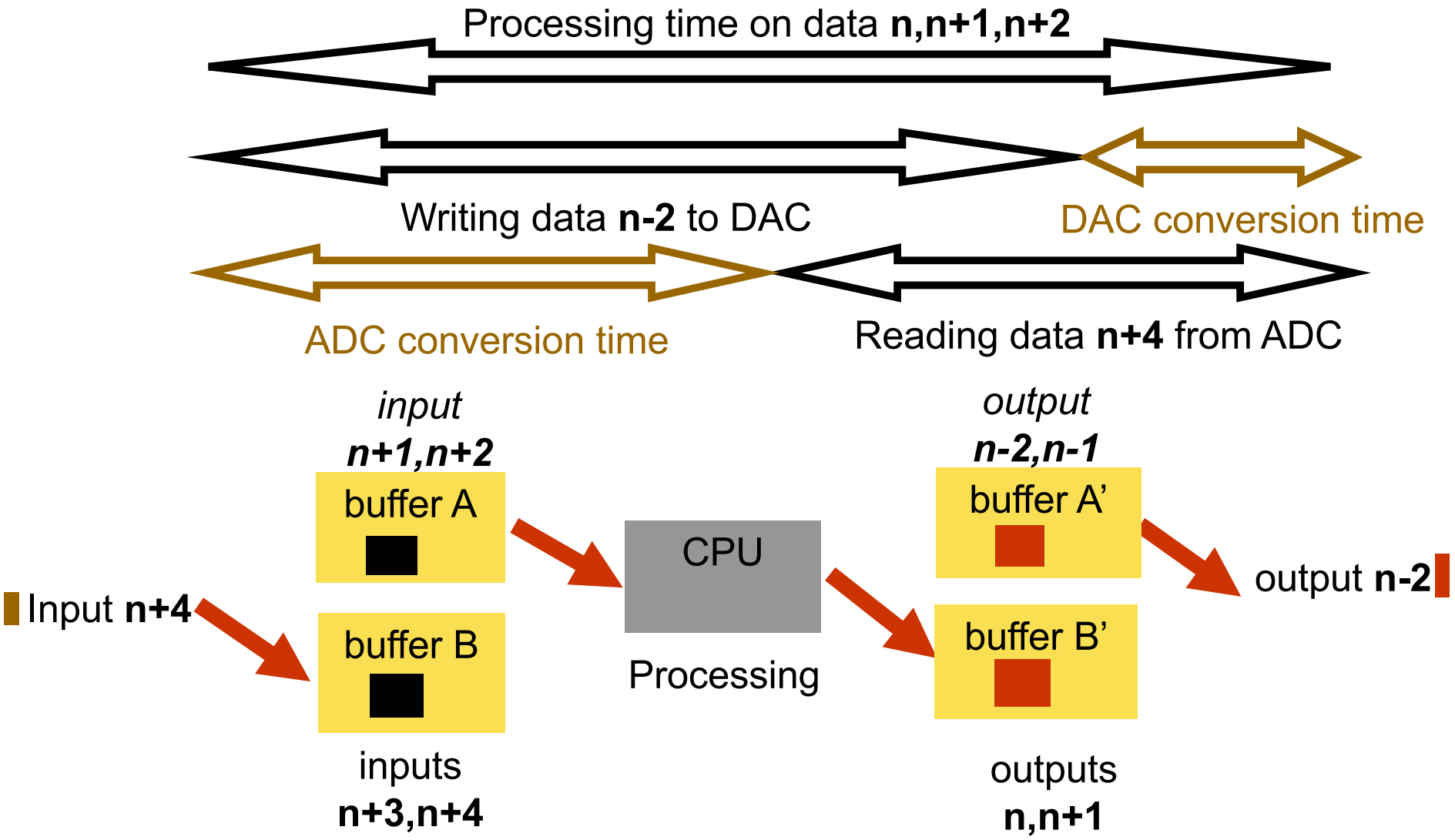
The switching is monitored by the **DMA**, allowing the CPU to use all time for processing



