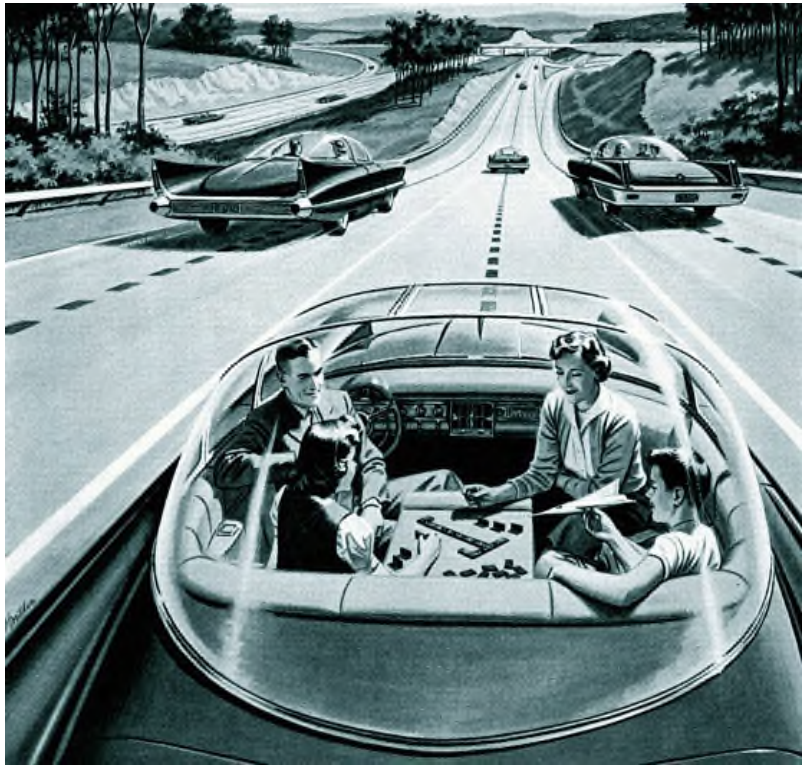


Connected Vehicle

H.Perrault
herve.perrault@ext.isep.fr

Disruptive innovation Part 3/3

Part1&2_Summarize

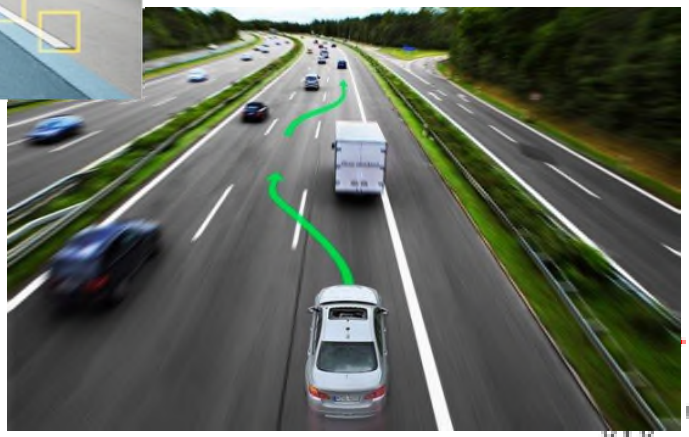
Autonomous vehicle ➔ Motivations :

- Fatalities mitigation ➔ Human errors lead to 90% of road accidents,
- Increase mobility (Seniors & disabled people, Traffic Jam,…)
- Energy print foot (Oil expected reduction ~15%, Health)
- Stringent regulators will

Experts forecast : In 15 years, 13% in EU-20% in US of journey will be done with autonomous vehicle.

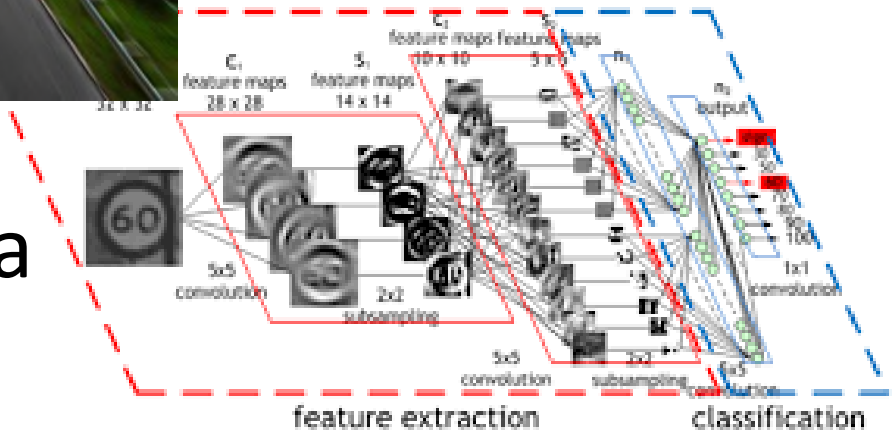
Autonomous Car - Challenges

Scan the scene : Almost mastered



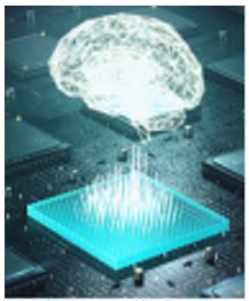
Path planning : Almost mastered

Shape recognition : Still a challenge !

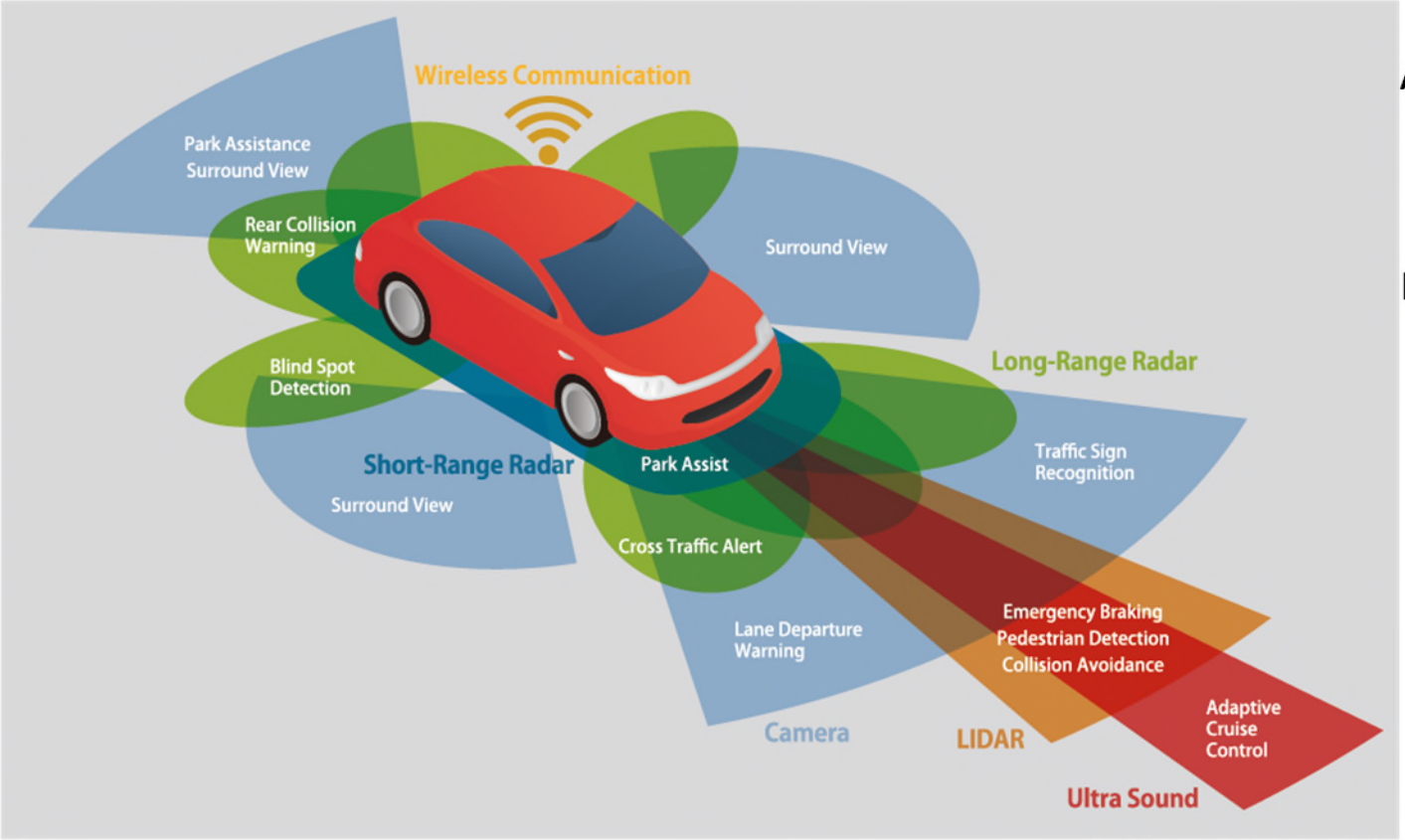


Level 2 is pretty mature, Level 3 in on the way…

There is still a long way to go beyond



Sensors Belt



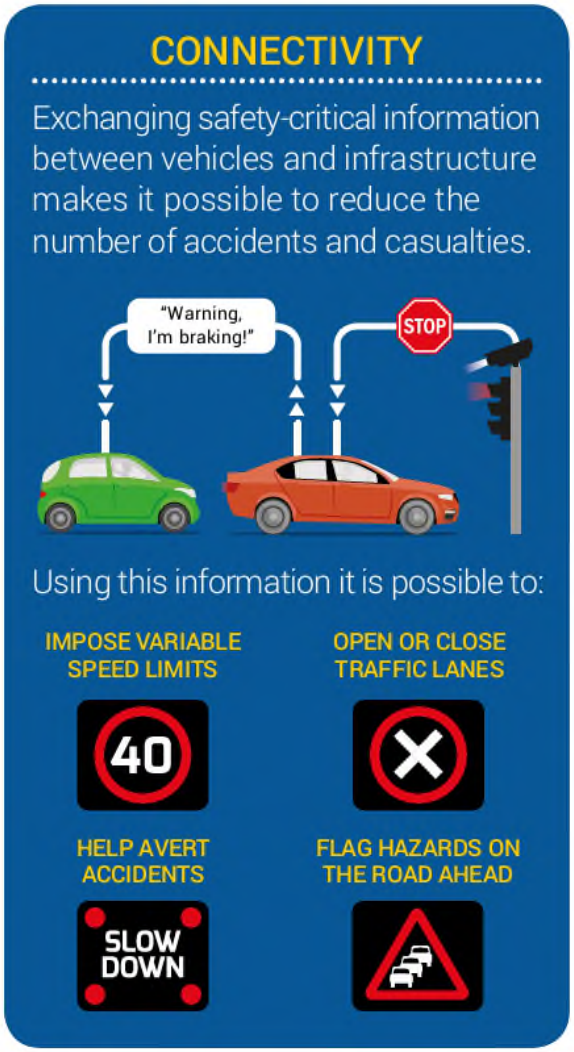
AV need to detect :

- Target range,
- Position and scene recognition,
- Shape & color detection,

CAMERA LIDAR RADAR GPS (RTK) Odometer HD Maps

Sensor fusion is a key to insure precision & accurate nearby and distant target.

Data flow – “See” what you can’t see !



Source : ACEA

Connected Car ➔ “Talking” with :

- ✓ Road,
- ✓ Infrastructure,
- ✓ Other cars,
- ✓ Pedestrians,
- ✓ Factory & Workshop,
- ✓ Services (Parking, Google apps,…)

Data exchanges involve standardized I/F & Messages

V2x definition

V2V

Vehicle to vehicle communication allows safety and harmless improvement

(ex : Intersection incoming situation lead to Camera & Radar blind situation).

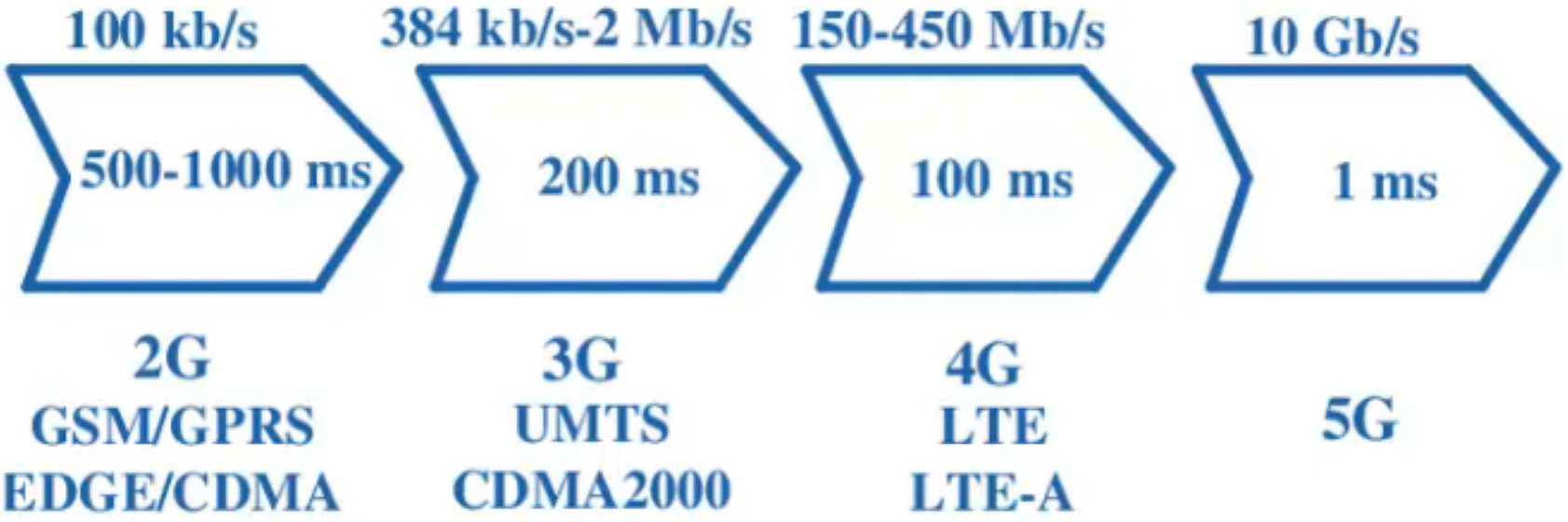
This communication is also powerful to insure bubble safe area all around the car with up/down stream).

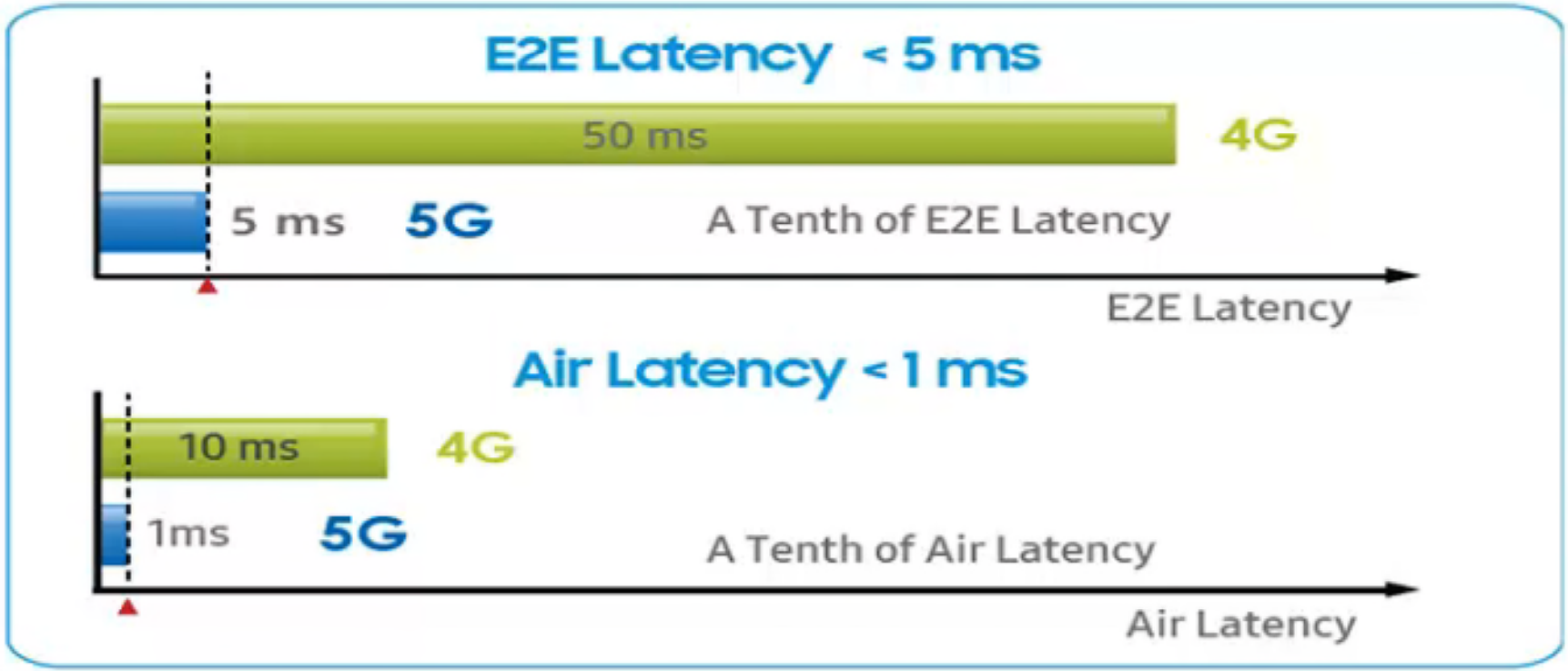
Studies are running for V2V infrastructure & road detection exchanges between vehicles (peer data flow).

V2I

Will lead to massive cost investment, however, these services will allow to receive traffic information, signalization, weather conditions, on work situation, closed danger,…

4G/5G Cellular - Latency





Quiz

Autonomous vehicles are required for :

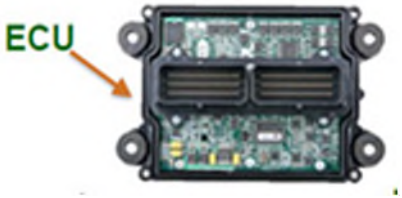
- Traffic jam ☐
- Car maker business ☐
- Health protection ☐

Accident massively belongs on human errors for :

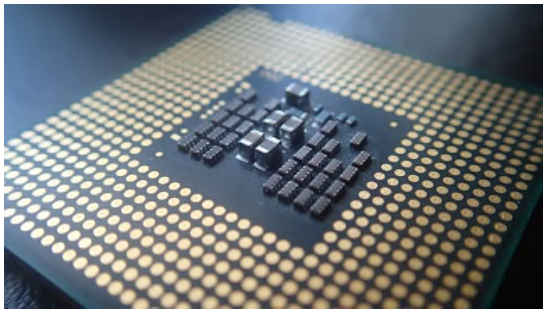
- 40% ☐
- 60% ☐
- 90% ☐

Massive autonomous car introduction is expected for :

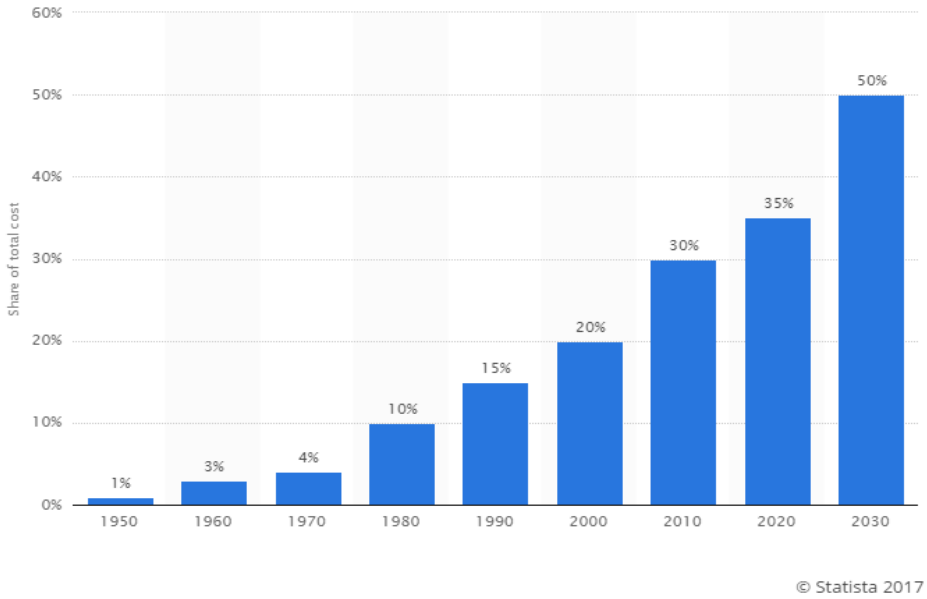
- 2040 ☐
- 2030 ☐



Embedded ECUs



Embedded electronic systems rise up in 70's, at first to reduce fuel consumption and gas emission.
Microcontroller appear in 80's, and 10 years after, a massive ECUs integration occurs



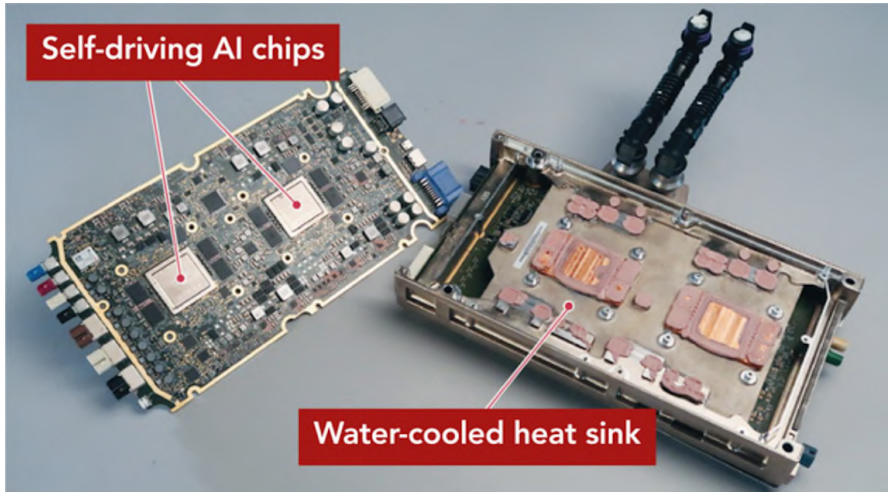
Electronic & Electrical equipment that was at 1% ratio in the 50's rise up to 30% today… specialist forecast 50% in 2030.

