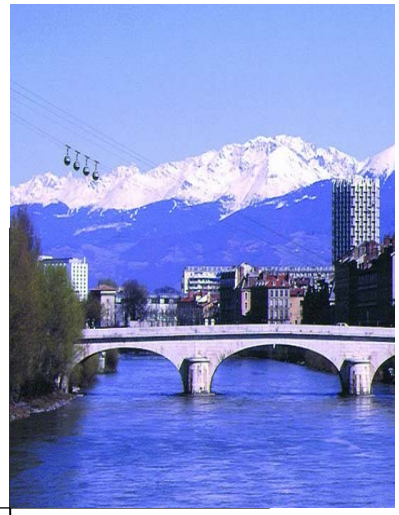
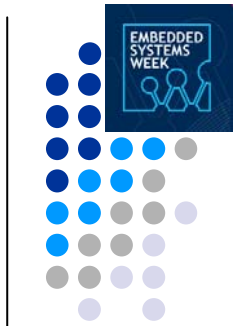
# Smart Cameras and Visual Sensor Networks



Embedded computer vision and image processing

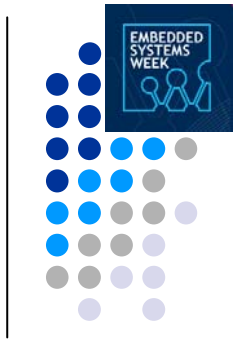
François BERRY
Joel FALCOU
Bernhard RINNER

# Embedded Computer Vision




- Embedded computer vision system
- Hardware considerations
  - Programmable devices (FPGA, CPLD)
  - Signal Processor (DSP)
  - General purpose processor (Cell)
  - Graphics Processor Unit
- Few ways for programming
  - SOPC approach
  - HDL versus High level synthesis language

# Embedded Algorithms vs processing hardware




Algorithm's performance in terms of speed on a DSP or FPGA often differs vastly as compared to its performance on a general-purpose processor.

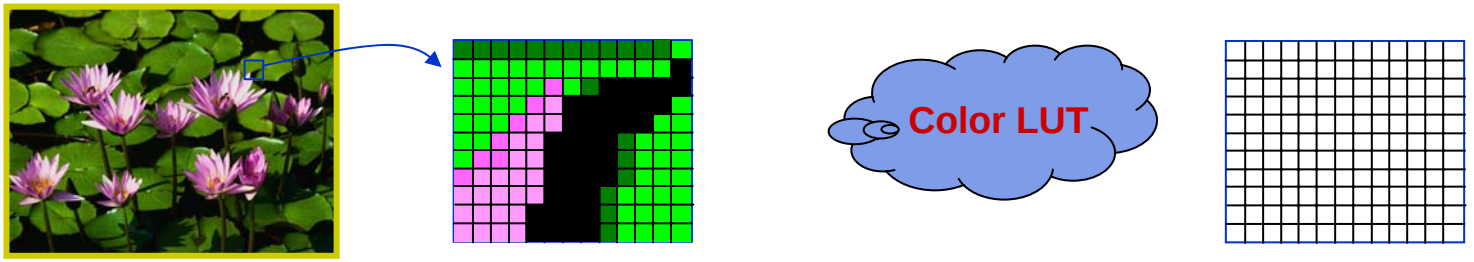
There are special characteristics of an algorithm which are taken in to consideration when applying that algorithm to special hardware:

- Sequential data access as opposed to random access
- In case of multiple streams of data, they must be largely independent
- Fixed data size packet
- Possibility to break up the stream computations into pipeline stages: The same set of computations can be applied to different set data
- Operations require only fixed-precision (or integer!) values
- Algorithms are parallelizable at instruction and module-level: It avoids "inter-stream" communication

# Operations vs complexity of data access

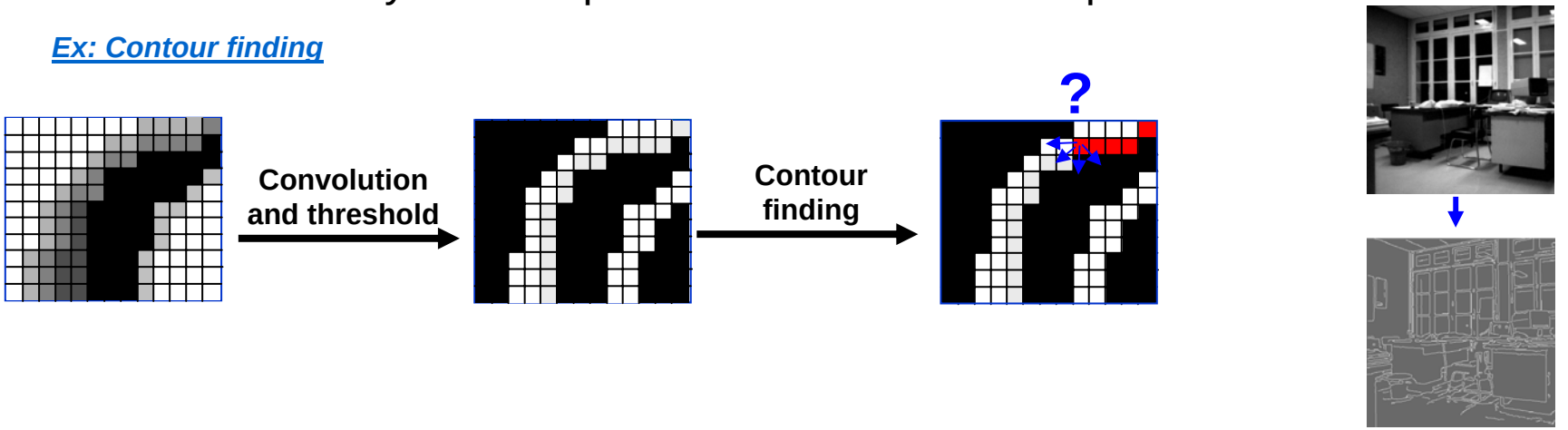
For the best speed and memory performance, the algorithm accesses only a single pixel at a time and the sequence of pixel access is known before-hand.

***Ex: Color Lookup-Table:***




The worst case are algorithms that require large amounts of data for the calculation of one result with a dynamic dependence on the result of previous calculations

***Ex: Contour finding***

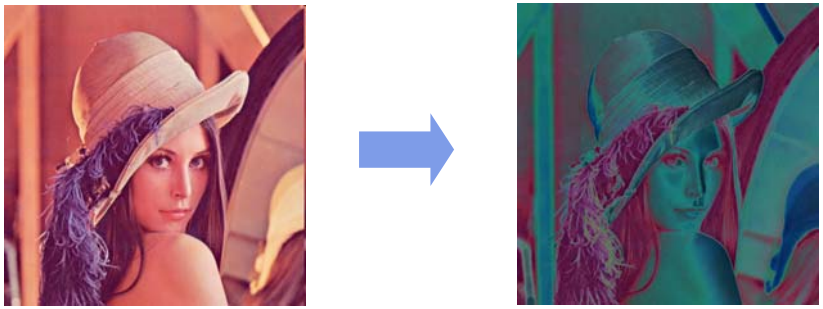


# Data interdependency in Image Space (I)

Common operations versus complexity of their data access patterns

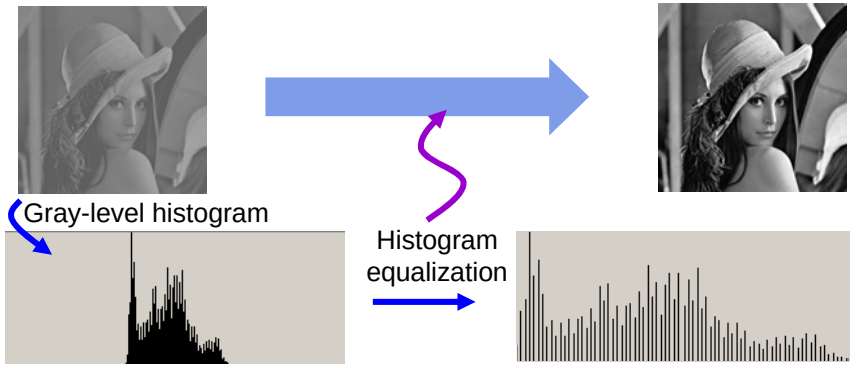
**Pixel Processing:** A single pass over the image is sufficient, and a pixel's new value is only determined by exactly one source pixel value.

*Examples :*
- Lookup Table
- Gray-level or color thresholding
- Color space conversion
- Brightness correction
- Arithmetic operations
- Logic operations



**N- pass:** Multiple passes over the image and data space are necessary; However, only one source pixel value determines the new pixel value

*Examples :*
- Min, Max, Average, Std Deviation
- Histogram equalization,…
- Hough transforms




***Source :*** *Hardware considerations for embedded vision systems - Mathias Kölsch and Steven Butner – Embedded Computer vision 2009 – APR – Springer.*

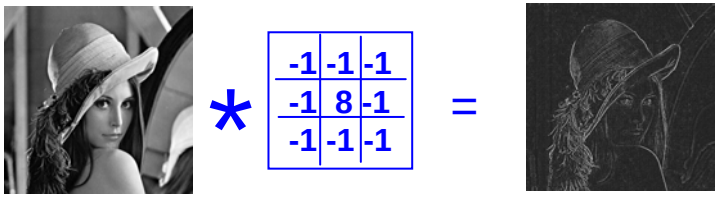
# Data interdependency in Image Space (II)

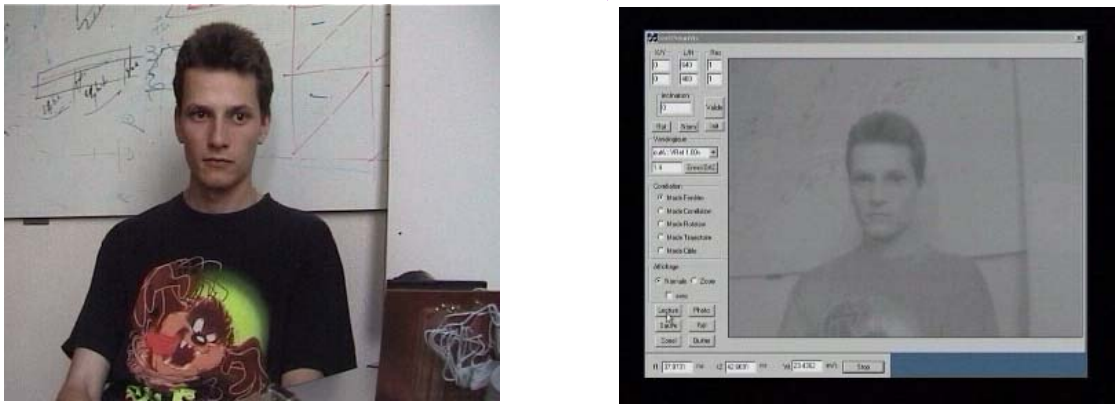
Common operations versus complexity of their data access patterns

**Fixed-size block access:** The values of pixels in an area of known and fixed size determine the output value

*Examples :*

- Morphology
- Convolution, filtering
- Wavelets
- Feature tracking (KLT tracker, **SAD**, SSD,….)



High-speed template tracking by SAD on ROI 32x32 @ 2000 fr/s (SeeMOS Demo)

***Source :*** *Hardware considerations for embedded vision systems - Mathias Kölsch and Steven Butner – Embedded Computer vision 2009 – APR – Springer.*

# Data interdependency in Image Space (III)

Common operations versus complexity of their data access patterns

**Data-independent, global access:** multiple source pixel values from pixels all over the image determine the outcome of the operation. However, the access pattern is known

*Examples :*

- Viola-Jones
- Warping or remapping for distortion correction


Tracking of deformable object

***Source :*** *Hardware considerations for embedded vision systems - Mathias Kölsch and Steven Butner – Embedded Computer vision 2009 – APR – Springer.*

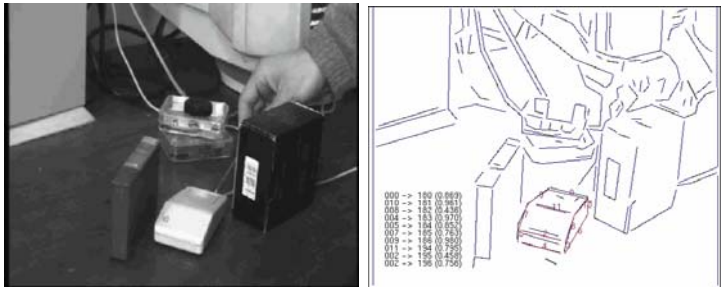
# Data interdependency in Image Space (IV)

Common operations versus complexity of their data access patterns


**Data-dependent, random access:** Multiple source pixel values from all over the image determine the outcome of the operation. The access pattern is determined by the values read from the source pixels

*Examples :*

- Dynamic vision
- Contour finding, flood fill

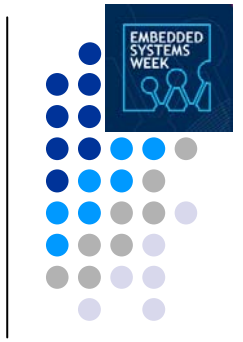


Tracking of segmented object based on contour finding

Only the motion parts are processed

***Source :*** *Hardware considerations for embedded vision systems - Mathias Kölsch and Steven Butner – Embedded Computer vision 2009 – APR – Springer.*

# Embedded Computer Vision

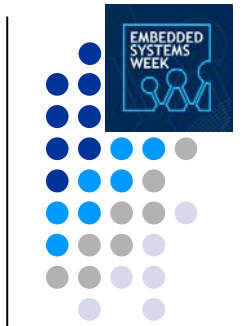



- **Embedded computer vision system**

- Hardware considerations
  - Programmable devices (FPGA, CPLD)
  - Signal Processor (DSP)
  - General purpose processor (Cell)
  - Graphics Processor Unit

- Few ways for programming
  - SOPC approach
  - HDL versus High level synthesis language

# Embedded Computer Vision

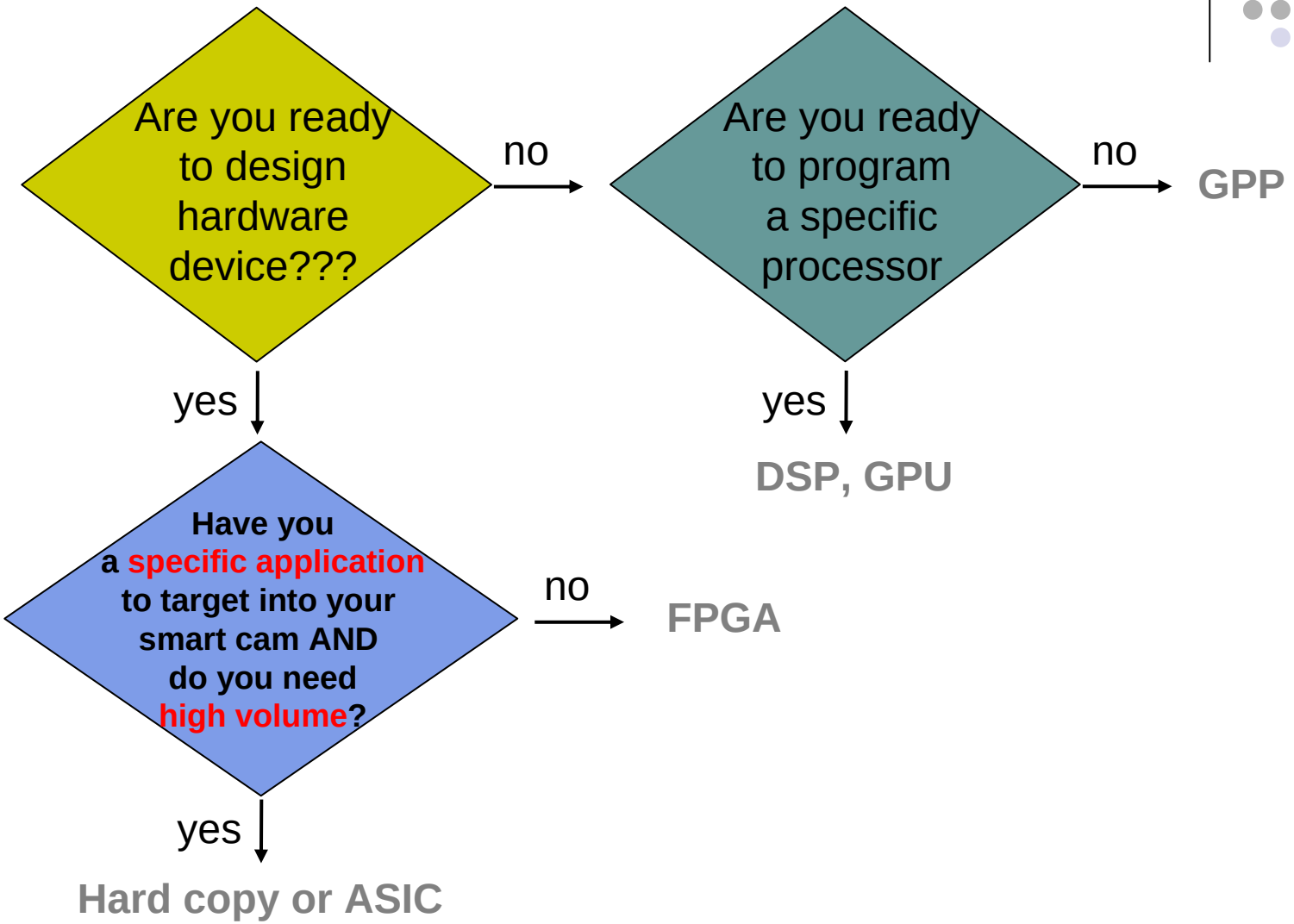



- Embedded computer vision system

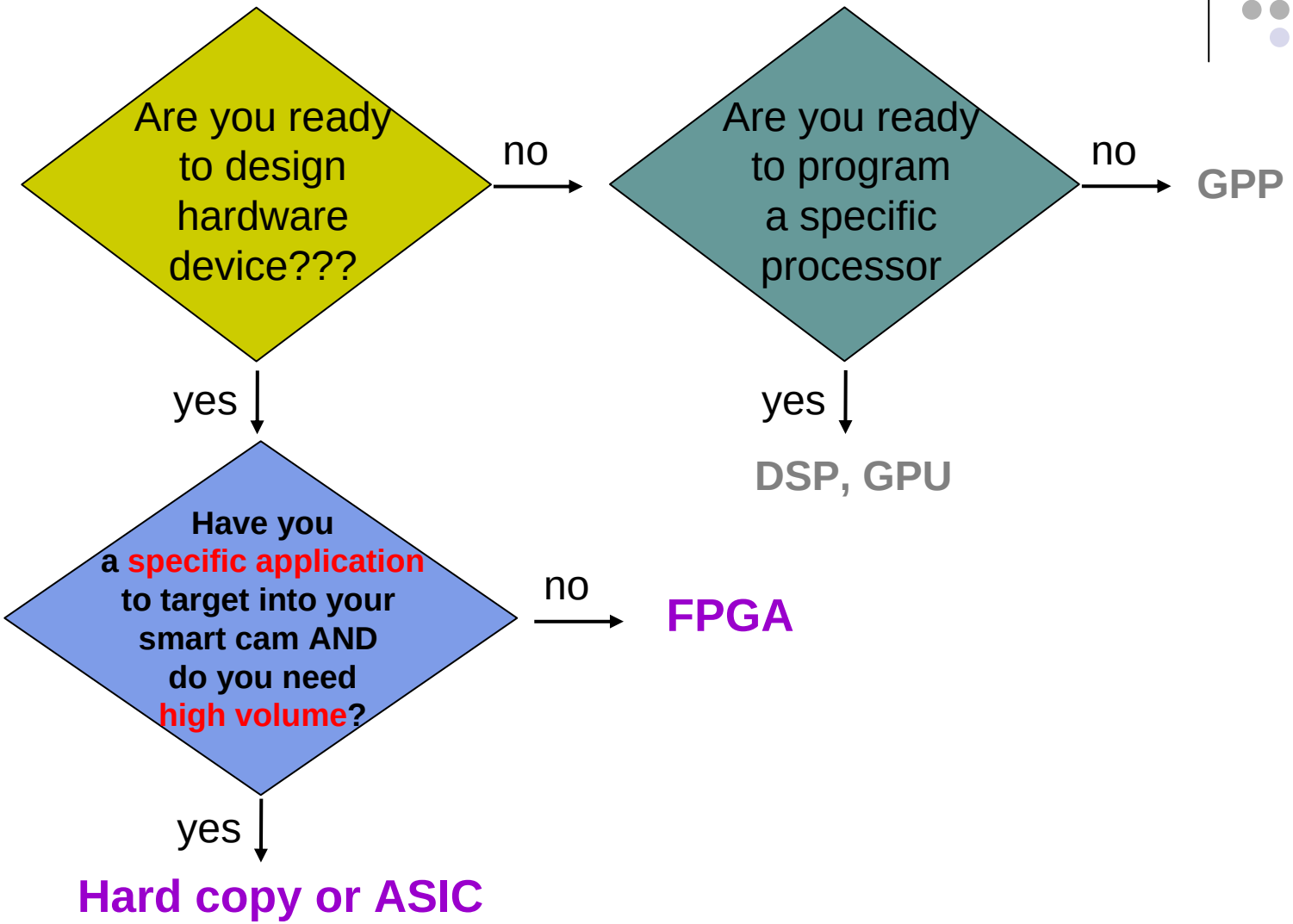
- **Hardware considerations**
  - **Programmable devices (FPGA, CPLD)**
  - **Signal Processor (DSP)**
  - **General purpose processor (Cell)**
  - **Graphics Processor Unit**