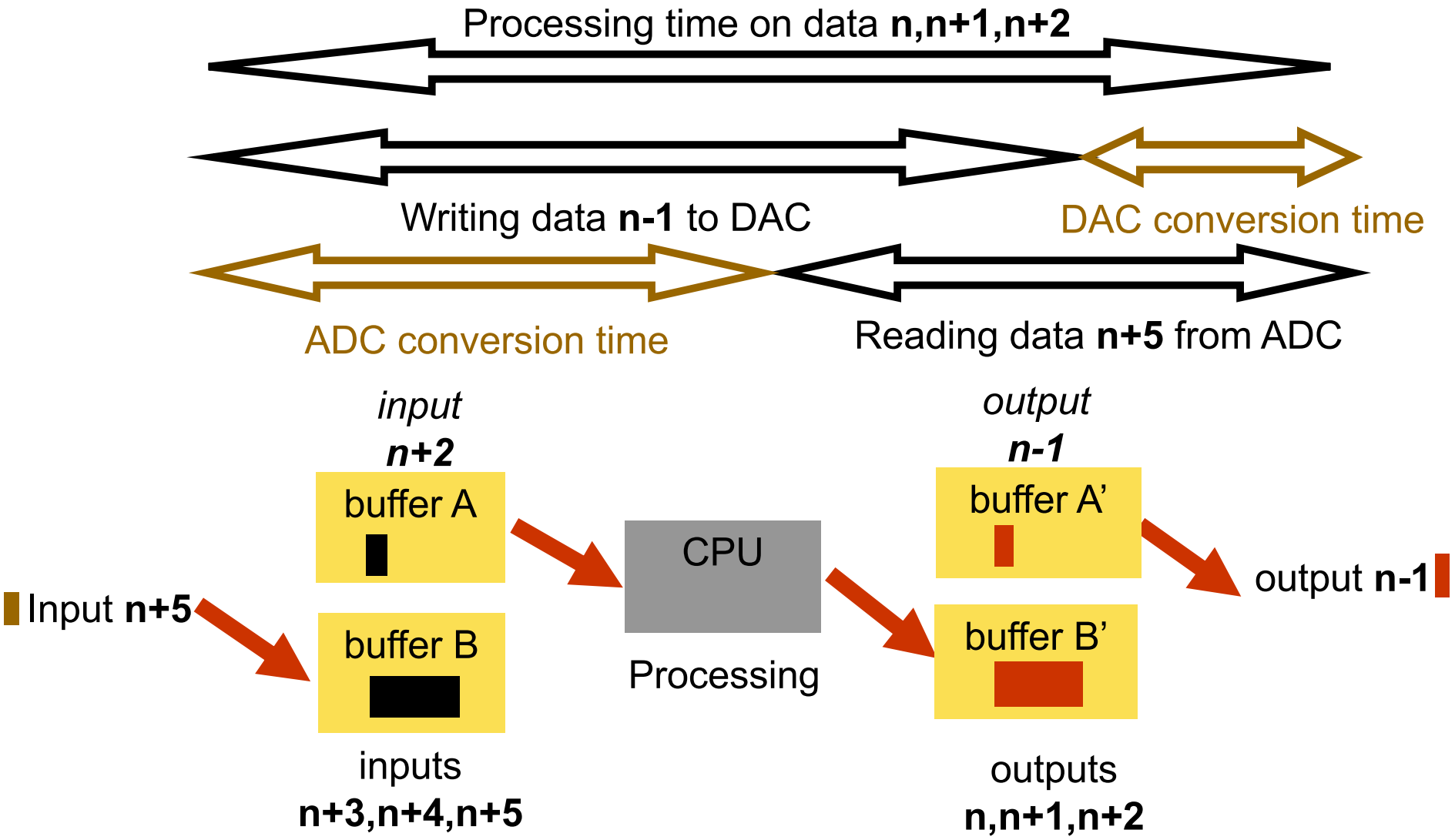
Ping-Pong buffering example



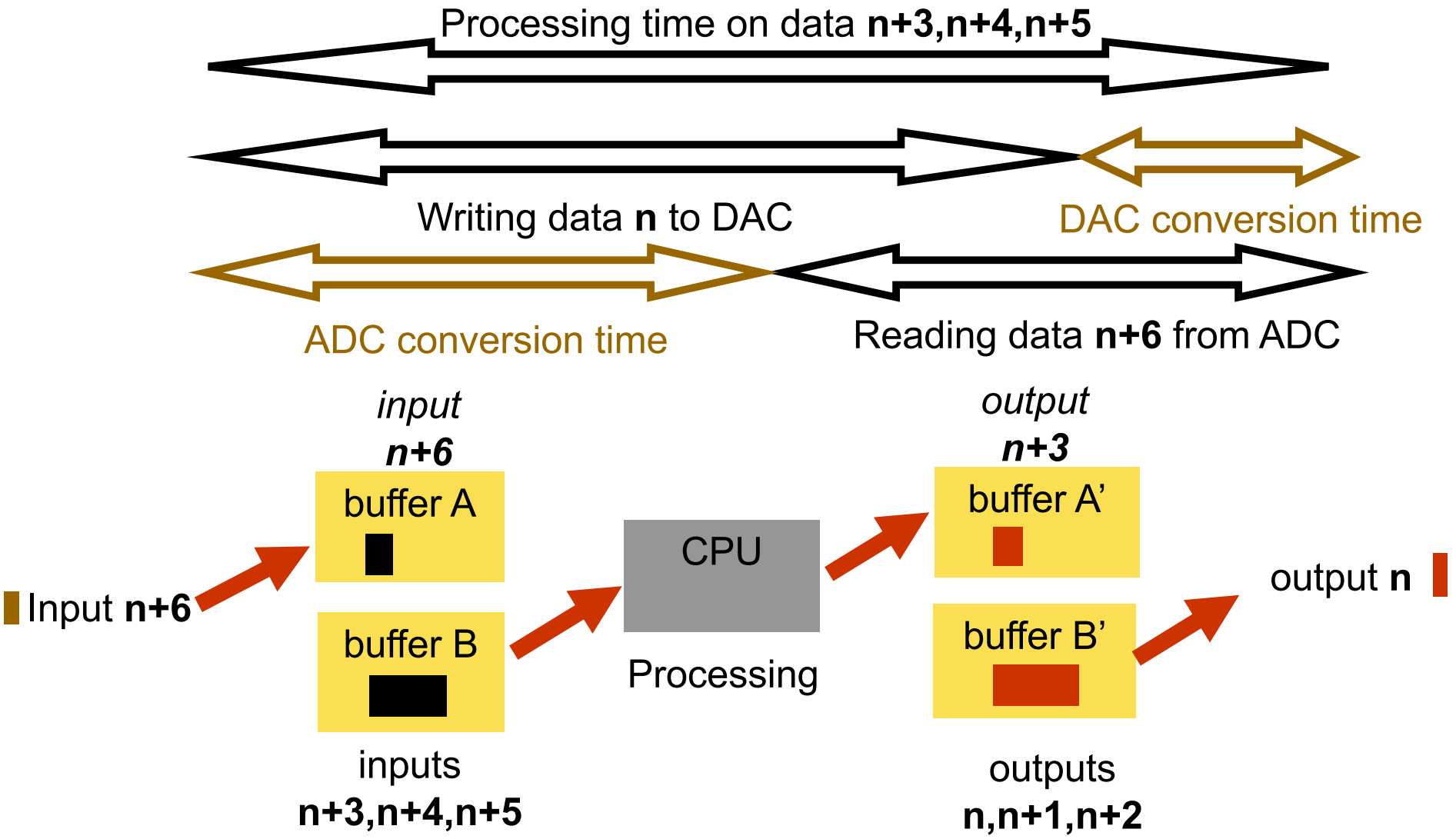
Ping-Pong buffering example



Ping-Pong buffering example



Ping-Pong buffering example

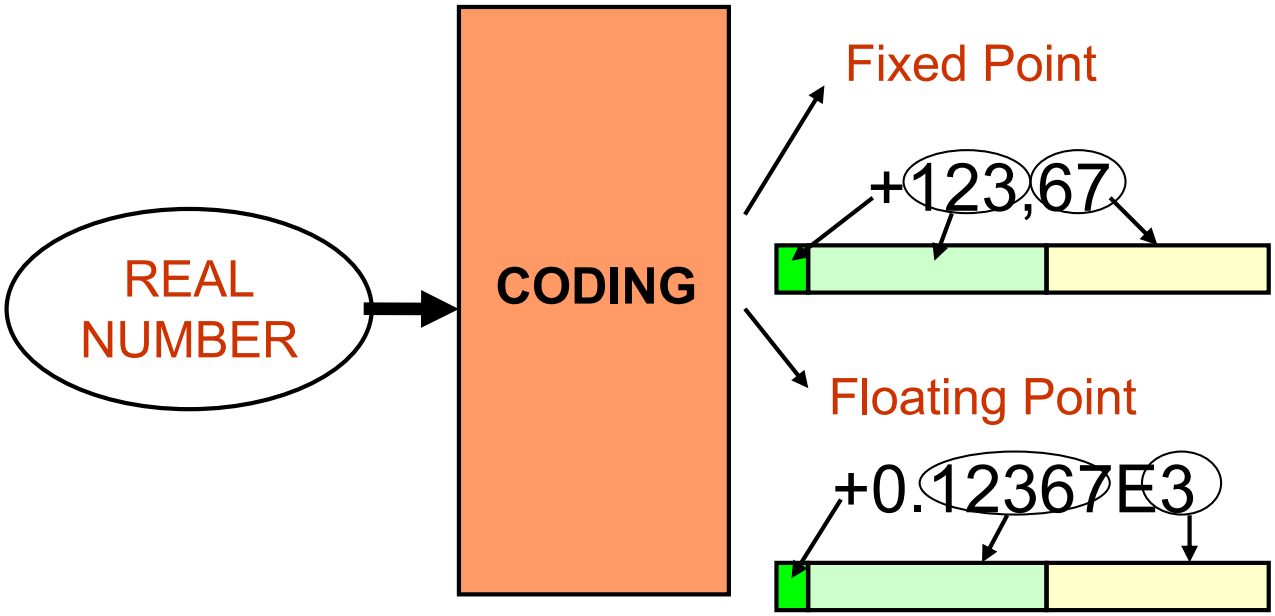


Become familiar with DSP

DSP Basics

- Introduction to Digital Signal Processing ?
- Solutions to reduce processing / sample time
  - MAC
  - Circular buffers
  - Harvard architecture
  - BSP, DMA
  - Pipeline
- Fixed- or Floating-Point precision ?
- Comparison between DSP and other solutions
  - Hardware solutions (GPP, µC, DSP, ASIC, ASSP, FPGA,GPU)

representation "IEEE 754"



Some definitions

- Resolution (quantum, quantification step) : higher change of the coded value that doesn't change the coding

- Definition interval: maximum and minimum values that can be coded

[Min(x), Max(x)]

- Dynamics : ratio (in dB) of the Max to the Min of the absolute values

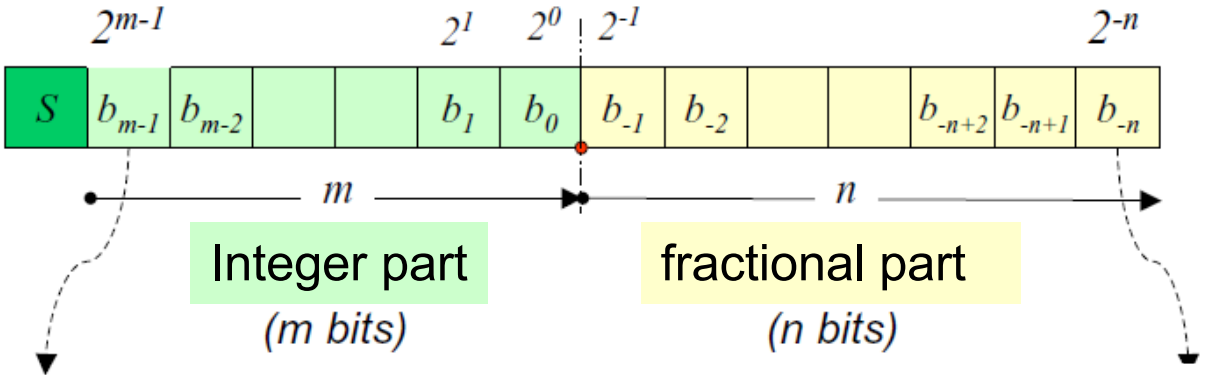
$$Dyn_{db} = 20.\log_{10}\left(\frac{Max(|x|)}{Min(|x|)}\right)$$