One main issue of imbedded system is that "intrinsic" resources are limited but also external : Energy, cooling, communications,…

Embedded ECU

Constraints :

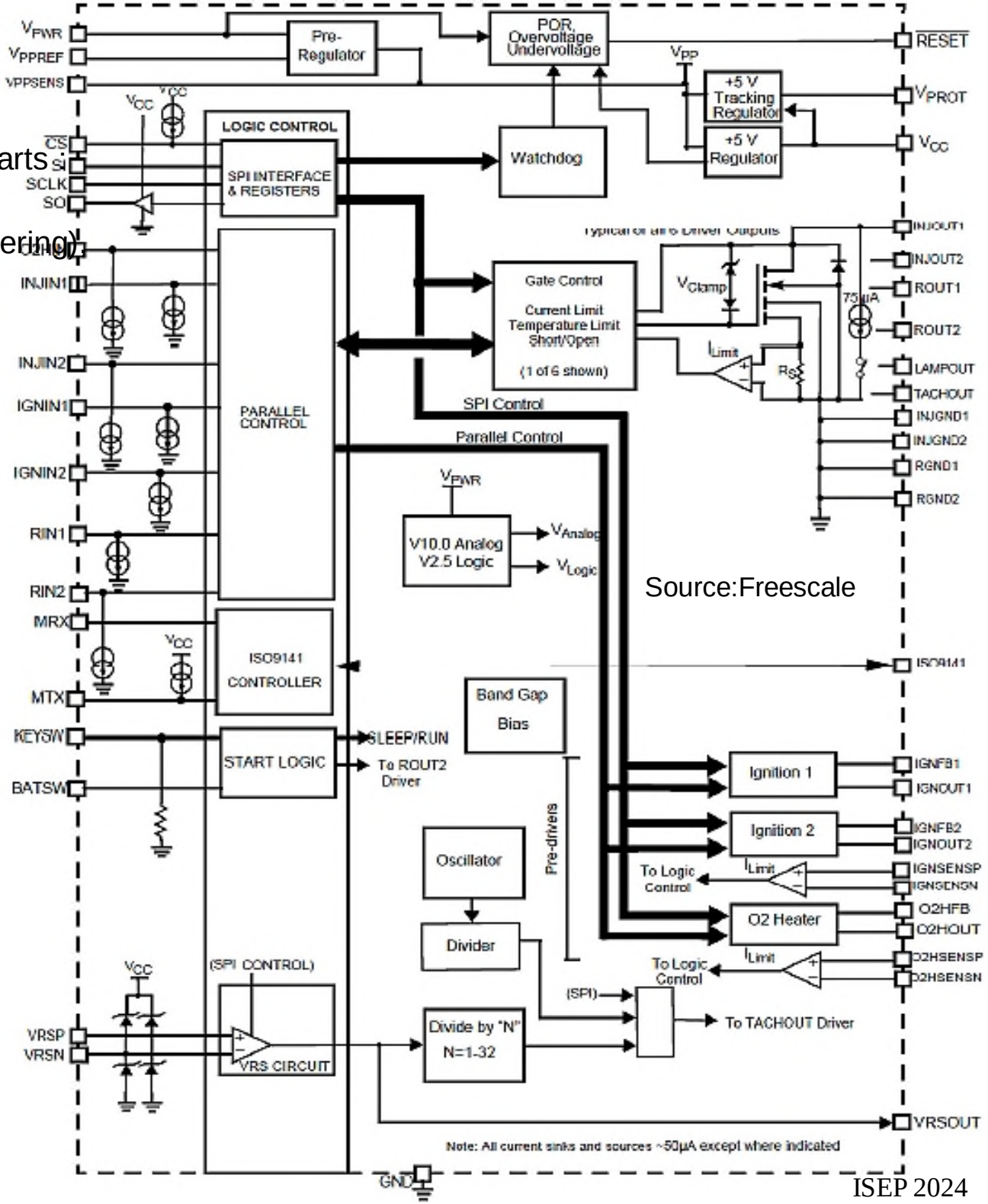
- ✓ Environment,
- ✓ Weight,
- ✓ Cooling,
- ✓ Energy
- ✓ Maintainability (including workshop services),
- ✓ Safety
- ✓ SW Updating
- ✓ Size



Embedded ECU

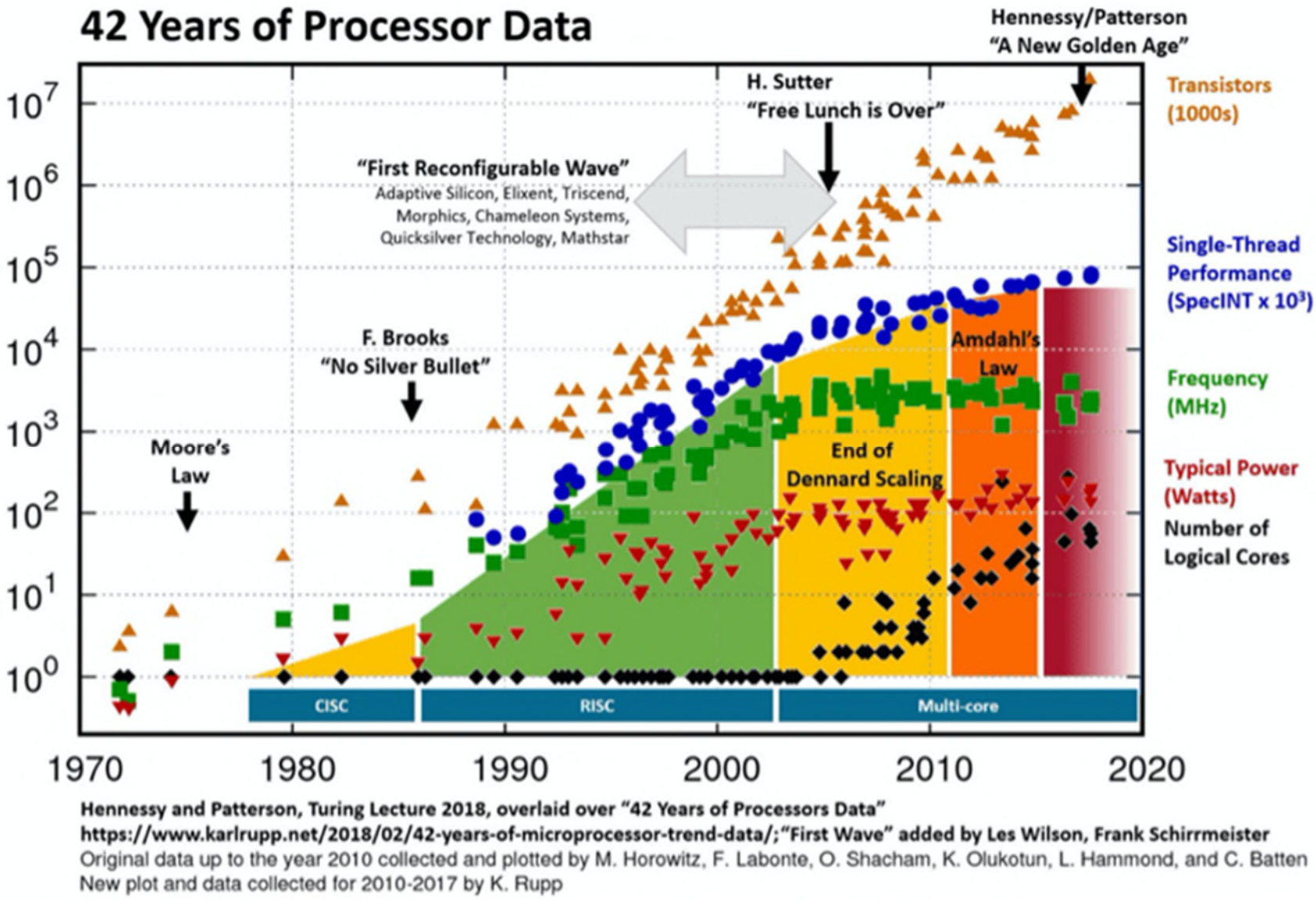
An embedded ECU includes the followings parts :

- Several power supply (yc protections & filtering),
- Voltage rails monitoring & Watchdog,
- Inputs : Filtering & buffering,
- Power stages (Smart power, H-Bridge),
- ASIC (and/or FPGA connected to uC),
- Transceiver & Phi (Networks)



Source:Freescale

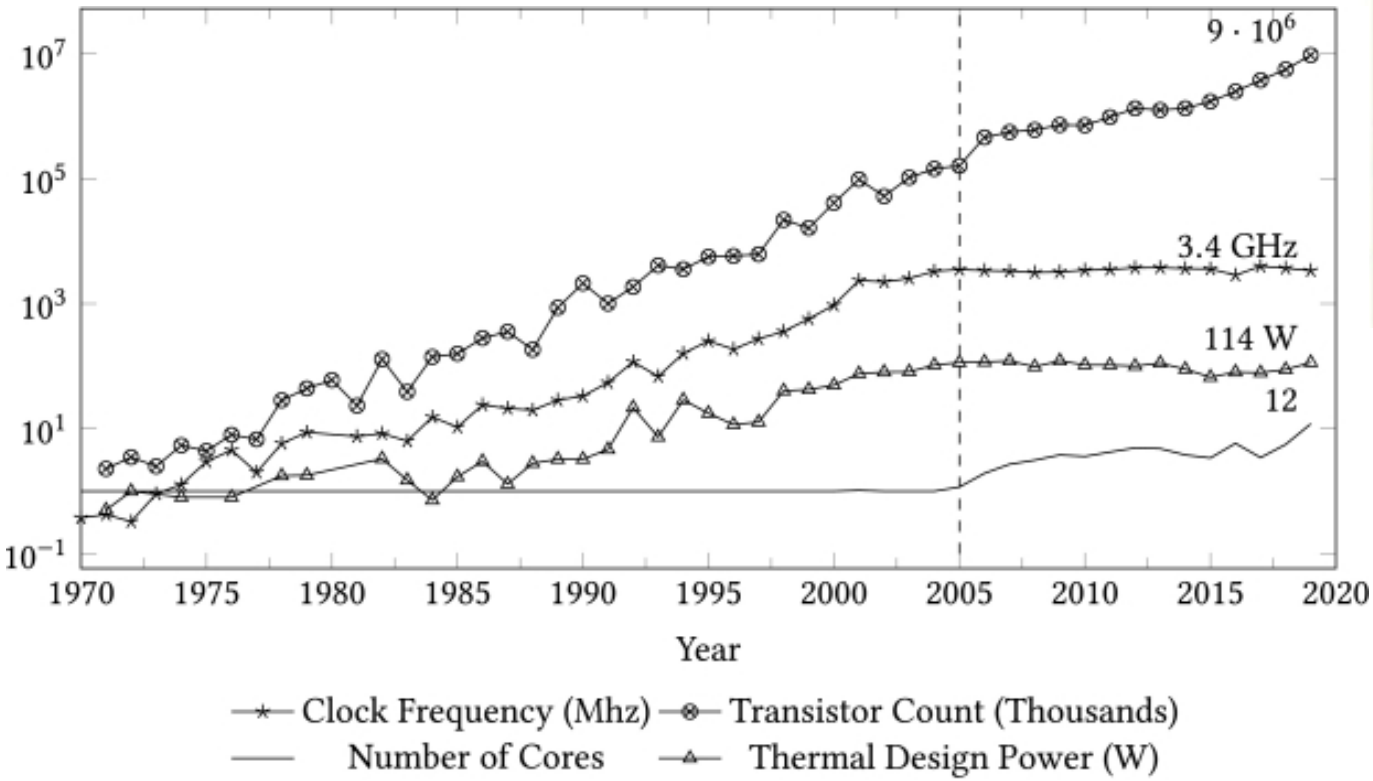
Embedded ECU – Computation capabilities

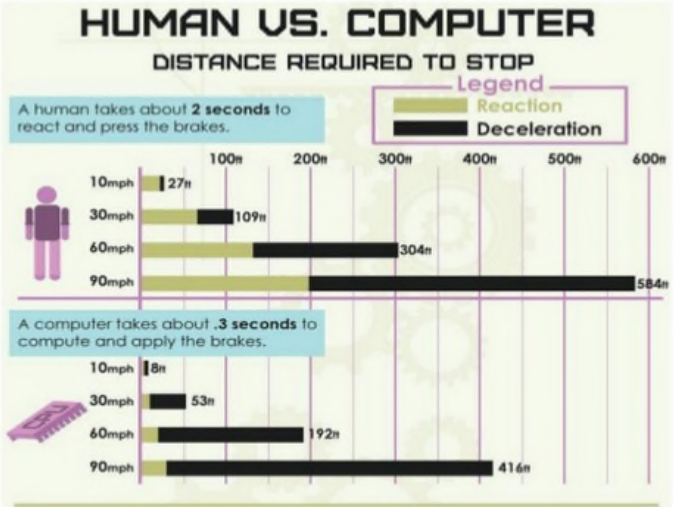


Hennessy and Patterson, Turing Lecture 2018, overlaid over "42 Years of Processors Data"
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/;"First Wave" added by Les Wilson, Frank Schirrmeister
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

80's : More transistors; 90's : More GHz; 2k : More Cores; 10's : Multithreading, pipeline,…
20's : Accelerators : GPU & FPGA

Processor - Trend





Key points :

- Nodes scaling down,
- Voltage decreasing,
- Clocking (stabilization due to current leakage),
- Multi Cores

Due to thermal wall & Nodes scale to decrease (shrink), clock speed racing has been replaced by multi-Cores, instructions optimization and parallelization.

ECU processing types per functional domains

Domain	Control loop time	Real time	ASIL	Processing type
Infotainment	ms	AVB, soft real time	Mostly QM, Up to B	µC with GPU
Body and comfort	ms	Soft real time	Mostly QM, Up to B	µP
Powertrain	µs	Hard real time	Up to D	µP Multi-core
Chassis	ms /µs	Hard real time	Up to D	µP Multi-core
ADAS domain	ms	Hard real time	Up to D	µC with GPU
ADAS sensors	ms	Hard/soft real time	Up to D (B and C common)	µC with GPU

Source: Siemens Digital

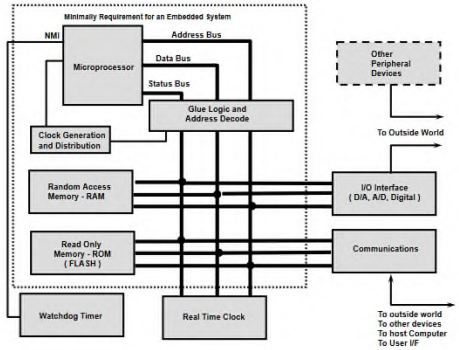
ECU processing types per functional domains

Domain	Control loop time	Real time		Processing type
Infotainment	ms	AVB, sc real		µC with GPU
Body and comfort	ms		y QM, Up to B	µP
Powertrain			Up to D	µP Multi-core
Chassis		Hard real time	Up to D	µP Multi-core
ADAS		Hard real time	Up to D	µC with GPU
AD	ms	Hard/soft real time	Up to D (B and C common)	µC with GPU

AV generate up to 10Gb of data every second

Source: Siemens Digital

Embedded – Main Silicon parts



CPU : General purpose

- ✓ Easy to use OS; highly flexible; huge catalog;

MCU : Specialized & high level of integration

- ✓ Specific OS; Safety issue; Real time; Energy; Thermal

FPGA

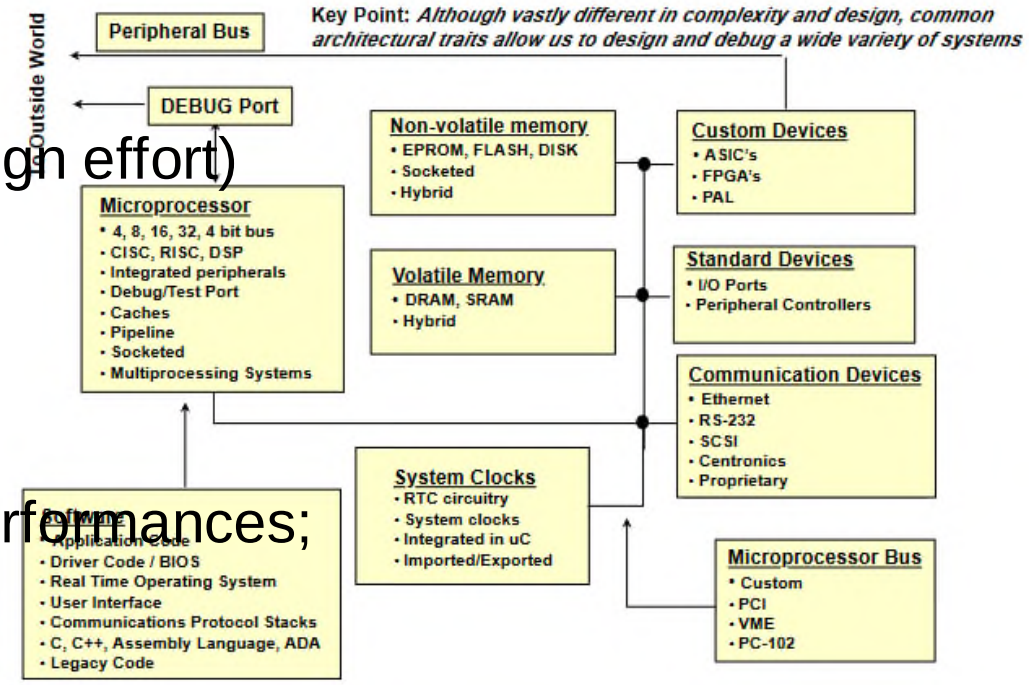
- ✓ Very flexible; performances (design effort)

Required specialized skills

GPU

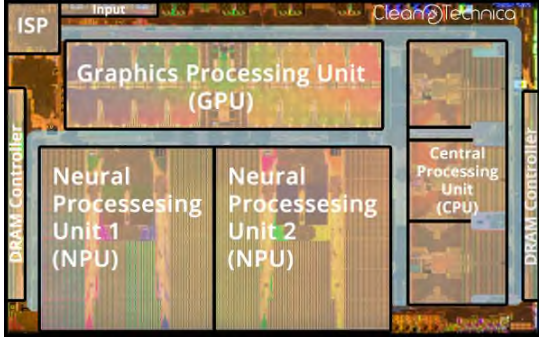
- ✓ Math & Parallelism oriented; Performances;

Efficient for image processing



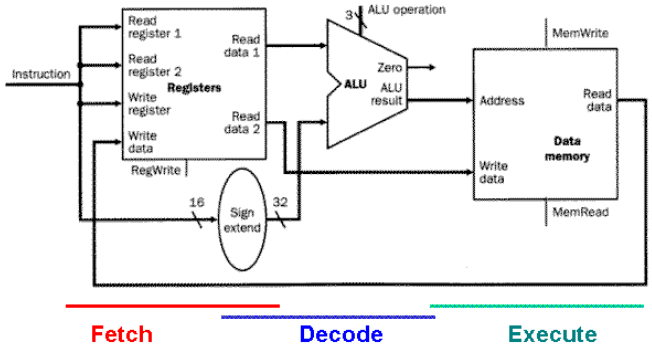
Autonomous vehicle require all of them !

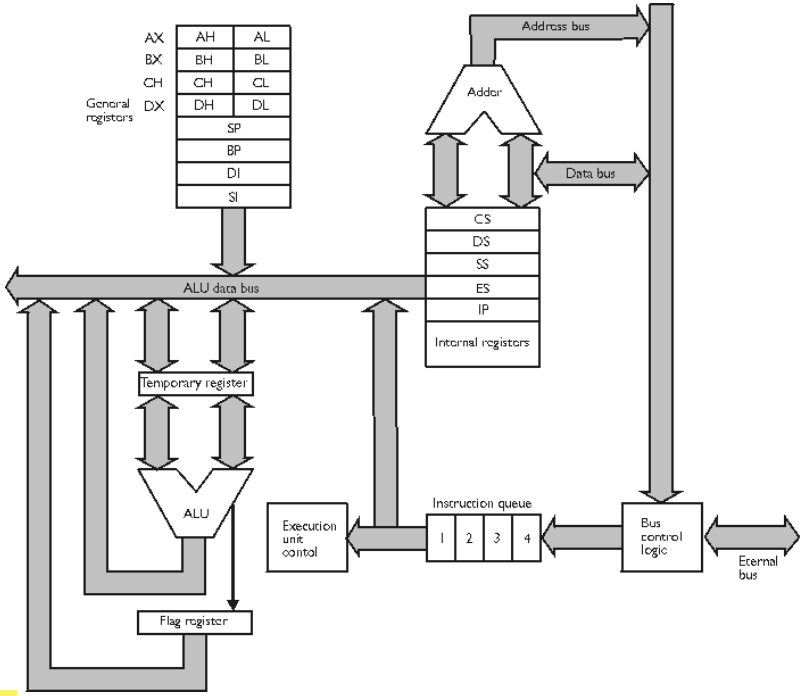
Introduction - Processor



A CPU is a set of circuitry and peripherals that executes instructions (program). To do this, a CPU includes :