- Few ways for programming
  - SOPC approach
  - HDL versus High level synthesis language

# Hardware considerations



# Hardware considerations



# Customized Hardware

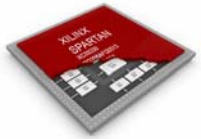
**Full-Custom ASIC**

**Hard Copy**

**FPGA**

**Building the hardware architecture**

It is an integrated circuit designed to be **configured by the customer** after manufacturing



**FPGA**

\+

Application

=

Customized IC


**Takes advantage from both: FPGA design methodology + ASIC technology**

Start from scratch! The devices are fully designed by the designer

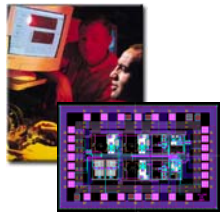
Application

\+

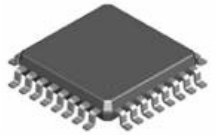
technology

founder

\+

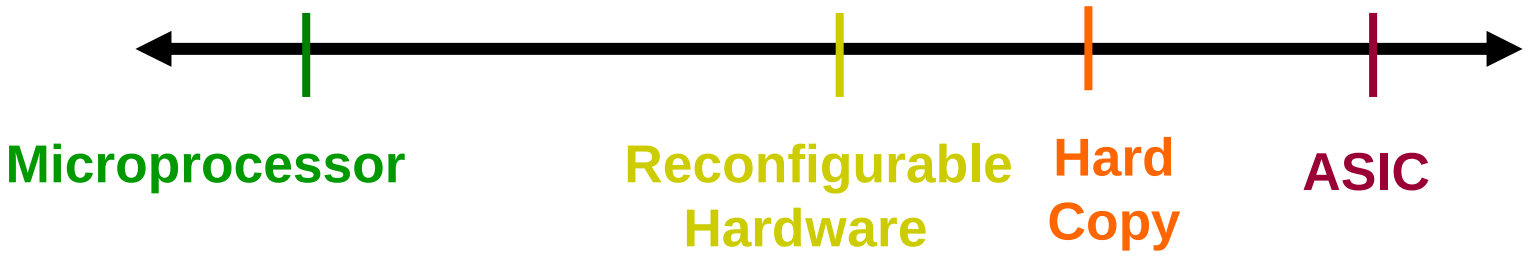


IC CAD



# Customized Hardware

## Implementation Spectrum



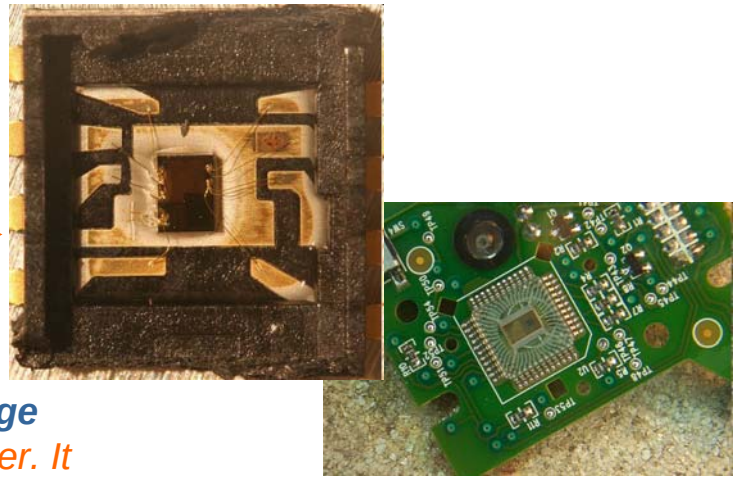

- ASIC gives high performance at cost of inflexibility.
- Processor is very flexible but not tuned to the application.
- Reconfigurable hardware is often a nice compromise.

# Full-Custom ASIC

ASIC means Application Specific Integrated Circuit.

It is an integrated circuit (IC) customized for a particular use, as opposed to a general purpose processor

One of the most popular smart camera based on ASIC : optical mouse!



*This sensor from a optical mouse contains a **15x15 image array**, a **digital signal processor** and a LED diode driver. It continuously **captures images**, analyzes them and **converts to the horizontal and vertical offsets**.*

# FPGA

## What is Reconfigurable Computing?


Computation using *hardware* that can be *adapted* at the logical level to solve *specific* problems

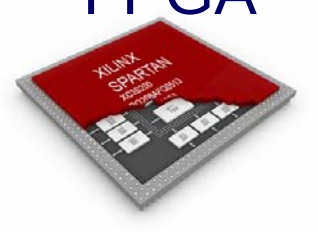
***Why is this interesting in embedded image processing?***

- Many applications are poorly suitable for microprocessors (low-level image processing → SIMD)
- VLSI “explosion” provides increasing resources. How can we use them? →“Field-programmable” devices
- Allows for high performance, bug fixes, and fast time-to market for a selection of applications

# FPGA (cont.)

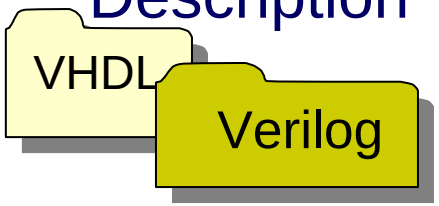

Integrated circuit is able to change interconnectivity of a large number of fundamental computing components via configuration information stored in onboard static RAM (or anti-fuse).

FPGA




\+

Hardware Description




=

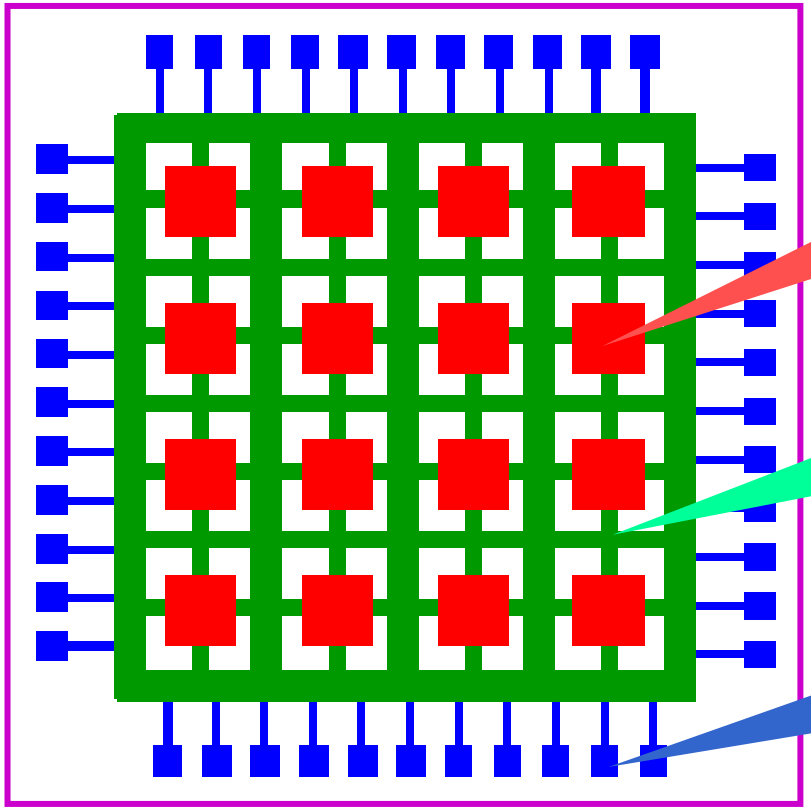
Customized IC

- Processor
- Specific Glue logic
- Specific applications

ALTERA. XILINX® Actel Lattice Semiconductor Corporation achronix SEMICONDUCTOR CORPORATION

# FPGA (cont.)

An FPGA is a device composed of 3 main parts.

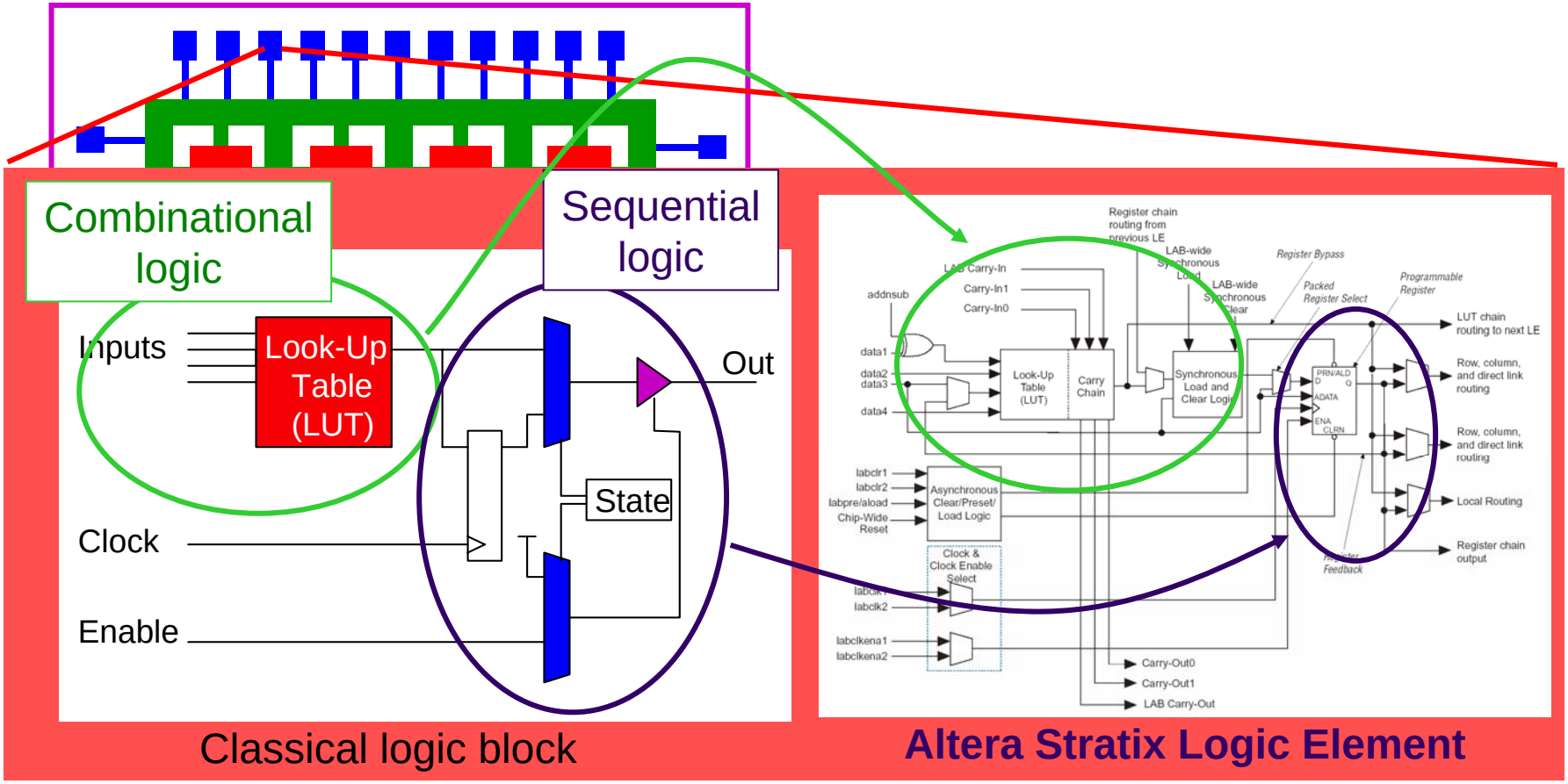
**Configurable Logic Block (CLB)**
Look-up table (LUT)
Register
Adder, Multiplier, Memory, Microprocessor

**Programmable interconnect**
Wires to connect inputs and outputs to logic blocks

**Input/Output Block (IOB)**
Special logic blocks at periphery of device for external connections

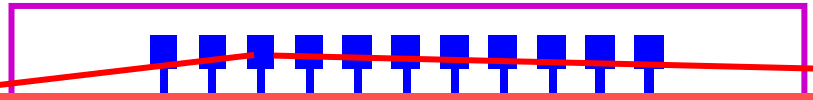
# FPGA (cont.)

**Configurable Logic Block (CLB)**
**Look-up table (LUT), Register**

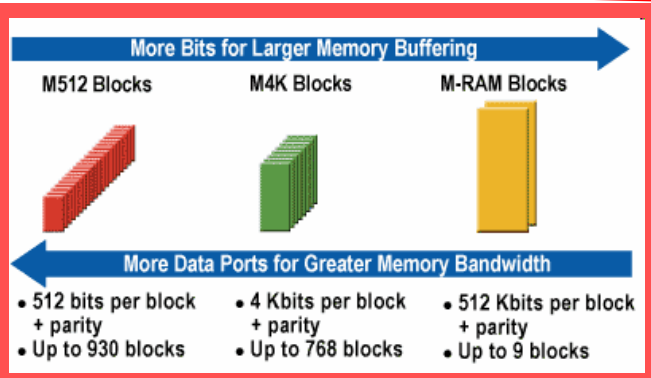



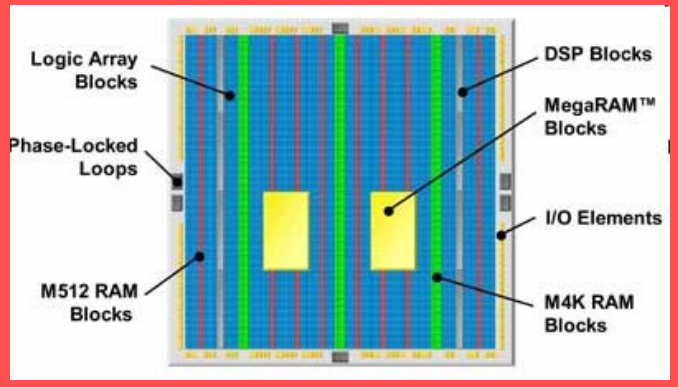
# FPGA (cont.)

## Configurable Logic Block (CLB)
## Memories

| Memory Block | Applications |
|---|---|
| **M512 / MLAB** | • Shift registers<br>• Small FIFO buffers<br>• Filter delay lines |
| **M4K / M9K** | • General-purpose memory<br>• Packet header or cell buffers |
| **M144K / M-RAM** | • Processor code storage<br>• Packet buffers<br>• Video frame buffers |




***Stratix Device Architecture***

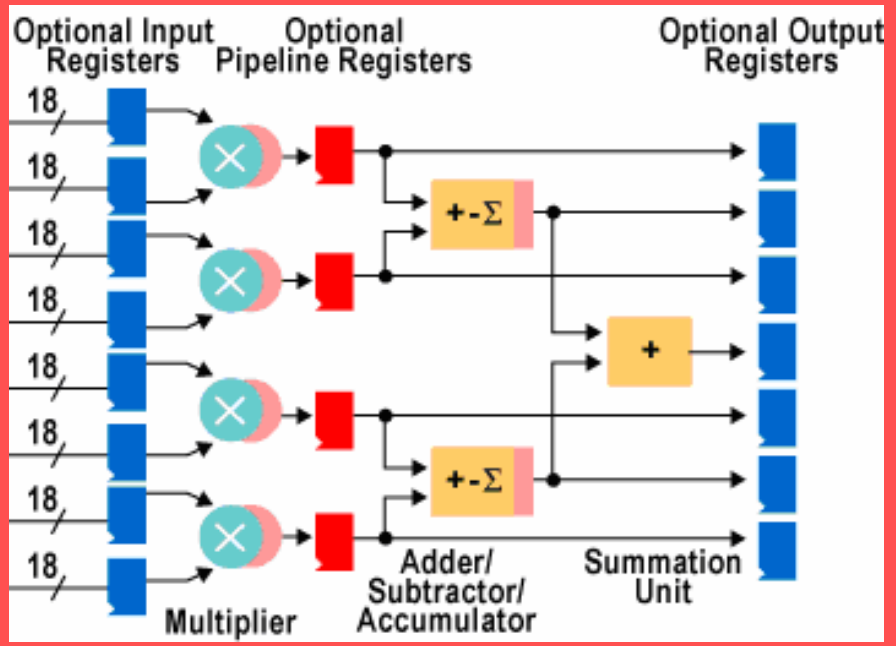
# FPGA (cont.)

**Configurable Logic Block (CLB)**
**DSP Blocks**

Embedded DSP blocks are available for many FPGAs

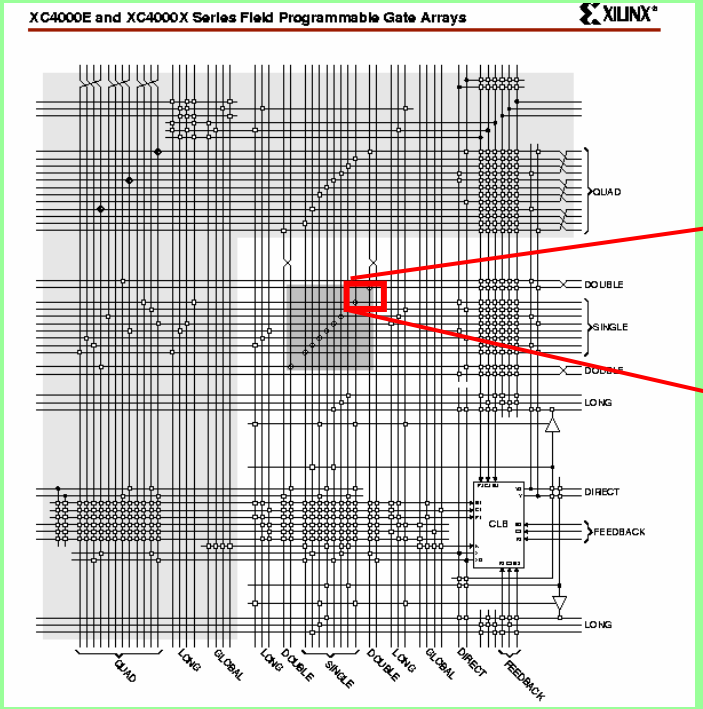
It consists of hardwired blocks to compute “MAC” operation such as:

**a ← a * b + c**

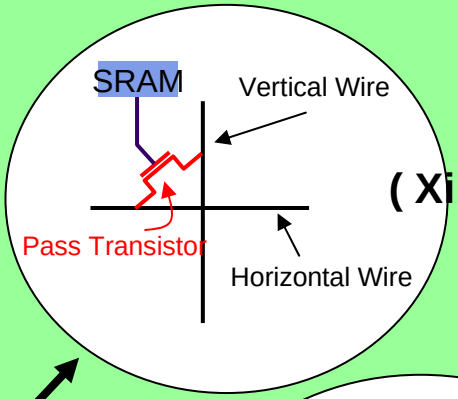


# FPGA (cont.)

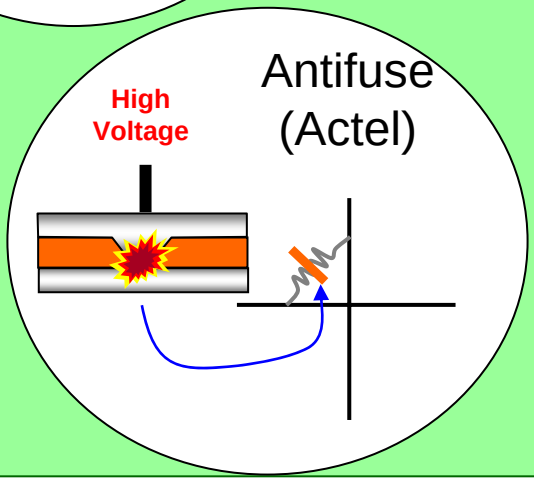
## Programmable interconnect



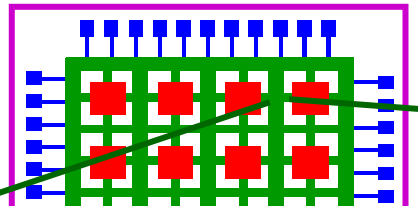

2 technologies:



