Power efficient software acceleration using programmable logic in the Exascale Era

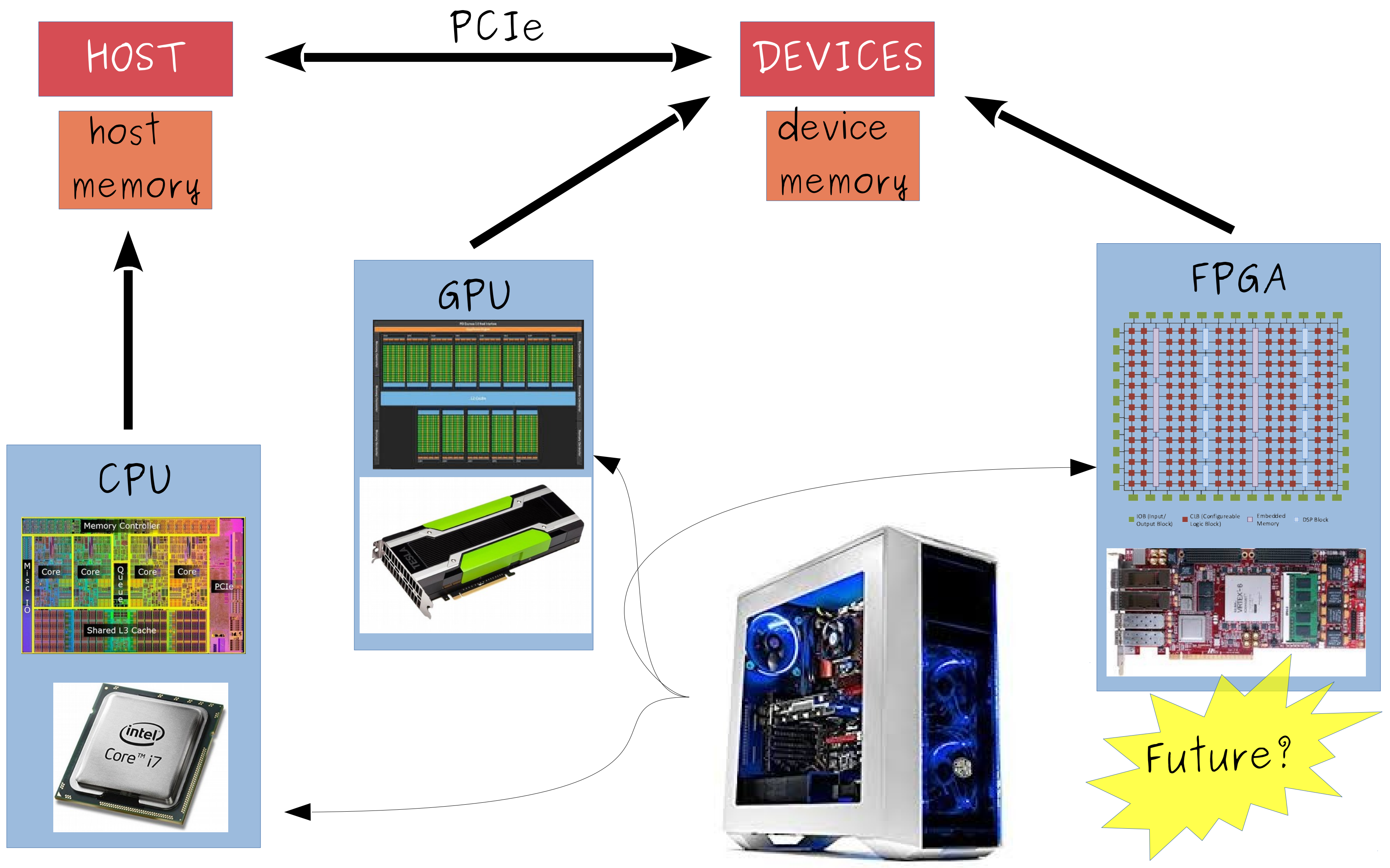
David Goz,
INAF – OATs

ASTRO@TS

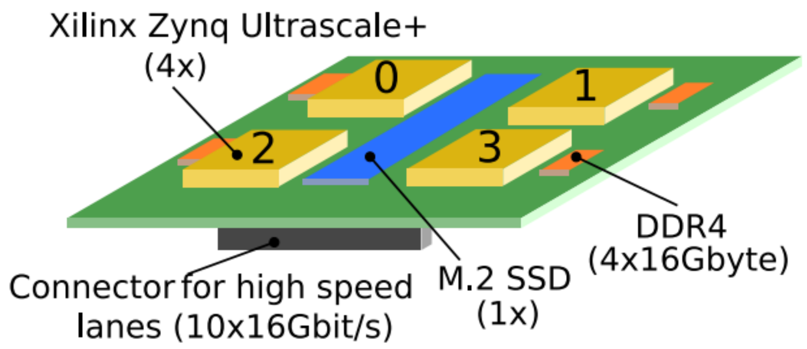
25 September 2017

Horizon 2020

Heterogeneous hardware



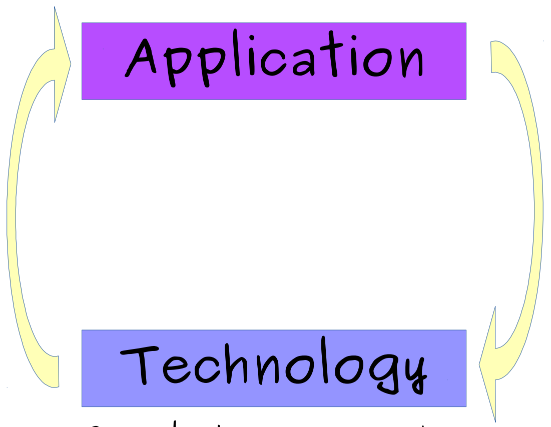
ExaNeSt hardware and co-design approach



The ExaNeSt compute Quad-FPGA daughter-board.

ExaNeSt compute unit:

- 4 Xilinx Zynq Ultrascale+ FPGAs;
- 4 ARMv8 cores @1.5GHz per FPGA (thus 16 cores);