- Registers,
- Memory (cache),
- I/Os capabilities,
- Clock,
- ALU,
- …

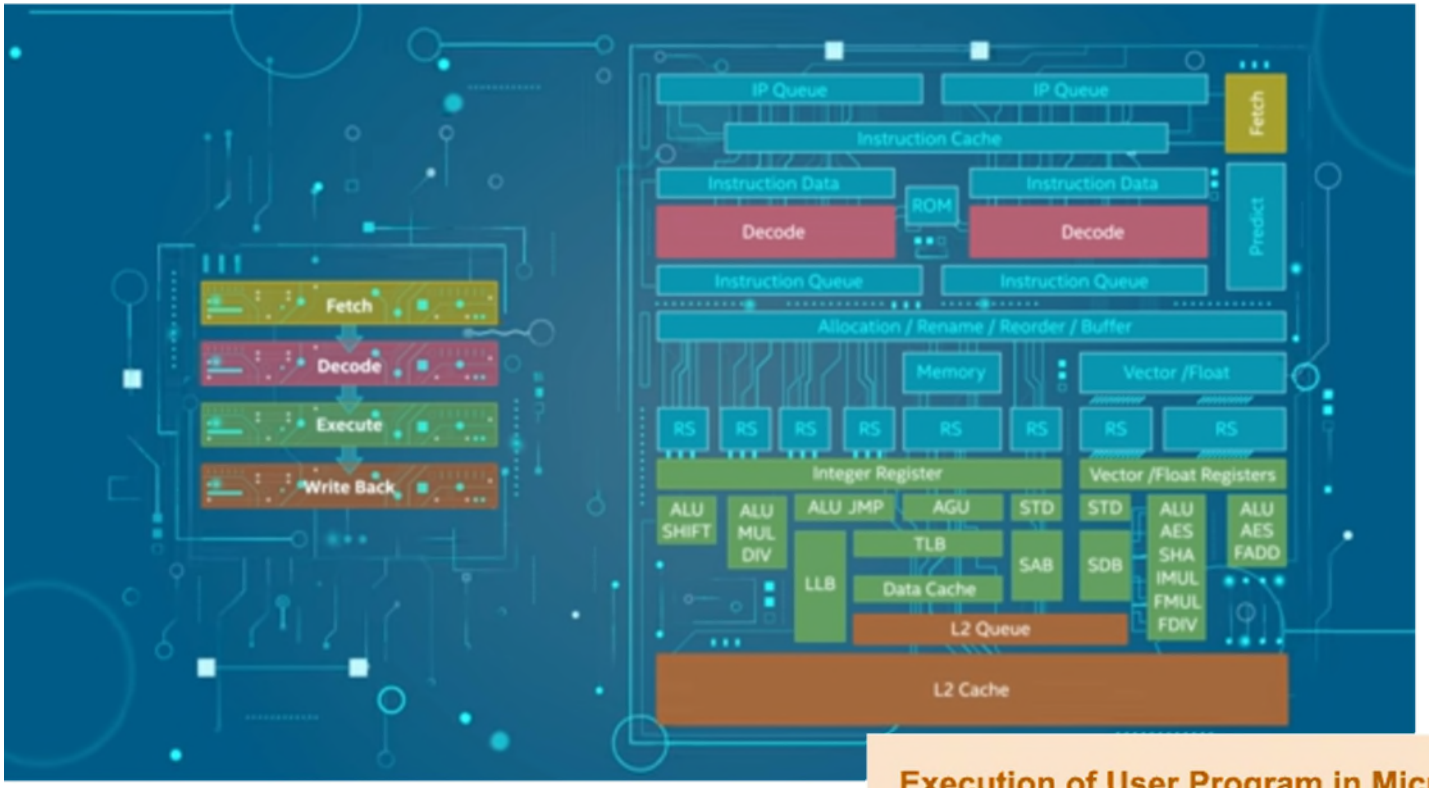


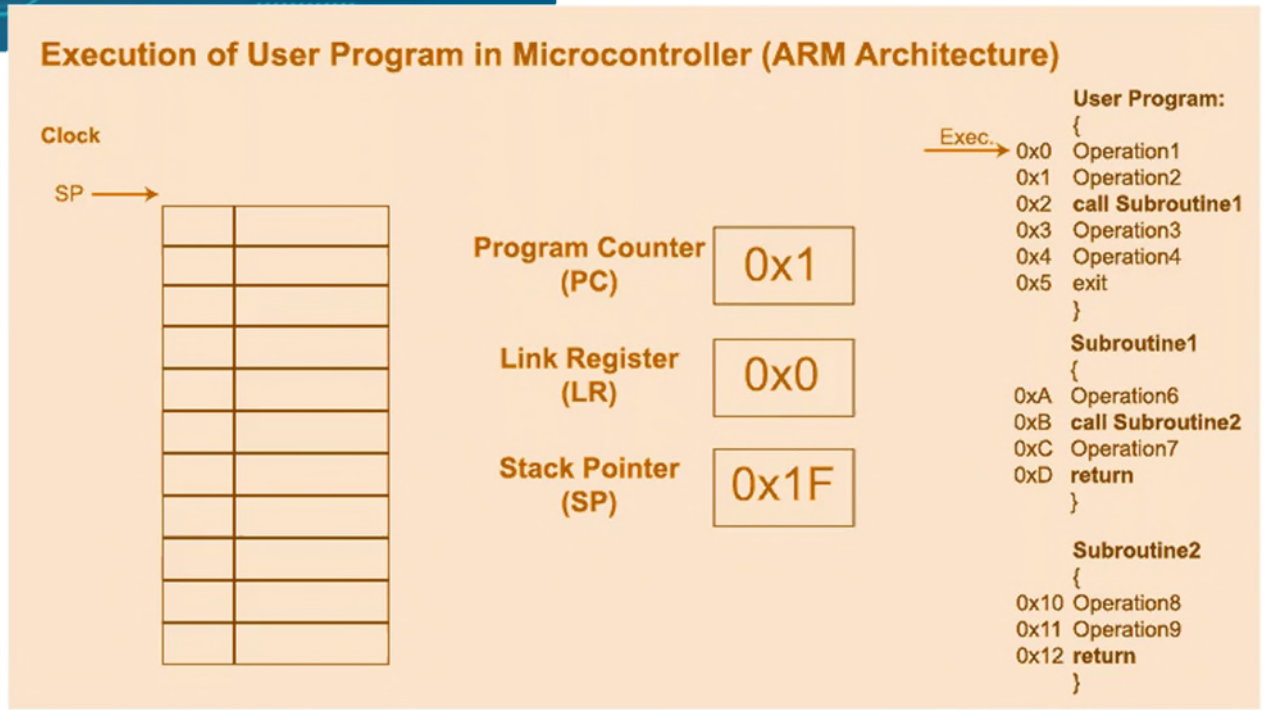


CPU flow is : Fetch->Decode->Execute->Store

CPUs are the brain of the supervisor, they are very powerful for control, mix computations but require additional ship and special design for rugged applications.

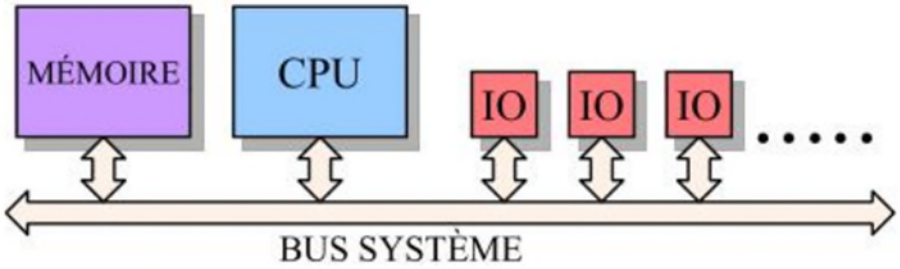
Processor – CPU cycle



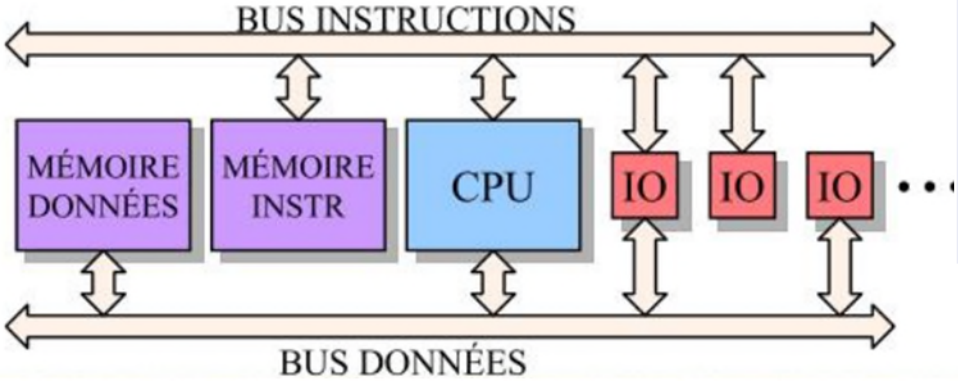


CPU - Architecture

- **Von Neumann**



- **Harvard**



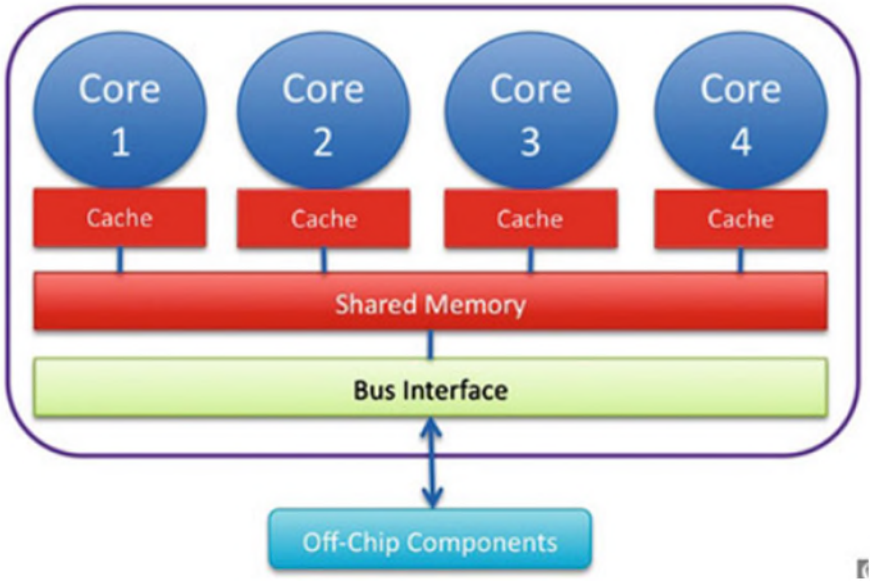
Advantages	Drawback
Simple, memory programmer handling	// executions not allowed*, CPU idling : only one bus
Data & instructions can be accessed at same time, Performances	Complexity, Cost

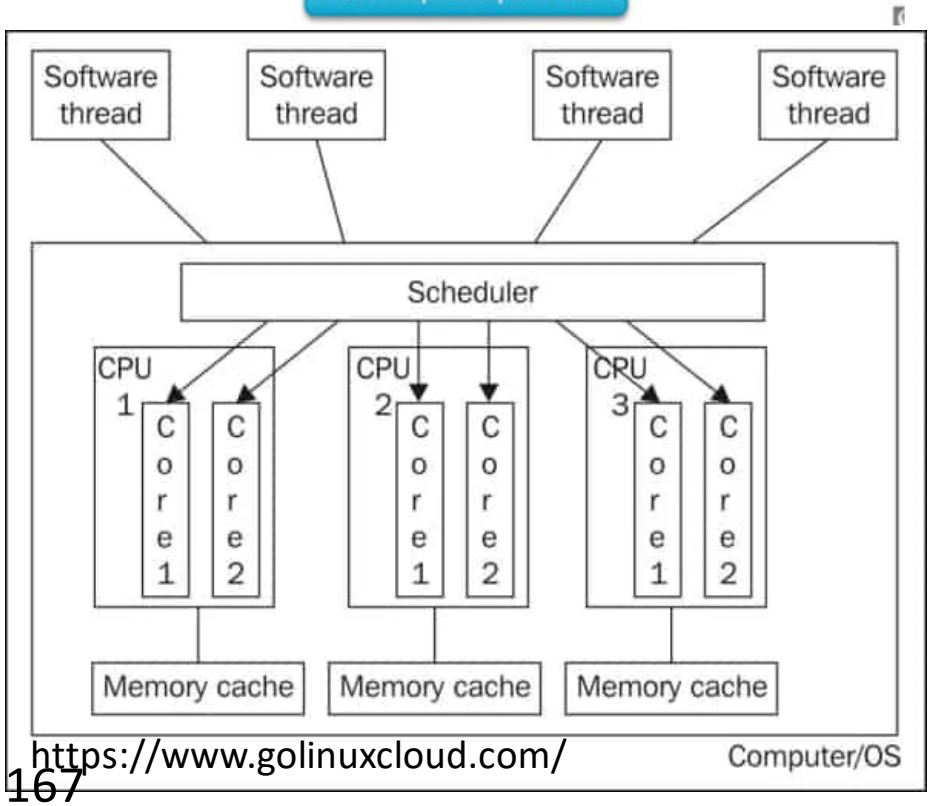
* : can be overcome with caching, prefetch, multithreading,…

Two types of CPU architecture coexist on processors : Von Neumann and Harvard.

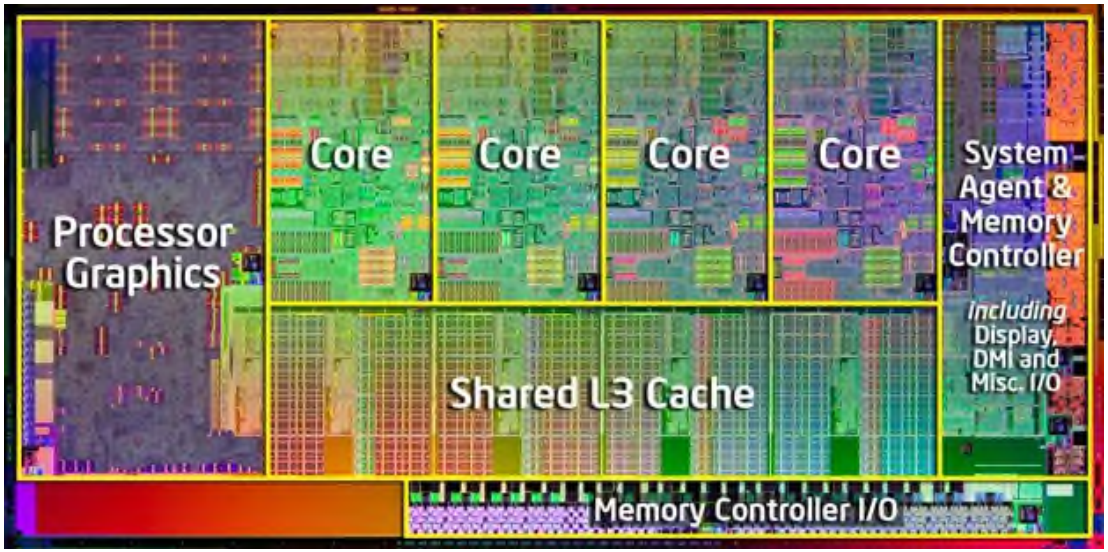
- Von Neumann Architecture uses the same bus to transfer data and instructions from the memory.
- Harvard architecture uses separate buses for instructions and data. The instruction address bus and instruction bus are used for reading instructions from memory. The address bus and data bus are used for writing and reading data to and from memory.

CPU – Single/Multi-Core(s)



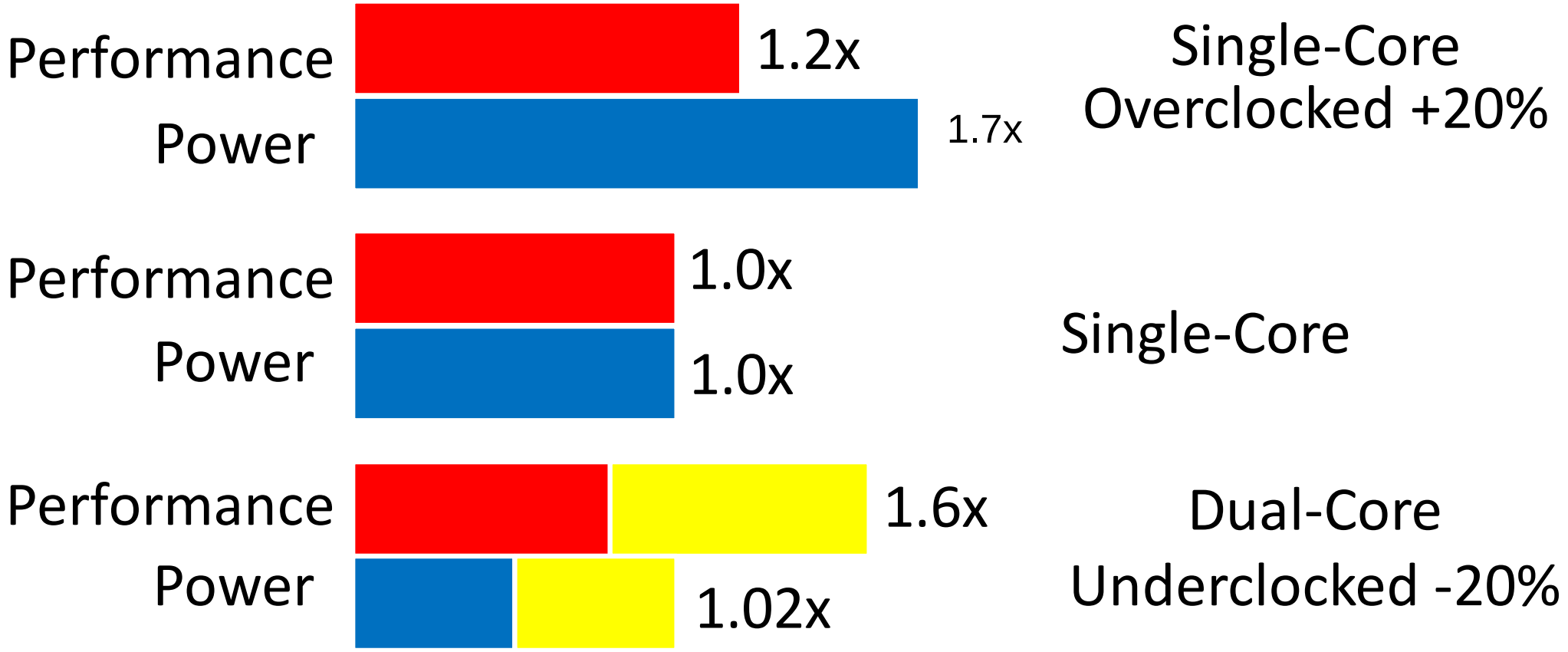


Due to thermal wall & high frequency handling, silicon vendors get around performance improvement with multi-cores (among other stuff).



M.C CPU : More than one physical processing unit (number of physical processing units that can be scheduled and run in parallel).

CPU – Single/Multi-Core(s)



Software must be written as parallel program

Multicore difficulties

- Partitioning work, Coordination & synchronization, Communications overhead
- Balancing load over cores
- How do you write parallel programs? … without knowing exact underlying architecture?

Single core vs Multi core

CPU : Optimized for serial tasks

- 1 to 32 Cores
- Run fast (ex : 1ns/instruction, so called IPC : Instructions/Cycle)
- Optimized for serial operations

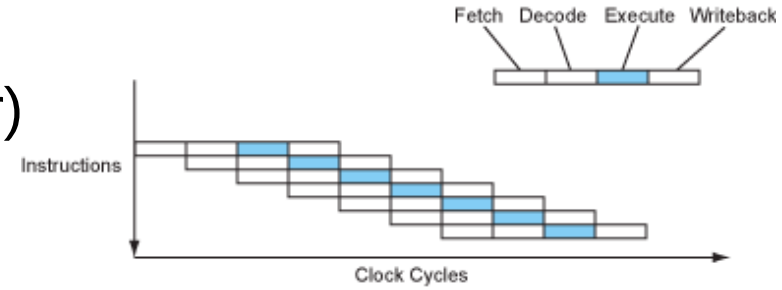
	Single Cores	Multi Cores
Cost	Depend on application, MC probably less expensive (global approach).	
Sw Design	+	-
Capabilities	- (could be)	+
Power consumption(1)	- (thermal wall)	+
Size factor (package)	-	+ (resources sharing)

(1) : $P = C.V^2.Freq$ (Capacitance; Voltage and Frequency)

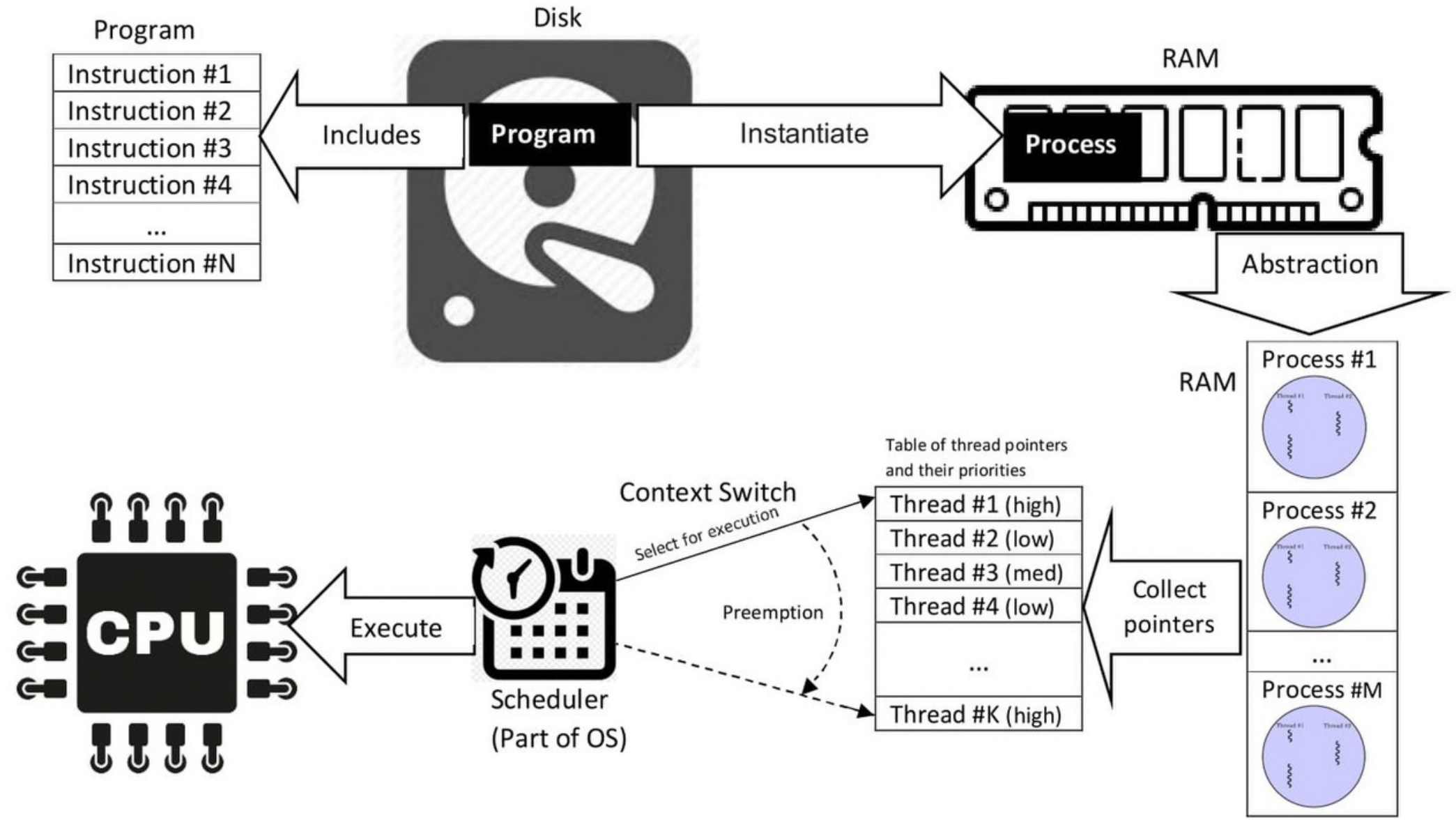
4 Guys in // to change wheels

Performances improvement :

- Cache RAM (Very fast : close to processor)
- Pipeline,
- Vcores.