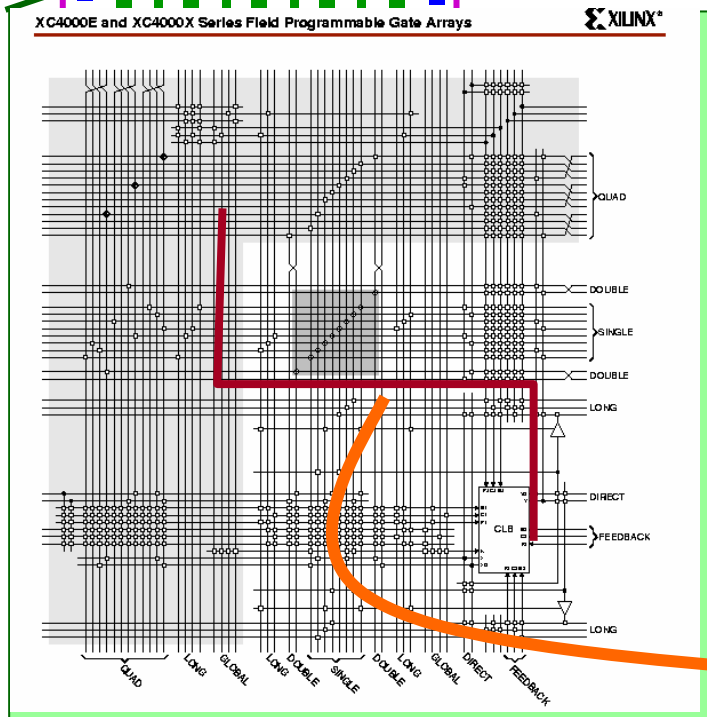

**SRAM**
**( Xilinx, Altera)**



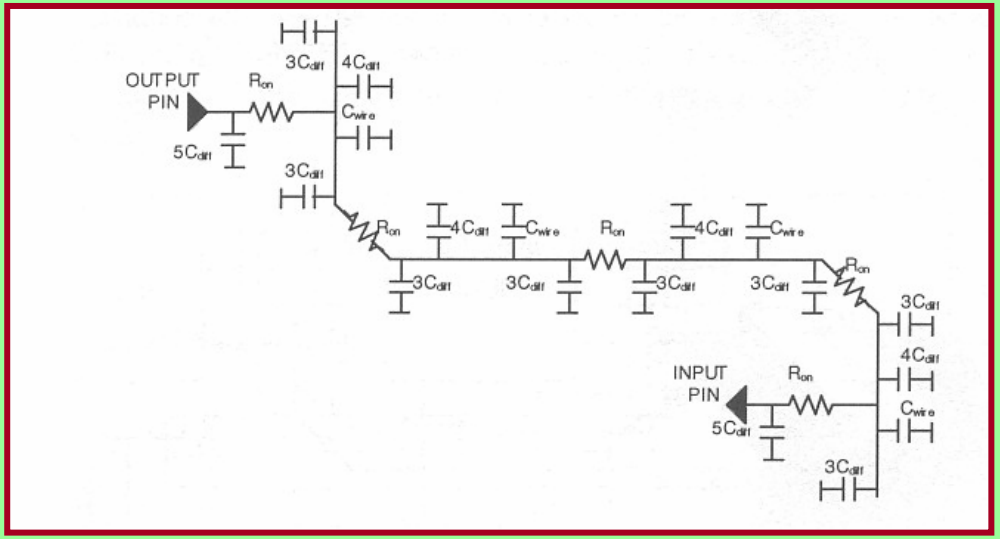
# FPGA (cont.)



**Programmable interconnect**

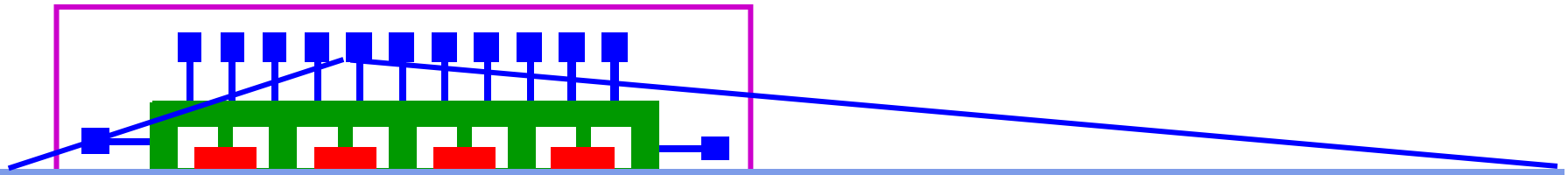




Based on the switch and parasitic wire, interconnect routes can be modeled as *RC* networks.

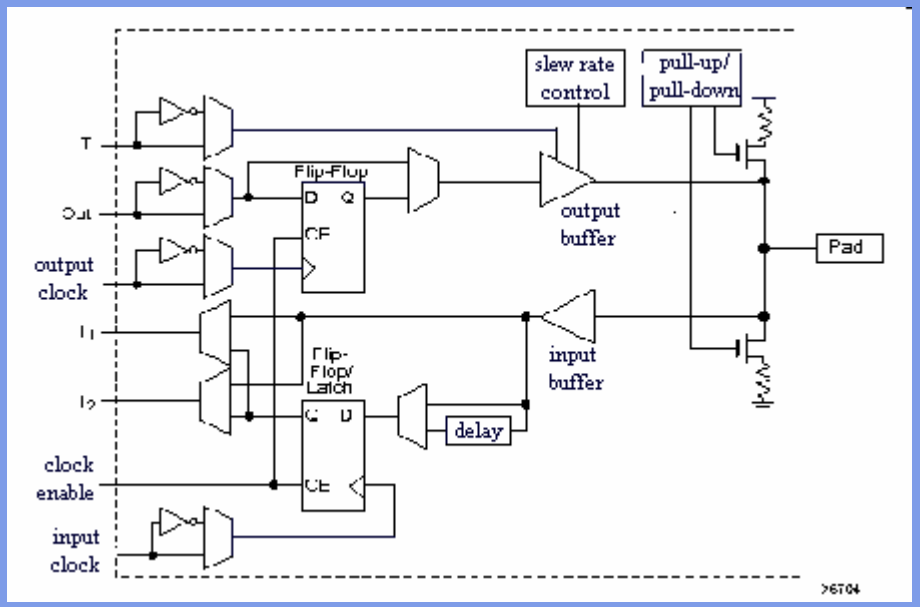


# FPGA (cont.)

## Input/Output Block (IOB)

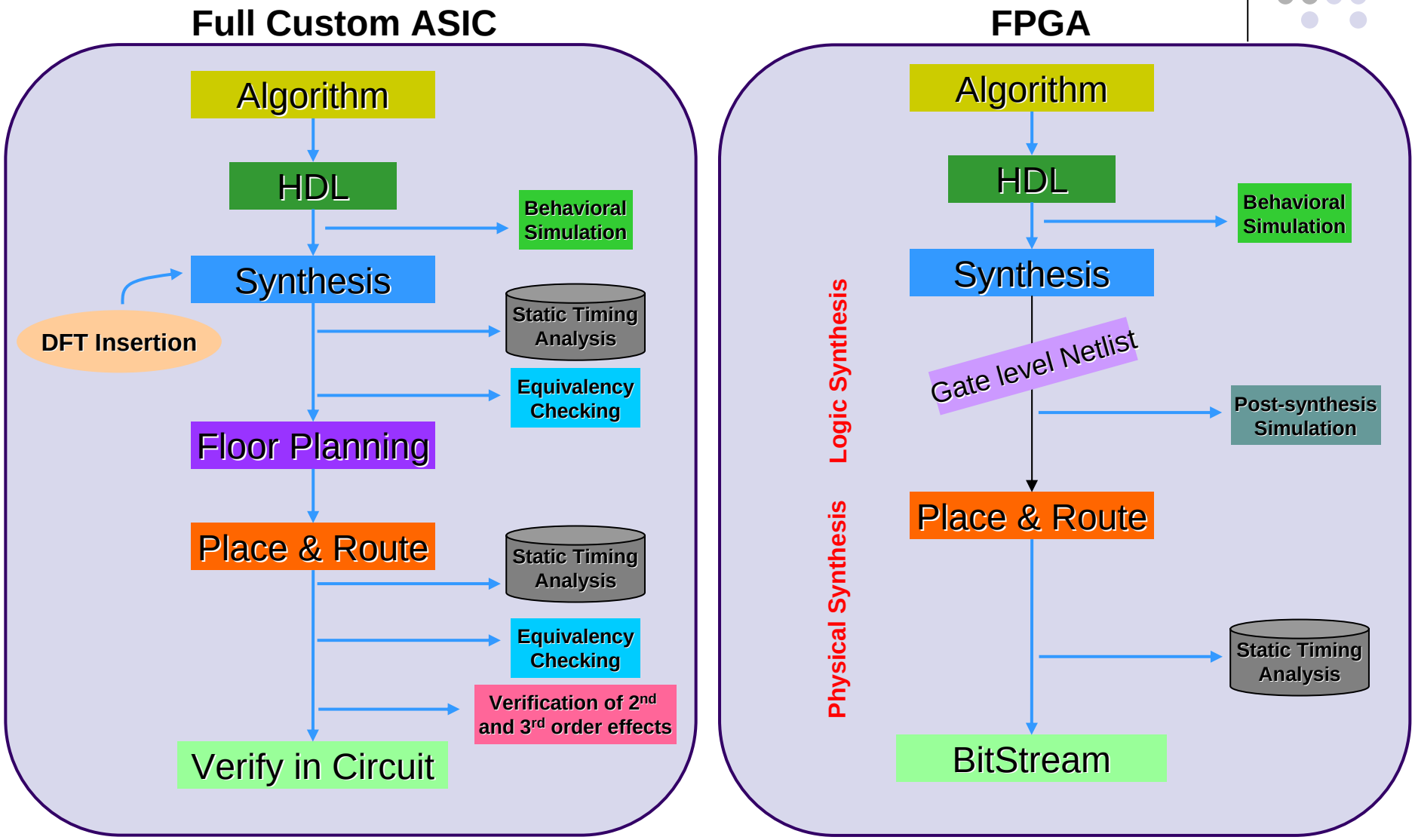



- FPGAs provide support for dozens of I/O standards (TTL, CMOS, LVDS, …)
- I/O in FPGAs is grouped in banks with each bank is independently able to support different I/O standards.



# Full-Custom ASIC vs FPGA

## Design flow



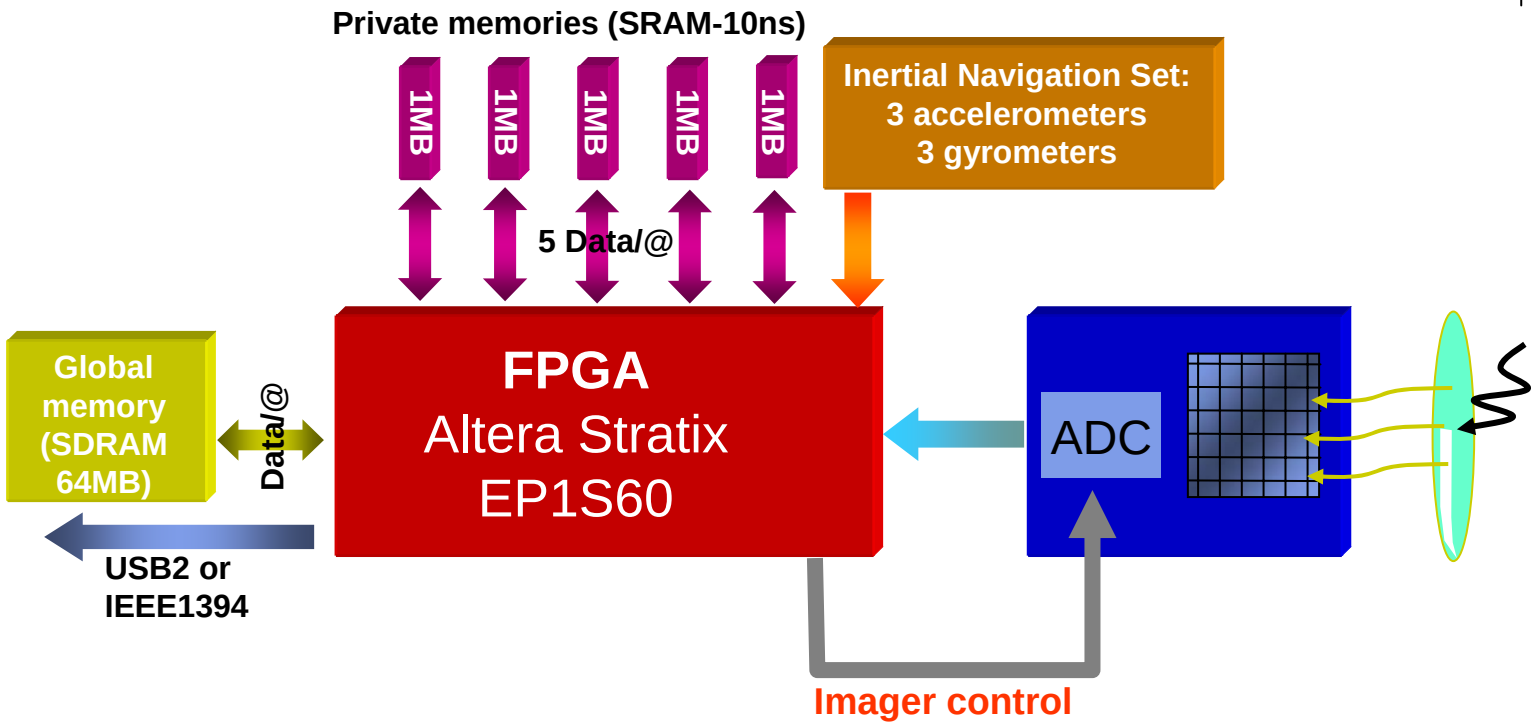
# Customized Hardware

## FPGA VS Full-Custom ASIC

| FPGA Design Advantages | ASIC Design Advantages |
|---|---|
| **Faster time-to-market** - no layout, masks or other manufacturing steps are needed | **Full custom capability** - for design since device is manufactured to design specs |
| **No upfront NRE** (non recurring expenses) vs costs typically associated with an ASIC design | **Lower unit costs** - for very high volume designs |
| **Simpler design cycle** due to software that handles much of the routing, placement, and timing | **Smaller form factor** - since device is manufactured to design specs |
| **More predictable project cycle** due to elimination of potential re-spins, wafer capacities, etc. | **Higher raw internal clock speeds** |
| **Field reprogramability** - a new bit stream can be uploaded remotely | |

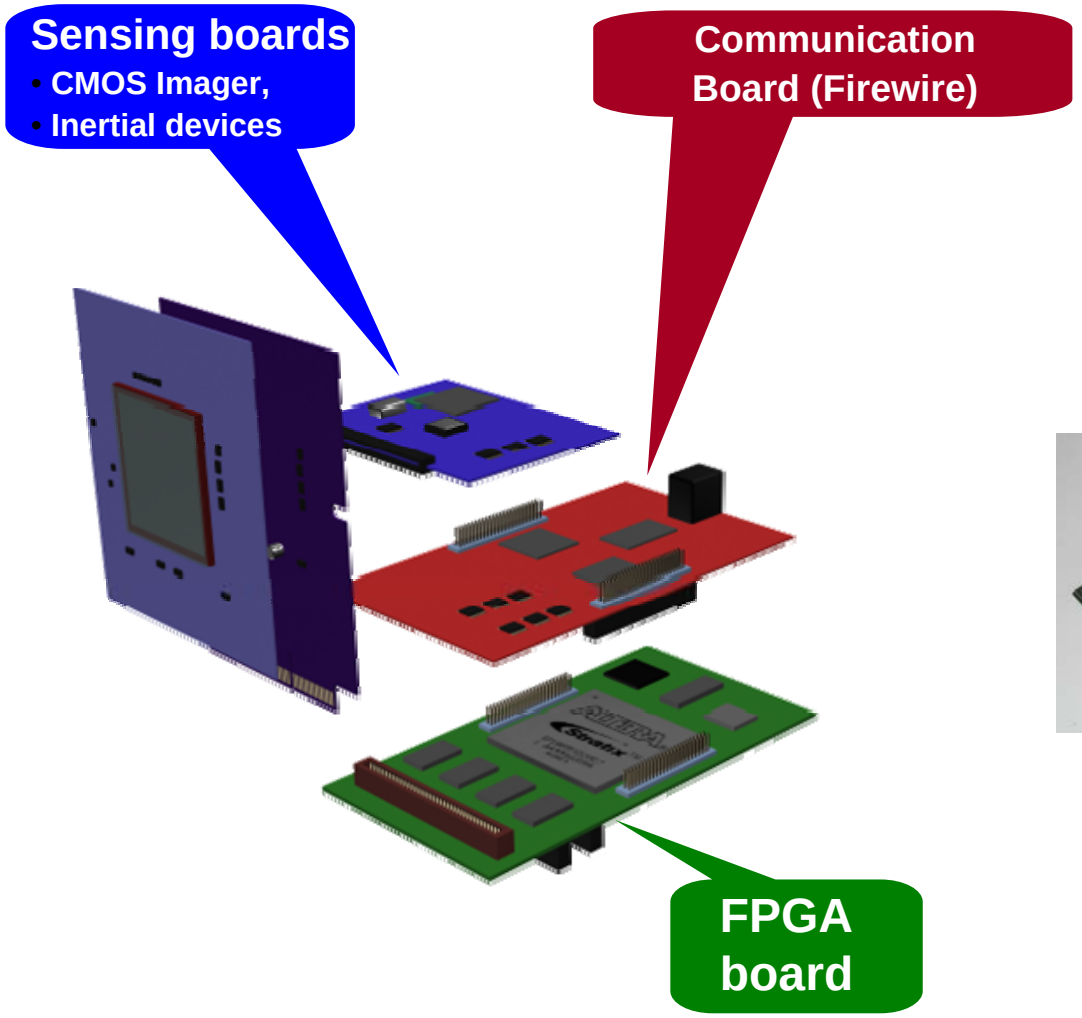
***Source :*** *Xilinx - http://www.xilinx.com/company/gettingstarted/fpgavsasic.htm*