- Signal to Noise Ratio : ratio (in dB) of the Signal Power to the resolution = depends of the RESOLUTION q

$$SNR_{db} = 10.\log_{10}\left(\frac{P_{signal}}{(q^2/12)}\right) = P_{signal\text{-}dB} + 10.7 - 20.\log_{10}(q)$$

Signed Fixed Point representation "IEEE 754"

Qm.n



Definition Interval

$$D = [-2^m, 2^m - 2^{-n}]$$

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i 2^i$$

Resolution (quantification step)

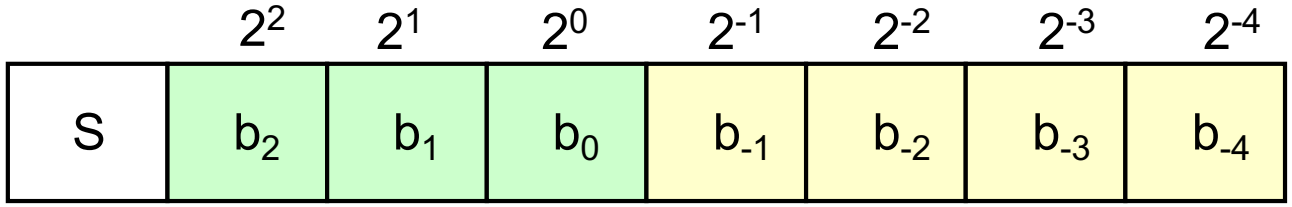
$$q = 2^{-n}$$

$$Dyn_{dB} = 20 .\log \left(\frac{2^m}{2^{-n}}\right) = 20\ (m+n) .\log 2 = 6.(m+n)$$

$$SNR_{dB} = Ps_{dB} - 10.\log(2^{-2n}/12) = Ps_{dB} + 10.7 + 6.n$$

Signed Fixed Point representation "IEEE 754"

Q3.4



$1000.0000 = -2^3 = -8$
$0111.1111 = 7.9375 = 2^3 - 2^{-4}$

Definition Interval

$D = [-8 , 7.9375]$

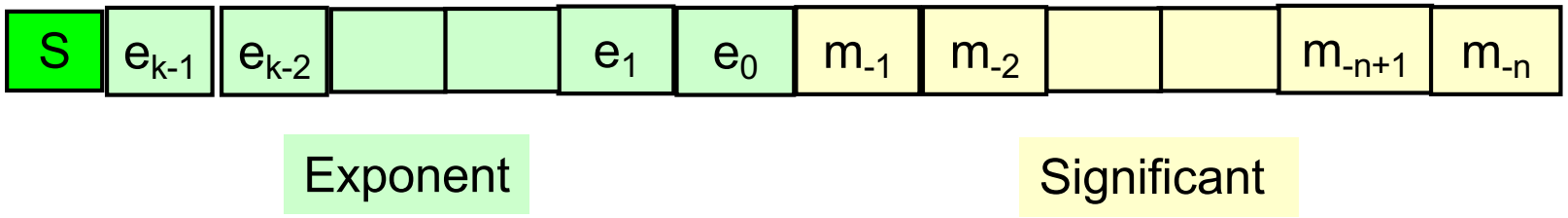
Resolution (quantification step)

$q = 0.0625$

$$Dyn_{dB} = 20 .\log \left(\frac{2^3}{2^{-4}}\right) = 42 \text{ dB}$$

$$SNR_{dB} = Ps_{dB} + 35 \text{ dB}$$

Signed Flotting Point representation "IEEE 754"



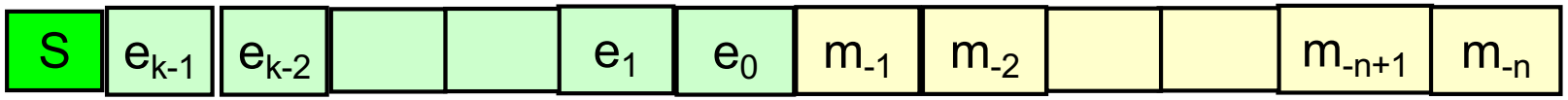
Single precision : 32 bits : k=8, n=23

$$x = (-1)^S.(1+\sum_{i=1}^{i=23} m_{-i}.2^{-i}).2^{E-127} \quad \text{with} \quad E = \sum_{j=0}^{j=7} e_j.2^j$$

Double precision : 64 bits : k=11, n=52

$$x = (-1)^S.(1+\sum_{i=1}^{i=52} m_{-i}.2^{-i}).2^{E-1023} \quad \text{with} \quad E = \sum_{j=0}^{j=10} e_j.2^j$$

Signed Flotting Point representation "IEEE 754"



Exponent

Significant

Definition Interval D
Single :≈ [-2^{128} à 2^{128}]
Double : ≈ [-2^{1024} à 2^{1024}]

Resolution q depends of the exponent:
Single: $2^{E-127}.2^{-23}$
Double: $2^{E-1023}.2^{-52}$

Special values
If E=255 (single) or E=2047(double)
if m=0 X= ± ∞
if m≠0 NAN (Not A Number)

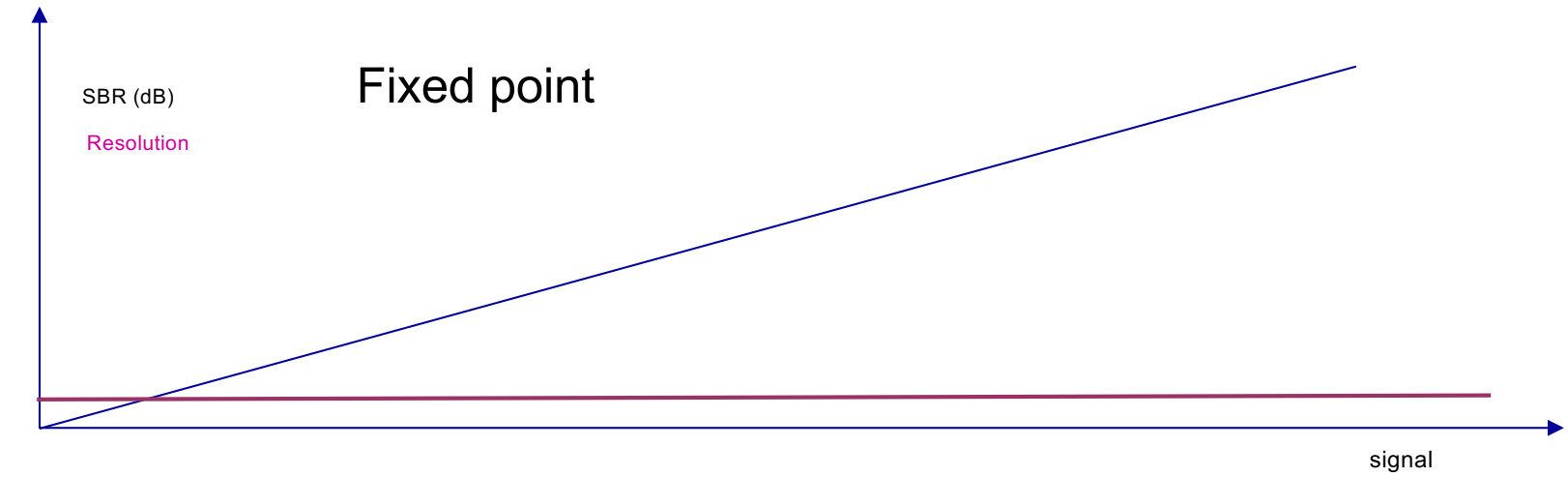
If E=0
if m=0 X = 0
Be careful if E=0 and m≠0