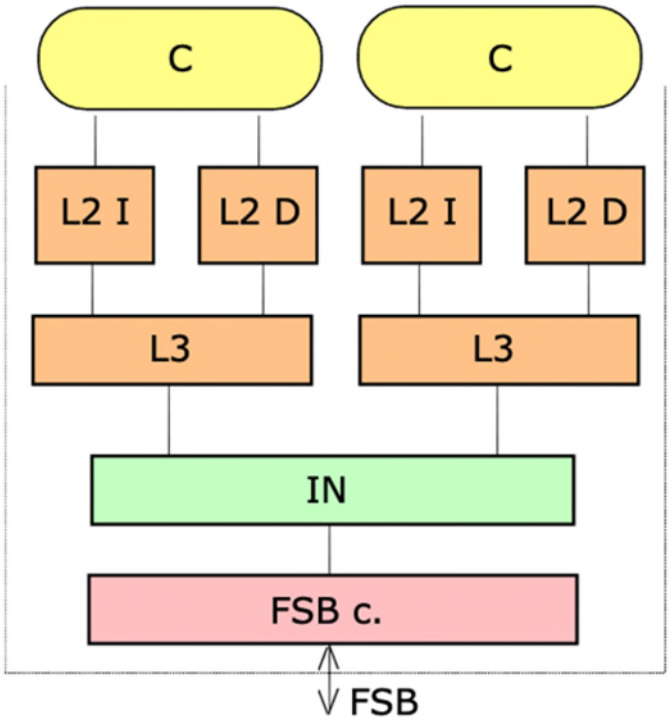


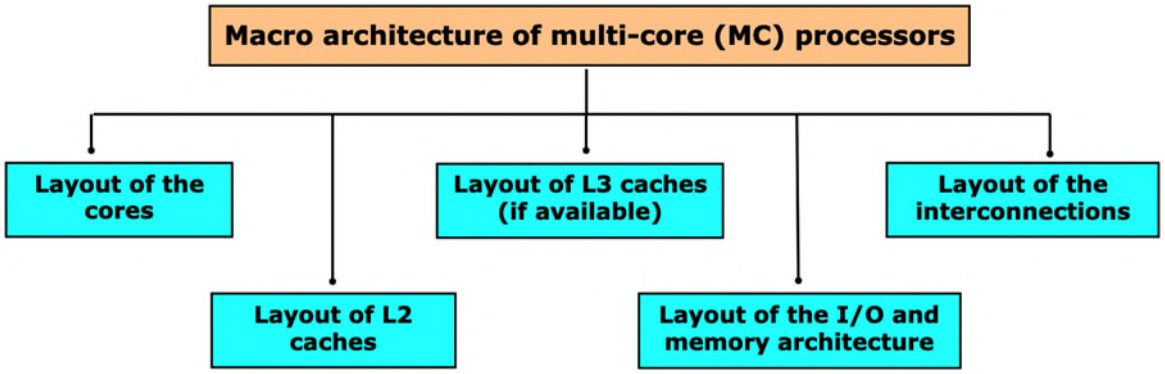
Multi-Threads



M.T allows “basic executions: threads” of a single process (ex: printing) in //. These threads share code & data of the process.

CPU – Caches

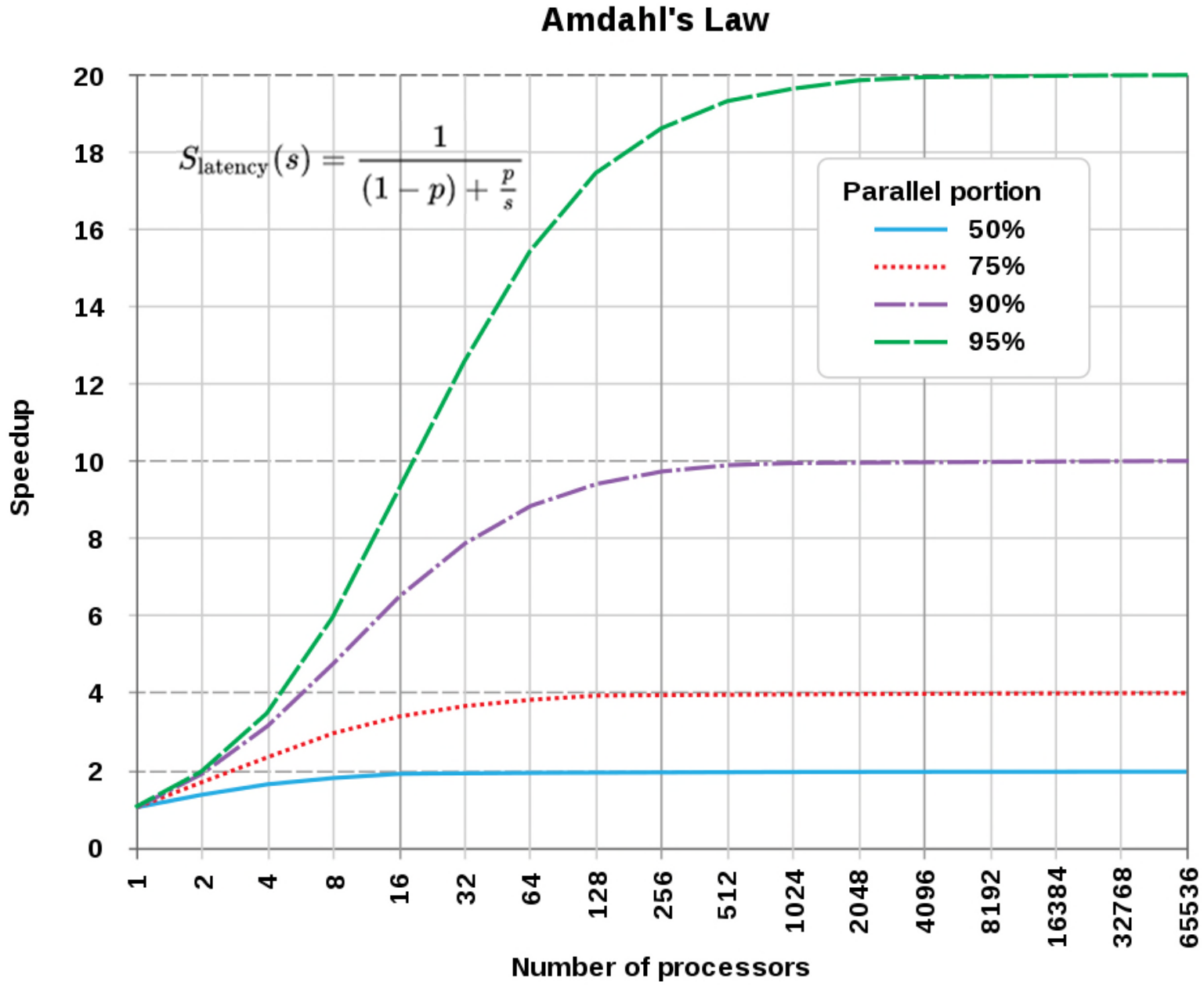




L2 I : Cache L2 for Instructions
L2D : Cache L2 for Data
FSB : Front Side Bus

A multi-core processor implements 3 levels of caches, these memory enable inter-core communications.

Amdahl law : The theoretical speedup of the latency of the execution of a program as a function of the number of processors executing it. The speedup is limited by the serial part of the program.

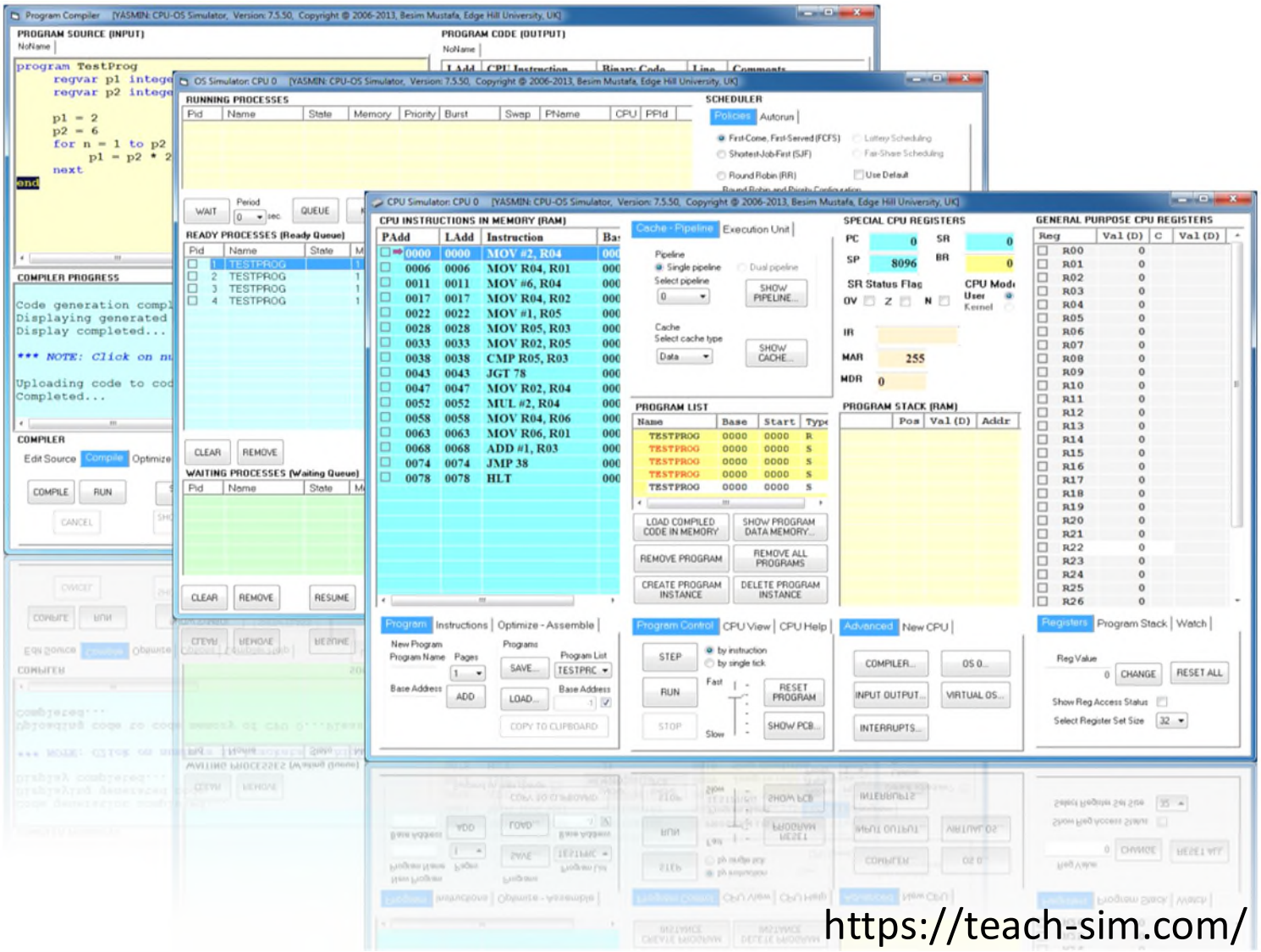


A m
me

Am
exe
exe
program.

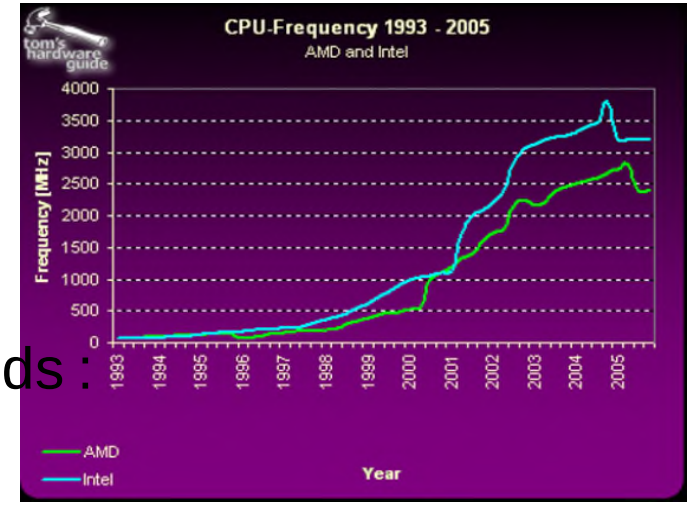
ors

CPU – Caches



Let’s play with CPU-OS Simulator

Computation needs

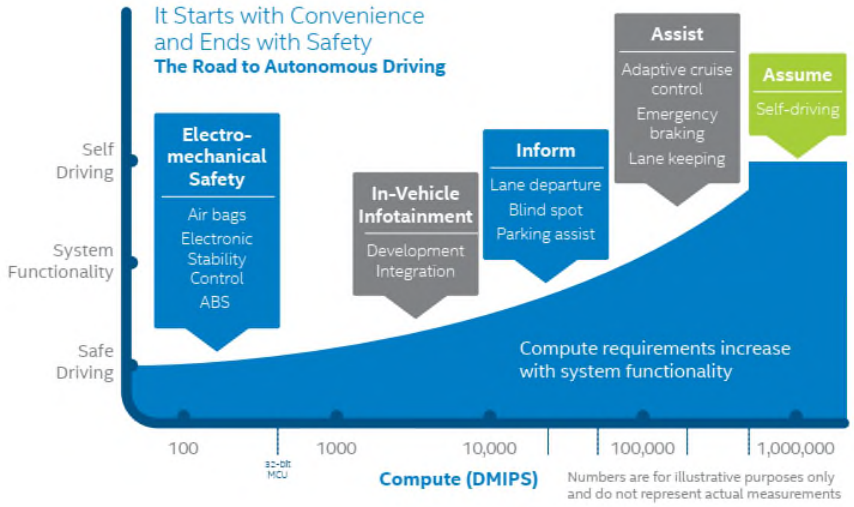


Computing capabilites are required from severals needs :

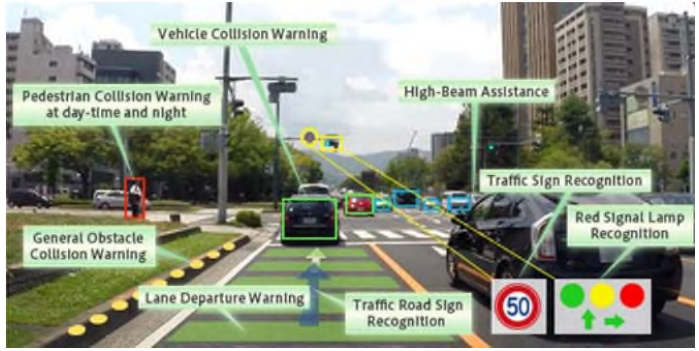
- Sensors treatement (Filtering, measurement,..),
- Signal & Information consistency (Safety)
- Targets handling (and recognition),
- Monitoring and sub domains control,
- …

All of this with embedded stringent constraints :

- Elecrical consumption, Fanless,
- Thermal robustness, Safety issues,...



Detection



Shape recognition

A Convolutional Neural Networks makes possible to identify objects with up to 95% accuracy.

Nevertheless, **CNN** have limitations:

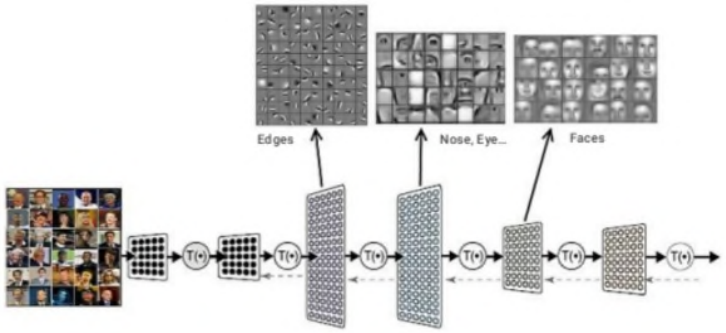
- Require high powerful processing power (Well supported by GPUs).
- Capabilities could be influenced if image is deformed, rotated or faked (sub-parts twisted).

CNN play a set of weight & parameters, the classifier makes lot a errors and adjust weight in order to decreases errors (it plays slightly and try to see in what direction errors decrease).

For a complex image, we split it into little blocs(convolution) and classifier analyze it.

where to play :
Objects library : https://codelabs.developers.google.com/

Neural Network

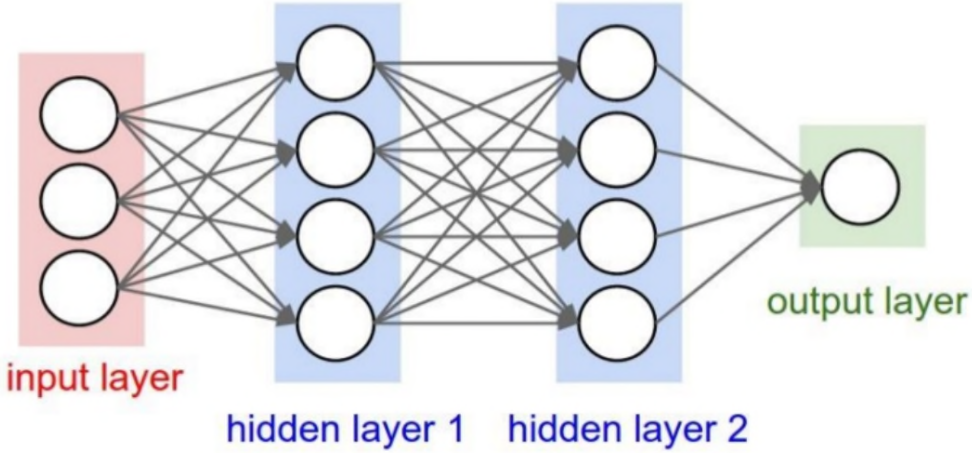


CNN : Convolutional Neural Networks

Enables Image recognition

Computation needs :

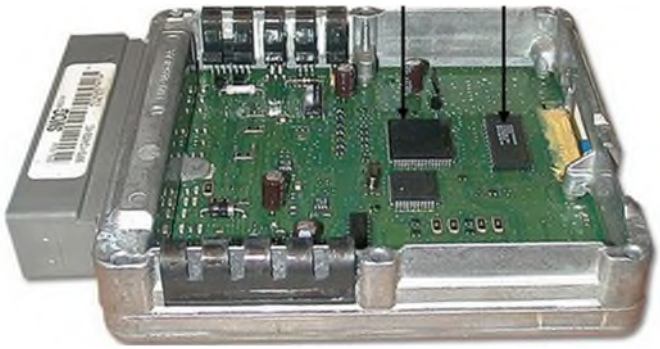
- Floating points,
- Parallel threads adapted



Neural network is a combination of inputs and output layers, connected.

Each layer is composed with a collection of neurons, left to right represent a “progression” in term of level.

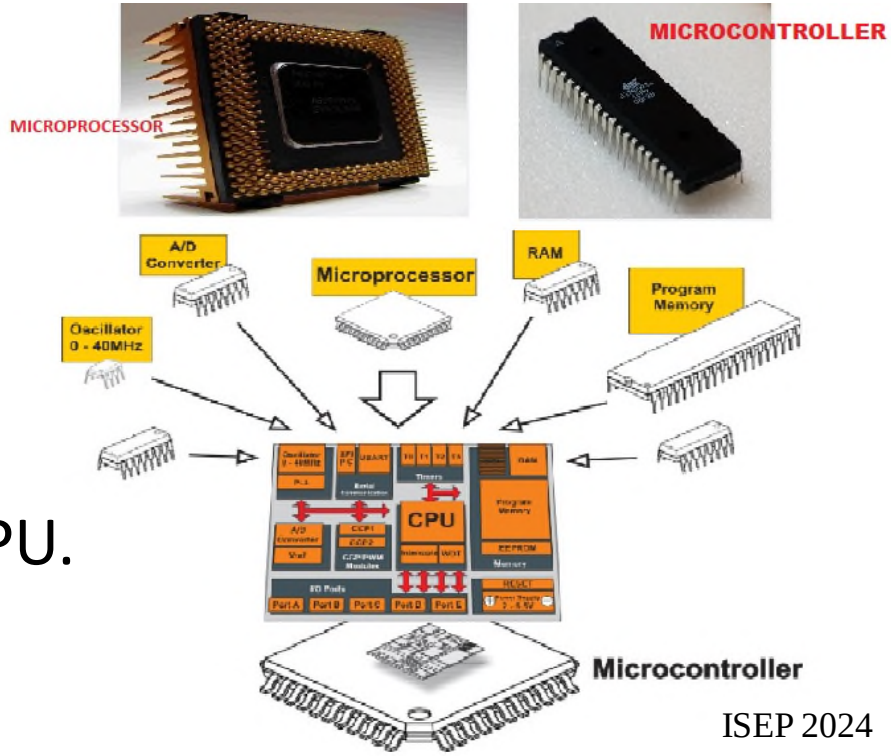
Them more layers the network has, the higher level features it will learn.