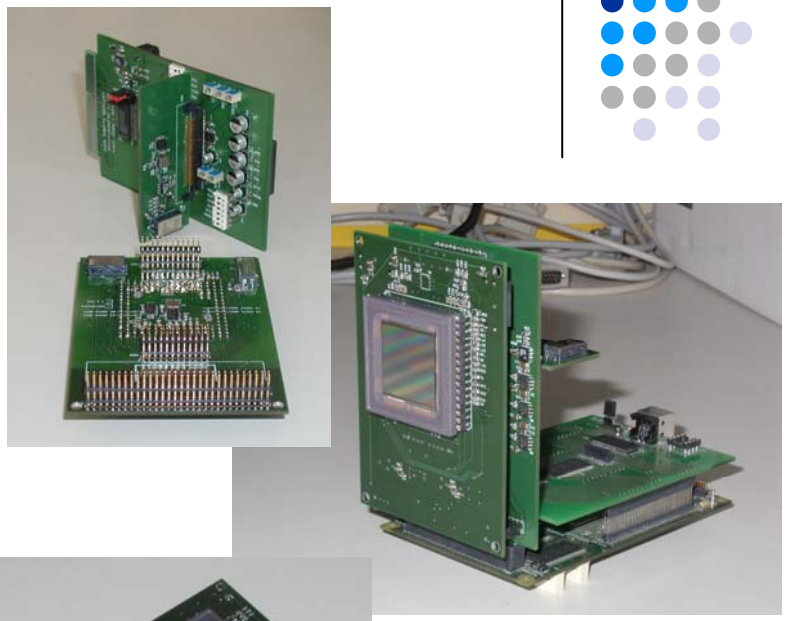
# SeeMOS: FPGA-based smart cam

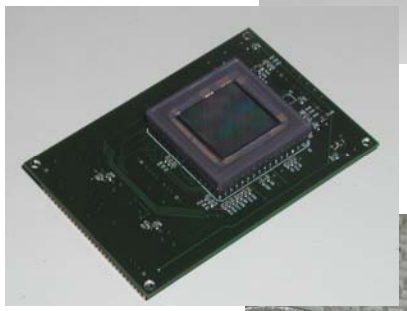


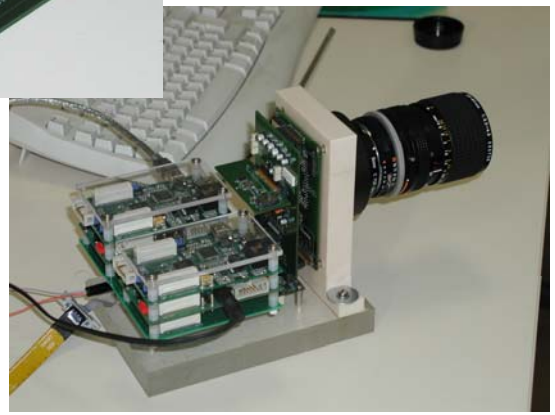

http://wwwlasmea.univ-bpclermont.fr/Personnel/Francois.Berry/seemos.htm

# SeeMOS: FPGA-based smart cam





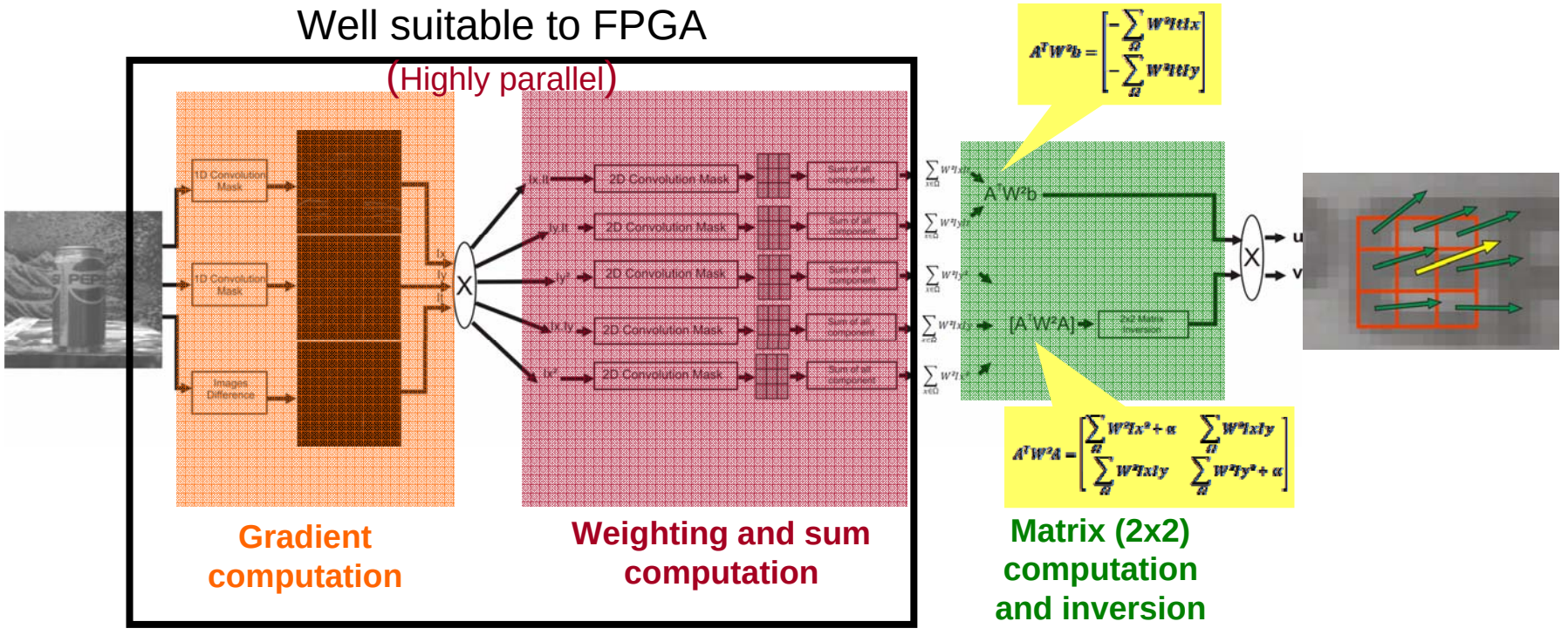


# SeeMOS: FPGA-based smart cam


**Example: Optical flow extraction (Lucas & Kanade)**

- **Local method which is only valid for small motions**
- **Gradient-based method, velocity is computed from first-order derivatives of image brightness using the motion constraint equation.**

Well suitable to FPGA
(Highly parallel)



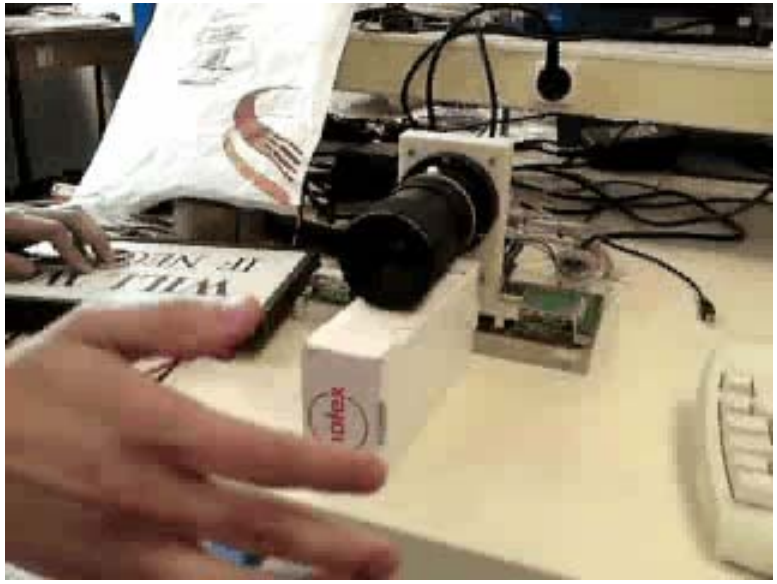

**Gradient computation**

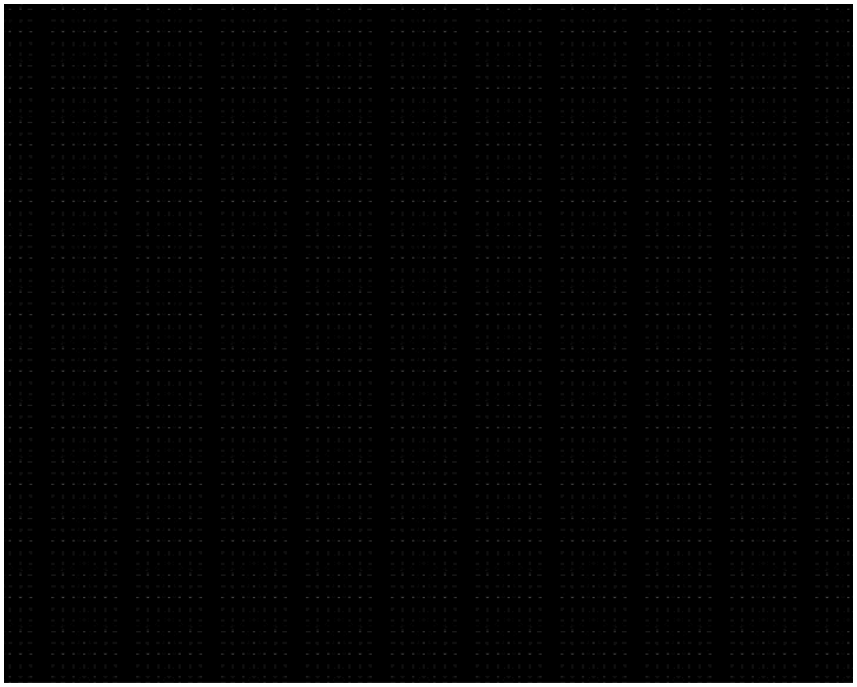
**Weighting and sum computation**

**Matrix (2x2) computation and inversion**

# SeeMOS: FPGA-based smart cam


## Example: Optical flow extraction (Lucas & Kanade)

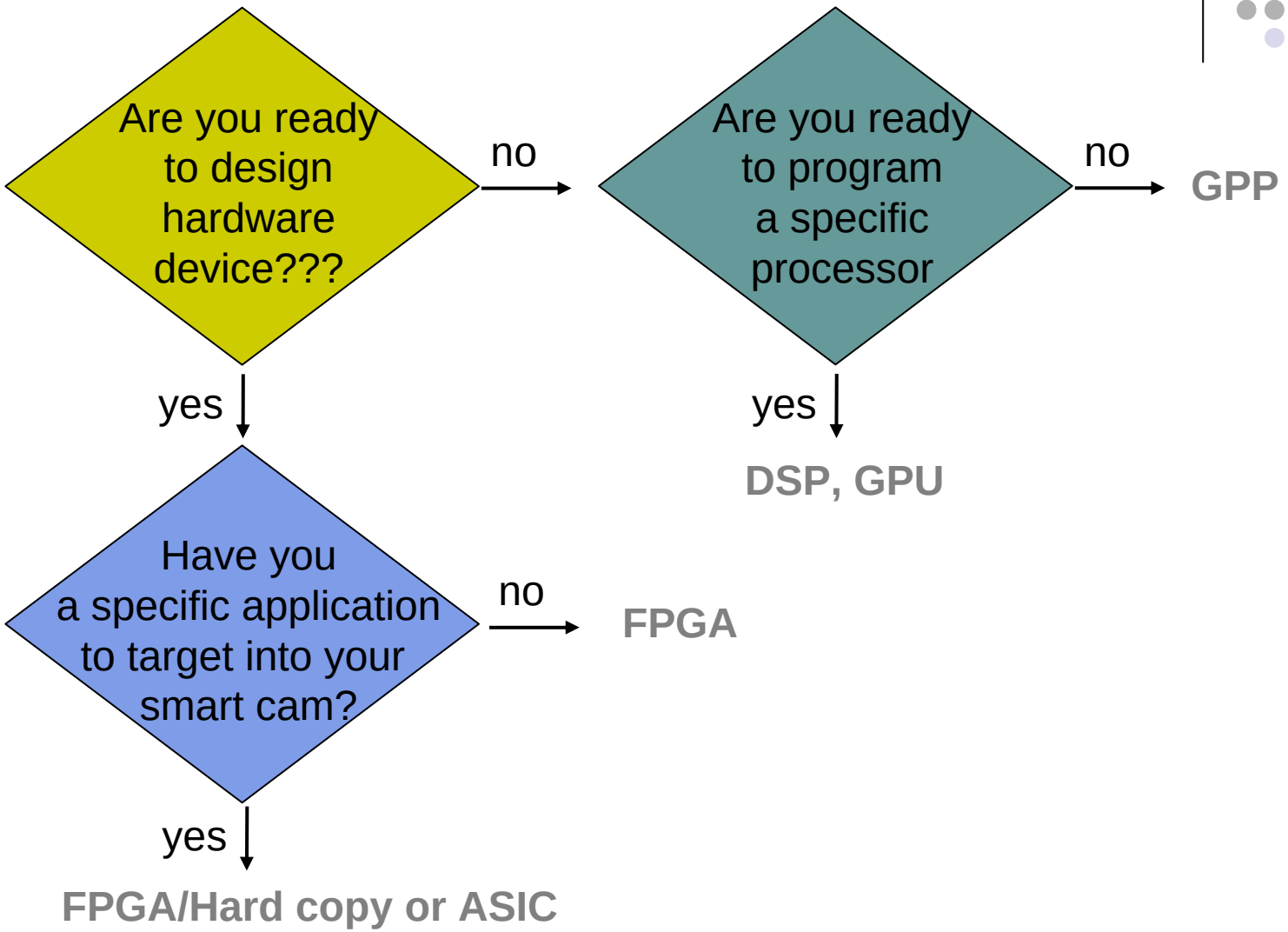




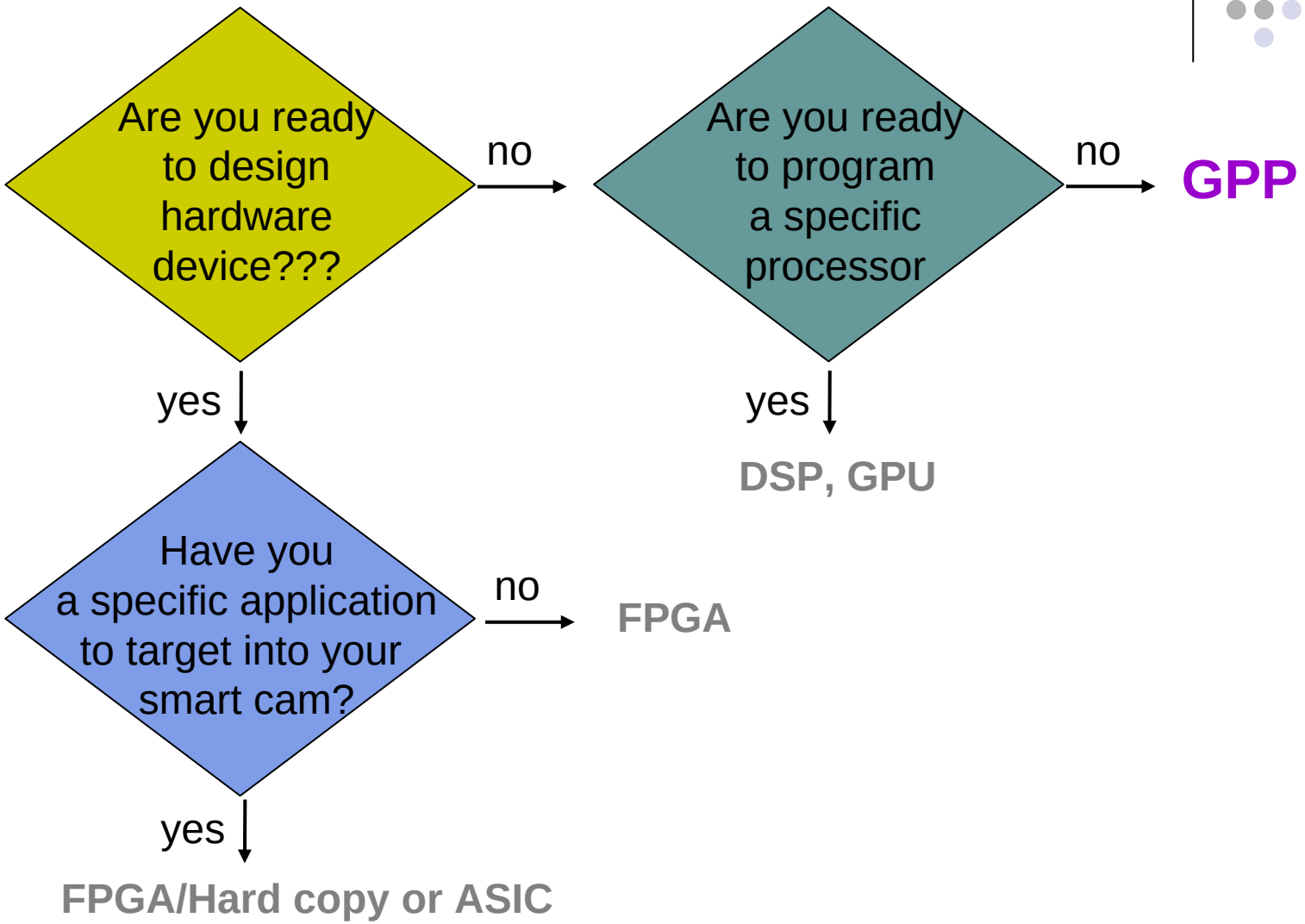
**FPGA used: Altera EP1S60**
57 120 Logic Element
5 215 kbits memory
144 DSP Blocks
782 I/O Pins

**Resolution : 800 x 600**
**Frame Rate : 30 FPS**
**Total Logic Elements used: 23645 (42% used)**
**Memory Bits used : 156 kbits (4% used)**
**DSP block 9-bits element used : 144 ( 100% used** )

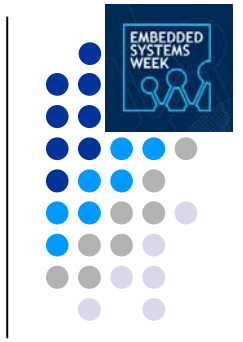
# Hardware considerations



# Hardware considerations

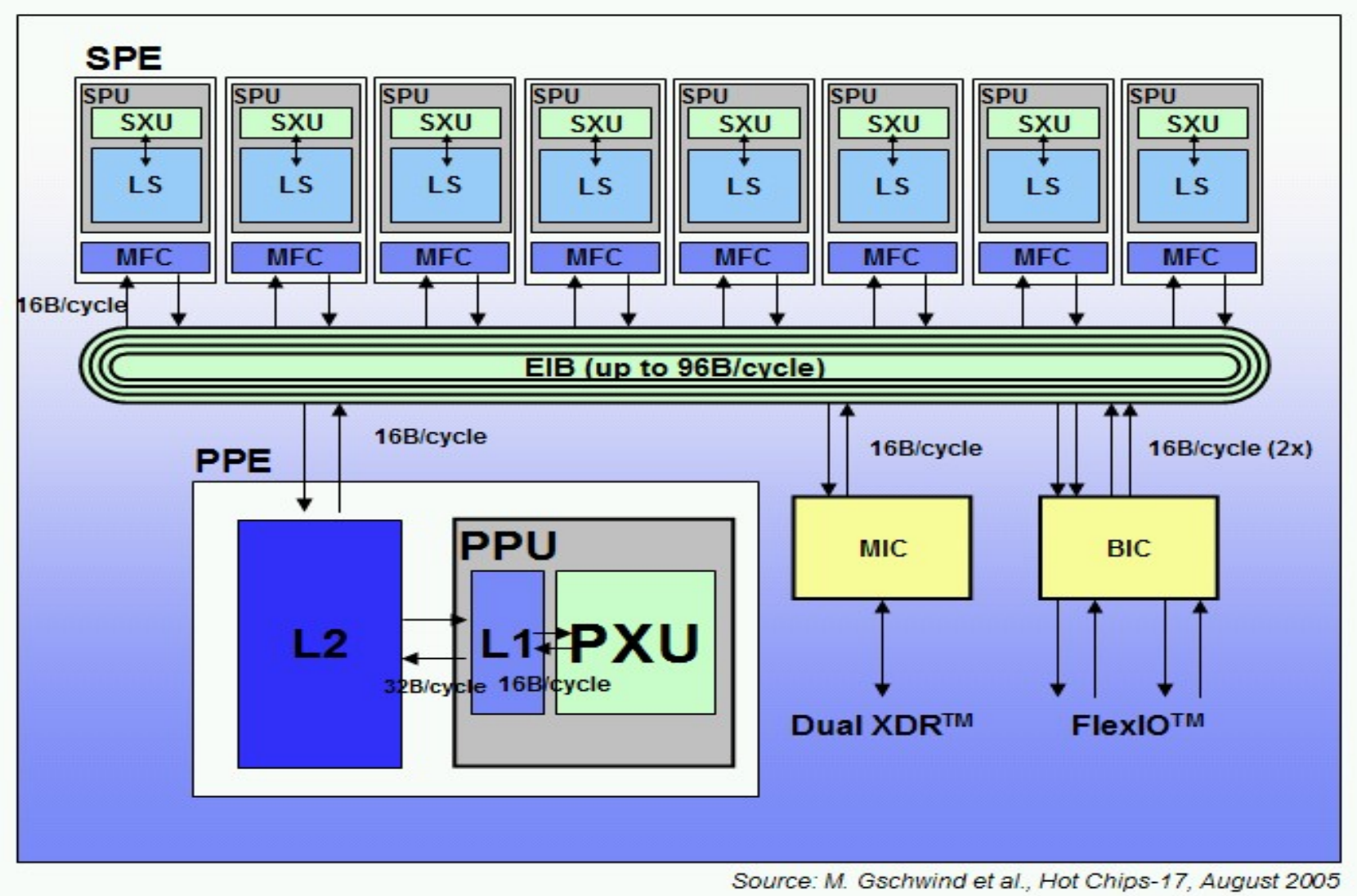


# The CELL Architecture



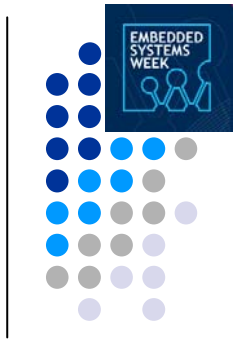

- Heterogeneous architecture:
  - 1 PPC core with Altivec+2 threads
  - 8 Synergetic Processing Units:
    - 256Kb scratchpad
    - Cacheless, branchless vector unit
- 200 GB/s Interconnexion DMA bus
- Memory Flow Controller
- Performance 250GFLOPS @ 70W