- 16 GB of DDR4 memory per FPGA (thus 64 GB);
- one NVM SSD storage device;



Co-design approach.

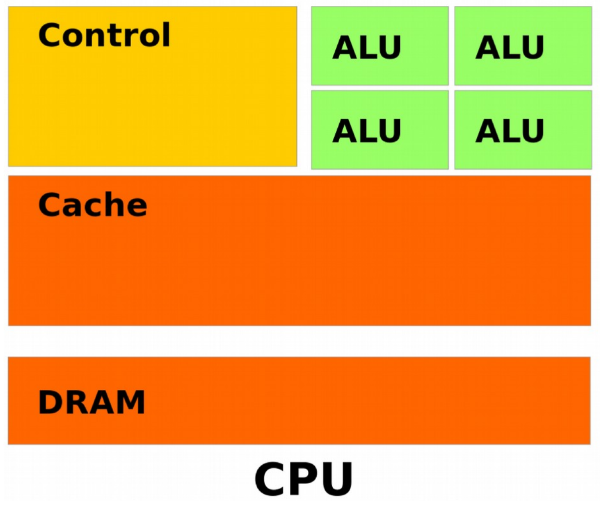
Co-design approach:

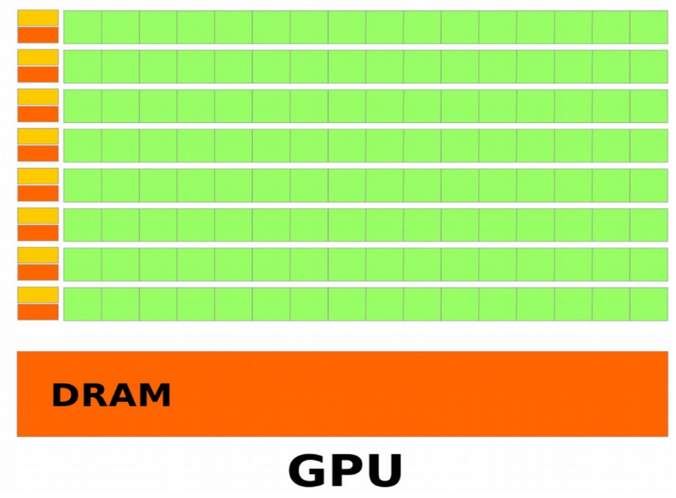
- new algorithms & applications;
- better technology and performance.

Sub-optimal approaches:

- application driven: find the best technology to run this code;
- technology driven: fit your application to this technology.

What are the differences between CPU and GPU?