Microcontroller

What we can find inside a µC :

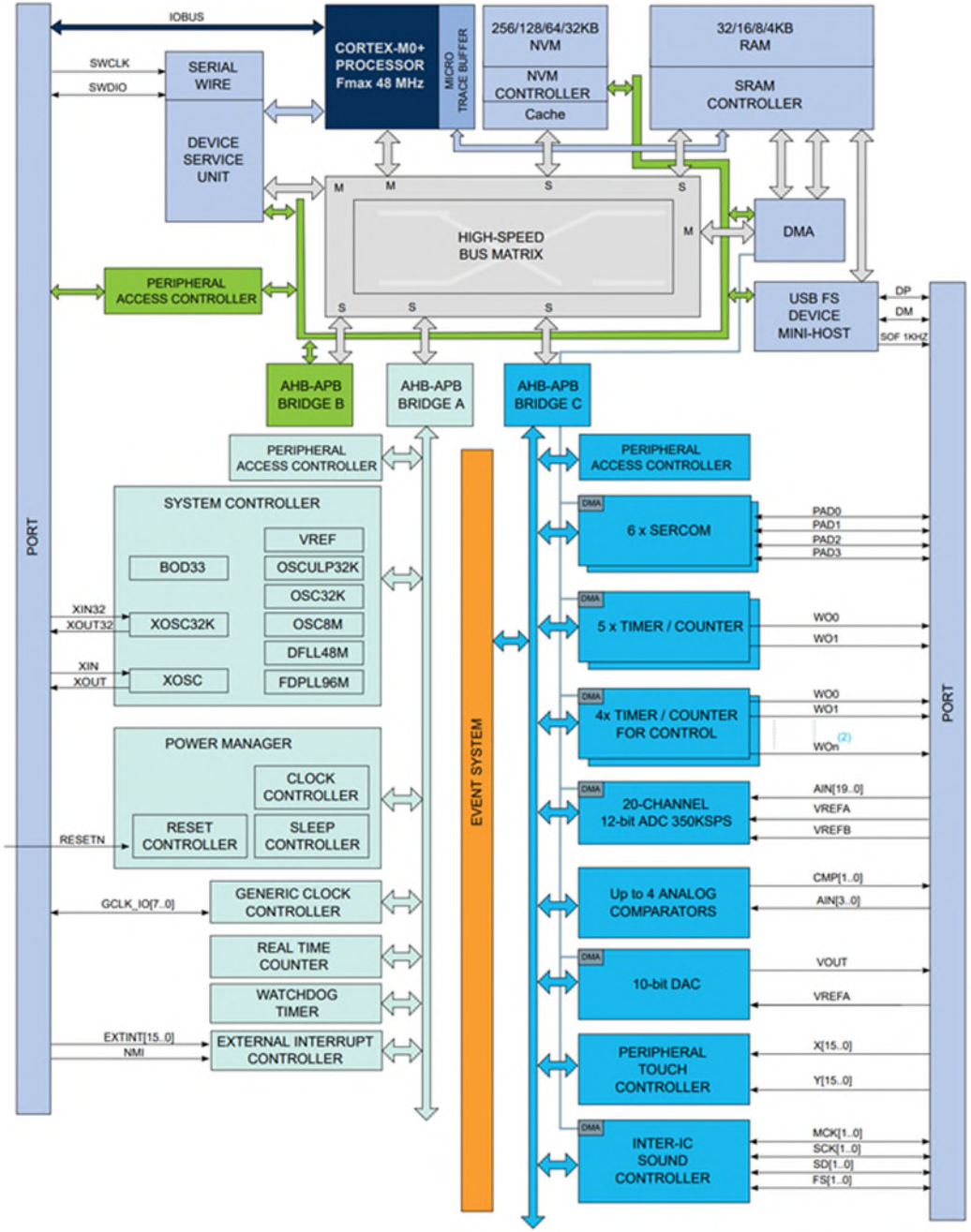
- Arithmetic unit,
- Buses,
- Memories (Ram, Flash,...)
- Peripherals,
- ...



For the biggest applications, we can find

Embedded microprocessor, FPGA and GPU.

Microcontroller



A microcontroller (MCU) has on a single integrated circuit :

- Simple central processing unit (CPU),
- Peripherals : Memories, I/O devices, timers, UARTs,...

Main Metrics to consider :

- Power consumption, Clock frequency, IO pins, Memory, Internal functions, Others
- On ship Memory : SRAM & Flash

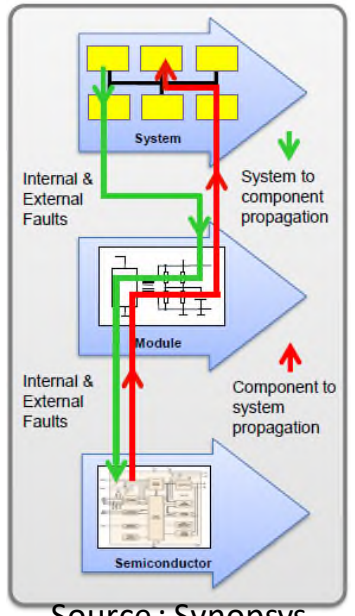
And also :

- Temperature range,
- Security,
- SW architecture & in house skills,
- Cost,
- Roadmap.

Processors & Software

Today's car has the computing power of 20 modern PCs, features about 100 million lines of code, and processes up to 25 gigabytes of data per hour.

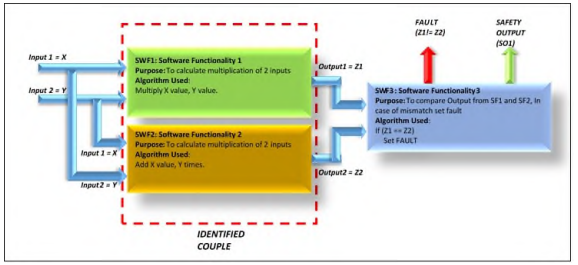
The question is : How & When we'll be able to deploy a fleet of fully autonomous driving systems, safe enough to leave humans completely out of the driving loop ?



Source : Synopsys

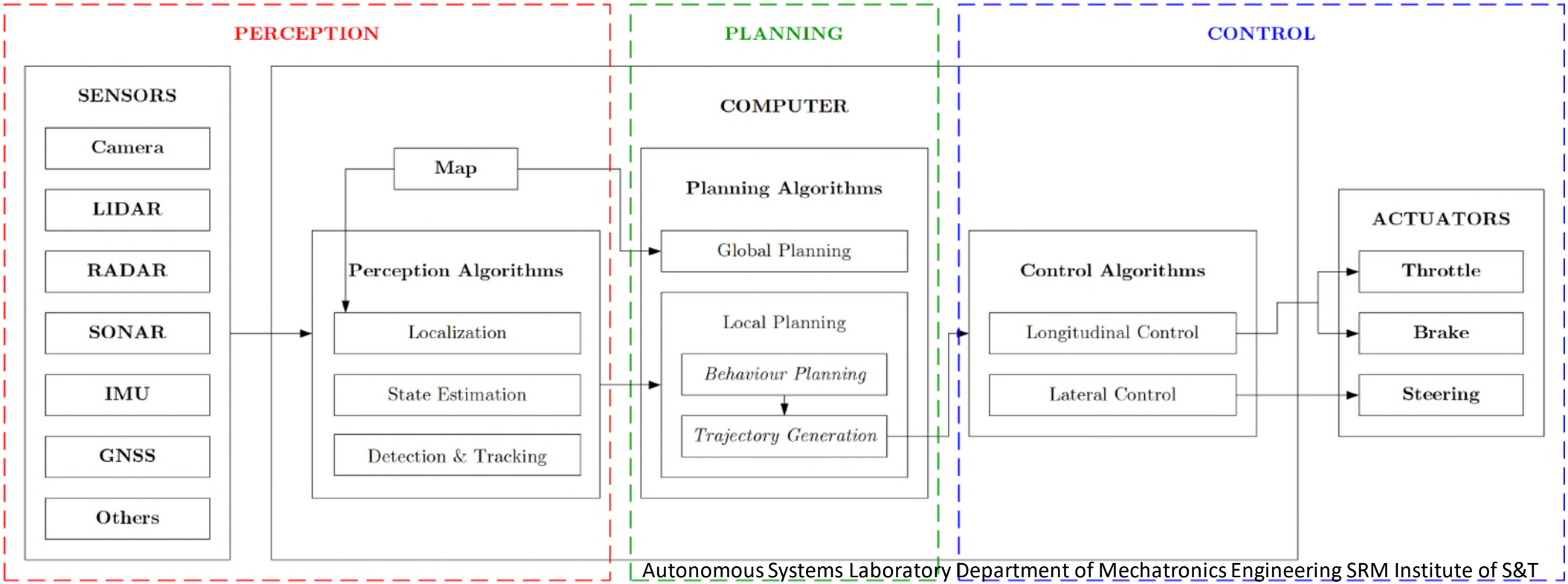
Process design & Components : verification (ISO 26262)

- ✓ Avoid blocking of execution, deadlocks, time budget violation,…
- ✓ Memory content corruption, unauthorized R&W access,
- ✓ Self monitoring software & Self-tested chips before each driving cycle with an extremely high level of diagnostic coverage,
- ✓ Step lock processors (Run time barrier).



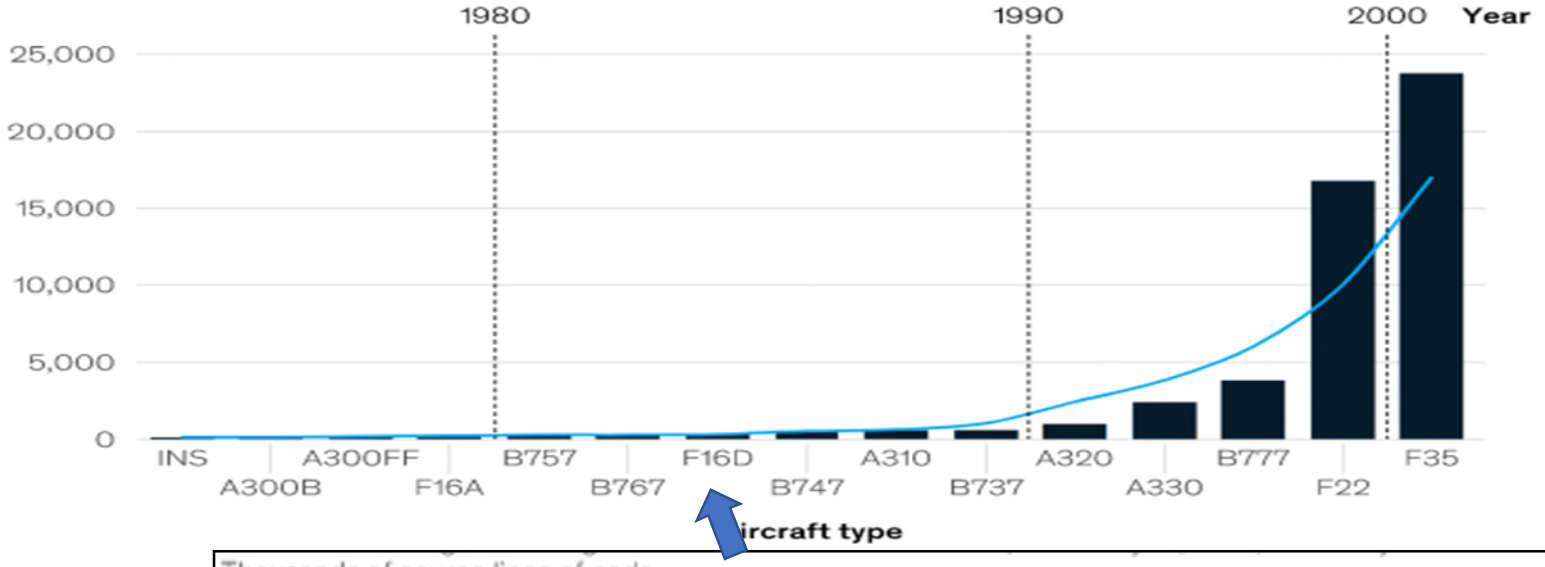
Automotive SW panorama : Multiple entities encounter multiple dependencies (Time to market constraint, cost mitigation, many third party,…).

Software



On the left side, sensors perceives the system's surroundings for extracting relevant features or from the incoming raw data.
Computer determine motion & a collection of actions which are processed by the last layer by using control algorithms.
For setting up interactive or unattended system simulations, processing chain must be closed.

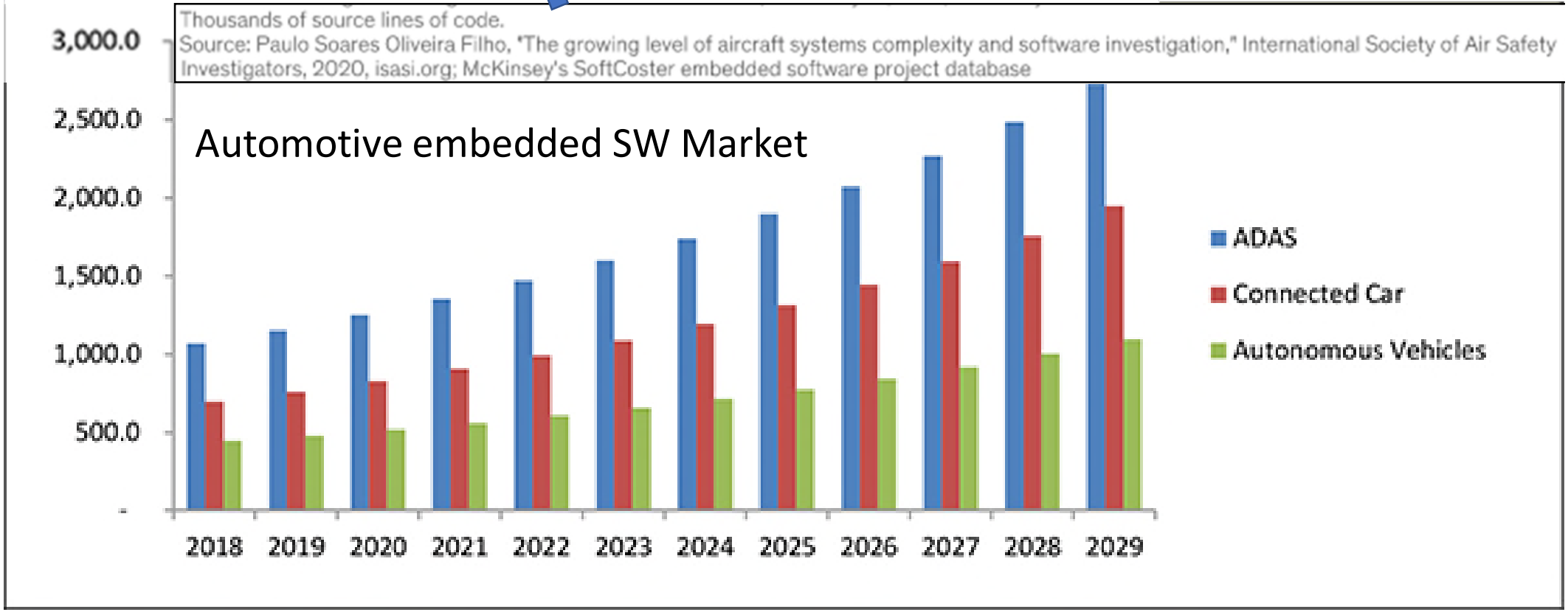
Software



Automotive embedded SW : Up to :

100 million lines of SW

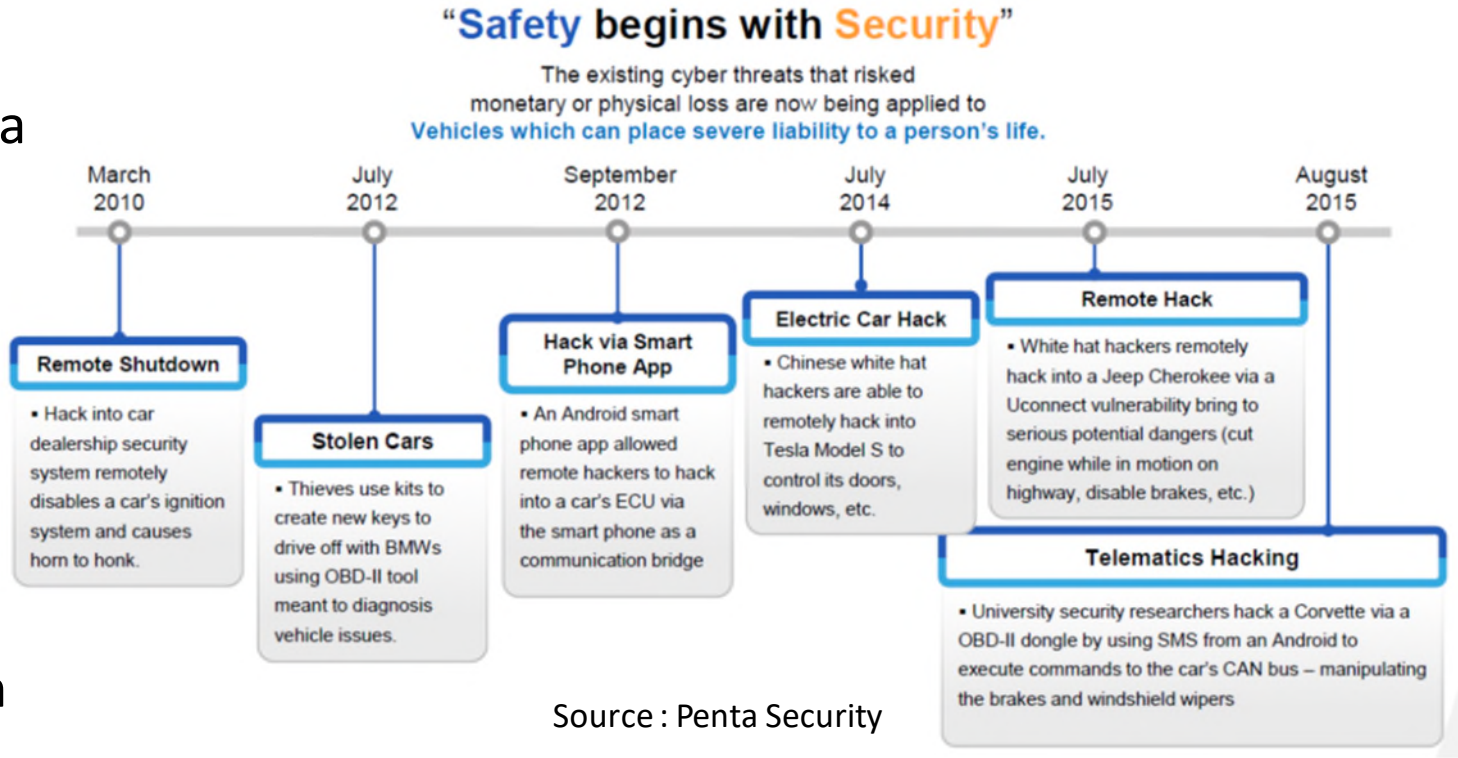
Thousands of source lines of code.
Source: Paulo Soares Oliveira Filho, "The growing level of aircraft systems complexity and software investigation," International Society of Air Safety Investigators, 2020, isasi.org; McKinsey's SoftCoster embedded software project database



Safety issues

Software

To avoid a car “crash” and a costly recall, automotive suppliers should be continually vetting their code to ensure its safety and impenetrability. Governments around the world have begun to help the automotive industry in establishing these practices.

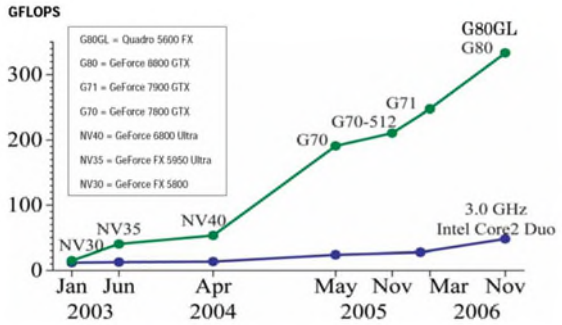


Source : Penta Security

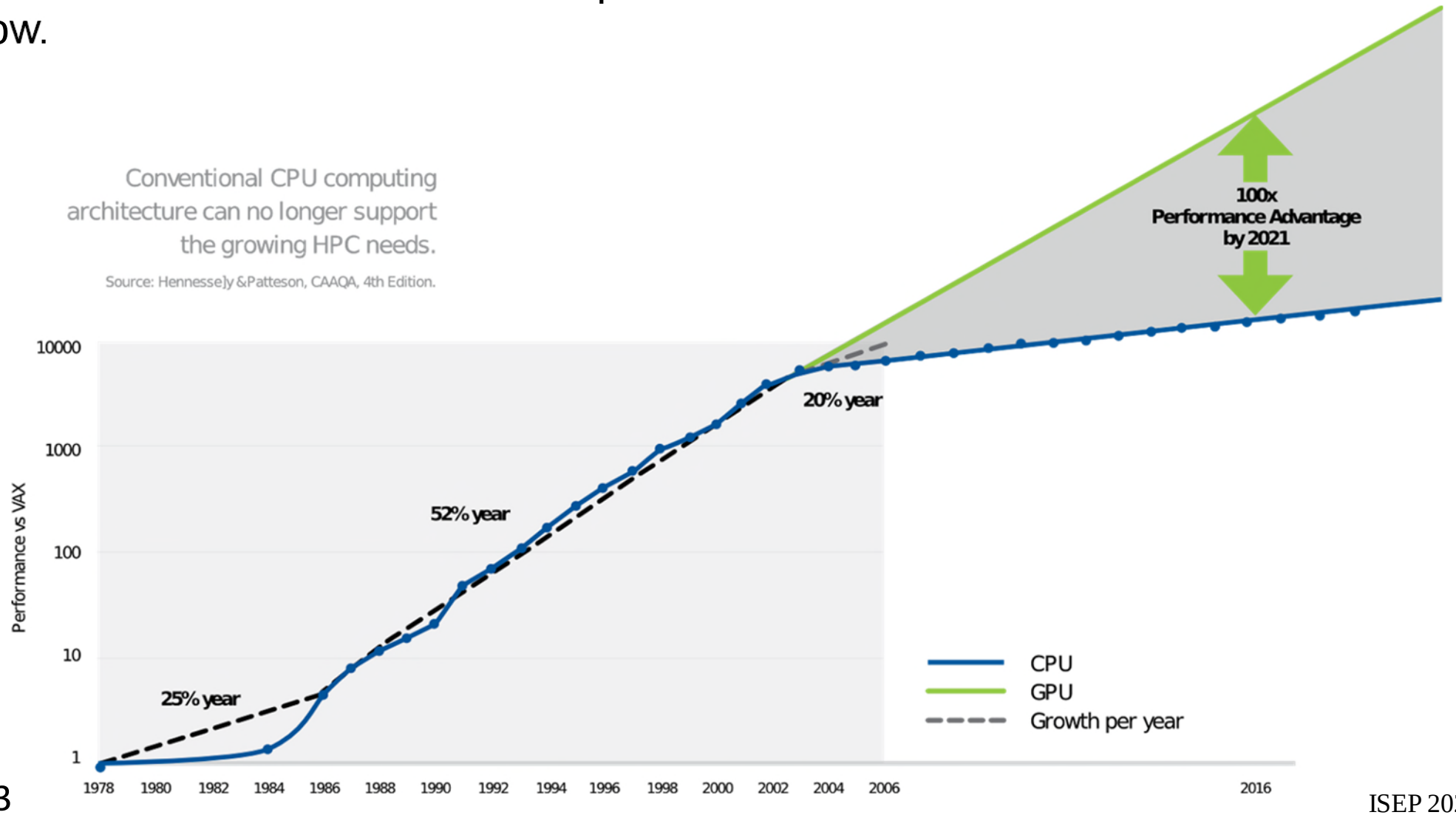
Whatever number of modules (and SW dvpt team),
SW cannot be developed independently and integrated later, even if the interfaces are well defined.