# The CELL Processor




*Source: M. Gschwind et al., Hot Chips-17, August 2005*

# Software Support

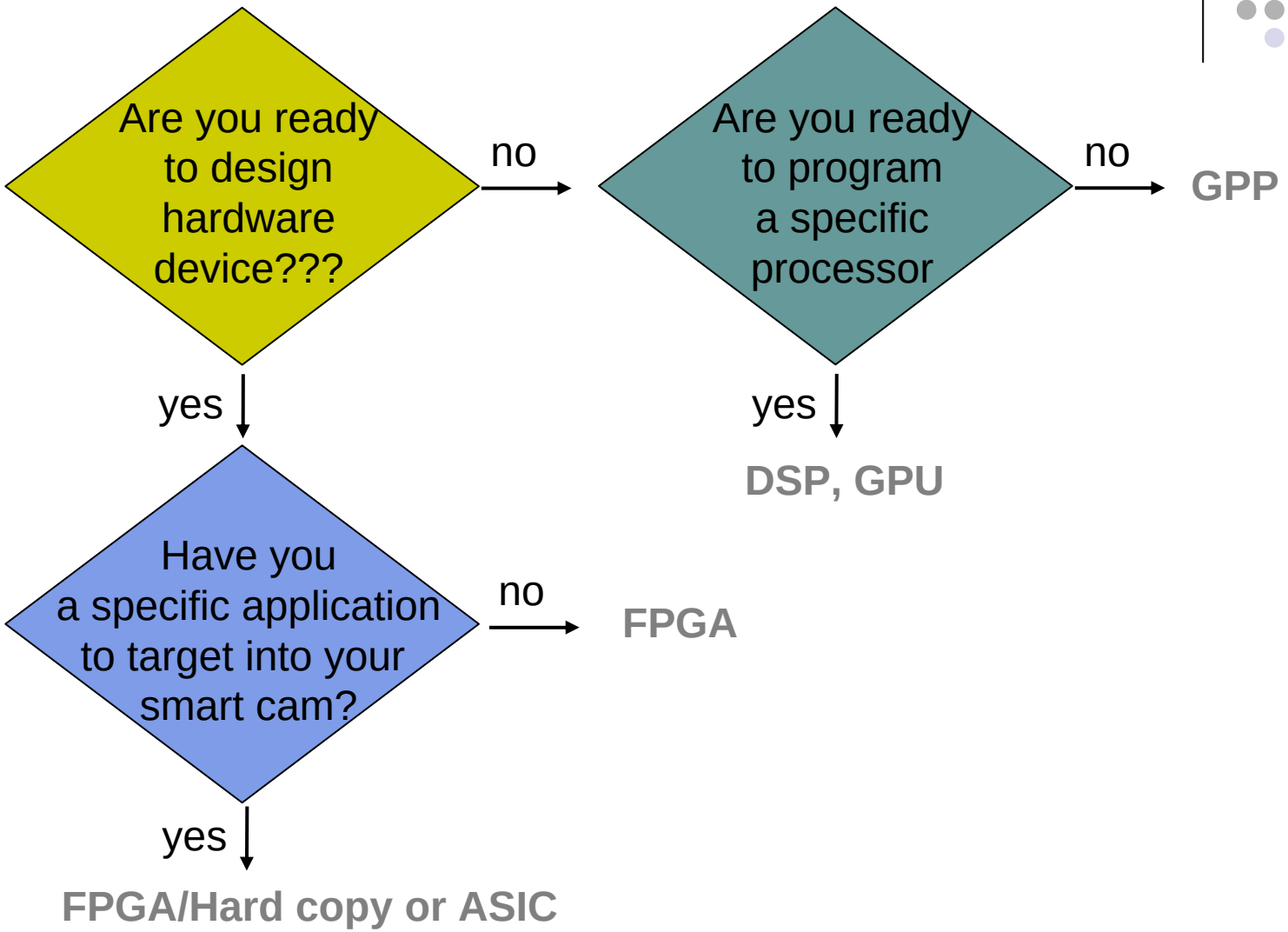



- Multi-sources cell-gcc :
  - AltiVec support
  - SPE are used via pthread
- Single Source xlc compiler:
  - SPE are used via OpenMP
  - No AltiVec support
  - No C++ support

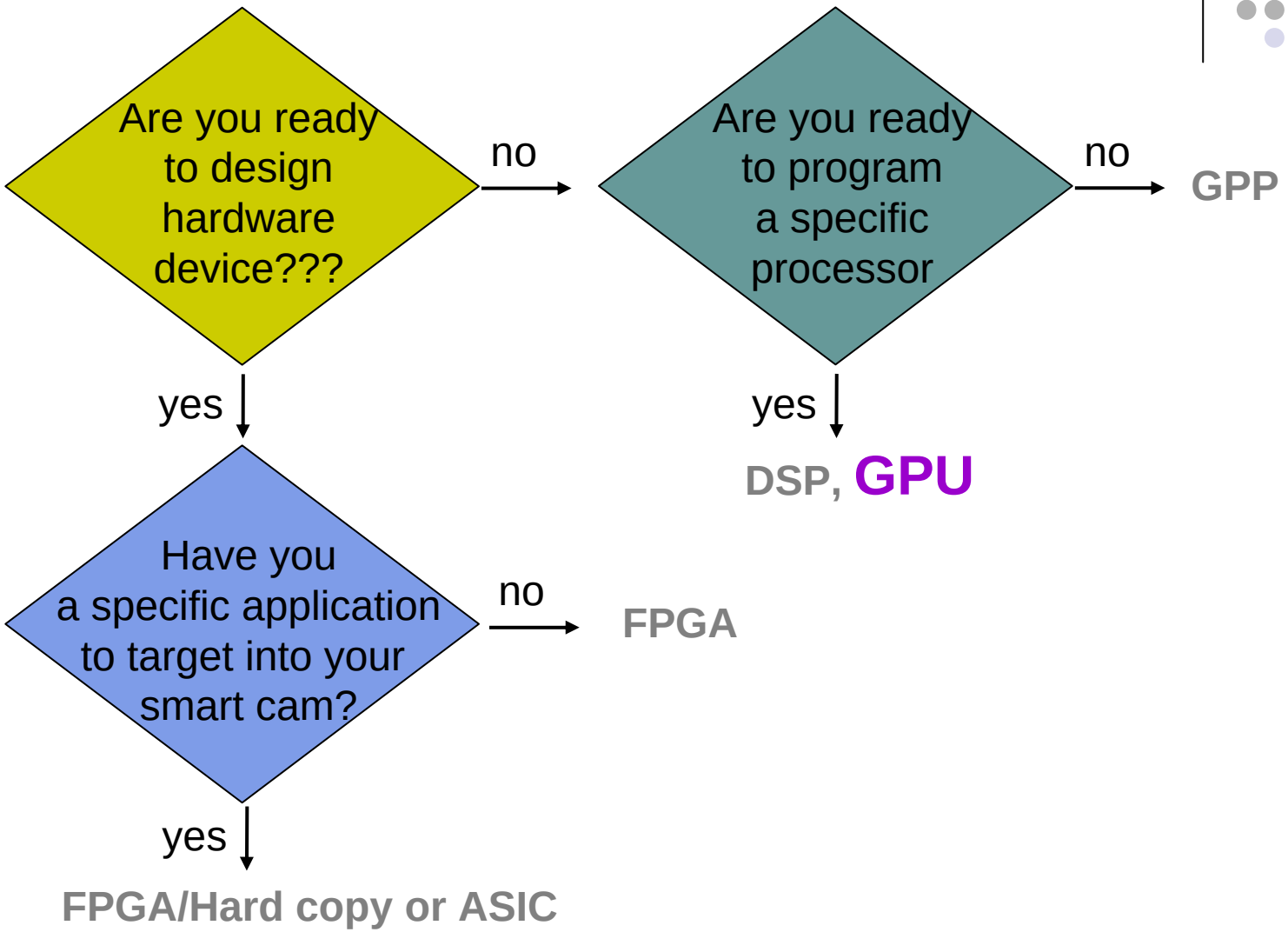
# Benchmarks Results


| | Fréquence | Taille | *cpp* architecture | $cpp_1$ | Temps(ms) | FPS |
|---|---|---|---|---|---|---|
| Power PC G4 | 1 GHz | 256×256 | 3.25 | 52 | 0.2 | 4695 |
| Power PC G4 | 1 GHz | 512×512 | 12.25 | 196 | 3.2 | 311 |
| Power PC G5 | 2.5 GHz | 256×256 | 1.38 | 22 | 0.04 | 27642 |
| Power PC G5 | 2.5 GHz | 512×512 | 4.141 | 66 | 0.4 | 2300 |
| Maille Associative | 500 MHz | 256×256 | 35 | 35 | 0.0046 | 232558 |
| Maille Associative | 500 MHz | 512×512 | 35 | 35 | 0.0046 | 232558 |
| GeForce 8800Ultra | 1.5 GHz | 256×256 | 6.03 | 772 | 0.26 | 3826 |
| GeForce 8800Ultra | 1.5 GHz | 512×512 | 3.07 | 393 | 0.53 | 1879 |
| PowerXCell8i | 3.2 GHz | 256×256 | 2.32 | 18 | 0.047 | 21046 |
| PowerXCell8i | 3.2 GHz | 512×512 | 1.75 | 14 | 0.143 | 6975 |