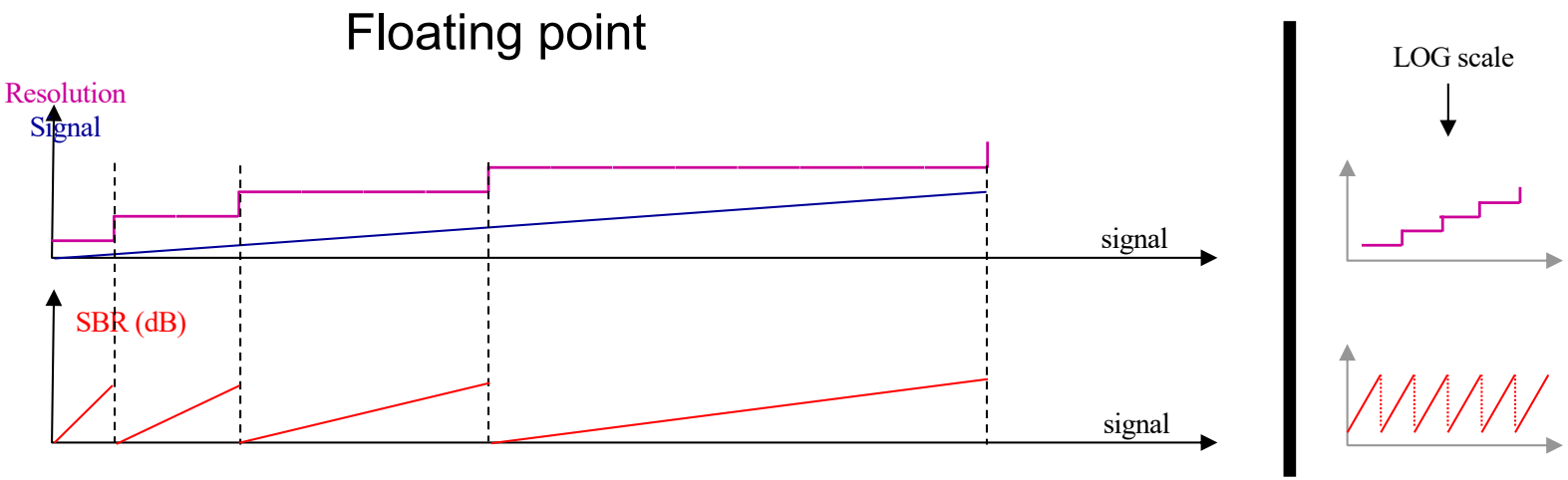
$$x = (-1)^S.(0+\sum m_{-i}.2^{-i}).2^{0-127}$$

$$Min = (0+2^{-22}).2^{-127}$$

Dynamics
$dyn_{dB\text{-}single}$ = 1530 dB
$dyn_{dB\text{-}double}$ = 12300 dB

Comparison between fixed and flotting coding





Comparison between fixed and flotting coding

To perform a Floating Point Algorithm, a Fixed Point based architecture DSP needs about 100 times more clock periods than a Floating Point based architecture DSP. But, Fixed Point DSP are faster and the execution time ratio is much lower than 100.

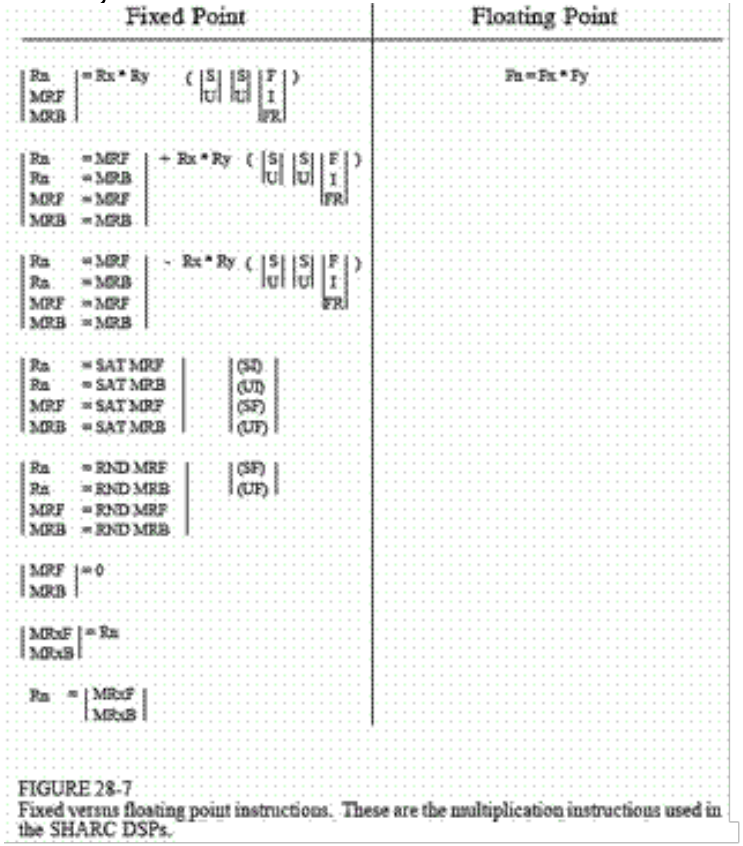
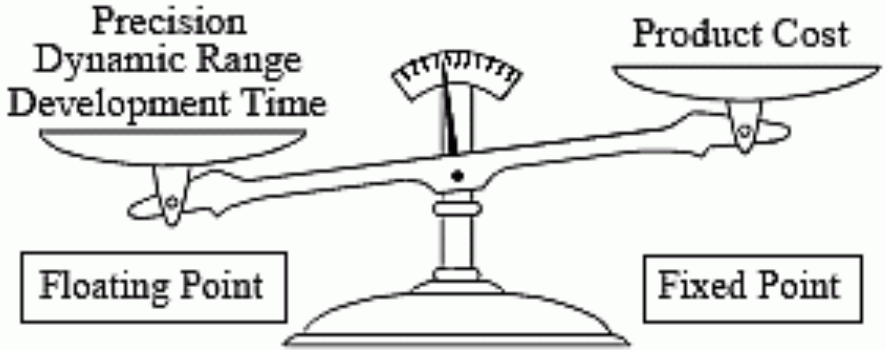


FIGURE 28-7
Fixed versus floating point instructions. These are the multiplication instructions used in the SHARC DSPs.

AUDIO
HIFI
(DAC ≥ 20 bits)

VIDEO
PROCESS
(DAC ≤ 16 bits)



Become familiar with DSP

DSP Basics

- Introduction to Digital Signal Processing ?
- Solutions to reduce processing / sample time
  - MAC
  - Circular buffers
  - Harward architecture
  - BSP, DMA
  - Pipeline
- Fixed- or Floating-Point precision ?
- Comparison between DSP and other solutions
  - Hardware solutions (GPP, µC, DSP, ASIC, ASSP, FPGA,GPU)

Solutions to reduce the processing time

Processors

– GPP : General Purpose Processors
– µC : Microcontroller
– DSP : Digital Signal Processor

Multiprocessor

– GPU (Graphic Processing Unit)

Non Reconfigurable Architectures

– ASIC (Application-Specific Integrated Circuit)
– ASSP (Application-specific standard products)