- Versioning & configuration tools for version are mandatory to achieve expected level of quality & maturity
- Validation & test effort : Snake analysis, testing trap, automated testing
  Hacking
- Telematics Hacking (Done by external Network, ex : 3G),
- Malware injection done through USB/SD port,
- Direct access through E-OBD connector (CAN – Packet injection)

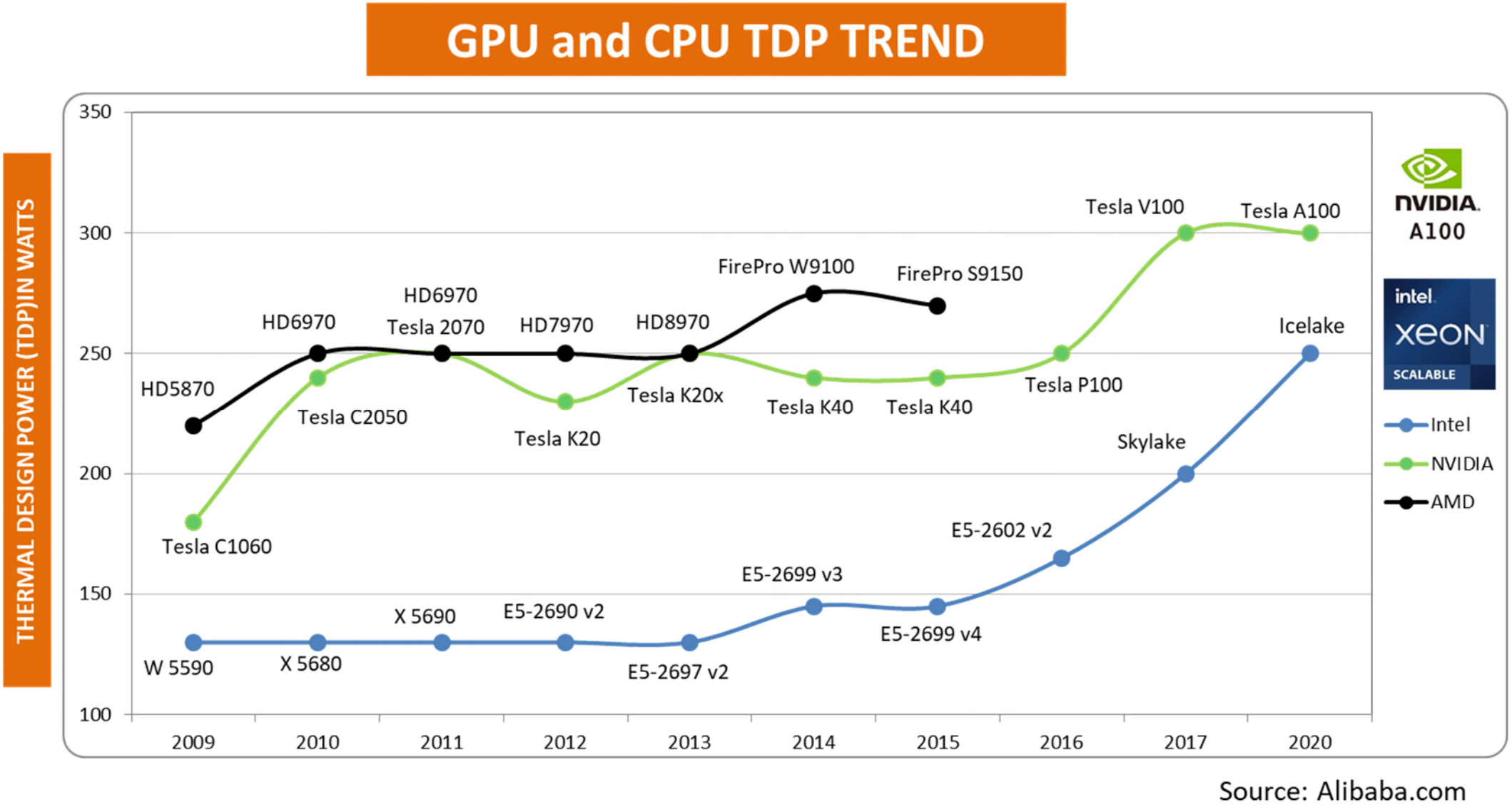
GPU - Roadmap



Race conditions (and needs !) for GPUs market lead to a very big rate to improve GPUs performances… much more CPUs.
Several hundreds cores can compute several Gb/s data flow.



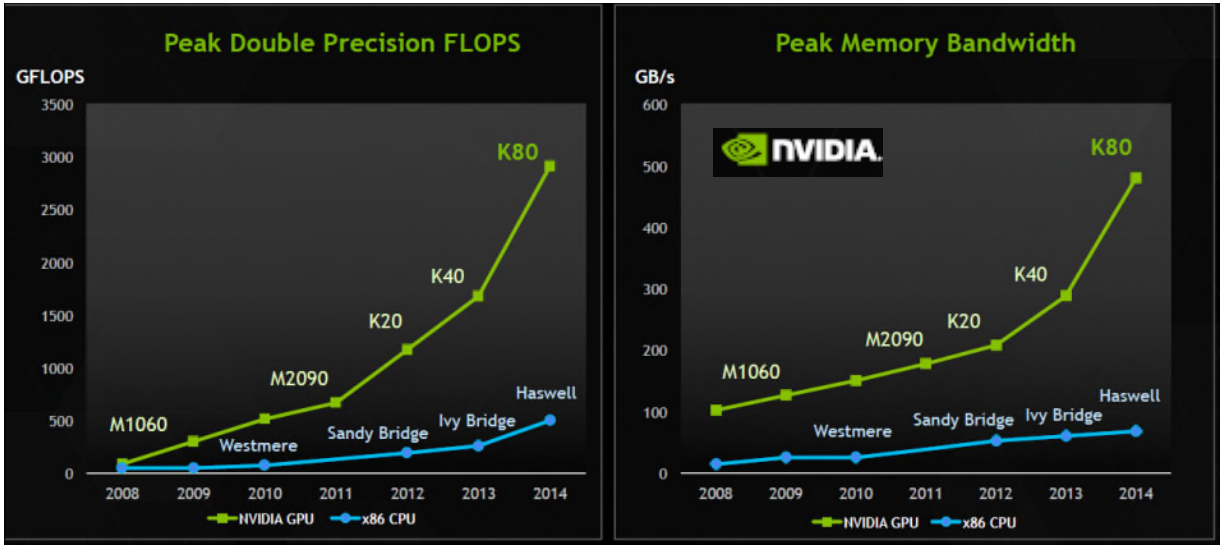
GPU – Thermal issue



Source: Alibaba.com

Latency reduction race (CPUs, GPUs, and shipset) push to increase physical density and temperature within ECUs.

Even with embedded applications, liquid cool is required to deal with tremendous amount of heat to spread out.



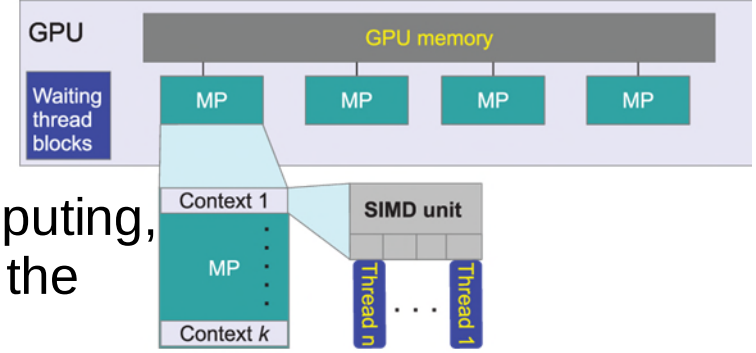
GPU

To obtain high performance, GPUs take benefits of **parallelism** in order to implement many parallel process to support many threads.

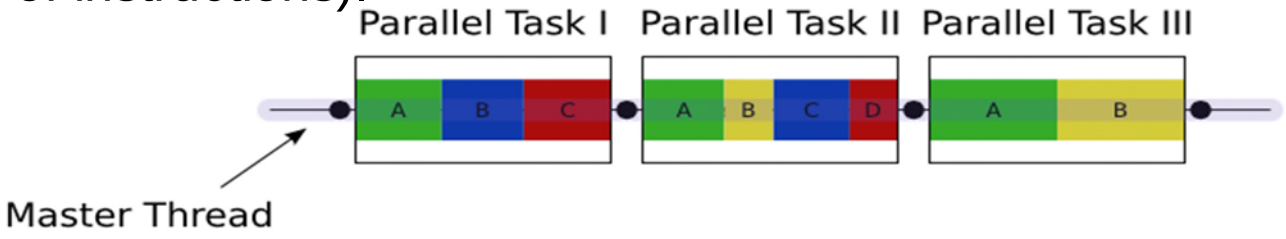
GPU is a **specialized** component, very powerful but for dedicated processing (mass data processing).

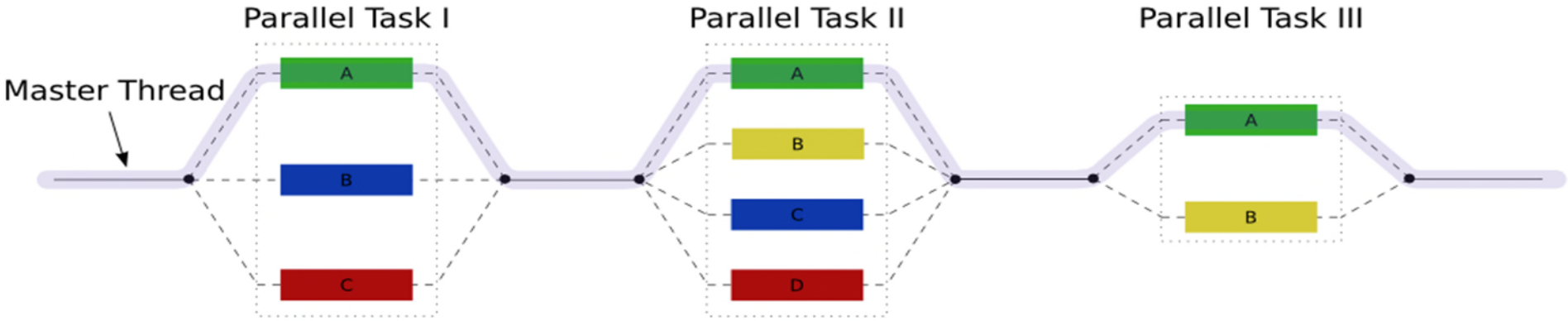
To meet these performances, designer have to use specialized peripheral IC (SRAM, Bus,…) and dedicated language (ex: CUDA).

GPU : Parallelism



Parallelism is a nice way to improve performance in computing, additional stuff can be added like pipelining (overlapping the execution of instructions).

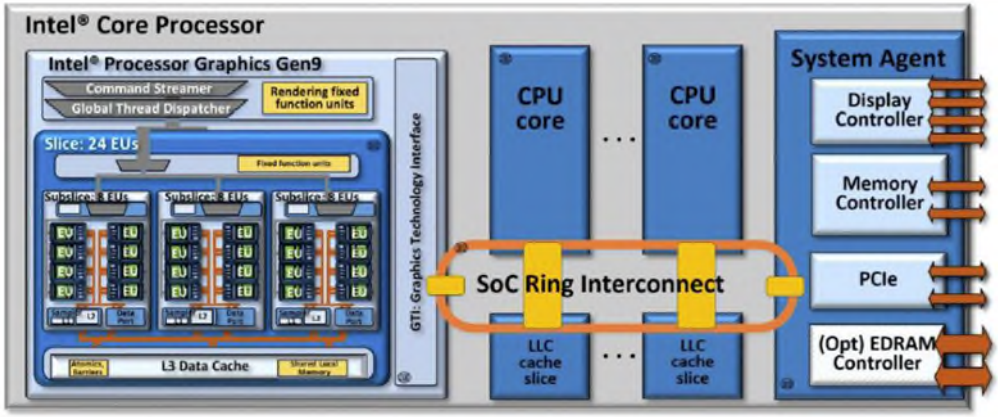


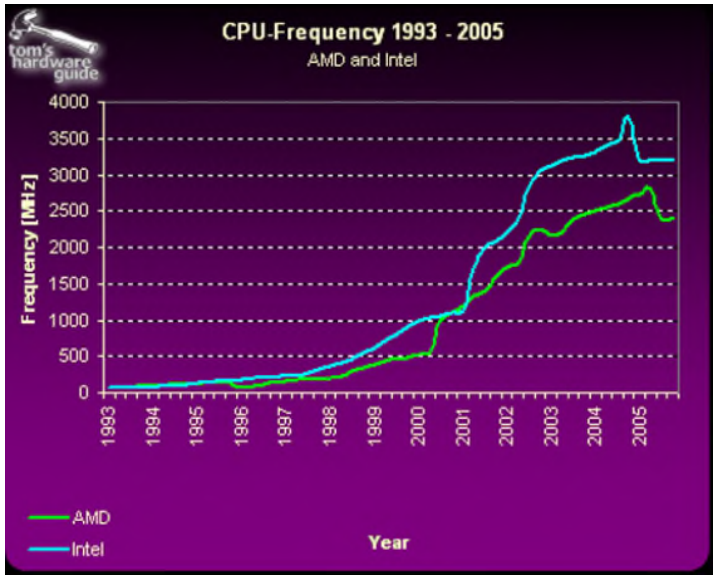


Parallelism software architecture is a completed different approach compared to a fully sequential way.

Improve performances with parallelism request to rewrite software, it's a real skill. In order to have all the benefits, the programmer must divide his application so that each processor has roughly the same amount to do at the same time, and that the overhead of scheduling and coordination doesn't waste the expected performance benefits of parallelism.

GPU





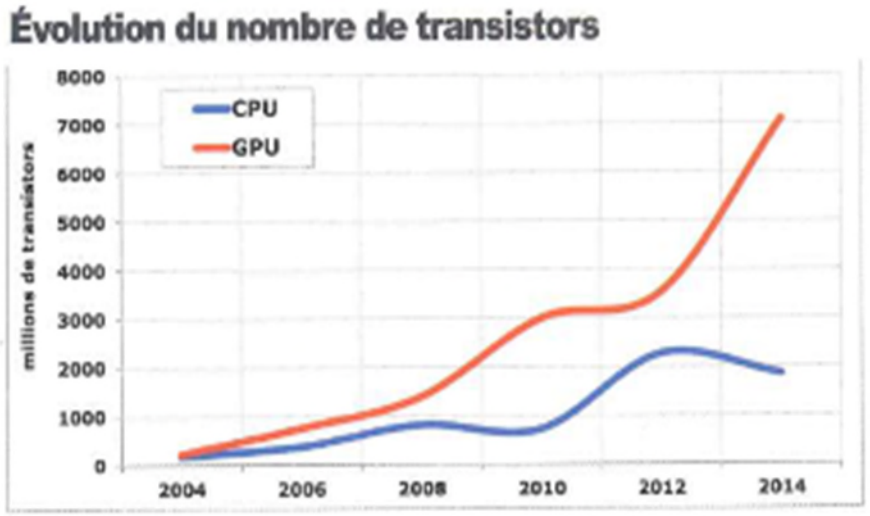
Silicon process for CPU and GPU are closed, but for architectural point of view, CPU have heterogeneous set up, that's completely different for GPU which are built up with similar blocs.

Inside the CPU, blocs density and thermal dissipation can be very different, these hot spots (~max ~90C) bring a limitation to increase frequency. GPUs are a set up of similar blocs relatively simples that's allow to spread thermal dissipation.

Die limit size = 600mm2, if not, huge problem of feasibility and cost could occur. Absolute thermal limit for an IC : ~130-150C, for GPU, margin included : 90C

GPU

GPUs are accelerators that supplement a CPU, so they do not need be able to perform all the tasks of a CPU. This role allows them to dedicate all their resources to graphics.



It’s fine for GPUs to perform some tasks poorly or not at all, given that in a system with both a CPU and a GPU, the CPU can do them if needed.

To enhance GPU performances, designer did not increase memory caches but had dedicated fast memory on ship.

Multiplication of cores allow to hide the memory latency.

- Between the time of a memory request and the time that data arrives, the GPU executes hundreds or thousands of threads that are independent of that request.

GPU – CUDA Matrix handling

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

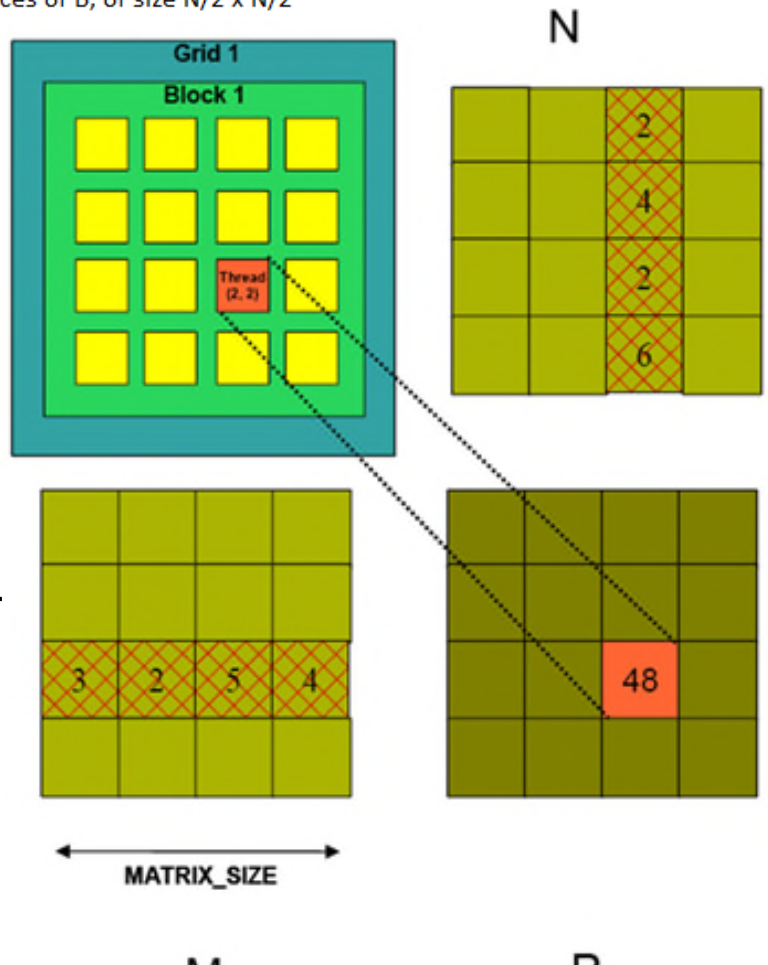
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + cz + d \\ ex + fy + gz + h \\ ix + jy + kz + l \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

A B C

A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2



One Block of threads compute a matrix P :
- Each thread computes one element of P

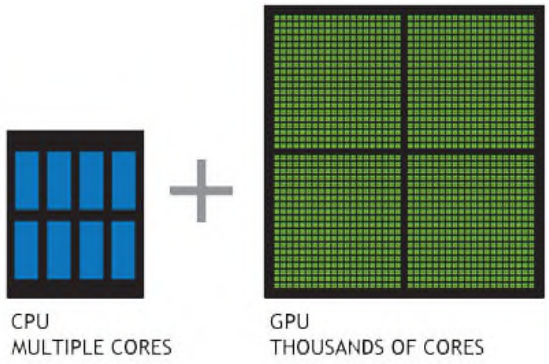
Each thread
- Loads a row of matrix M,
- Loads a column of matrix N,
- Perform one multiplication and addition for each pair of M and N elements
- Transfer to memory

Size of matrix limited by the number of threads allowed in a thread block

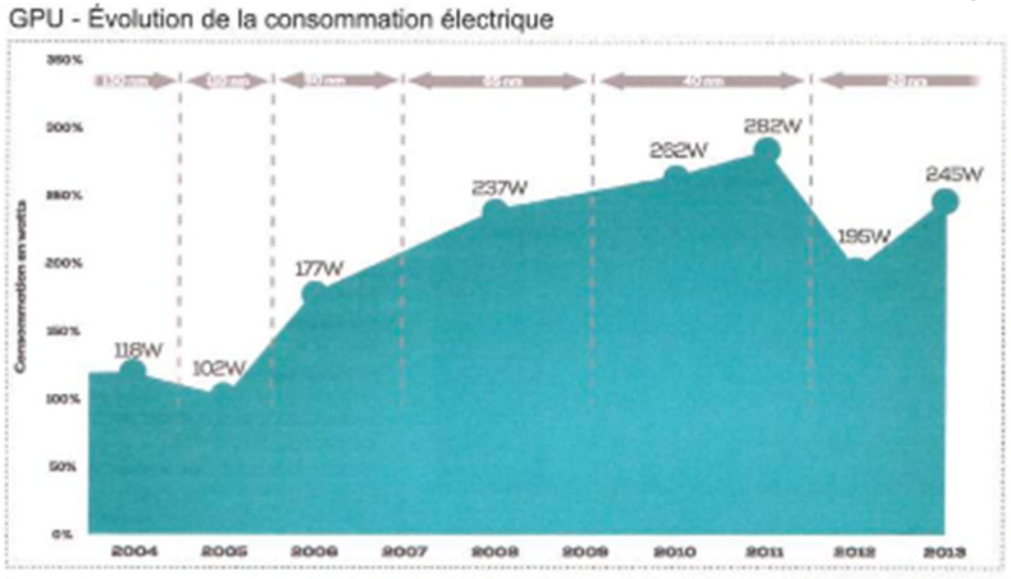
GPU vs CPU

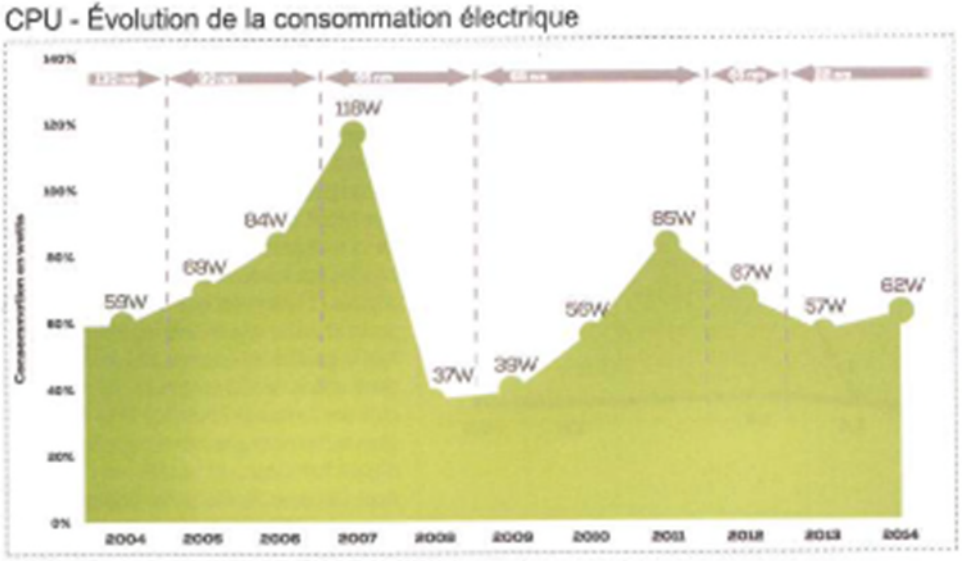
GPU and CPU are not in a race condition but much more complementary.
Anyway, using are different :

- CPU : Sequential oriented
- GPU : Intensive computing oriented



Last decade, silicon vendors have made huge efforts to decrease CPU power consumption





A GPU is composed of hundreds of cores that can handles thousands of threads simultaneously. Even if GPU cores runs slower than CPU, due to theirs numbers, thermal needs can reach several thousands of watts.