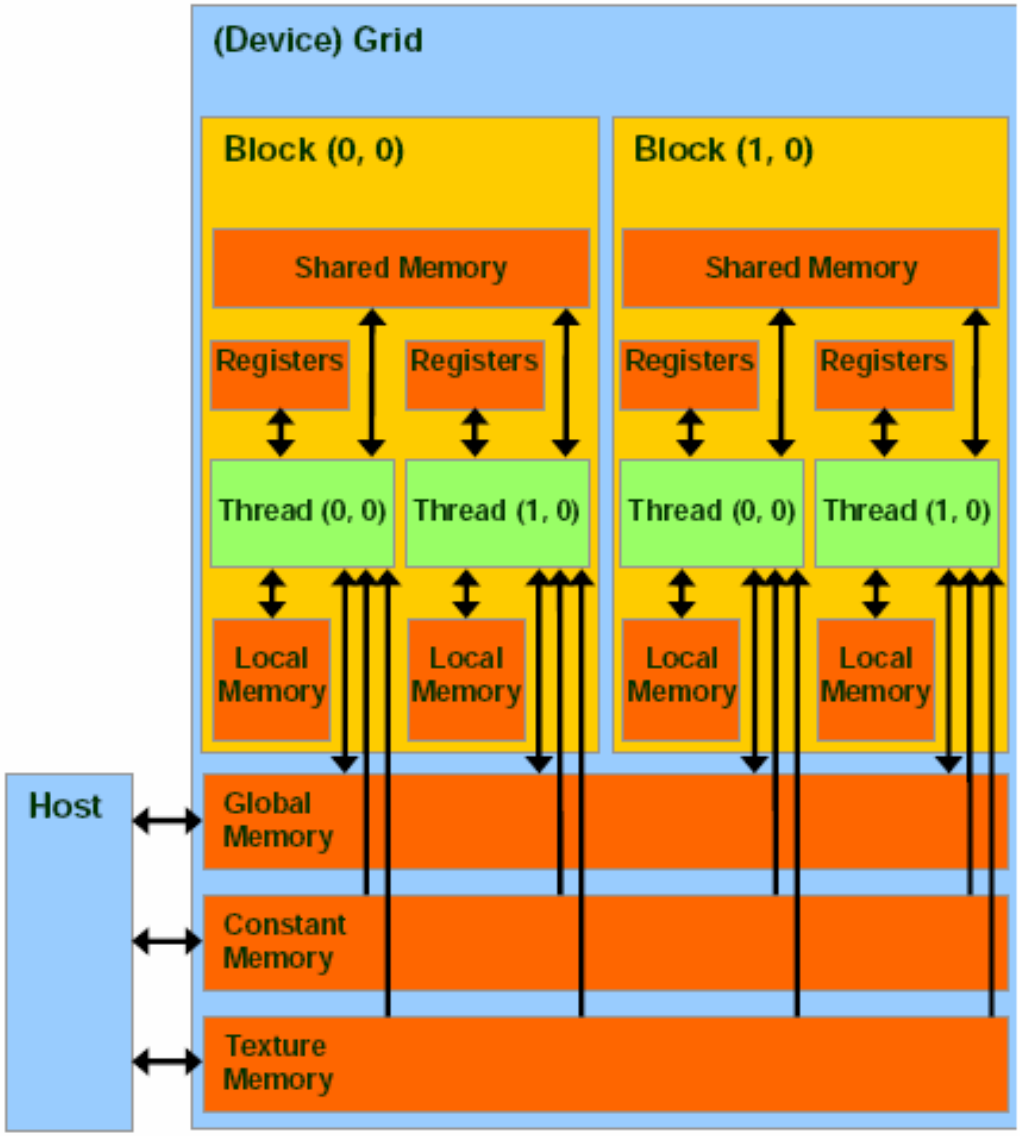
# Hardware considerations



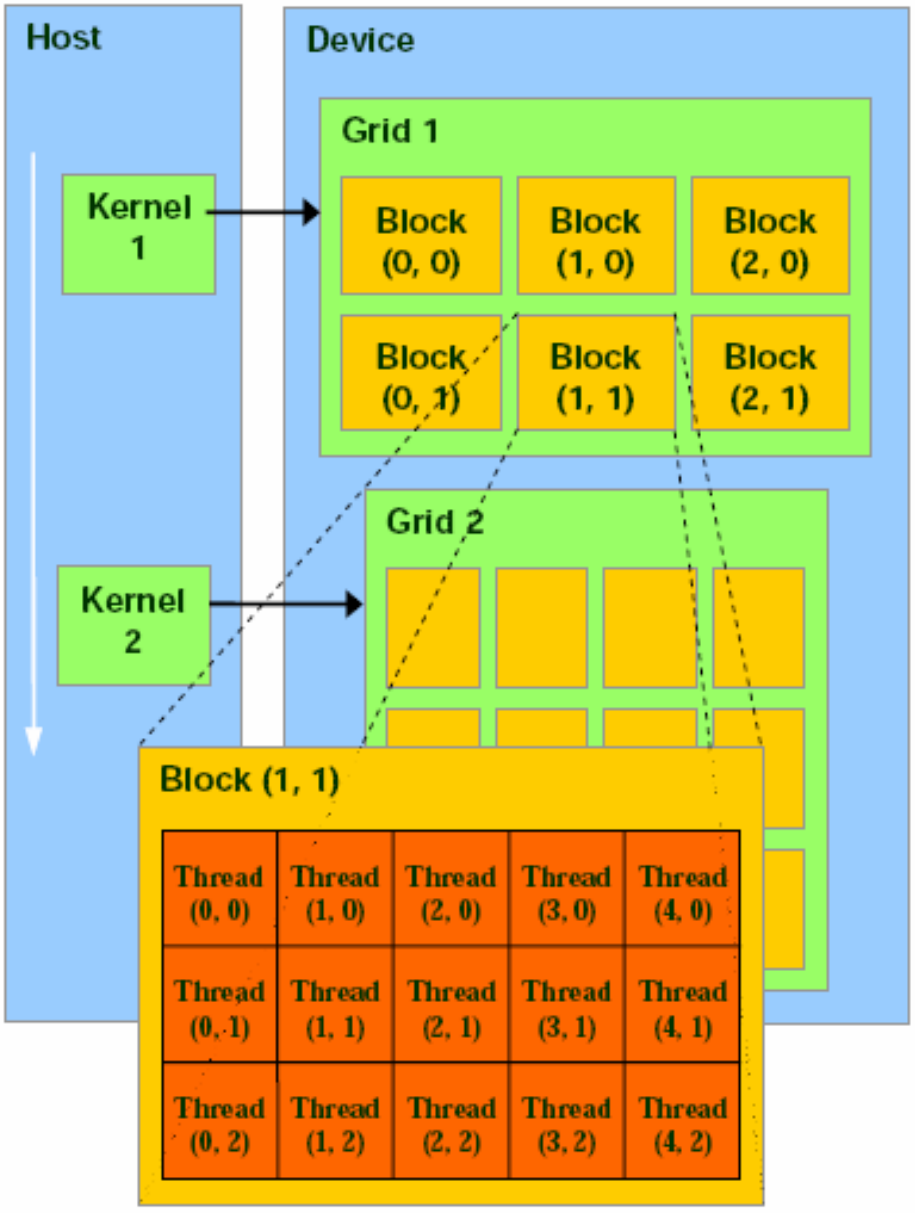
# Hardware considerations



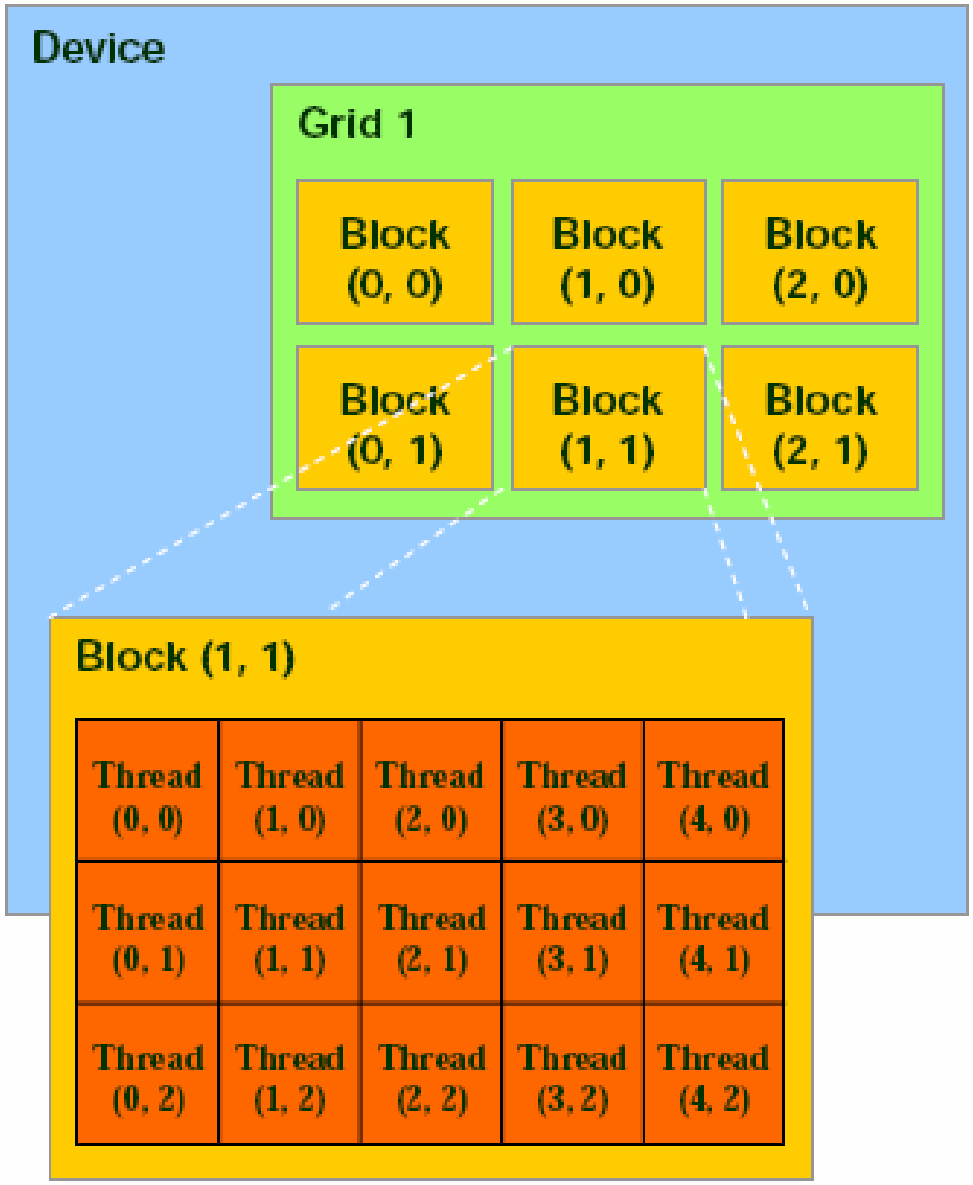
# NVIDIA GPU Architecture



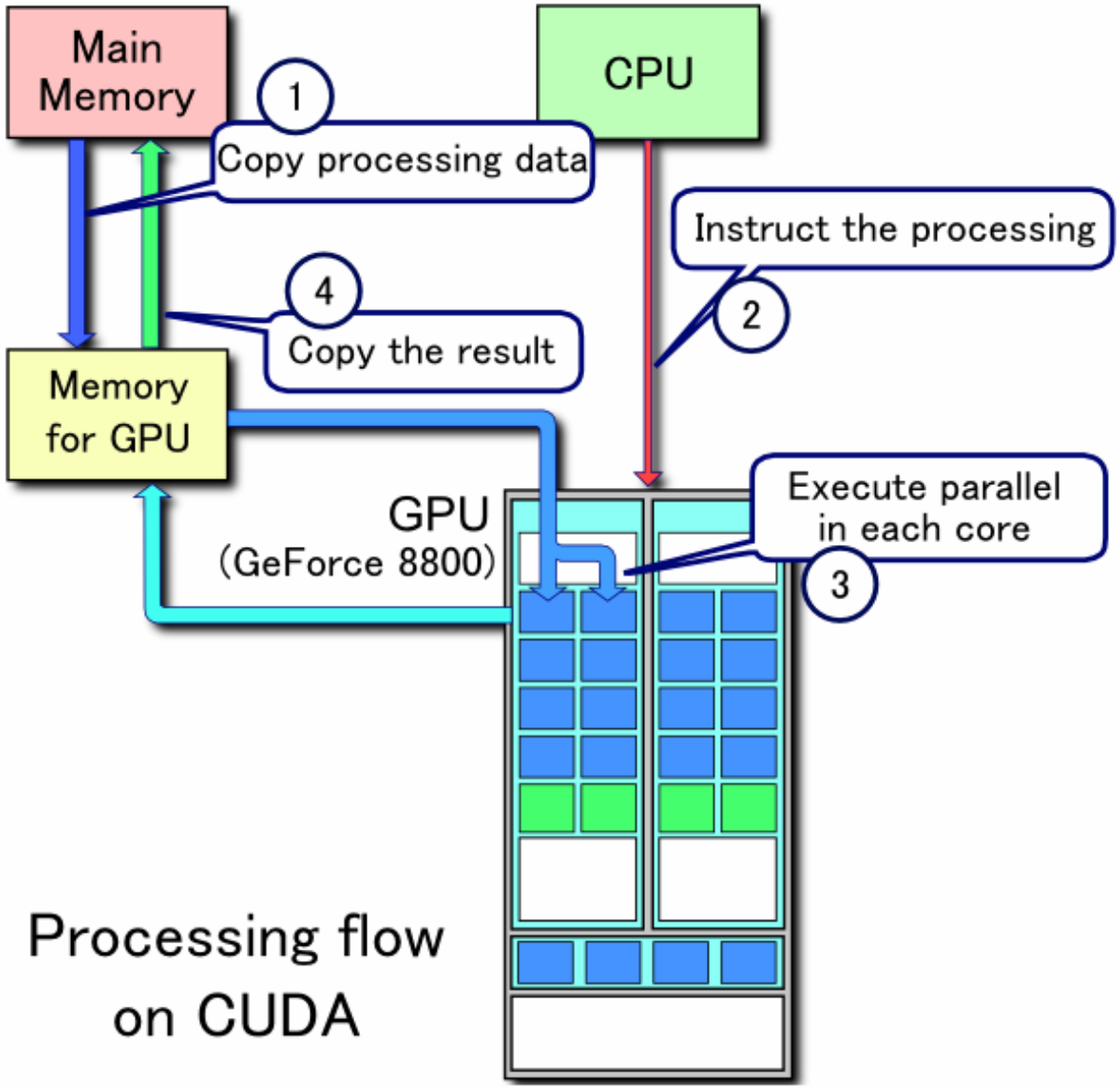
# NVIDIA GPU Architecture



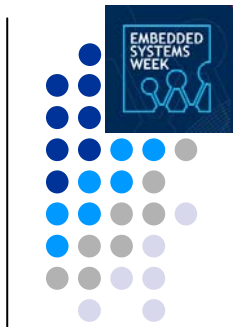
# NVIDIA GPU Architecture



# Execution Model

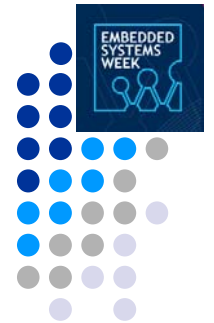


# Image Processing on accelerators



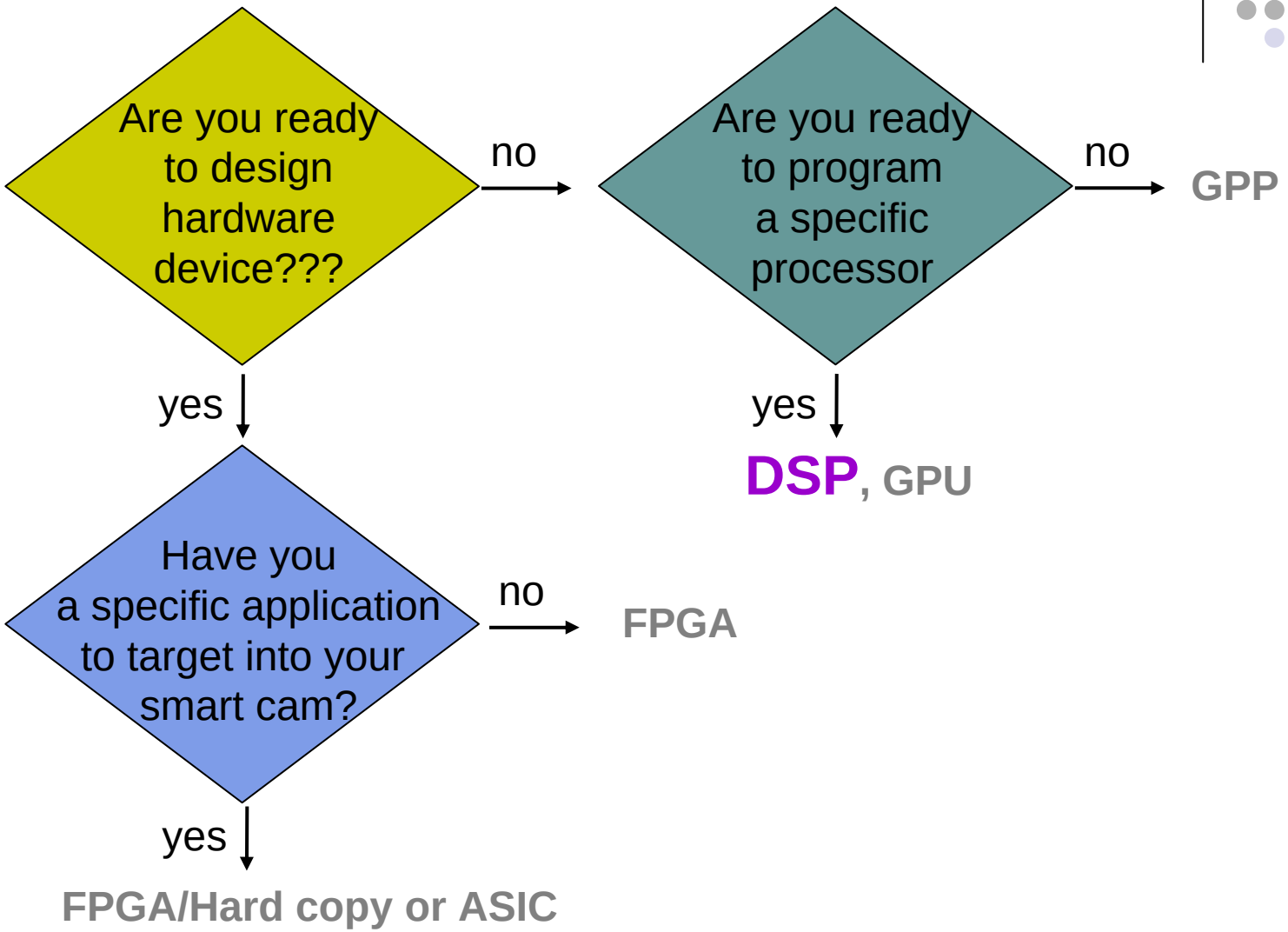

- Objectives:
  - Benchmark CELL & GPU in IP context
  - Find good candidates applications
  - Find architectural default
- Architectures:
  - CELL Blade
  - Gforce 880 Ultra
  - PowerPC
- (For)All results thanks to L. Lacassagne (IEF – U. Paris XI)

# Image Processing on accelerators



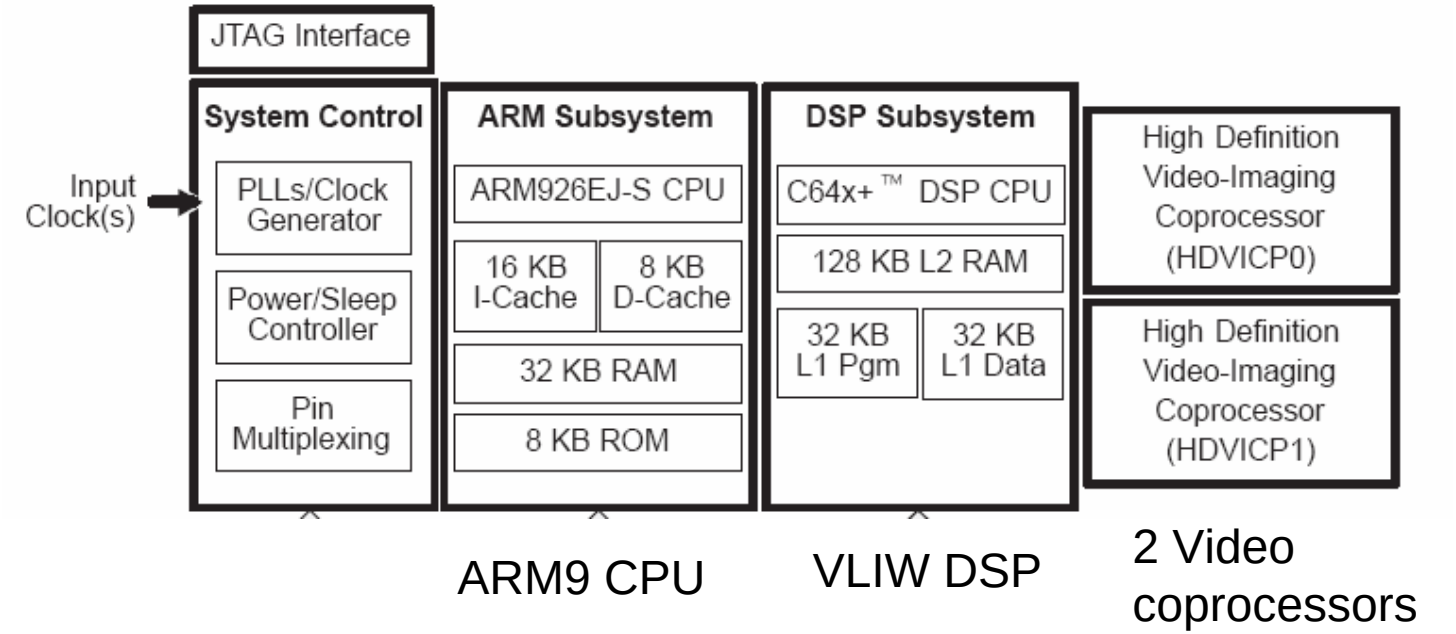

- Sigma-Delta Motion Detection
  - Proposed by A. Manzanera (2004)
  - Uses local variance to enhance image difference
  - Better detection
  - Less over-segmentation and false negative

# Hardware considerations

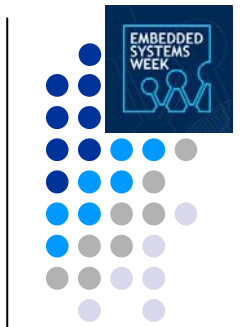


# Digital Signal Processors

- Transition from scalar processor (single MAC unit) to heterogeneous multi-cores
- Example: DaVinci platform form Texas Instruments1

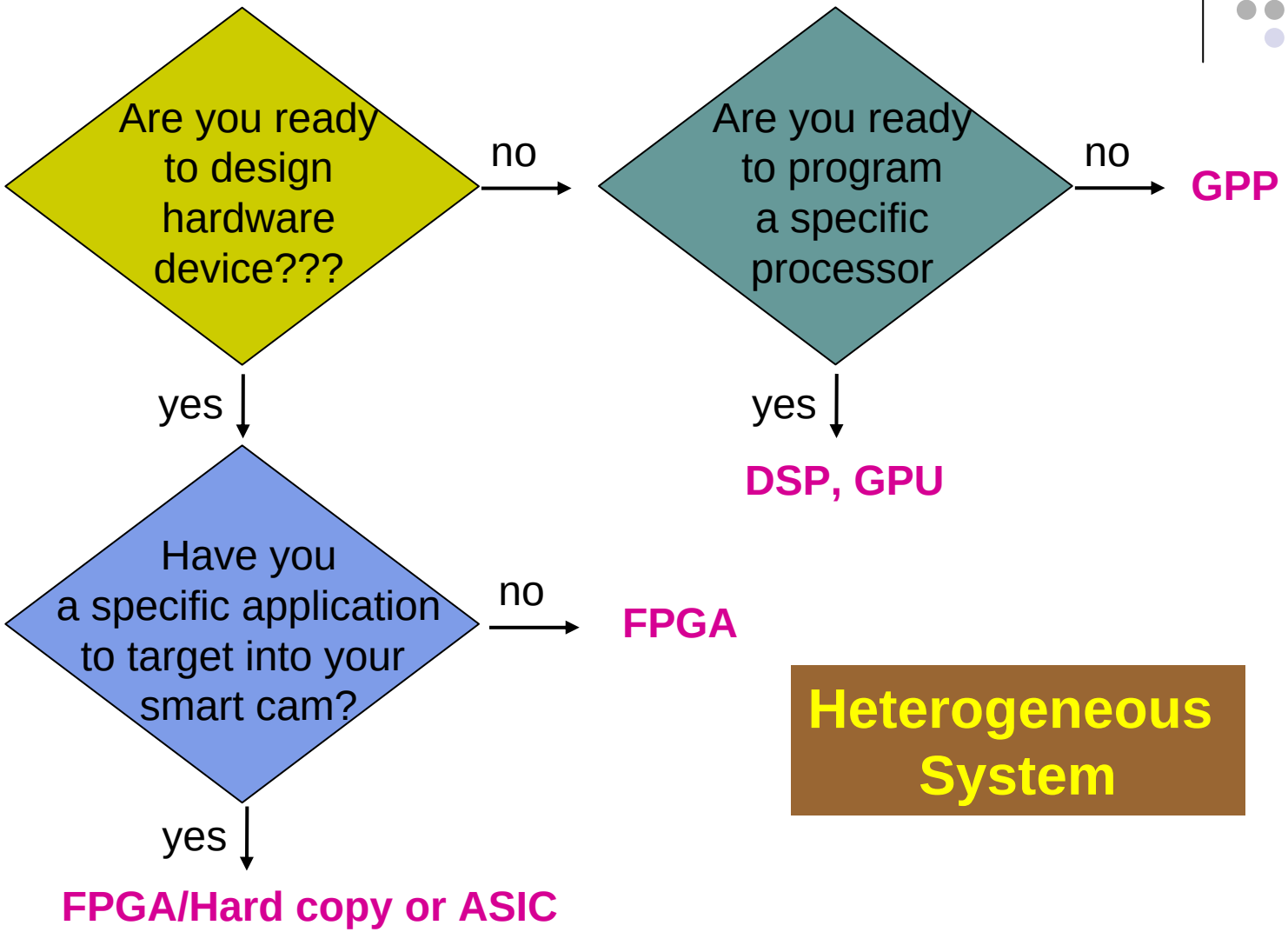


# Digital Signal Processors (2)



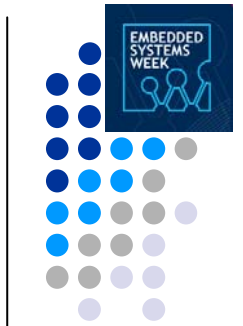

- Some design considerations
  - Fixed-point vs. floating-point DSP
  - Memory hierarchy and available memory
  - Interfaces (to sensor or other cameras)
  - Low-power
- Software development
  - Assembler (rarely)
  - High-level languages C/C++  Matlab/simulink (standard)
  - Operating system (eg. Linux  DaVinci)
  - Extensive usage of SW libraries/drivers

# Hardware considerations




**Heterogeneous System**

# Heterogeneous system: Processor + FPGA




## What ?

**An heterogeneous system comprises heterogeneous processing units such ASIC, FPGA, DSP, CPU,…**

**The heterogeneity is often due to assembly of programmable and configurable units.**

## Why ?

**Low-level algorithms such as filter kernels (image denoising, enhancement, pixel matching,…) which can be run in parallel (where the pixels are processed on-the-fly) fit best to FPGA.**

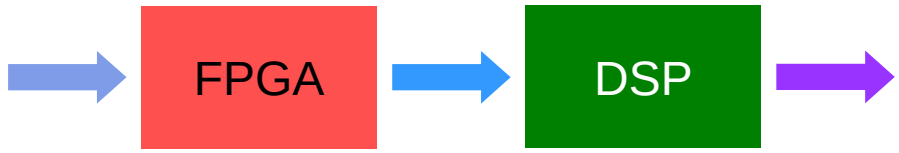
**High-level algorithms which consists of complex branches (if, else) or control loops (for, while,…) and operate on data widths which are a multiple of 8 bits are preferred on DSP for implementation**

**Powerful image librairies which are available for Mobile PC platforms enable short design cycles.**
**However, when compared to FPGA and DSP implementation, the performance is rarely best.**

*Source : Benchmarks of Low-Level Vision Algorithms for DSP, FPGA and Mobile PC Processors - D. Baumgartner, P. Roessler, W. Kubinger, C. Zinner and K. Ambrosch – Embedded Computer vision 2009 – APR – Springer.*

# Heterogeneous system: Processor + FPGA

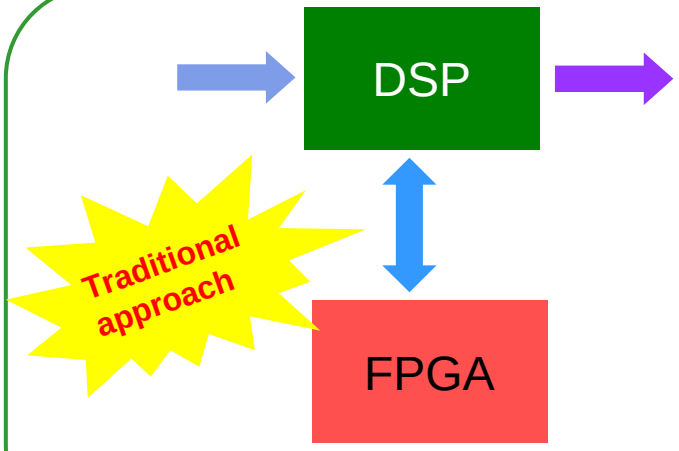
**3 Possibilities:**




**Dataflow structure where:**

- **FPGA performs low-level task**
- **DSP execute high-level algorithms**

**Sequential approach**




**DSP-based structure where:**

- **DSP manage the processing**
- **FPGA is a coprocessor**

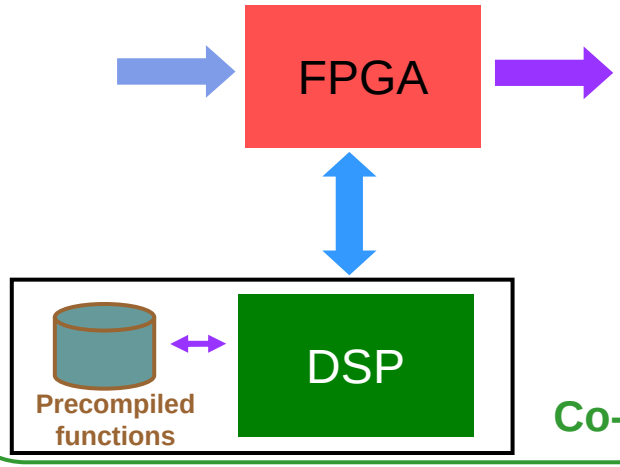
•To realize the benefits of FPGA co-processing, the **data path must be re-architectured** and implemented in a parallel manner, not in a serial or sequential DSP processor coding style.