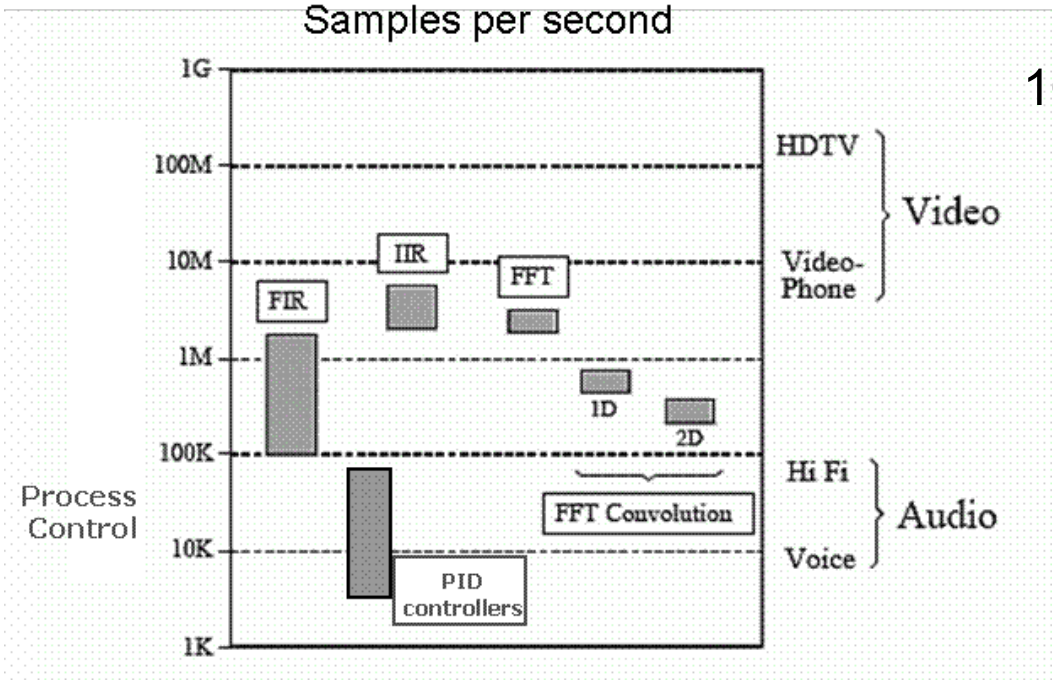
Reconfigurable Architectures

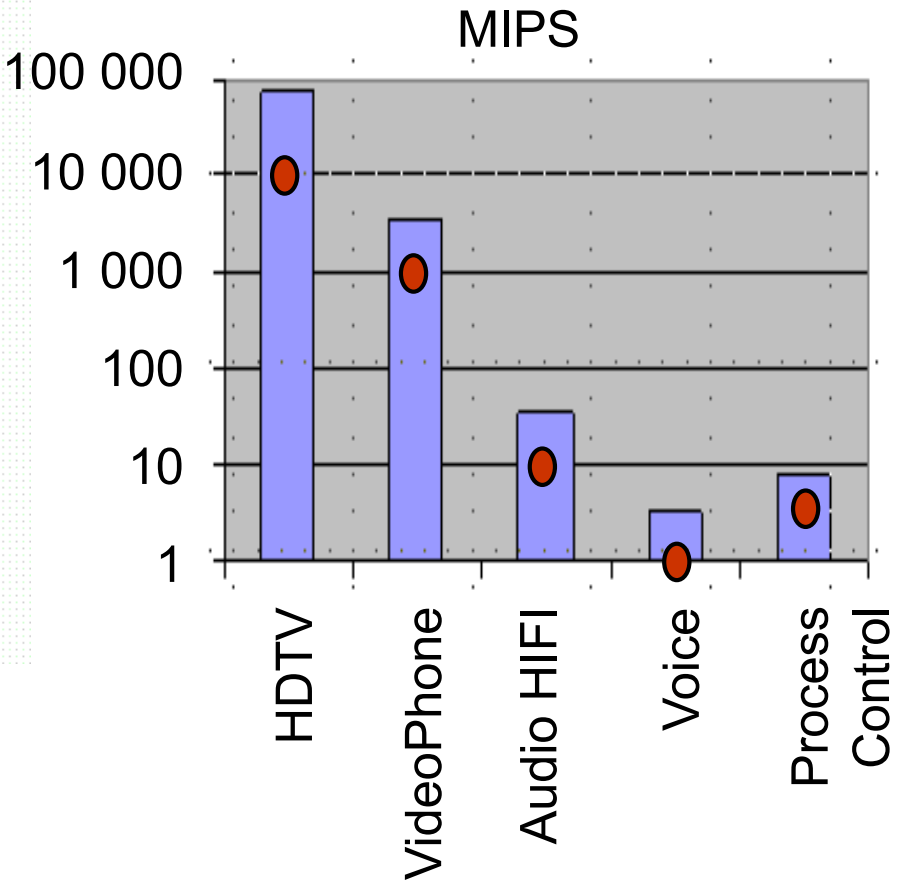
– FPGA (Field-Programmable Gate Array)

Evaluation of Processing Speed

MFLOPS	Million Floating–Point Operation Per Second	*Maximum number of Floating -Point arithmetic operations / second*
MOPS	Million Operations Per Second	*Maximum number of Operations : processing, DMA, Data transfer, I/O operations… / second*
MIPS	Million Instructions Per Second	*Maximum number of machine Codes (instruction) / second*
MMACS	Million of MAC per Second	*Maximum number MAC operations (Multiply + Accumilate)/ second*
MBPS	Mega-Bytes Per Second	*Band Width of a particular bus nr of an I/O port*

DSP applications constraints





Hardware platforms 1 : GPP

GPP :

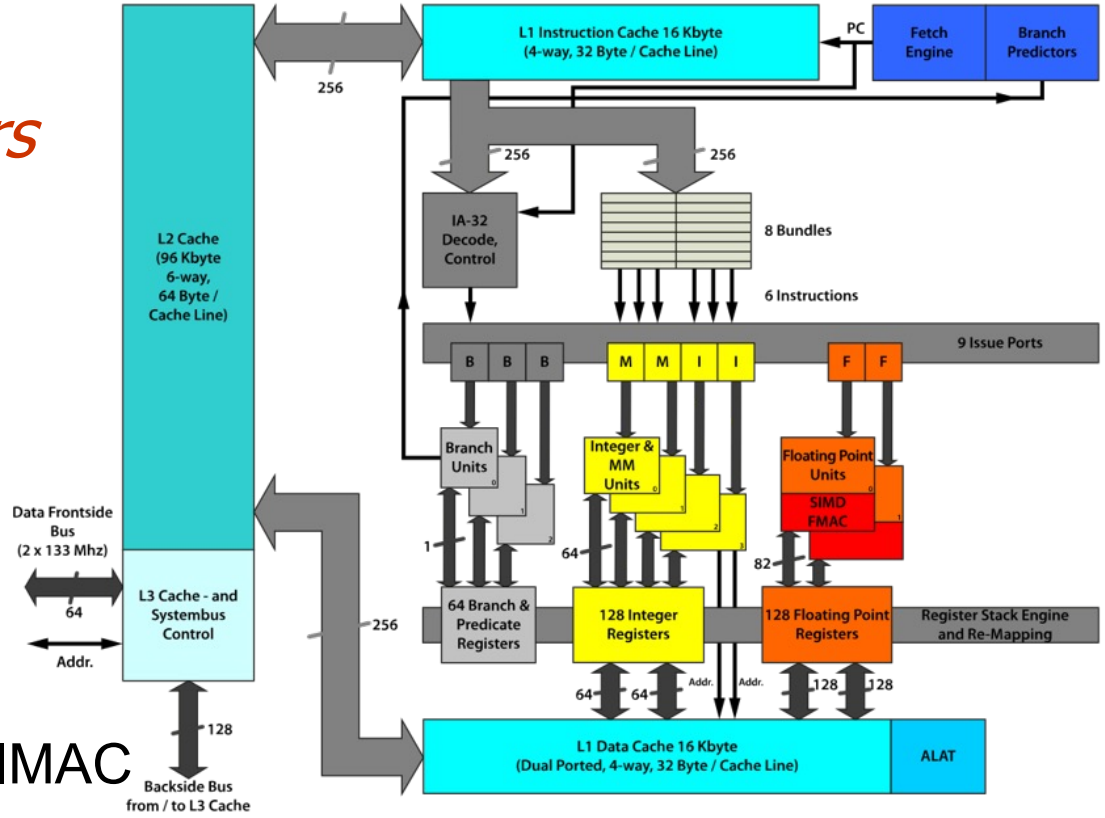
General Purpose Processors

- Intel
- AMD ex. Ryzen 5 >>>>>>>>>>
- **NXT S32R45** ARM (Advanced RISC Machines)
- **Apple M2 (ARM)**