CPU vs GPU: What's the difference?

Autonomous vehicles require huge computation capabilities…

These are granted thru CPUs (Central Processing Unit), GPUs (Graphics Processing Unit) or FPGA. CPU run faster but GPU run bigger.

Central Processing Units (CPUs)

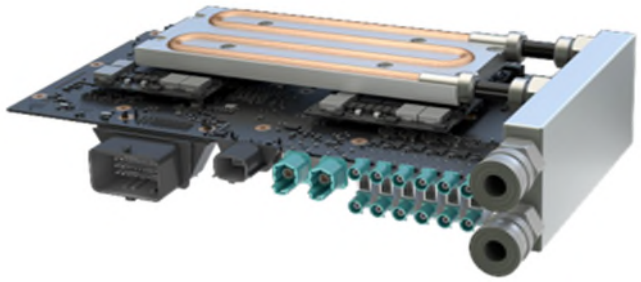
CPUs have 1 to > 24 cores, they have the capabilities to compute on integers, floats, etc.,...(ex: Intel x86 instructions set). CPU allow a very large span of computing, components are designed for standard thermal environment.

Graphic Processing Units (GPUs)

GPUs are very popular for desktop applications, dedicated for massive video computing, specially 3D processing. GPUs include several hundreds of specialized cores, which are very powerful for repetitive computations (ex : nVidia sous instructions set CUDA). GPUs are optimized for parallelism computation.

GPUs and Autonomous Vehicles

GPUs allow massive data computing, these are required for RADAR, LIDAR and CAMERAS data flow... Common situation is to handle ~1Gb/s data flow



GPU - NVIDIA DRIVE PX 2

12 CPU Cores, GPU Pascal, 8 Tflops (Core i7=280Gflops), 24 DL* TOPS, 16nm, 250w (Liquid cooled : Opérationnel jusqu'à 80° C ambiant). *: 24.1012 **d**eep **l**earning operations per second

2x Tegra *X2*

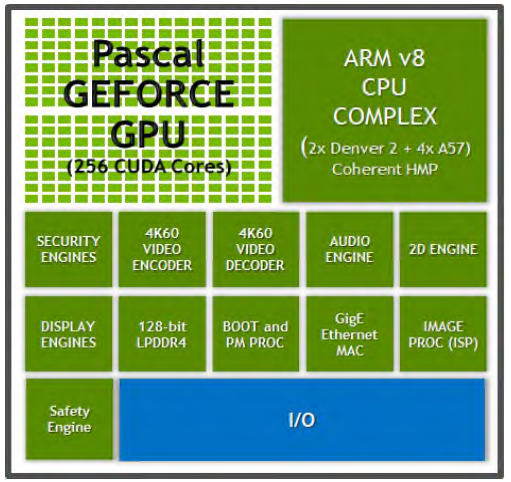
4x ARM Cortex-A57

2x NVIDIA *Denver2*

2x 3840-core Pascal GPU

8 TFLOPS (64-bit FP) 24 TFLOPS (16-bit FP)

1 GbE cluster interconnect



Integrated, a trained neural network model serving as deep learning platform, Drive PX 2 system is able to recognize up to 2,800 images per second

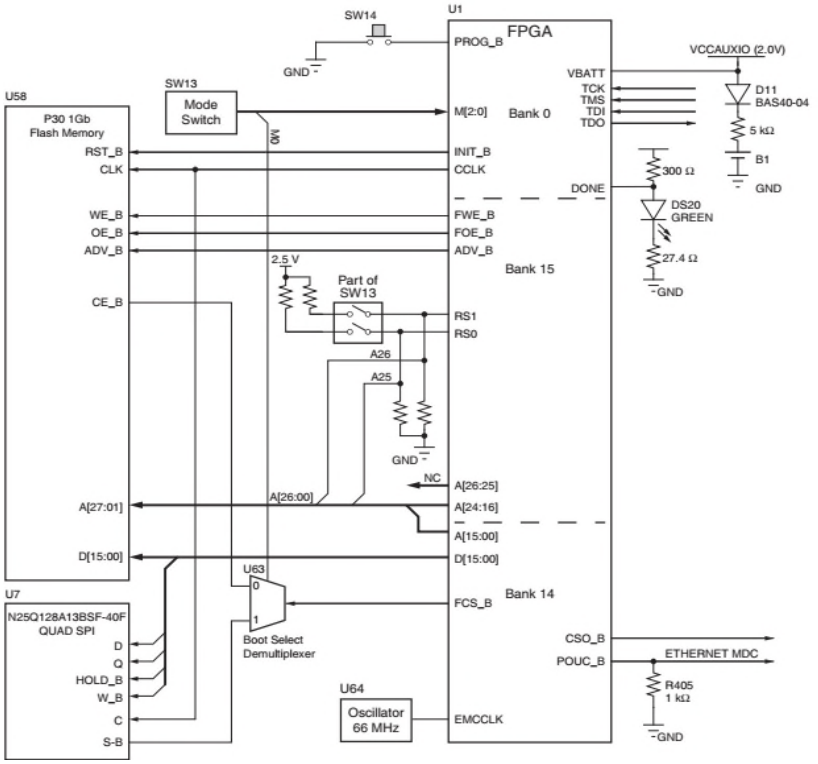
FPGA

FPGA (Fields Programmable Gate Array)

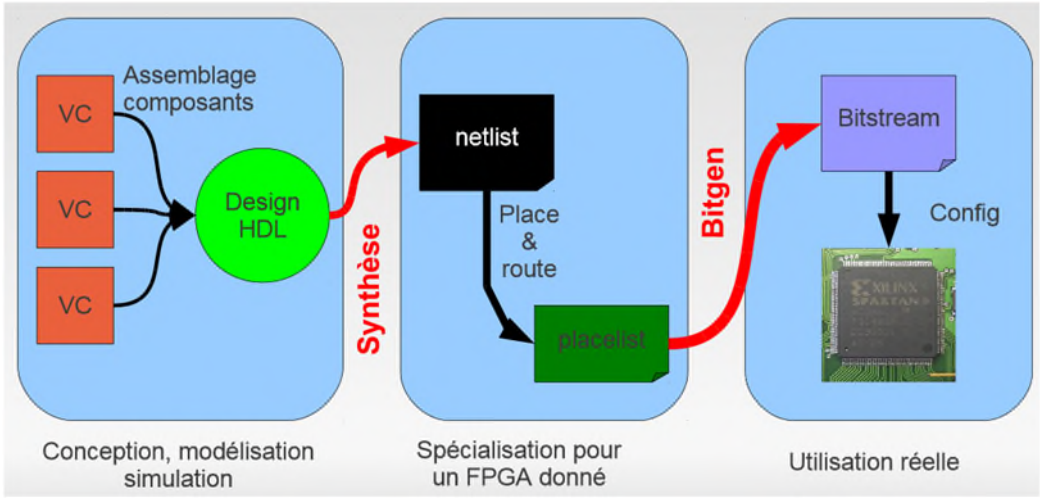
Configurable Integrated circuit that can emulate any Combinatorial and/or sequential logic circuit.

An FPGA is a “sea of gates” where a big amount of Logic Blocks can be connected.

Many FPGA include several complex blocks like : Memory, PLL, DSP, Com interfaces,…



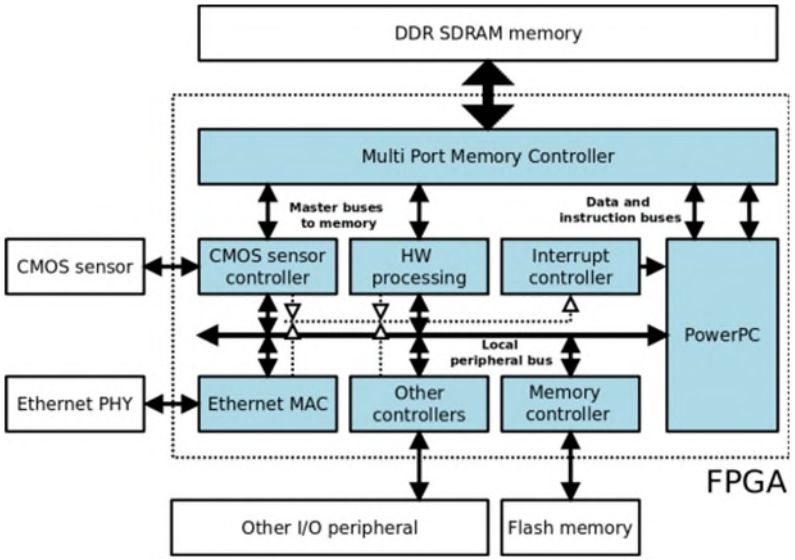
FPGA are reprogrammable, very suitable for prototyping, available on the shelf, Economical (for small volume of products) and more and more powerful.

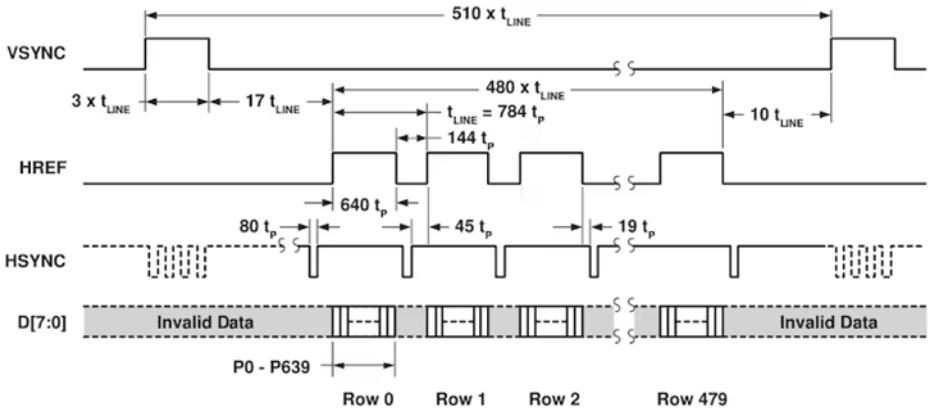


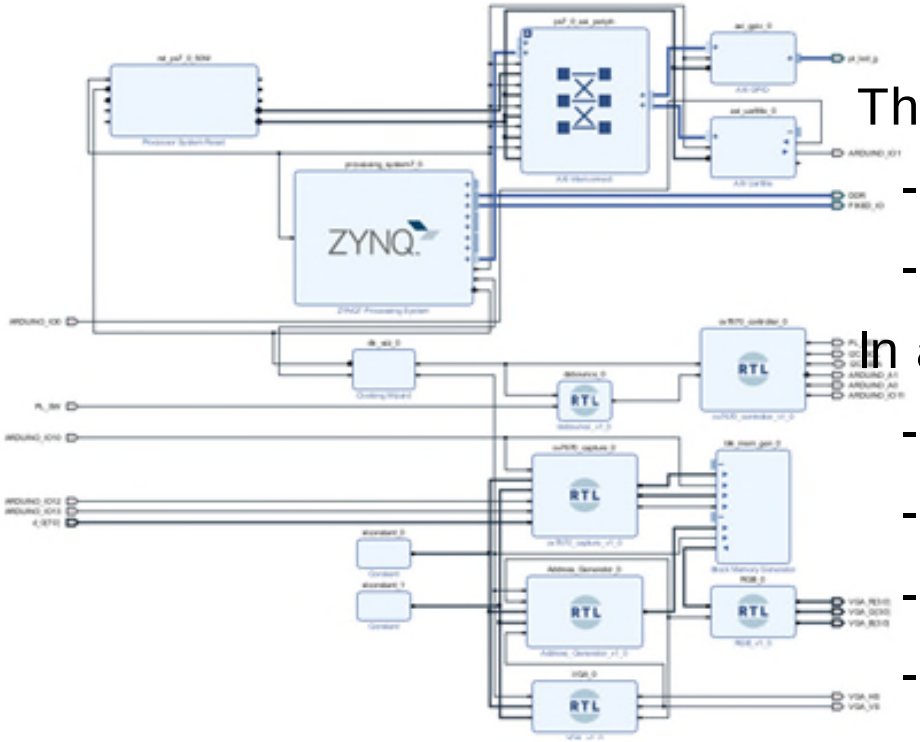
https://fr.wikipedia.org/wiki/VHDL

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

entity signed_adder is
  port
  (
    aclr : in   std_logic;
    clk  : in   std_logic;
    a    : in   std_logic_vector;
    b    : in   std_logic_vector;
    q    : out  std_logic_vector
  );
end signed_adder;

architecture signed_adder_arch of signed_adder is
  signal q_s : signed(a'high+1 downto 0); -- extra bit wide

begin -- architecture
  assert(a'length >= b'length)
    report "Port A must be the longer vector if different sizes!"
    severity FAILURE;
  q <= std_logic_vector(q_s);

  adding_proc:
  process (aclr, clk)
    begin
      if (aclr = '1') then
        q_s <= (others => '0');
      elsif rising_edge(clk) then
        q_s <= ('0'&signed(a)) + ('0'&signed(b));
      end if; -- clk'd
    end process;

end signed_adder_arch;
```

FPGA – Use Case (Camera)







The FPGA include :

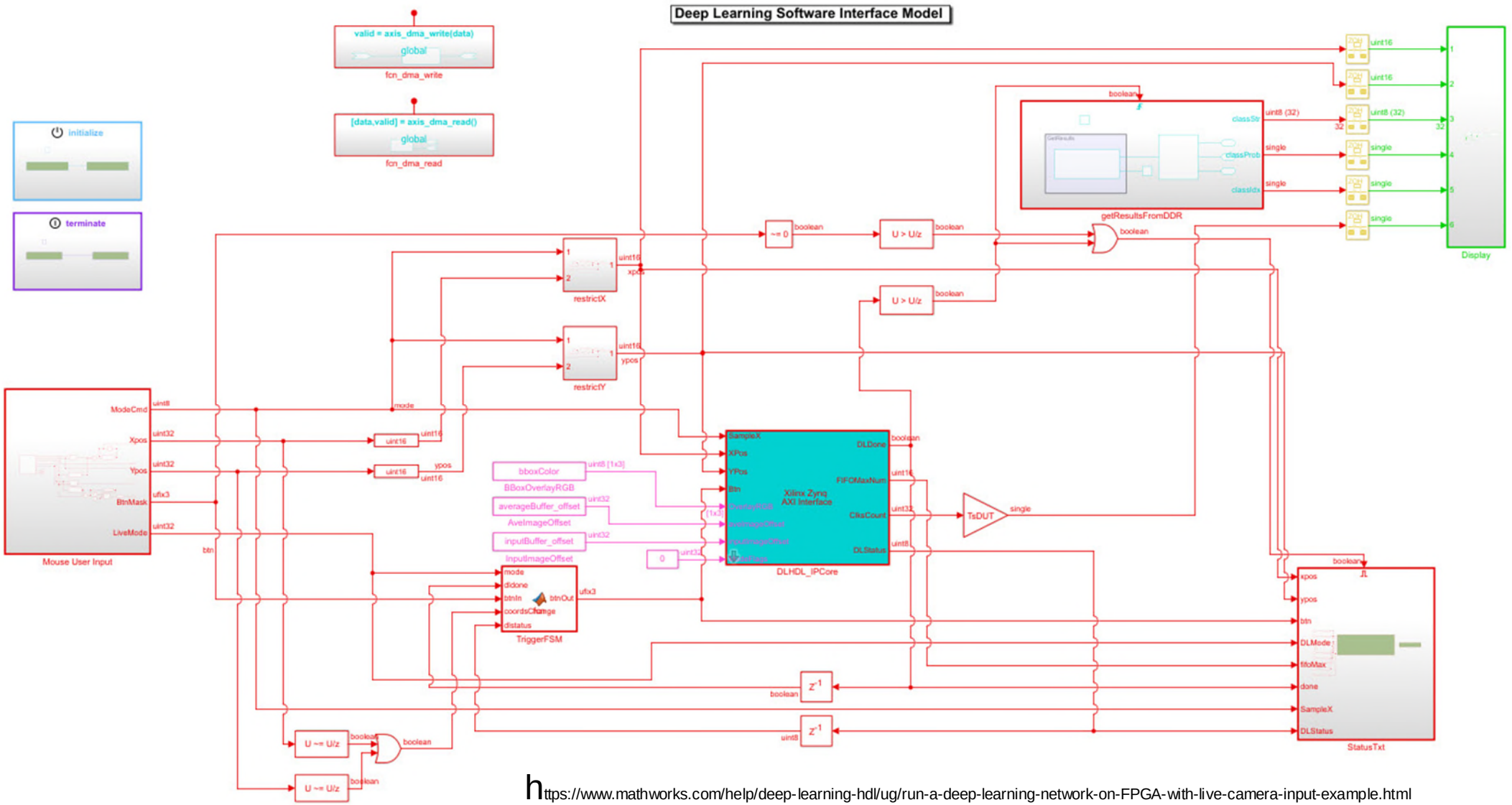
- Input capture image (transmit pixels Buffer),
- Camera configuration (via SCC Bus).

In addition, the FPGA can include

- Image shape recognition,
- Computing capabilities,
- Complex I/F (Ethernet, Flexray,…),
- …

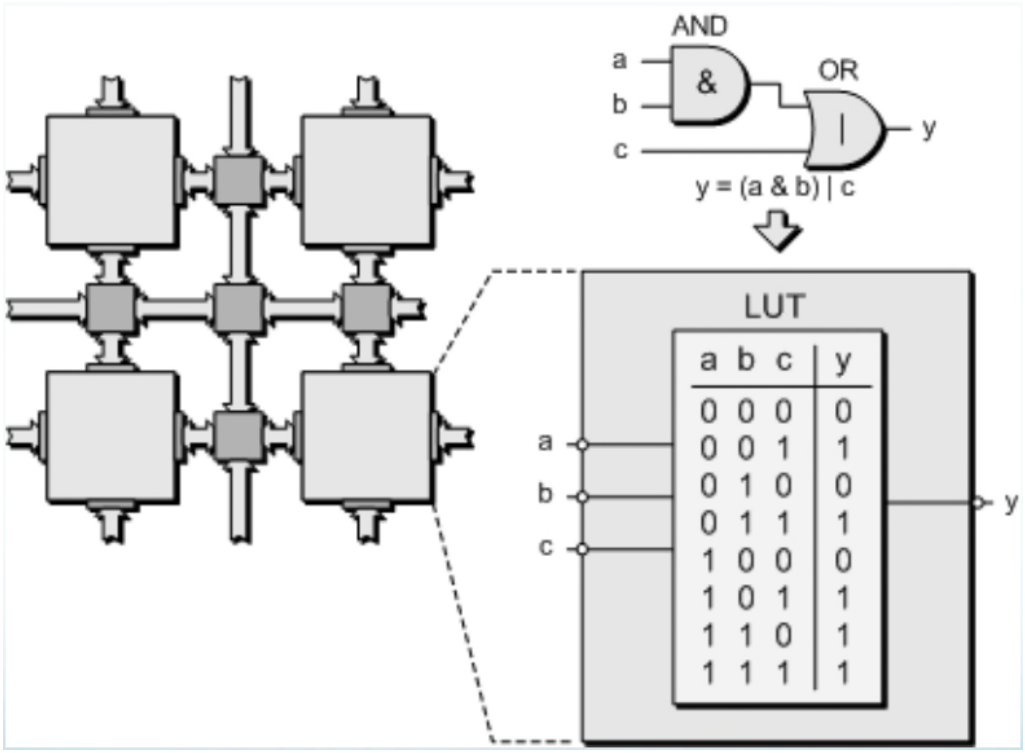
https://www.hackster.io/dhq/fpga-camera-system

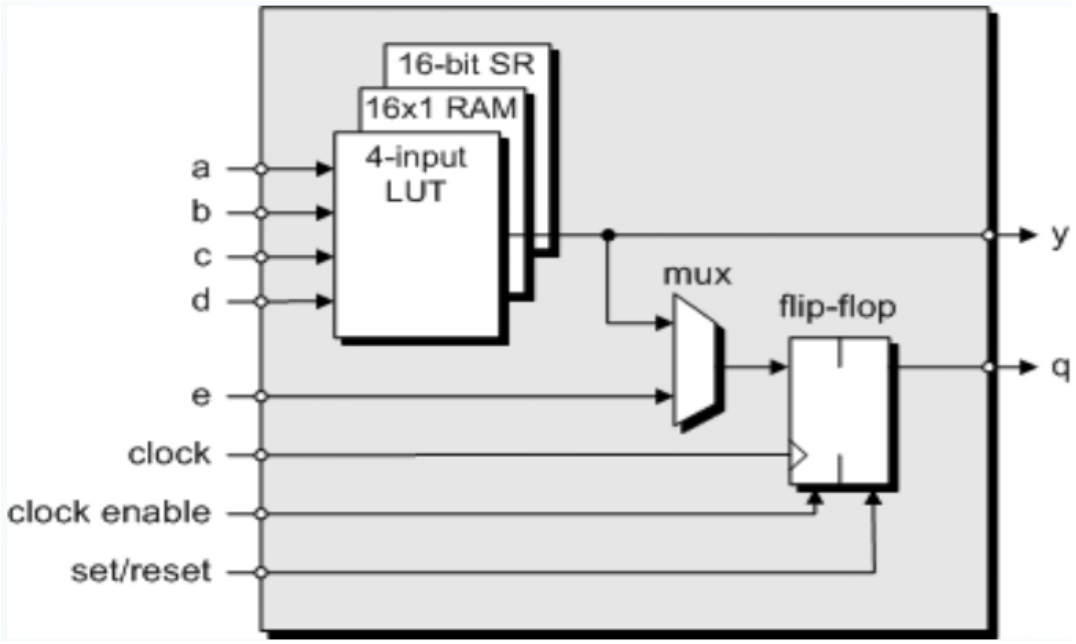
FPGA – Use Case (Deep Learning)

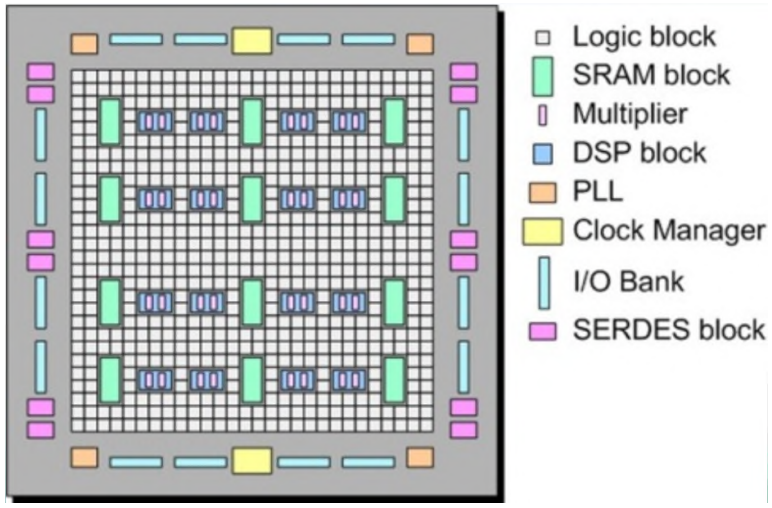


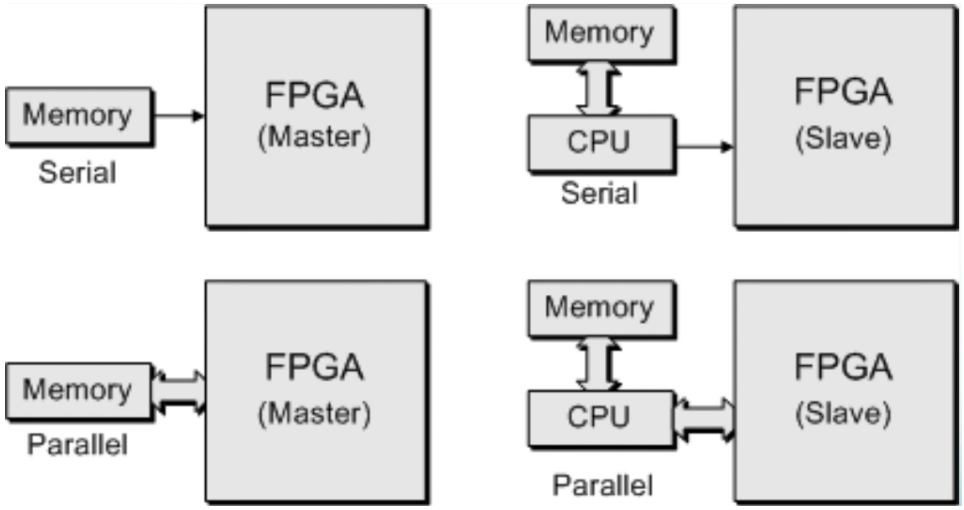
https://www.mathworks.com/help/deep-learning-hdl/ug/run-a-deep-learning-network-on-FPGA-with-live-camera-input-example.html

FPGA - LUT


AND
OR
a
b
c
&
|
y
y = (a & b) | c
LUT
a b c y
0 0 0 0
0 0 1 1
0 1 0 0
0 1 1 1
1 0 0 0
1 0 1 1
1 1 0 1
1 1 1 1


16-bit SR
16x1 RAM
4-input LUT
a
b
c
d
e
clock
clock enable
set/reset
mux
flip-flop
y
q


Logic block
SRAM block
Multiplier
DSP block
PLL
Clock Manager
I/O Bank
SERDES block


Memory
Serial
FPGA
(Master)
Memory
CPU
Serial
FPGA
(Slave)
Memory
Parallel
FPGA
(Master)
Memory
CPU
Parallel
FPGA
(Slave)

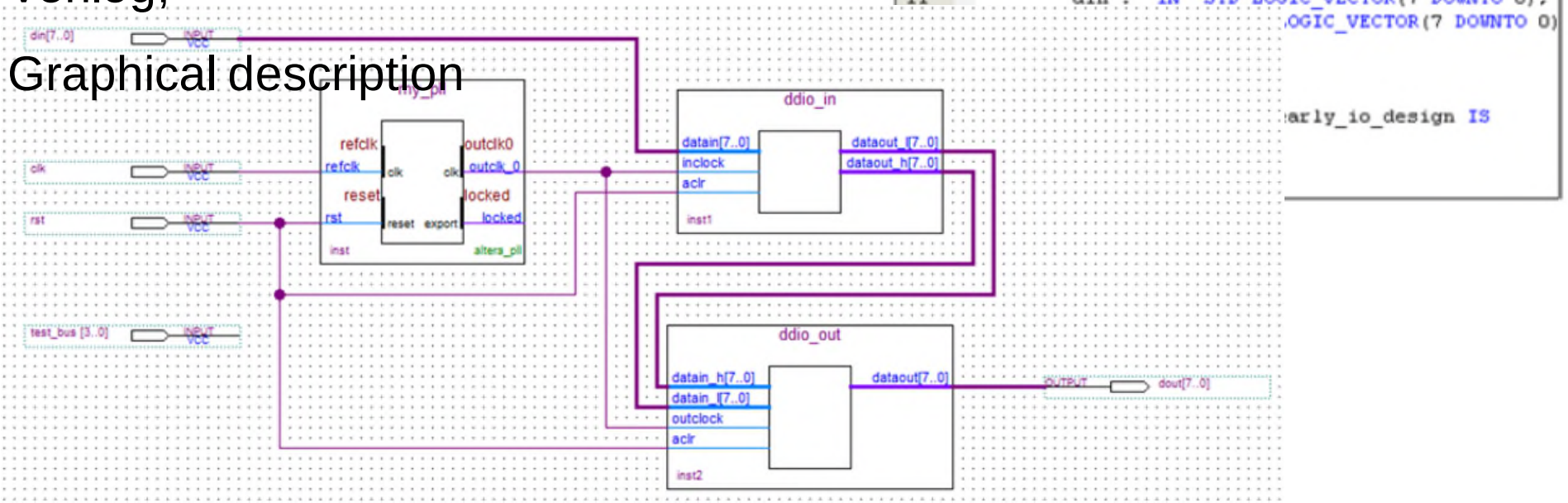
FPGA - Tool

Languages :

- ✓ VHDL (HW Descrip. Language),
- ✓ Verilog,
- ✓ Graphical description

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  LIBRARY work;
5
6  ENTITY early_io_design IS
7      PORT
8      (
9          clk :   IN  STD_LOGIC;
10         rst :   IN  STD_LOGIC;
11         din :   IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
```