•Decide how the DSP algorithms will be partitioned to DSP processor/FPGA co-processor architecture

**Co-processing approach**

# Heterogeneous system: Processor + FPGA

**3 Possibilities:**



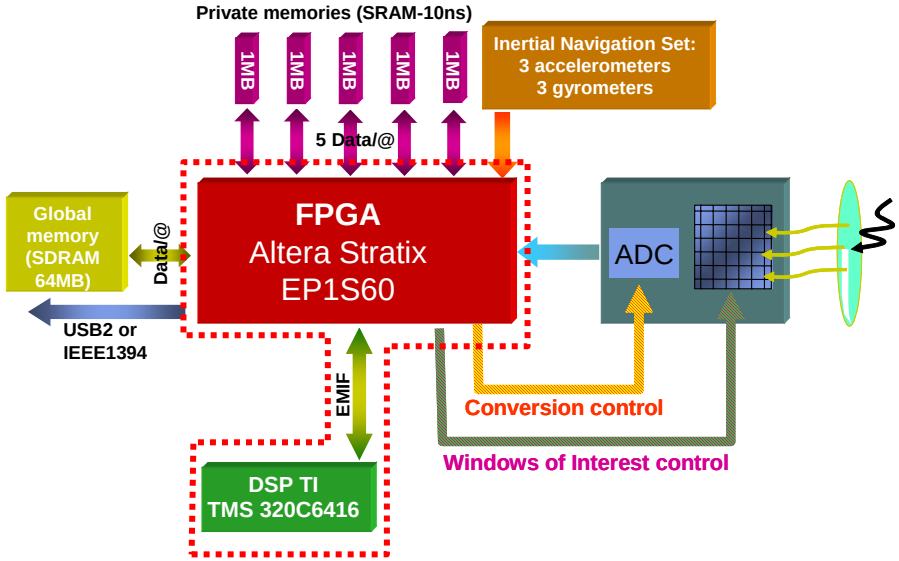

**FPGA-based structure where:**

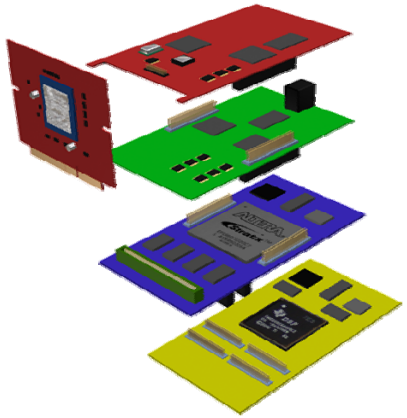
- **DSP is a high-level coprocessor**
- **FPGA performs global system management**

• Keeps versatility of an FPGA-based design

• The co-processor is based on DSP and a library of precompiled functions.

**Co-processing approach**



**Final structure of SeeMOS**

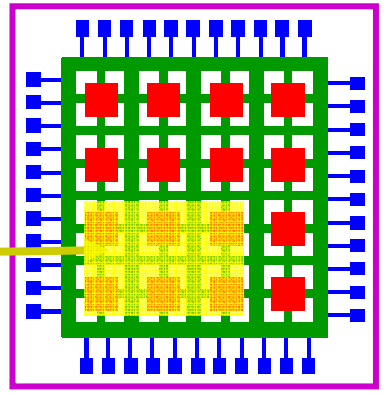
http://wwwlasmea.univ-bpclermont.fr/Personnel/Francois.Berry/seemos.htm

# Heterogeneous system: FPGA and embedded processors

## Soft Core vs Hard Core processor

A soft-core processor is synthesized onto the FPGA's fabric, just like any others circuits.

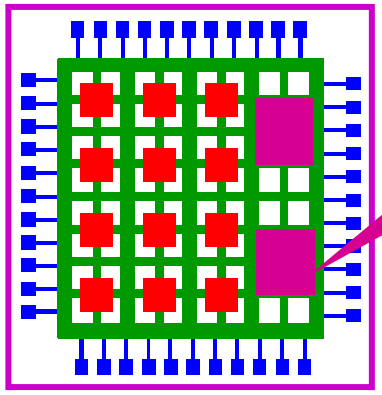
Processor HDL description



Nios II from Altera
Microlaze from Xilinx



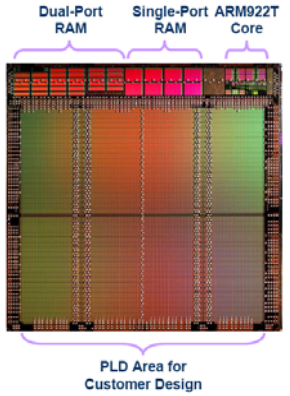
A hard-core processor is laid out on the chip next to the FPGA's configurable logic fabric



Processor is in silicon!

**Arm on Excalibur (Altera)**

**PowerPC on Virtex (Xilinx)**



# FPGA and embedded processors

## Soft Core vs Hard Core processor

FPGA platforms with hard-core processors offer better performance in terms of operating frequencies and reduced energy consumption, while soft-core processors offer a higher degree of configurability, implementation options and lower cost per device.

Actually, hard cores tend to disappear
(Excalibur has stopped since 5 years, No HC in virtex 6)

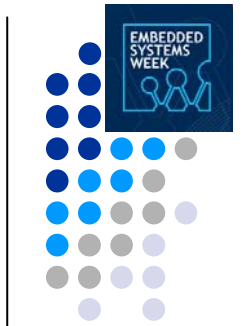
**Soft-core processors advantages:**

- Utilizing standard mass-produced
- Enabling a custom number of processor cores

**Soft core processors disadvantages:**

- Reduced processor performances
- Higher power consumption
- Larger size.

# Embedded Computer Vision

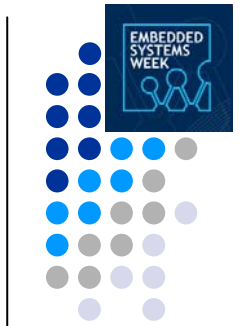



- Embedded computer vision system

- **Hardware considerations**
  - **Programmable devices (FPGA, CPLD)**
  - **Signal Processor (DSP)**
  - **General purpose processor (Cell)**
  - **Graphics Processor Unit**

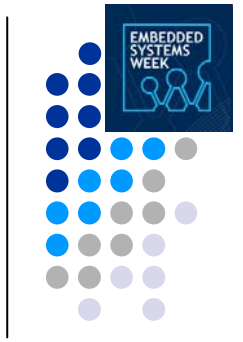
- Few ways for programming
  - SOPC approach
  - HDL versus High level synthesis language

# Embedded Computer Vision




- Embedded computer vision system
- Hardware considerations
  - Programmable devices (FPGA, CPLD)
  - Signal Processor (DSP)
  - General purpose processor (Cell)
  - Graphics Processor Unit
- **Few ways for programming**
  - **SOPC approach**
  - **HDL versus High level synthesis language**

# Introduction




**Smart camera = Vision task + Suitable Hardware + "Embedded" constraints**

Architecture design

Heterogeneous system based on:
- Hardware (DSP, FPGA, GPP, ASIC,…)
- Embedded software modules (embedded OS)

Memory sizing

Response time

Power consumption

Complex Design (Footprint)

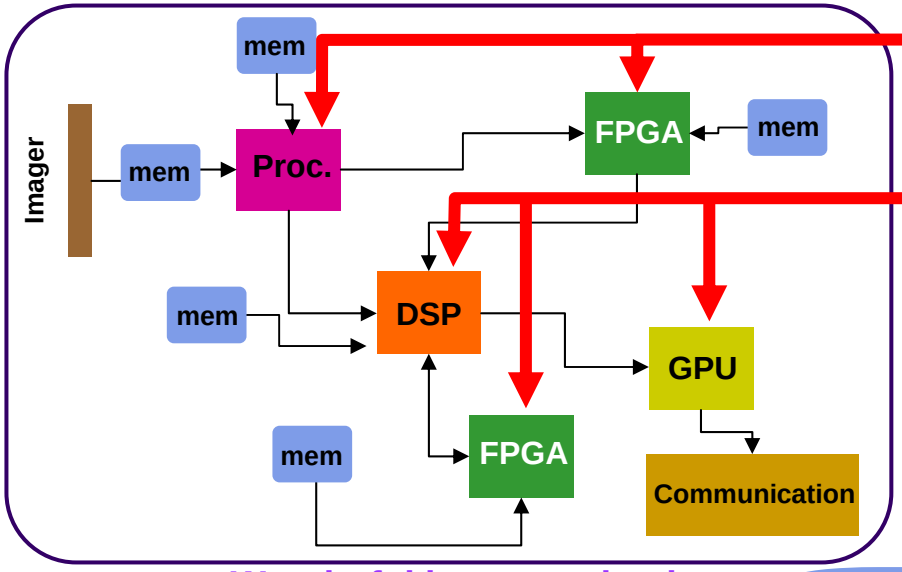
Algorithms selection/development

- High Computational demands
- Large volume of data

Programming tools

Hardware constraints (Memory, Low-level programming languages,…)

# Methodology for Embedded Computer Vision System




Algorithms

**Wonderful but apocalyptic Smart Cam architecture**

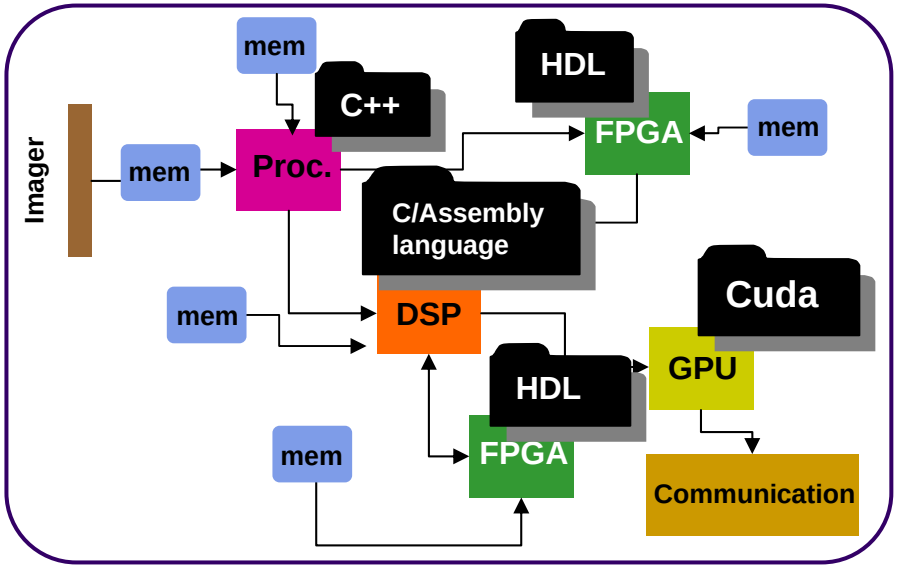
Is the end-user HDL-aware???

**Open Problems:**

- How to program the FPGA?
- How to manage the glue (interface, memory, communication, sensors,…)?
- How to well-fit the algorithms on the hardware?

Traditional partitioning problem

# Methodology for Embedded Computer Vision System




Nice Developer

An heterogeneous hardware architecture

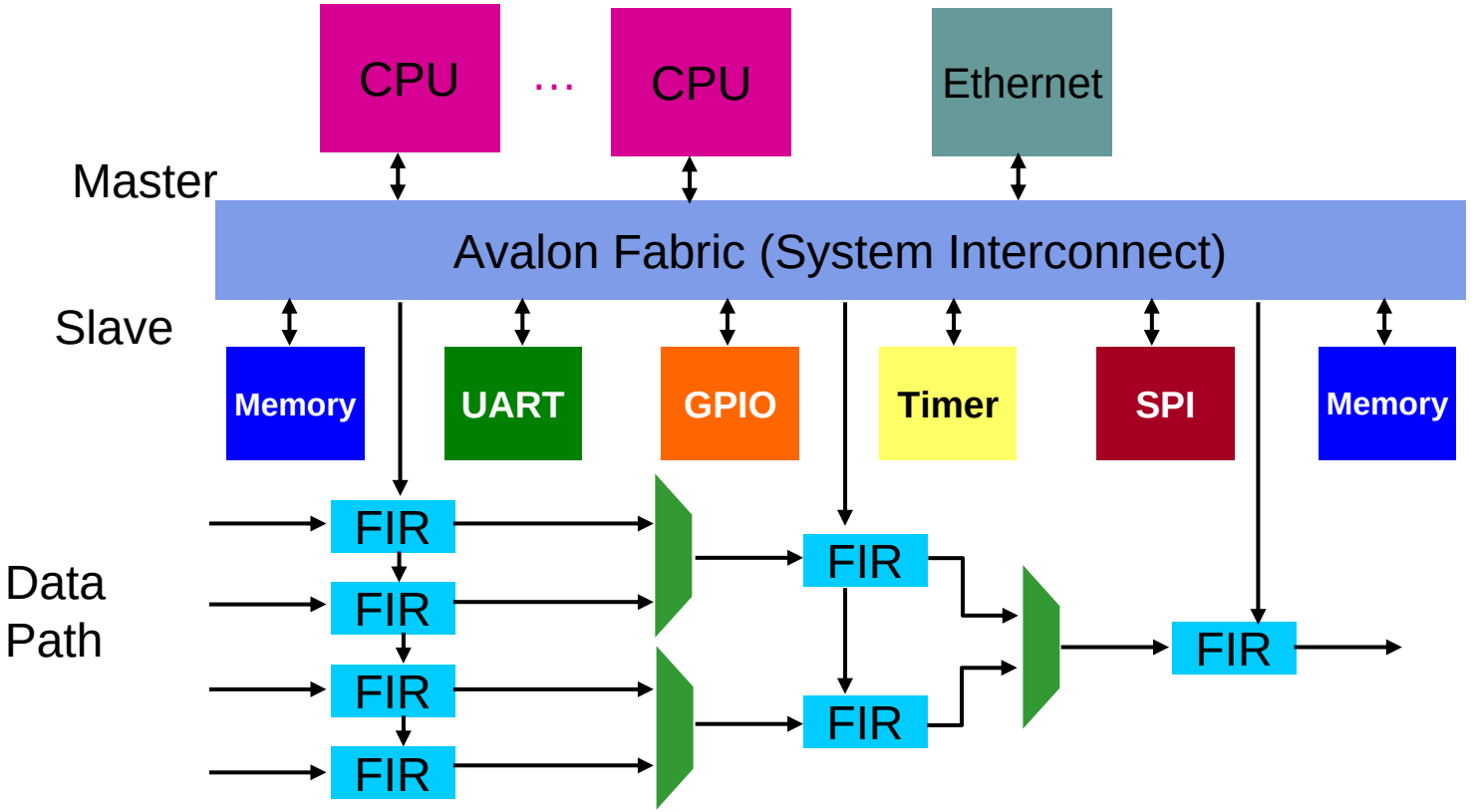
Host of programming tools and languages

# FPGA and embedded processors

## Soft Cores : SOPC approach

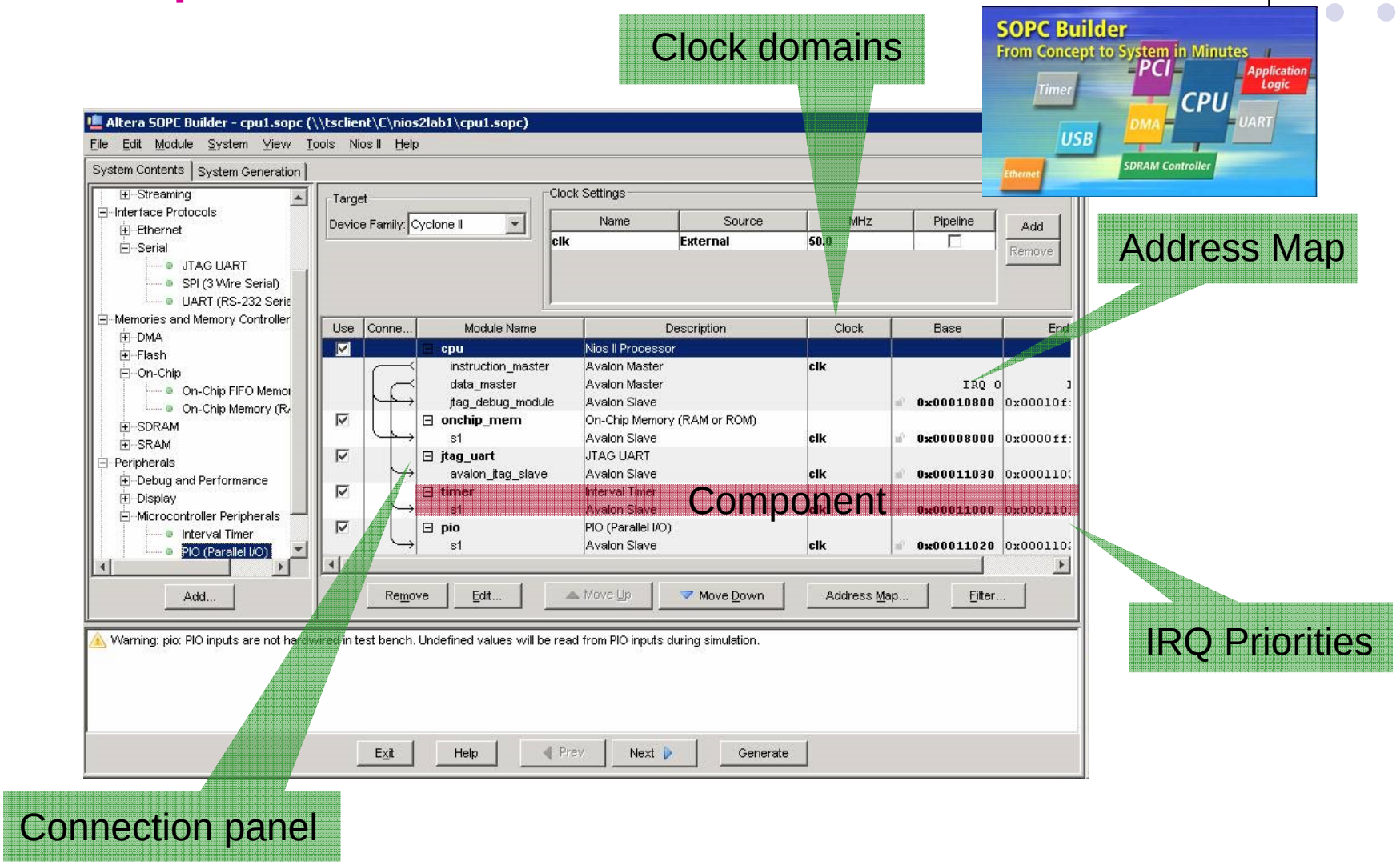
- SOPC means System On Programmable Chip
- In a spirit of reconfigurable device, it's a micro-controller-like approach.