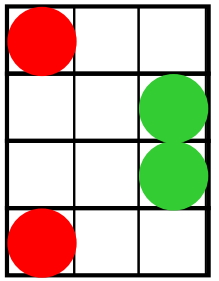
DSP architecture type of

AMD Ryzen

- Plusieurs CPU (coeurs)
- Dans chaque CPU : FMAC IMAC
- GPU
- Codec Video et Audio
- Protocoles de communications “standards” USB; SATA…



Cost
Processing Power
Flexibility
Power Consumption



Hardware platforms 1 : GPP

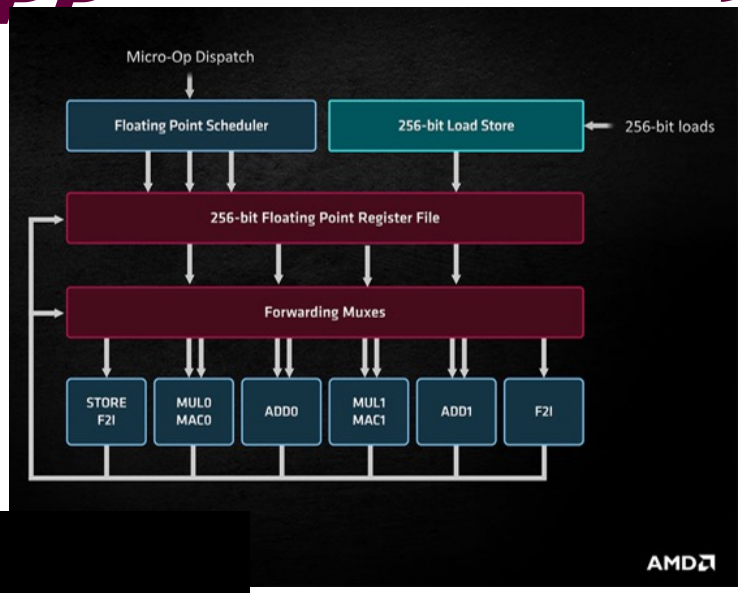
GPP :