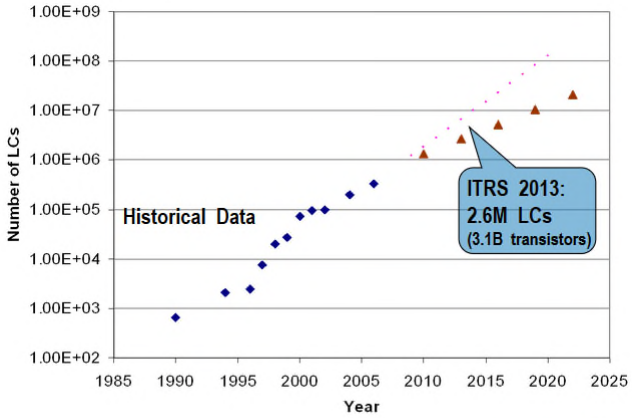
VHDL



VHDL is very popular, it describes what the hardware should do. Phases are :

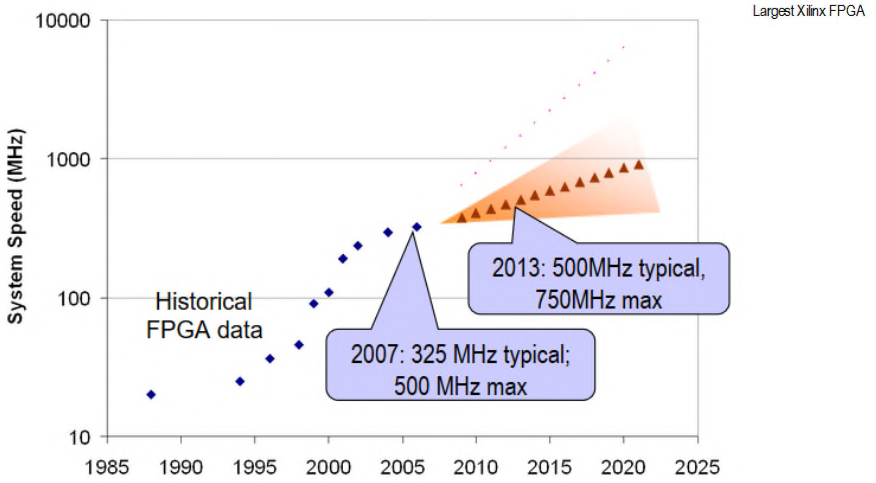
- Description, Syntax check,
- Compilation (Netlist),
- Translate & Map (Fits the design to target architecture)
- Place & Route (Functions Gates)

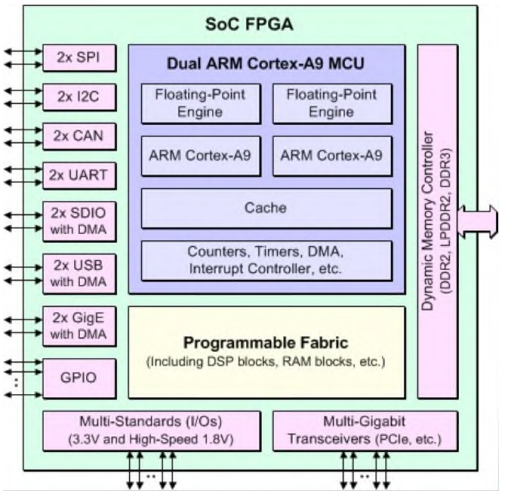
FPGA - Trends



Main improvement available (or incoming) to deal with designer expectations :

- ✓ Peripherals integration (SoC),
- ✓ IPs (ex: PCI express bus, processor,…),
- ✓ Graphical description
- ✓ Tools to mitigate development
  - 50% of the time spent for verification
- ✓ Configuration loading security





FPGA are powerful (several billions of transistors), faster than GPU while they are (HW)reprogrammable.

FPGA become more and more popular as accelerators.

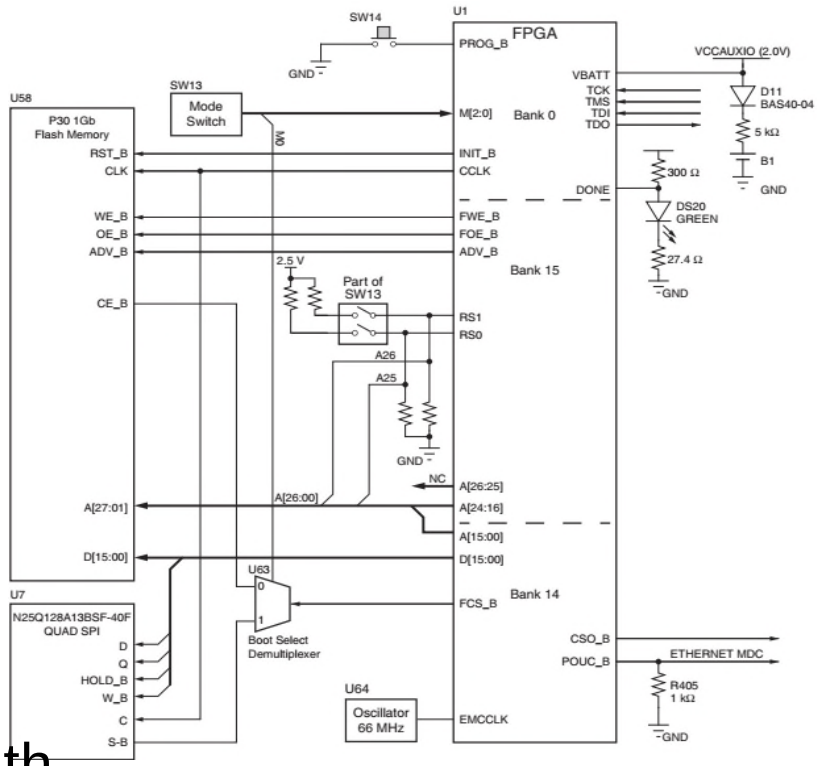
Rmk : It's a hard stuff to bench FPGA and choose it !

FPGA - Schematic

The FPGA require several additional ICs :

- ✓ Memories (and ext memory controllers),
- ✓ Control flow (ex :µC),
- ✓ Clock,
- ✓ Converters
- ✓ Processor (ARM, PowerPC,…)
- ✓ …

Modern FPGA integrate more an more peripherals.



Due to huge numbers of cells, interconnexion path, on-ship peripherals,… FPGA have a lot of pins (constraint for PCB layout design)

One important issue :

- The FPGA don’t save their configuration, at power on, an FPGA starts as a sea of gates (dead brain effect).

This limitation imposes additional stuff to download the configuration

- Can be done with a microprocessor or CPLD (bit-stream).

CPU - Comparison

	CPU	MultiCores	GPU	FPGA	ASIC
Computing (FLOp/s)	-	+	+	+	++
Floats	-	+	++	+	++
Parallelism	-	+	++	++	+
Matrix handling	-	+	++	+	+
Throughput	-	+	++	++	++
Power in/Flops	-	-	+ (cooling)	++	++
Libraries	+	+	++	-	--
Cost	$	$$ (scalable)	$$	$$ (prg)	$$$ (dvpt)

Criteria to choose :

- Low cost applications, no need to use an accelerator CPU,
- Application where some of computation need to get a boost CPU+GPU,
- High latency processing / direct Hard interface FPGA

For the most critical application : FPGA covers the front-end of data, GPU computes these data and CPU control the system.

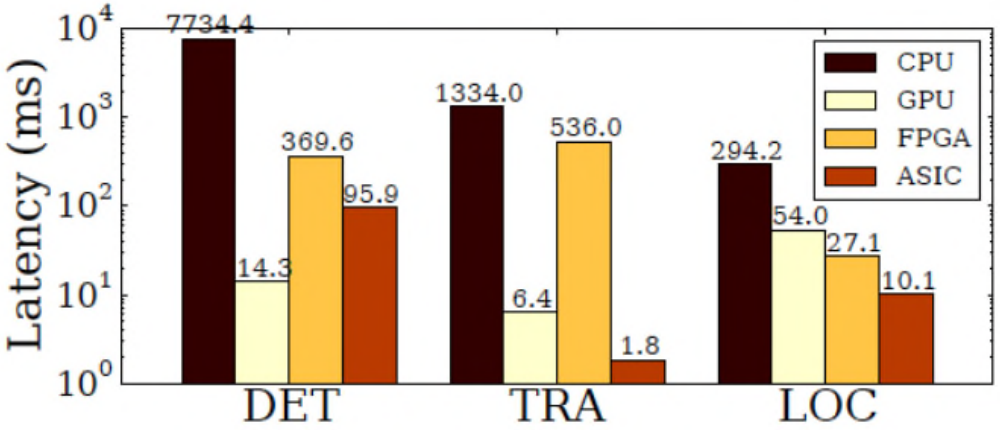
CPU – Comparison – Object tracking

ASIC : Power consumption & Latency

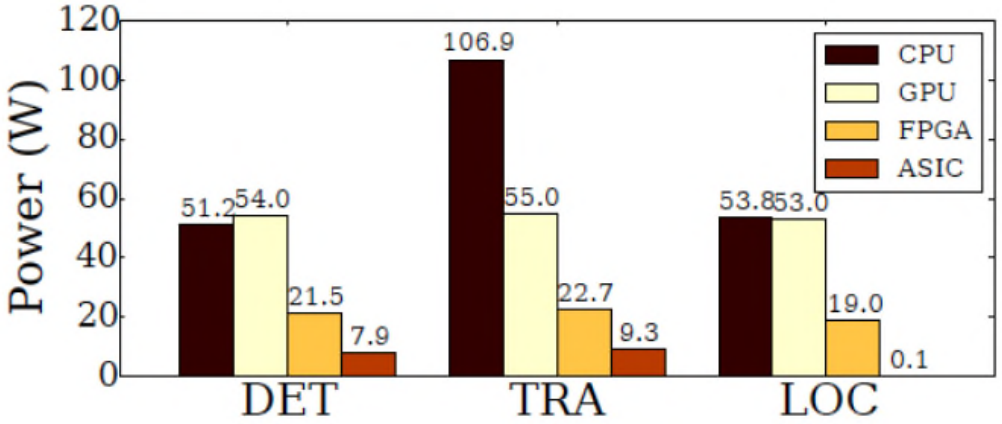
GPU : Latency

FPGA : Power consumption

CPU : Worst everywhere but…
can compute everything.

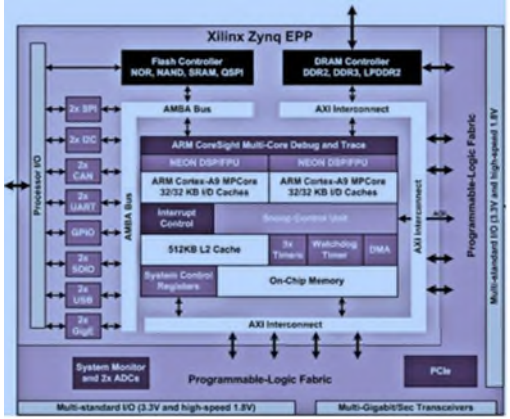


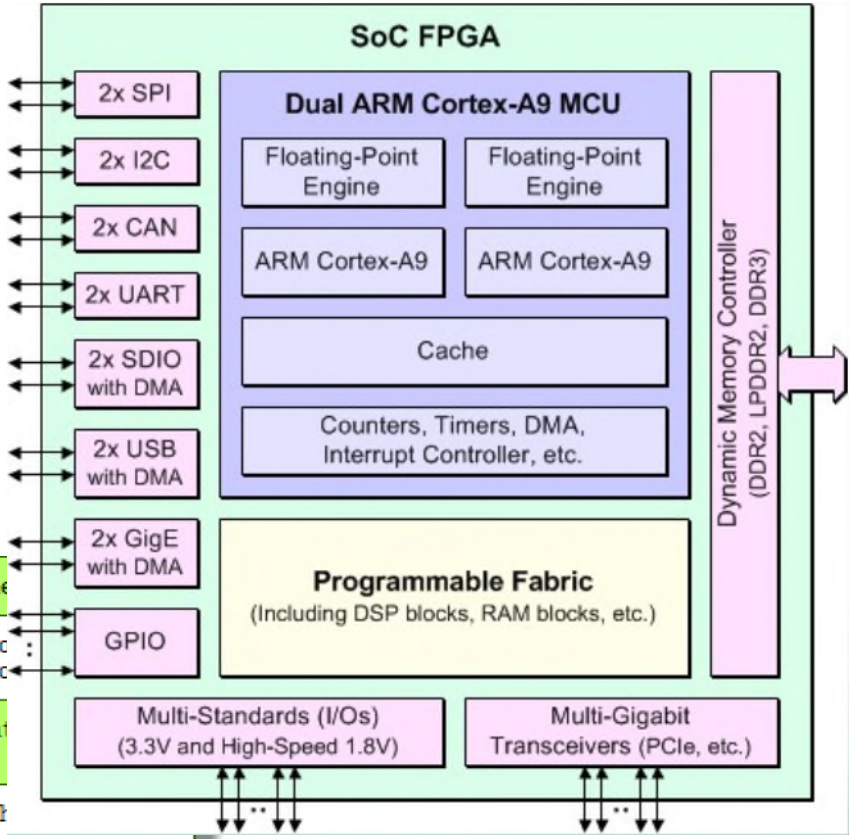
Object **Det**ection,
Tracking,
LOC : Feature Extraction Tasks



Source : "The Architectural Implications of Autonomous Driving." 2018

AV computation platform : Highly heterogeneous

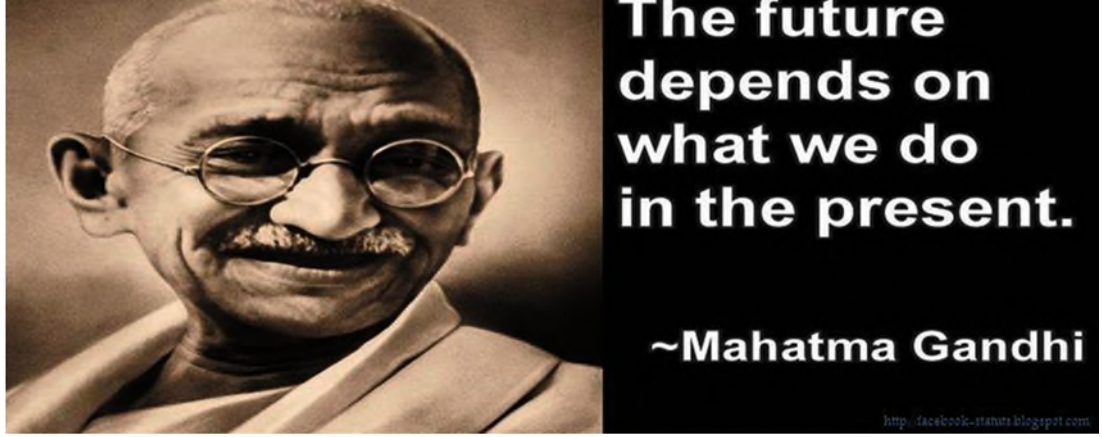




FPGA vs µController

Microprocessor is heart of Computer system.	Micro Controller is a heart of embedde
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processo with internal memory and i/O compo
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present in the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and i an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

Conclusion



- ADAS systems are more and more present in several sectors: drones, Subway, Space probes, ground Vehicles...,...
- The electronic complexity of the systems increases in a exponential way with the expected level of autonomy,
- The necessary embedded electronics parts in the highest level of autonomy requires the most successful technologies of the market, specially for computing capabilities and safety.
- Cost target : Shift to current price of ~10k€ to 5k€ in 2030.