***Example : SOPC Builder from Altera***

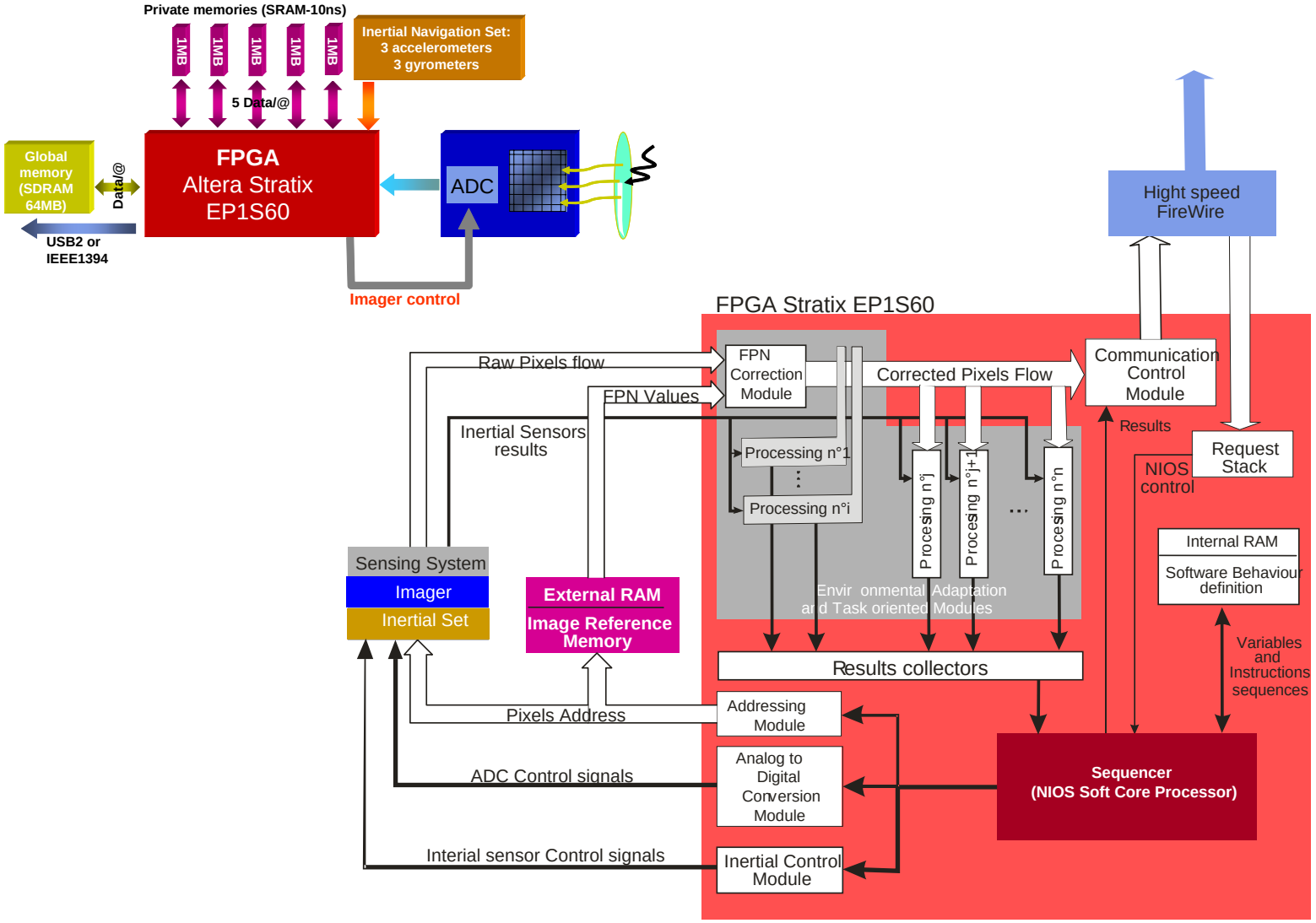


# FPGA and embedded processors

## Example: SOPC Builder from Altera

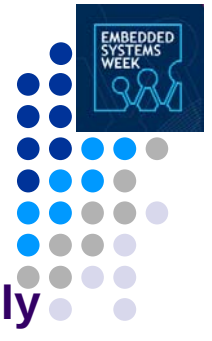


# FPGA and embedded processors




**SOPC-based methodology (SeeMOS 2004)**

# HDL vs High level synthesis language




**Processor (DSP, GPU, GPP,…)**

**Programming language: C-like, assembly**

**GUI: Simulink, Labview,…**

**Programmable Logic (FPGA,…)** → **Description language: VHDL, Verilog,…**

## What is a Hardware Description Language (HDL)?

- A textual approach to describe electrical circuits.
- Not limited to circuit structure, can also describe temporal/operational behavior.
- Can be very efficient (RTL level), but not really easy to use!!

  - Thinking in hardware terms and not in algorithmic terms
  - Needs a very good knowledge of the low level hardware

# Example of VHDL

## Synthesis Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
```

Library declaration

```
entity counter is
Port
(
  clk        : in std_logic  ;
  rst        : in std_logic  ;
  output      : out std_logic_vector(2 downto 0)
 );
end counter;
```

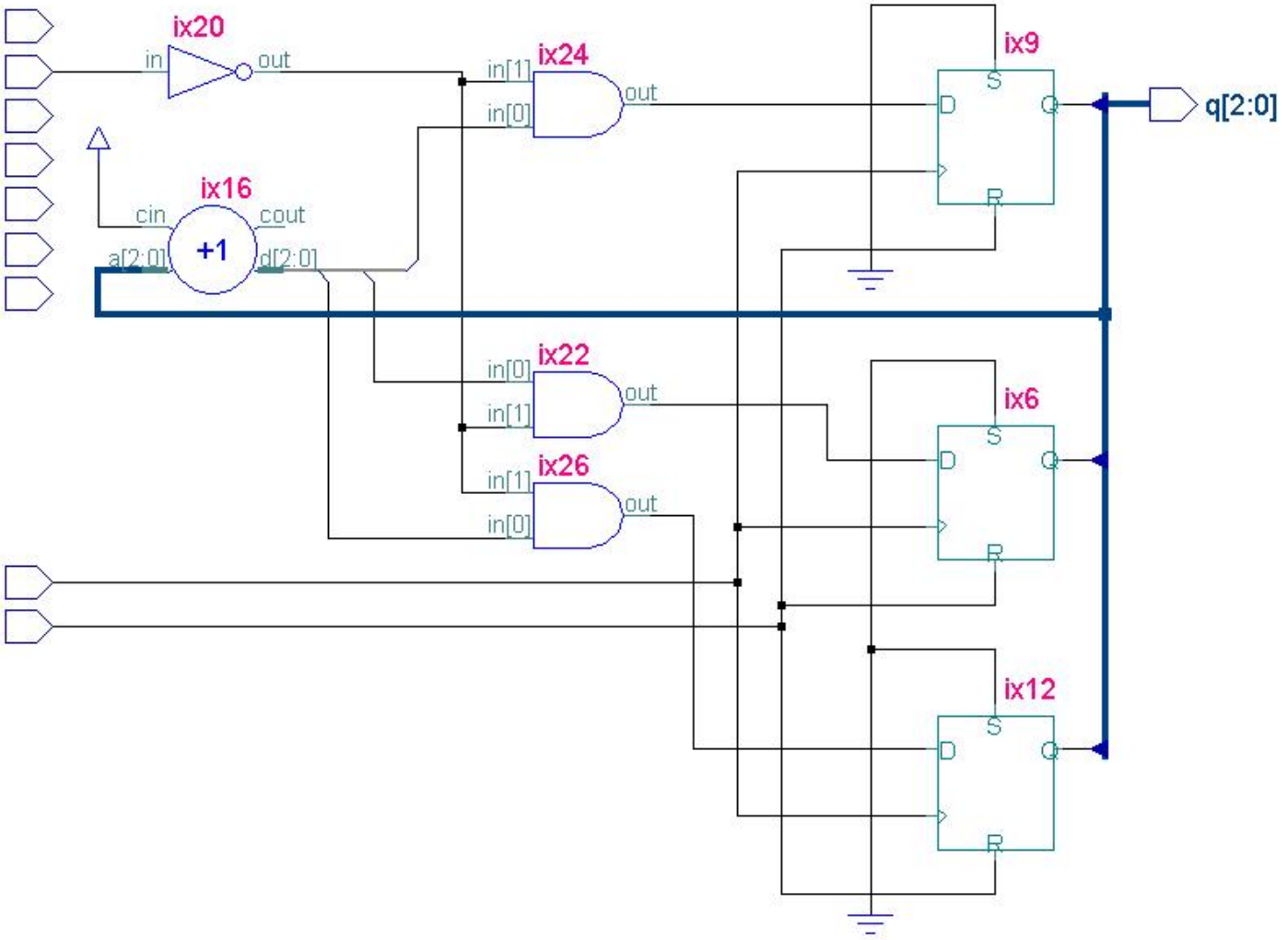
**Entity** - Specifies the interface of the underlying design with the external world. It basically defines the IO's of the system.

```
architecture counter_a of counter is
signal q :std_logic_vector(2 downto 0) ;
begin
process(clk,rst)
begin
if (rst='1')then
                    q<="000";
elsif (clk'event and clk='1') then
                    if (q="100")then q<="000";
                    else q<=q+'1';
                    end if;
end if;
end process;
output<=q;
end counter_a;
```

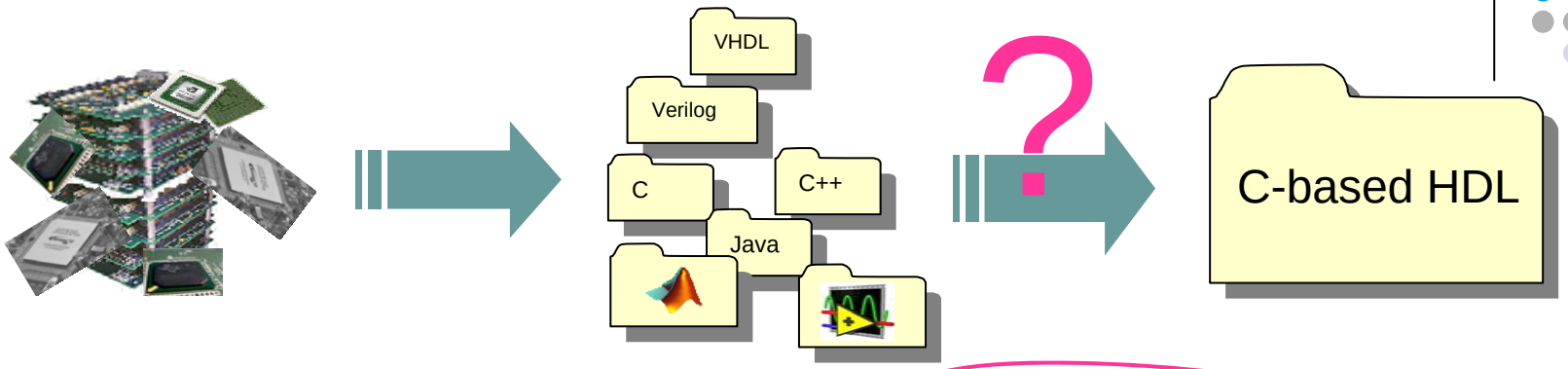
**Architecture** - Describes a design's behavior and functionality. It has no existence without an entity.

# Example of VHDL

**Synthesizer output**



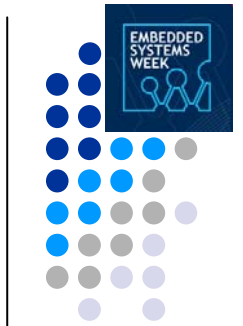
# HDL vs High level synthesis language




Ex: A Smart cam based on FPGA (with 1 soft core) + DSP = 1 HDL code + 2 C codes !!!!

- Simplify hardware/software partitioning by describing both using a single, C-based language.
- Work at varying levels of abstraction from the underlying hardware.
- Enhanced simulation and debugging performance.

# High level synthesis language Round Up

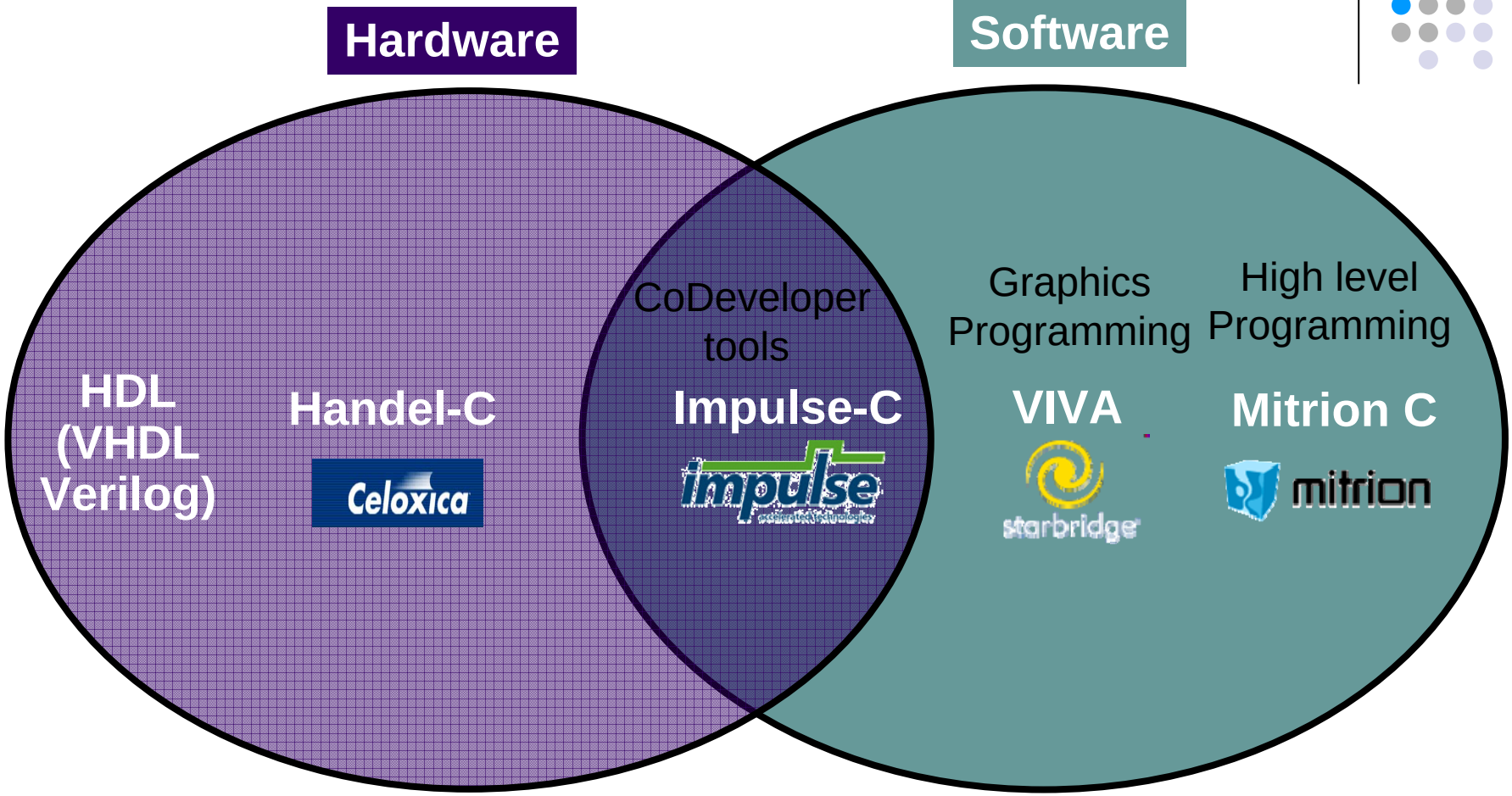



- **Celoxica Handel-C**
  - Augmented C syntax
  - Little underlying architectural assumptions-High level of flexibility
  - Refinement from C specification to FPGA hardware
- **Starbridge VIVA**
  - Graphical construction utilizing polymorphic components
  - Little underlying architectural assumptions-Hint of data flow
  - User translation from C specification to graphical description (i.e. schematic)
- **Mitrion-C**
  - Modified C syntax
  - Utilizes a processor/cluster abstraction
  - User translation from C specification into Mitrion C
- **Impulse-C**
  - Augmented C syntax
  - Optimized for streaming applications
  - User translation from C specification into Impulse-C

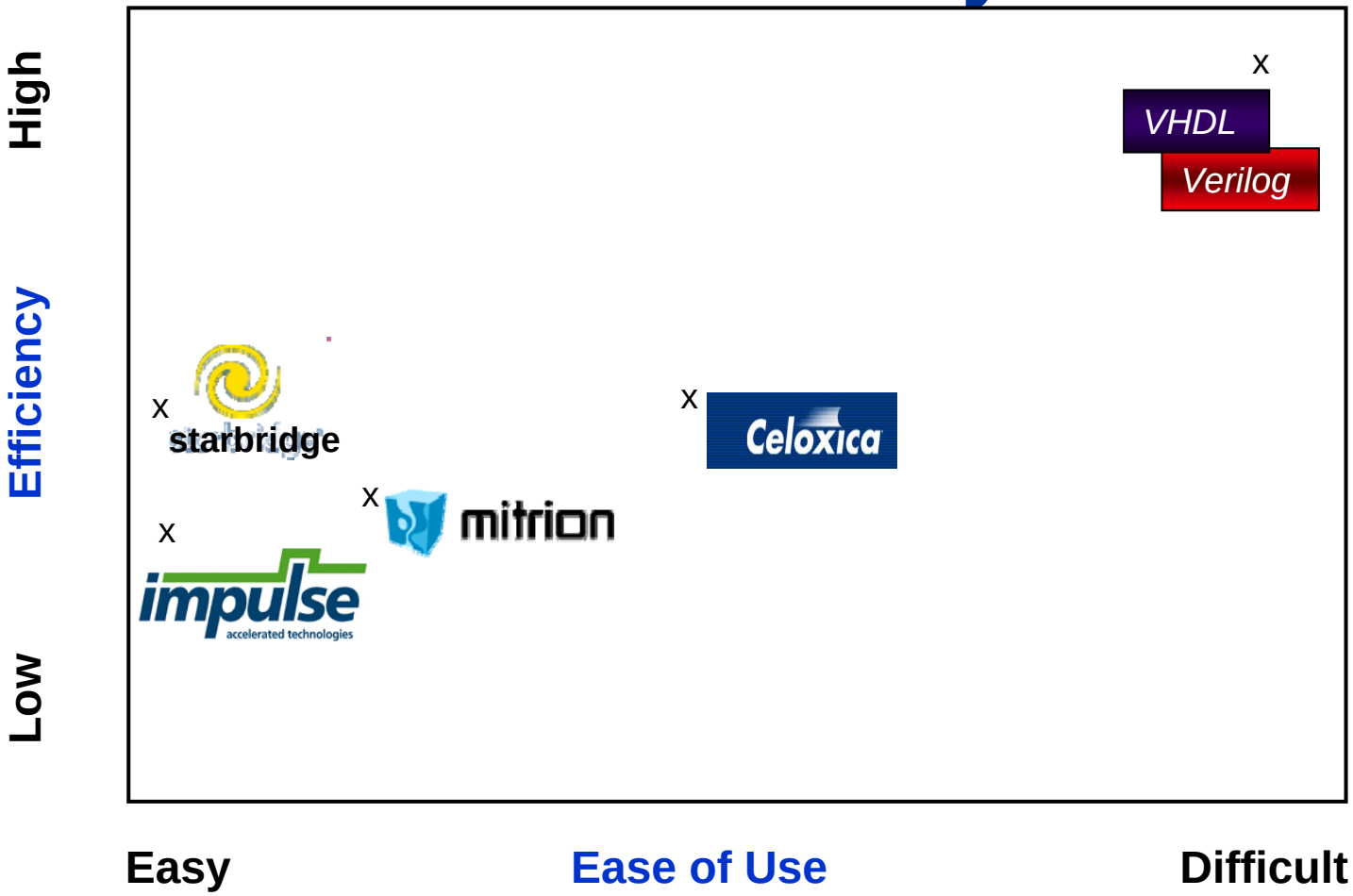
# Other HLLs

- SRC -> Carte
- Stoneridge -> Frontier
- Mentor Graphics -> Catapult C
- Nallatech -> DIME-C
- AccelChip –> MATLAB DSP Synthesis
- National Semiconductor -> Napa C
- Colorado State University -> SA-C
- Los Alamos National Laboratory -> Streams C
- Open SystemC Initiative  -> SystemC

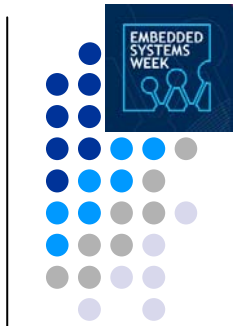
# HW versus SW


Hardware
Software
HDL
(VHDL
Verilog)
Handel-C
Celoxica
CoDeveloper
tools
Impulse-C
impulse
Graphics
Programming
VIVA
starbridge
High level
Programming
Mitrion C
mitrion

# Ease of Use vs Efficiency



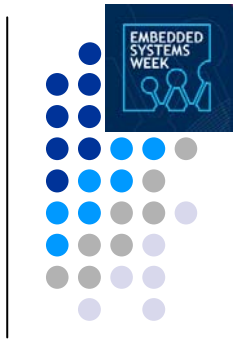
# C-based HDL




- **Advantages of C-based application mappers**
  - Far broader audience of potential RC users with high-level languages
  - Required HDL knowledge is significantly reduced or eliminated
  - Time for preliminary results is much less than manual HDL
  - Software-to-hardware porting is considerably easier
  - Visualization of C hardware is far easier for scientific community

- **Disadvantages**
  - Mapper instructions are many times more powerful than CPU instructions, but FPGA clocks are many times slower
  - Mappers can parallelize and pipeline C code, however they generally cannot automatically instantiate multiple functional units
  - Optimized C-mapper code is obtained through manual parallelization of existing code using techniques pertinent to algorithm's structure
  - Reduced development time can come at cost of performance

# Open Problems




- **Efficient HW/SW partitioning (Codesign)**
  - **Boundary between SW and HW are fixed**
  - **FPGA performance is dependent on high level architectural and low level details**
  - **Compilation times for FPGA changes can be long**
  - **Abstractions are not mature**