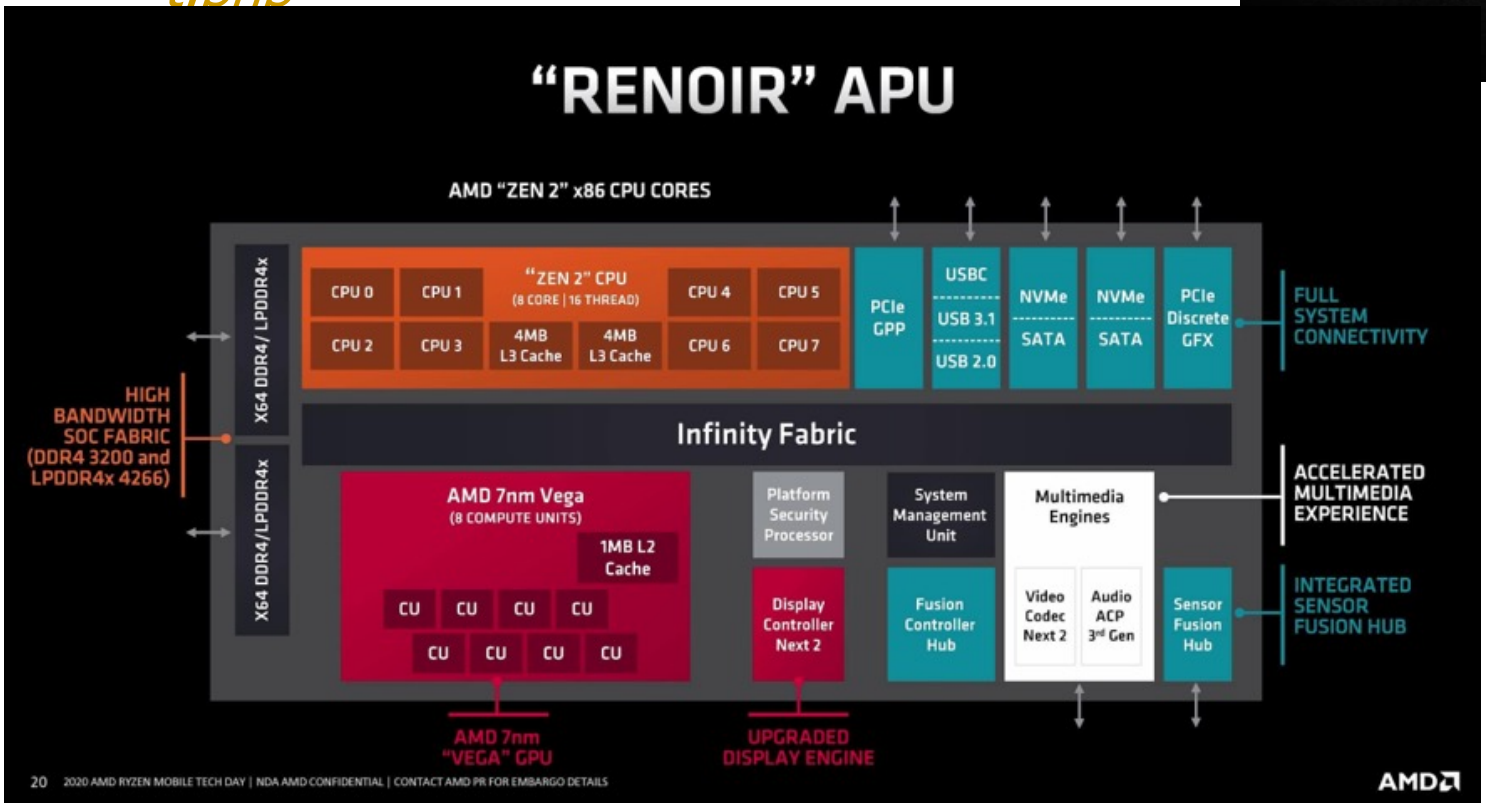
General Purpose Processors

DSP architecture Ryzen 9

Benchmarks : max Ryzen 9 170GFLOP

https://setiathome.berkeley.edu/cpu_list.php



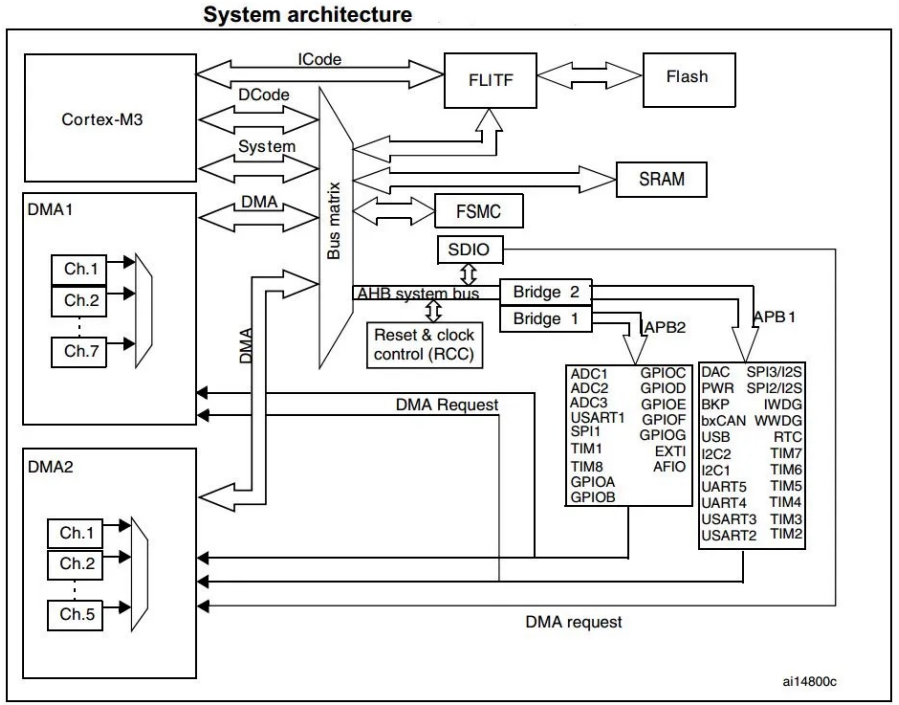


PR 2024

Hardware platforms 2 : µC

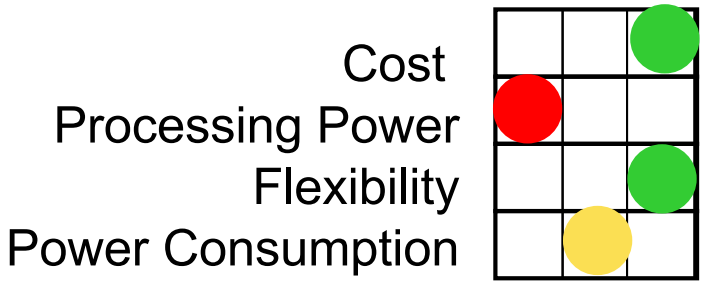
µC : *Microcontroller*

- Microchip / Atmel
- ST
- NXP **S32Z2**
- Renesas
- Cypress
- Intel, Ti …



DSP architecture type of Cortex M4 – STM32 F4

- Architecture Harvard (SIMD)
- DMA + Circular buffers
- 225DMIPS (Dhrystone benchmark)
- 2 MAC 16bits en 1 cycle
- FMUL



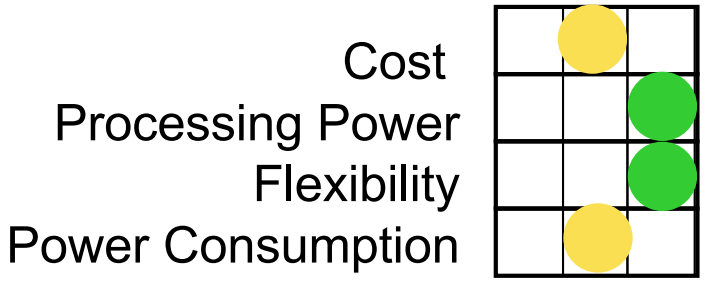
Hardware platforms 3 : DSP

DSP : *Digital Signal Processors*

- Ti **2022 : TMS320F2800137**
  2021 : single chip, 2020 : TDA3x : SOC 2 DSP + 2 Cortex M4)
- Analog Devices **2023 : ADSP-SC592**
- NXP (dernier en date 2013 MSC82XX)

DSP architecture :

- Architecture designed for digital signal processing
  - specialized instruction set
  - specialized hardware processing units
- High flow rate for input/output data



Hardware platforms 3 : DSP

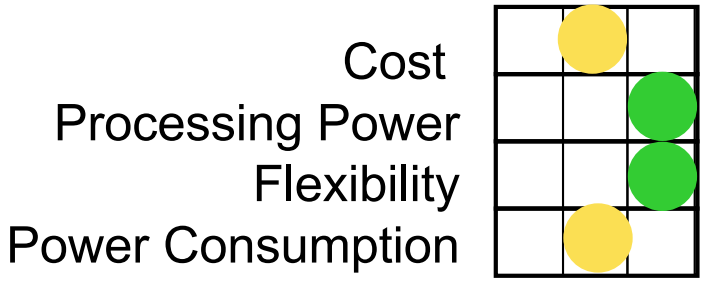
DSP CORE on System on chip

Ti **TDA3MV** : **2x 1000 MHz C66x DSP and 2 dual Arm Cortex-M4**

Ti **DRA785 : 2x 1000 MHz C66x DSP and 2 dual Arm Cortex-M4**

AD **ADSP-2156x** : **Quad Cortex + DSPs**

- Architecture designed for digital signal processing
  - specialized instruction set
  - specialized hardware processing units
- High flow rate for input/output data



DSP versus GPP

A DSP (Digital Signal Processor) is better for

- Size optimization
- Electric Power minimization
- Real Time processing at high flow rate of data

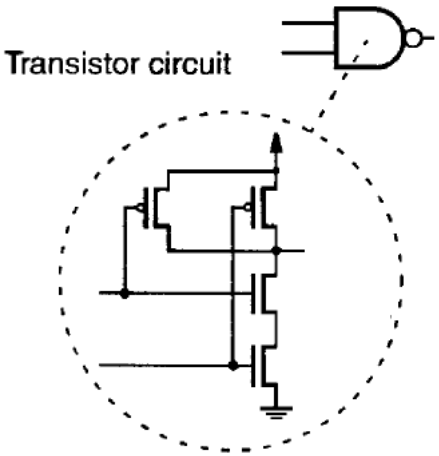
A GPP (General Purpose Processor) is better for

- Use of a large amount of memory
- Development and exploitation under advanced Operating Systems
- Mixing digital processing and other tasks

Hardware solutions 4 : ASIC

ASICs: Application-Specific Integrated Circuits

- Full custom design chips, "sea of transistors"
- Higher performances (speed, integration, power consumption).
  - State of the art:

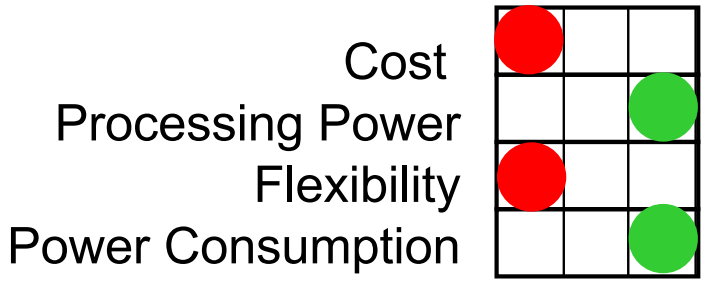


- Specific manufacturing tools, integration not done by the end user, no flexibility.
- High cost of the "first" one, reserved for very massive production (>100 000 / year).

Hence, use limited to:

1/ large scale diffusion product (mobile phones, ADSL modems,…)

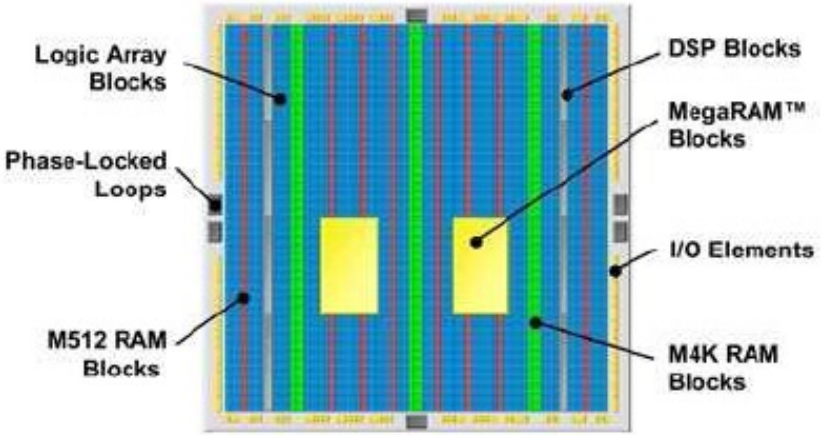
2/ well defined and widely used algorithms (FFT, Reed-Salomon codes,…)



Hardware solutions 5 : FPGA

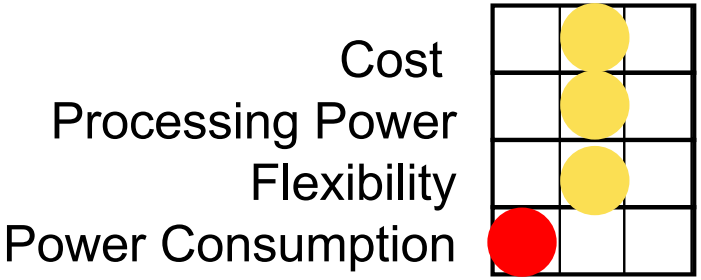
FPGA: Field Programmable Gate Array

- Xilinx : **DSP48E**
- Intel : **Arria® 10 FPGA and SoC FPGA**



Altera "Stratix"

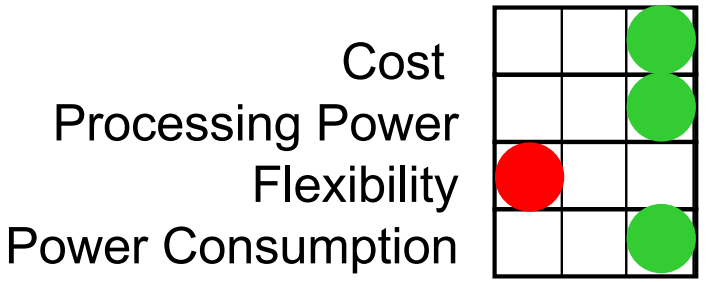
- Full custom design chips, “sea of cells"
- RAM technology : Architecture (VHDL description) stored in memory and download at power on.
- Development using IP (customisable µP, DSP, interfaces)
- Internal co-design (processor and hardware accelerators on the same chip)



Hardware solutions 4b : ASSP

ASSP: Application-specific standard products

- ASICs that can serve a wide market.
- Include dedicated processing engines (MPEG-2 decoders…) or communications links (USB, Ethernet …)
- Low in cost because of the high market volume
- Lack flexibility. For example, an MPEG-2 ASSP will only support a limited range of display resolutions; to introduce a new resolution requires a new ASSP and accompanying hardware redesign.



ASIC or ASSP versus DSP

Example : implementation of WIFI 802.11b standard

- The required processing power for a DSP is 4 GOPS (i.e. the most powerful DSP families)
- 200K gates are enough in ASIC or ASSP (i.e. very lower than the maximum integration capabilities)

Manufacturer usually choose **co-design** :
DSP for flexible signal processing (adaptive algorithms)
+ ASSP as accelerators for heavy standard processing (compression, communication…)

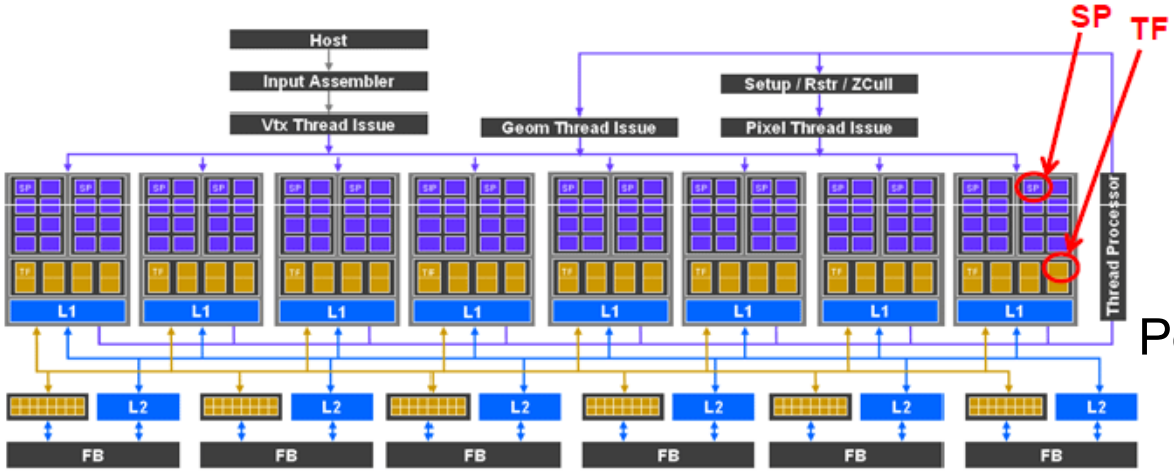
Hardware platforms 6 : GPU

GPU : *Graphics Processing Unit*

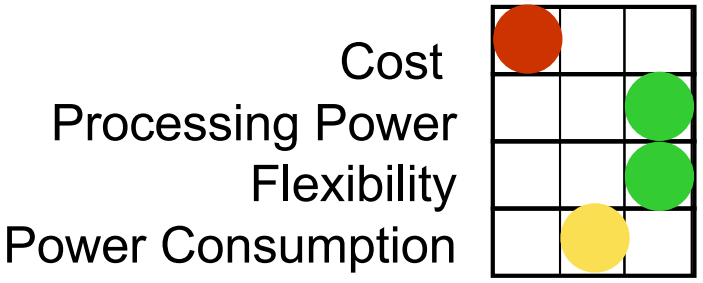
Several independent processing units working in parallel
- Streaming Processors
- Texture Filtering Units

Specific programming language (CUDA, OpenCL…)

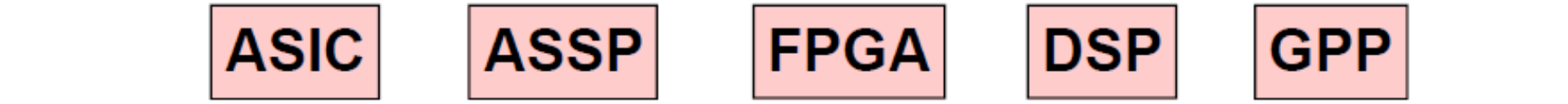
Very efficient for image processing



Nvidia GeForce 8800 Series GPU Block Diagram



Compromises for hardware solutions



Specific hardware equipment

Generic hardware equipment
Specific software

Efficiency for a specific application
Real time constrains
Power consumption constrains

Flexibility
Short delay for marketing
Low cost for development

Study plan

- **Part 1 : Become familiar with DSPs : Lec. 4H (19/03) + presentation 2H (23/04)**
  - Lecture "**DSP Basics**"
  - **Presentations "What's inside a DSP component ?"** (2H) : 15 min by pair
- **Part 2 : How to program DSPs : Lab 6H (02/04 + 23/04) + Project 6H (30/04 au 30/05)**
  - Application
  - Target
  - Demo

Study plan

- **Part 1 : Become familiar with DSPs : Lec. 4H (19/03) + presentation 2H (23/04)**
  - Lecture "**DSP Basics**"
  - **Presentations** (2H) : 15 min by pair
- **Part 2 : How to program DSPs : Lab 6H (02/04 + 23/04) + Project 6H (30/04 au 30/05)**
  - Application
  - Target
  - Demo

Become familiar with DSP

Presentation

- What's inside DSPs ?
  - What your presentation must include ?
  - Choose your component

What's inside a DSP component ?

your state of the art presentation should contain Specific information about...

For SOCs and FPGAs you may focus on one heart only

- Architecture schematic
- Benchmark (unit definition and values)
- Instructions (SIMD, MIMD, exemple
- DSP library (functions ?)
- Real-time (deterministic instructions ? Nb of cycles ?)
- Cache (number, capacities)
- Circular buffers (on witch periferals)
- Accelerators
- DMA (between ?)
- Communication interfaces
- Precision (ADC/DAC, ALU fixed or float on n bits...)
- Applications
- Prices

Choose your component !

References
NXP S32Z2
FPGA INTEL
FPGA XILINX
STM32F…
TMS320C66x (dans SOC)
ADSP-2156x (SOC)
AMD Ryzen
ADSP-SC598
TMS320F28075

Study plan

- **Part 1 : Become familiar with DSPs : Lec. 4H (19/03) + presentation 2H (23/04)**
  - Lecture "**DSP Basics**"
  - **Presentations** (2H) : 15 min by pair
- **Part 2 : How to program DSPs ? Lab 6H (02/04 + 23/04) + Project 6H (30/04 au 30/05)**
  - Target
  - Application
  - Demo

DSP in the Lab

What your application must include.

- ADC
- DAC
- DMA
- Circular buffering
- Some functions of the DSP Library (based on library's examples)
- A concret use case
- An algorithm's benchmark (visualisation of the algorithm delay per data)
- Benchmark vs datasheet information