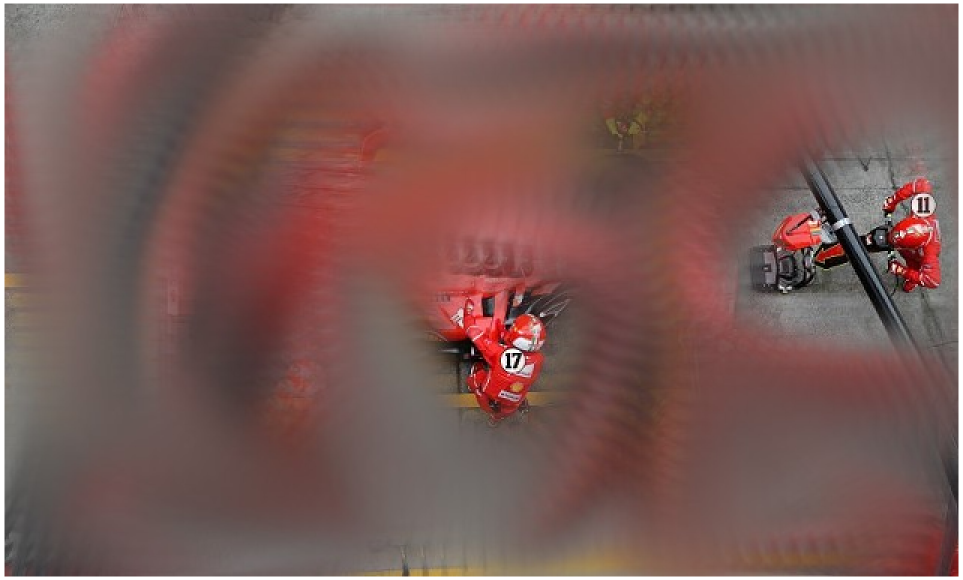
- CPU is latency-optimized (each thread runs as fast as possible, but only few threads);
- CPU has few cores (≤ 16);
- CPU excels at irregular control-intensive work (lots of hardware of control, few ALUs);
- Programming languages: C/C++, Fortran, Python, IDL, …
- Parallel libraries/directives: MPI/OpenMP.

- GPU has highly data-parallel fixed architecture (SIMD);
- GPU is throughput-optimized (thousands of threads, hundreds of cores);
- GPU excels at regular math-intensive work (lots of ALUs for math, little hardware control);
- Very high memory bandwidth (drawback for power consumption);
- Parallel programming: OpenACC (directives), CUDA, OpenCL (low level programming for high performance).

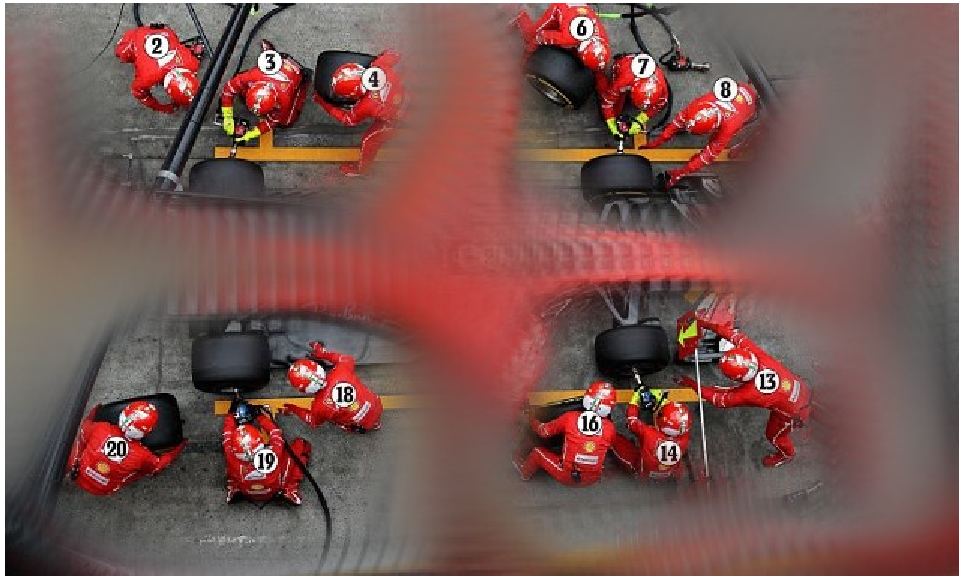
A high-performance problem solved in parallel

Two types of parallelism

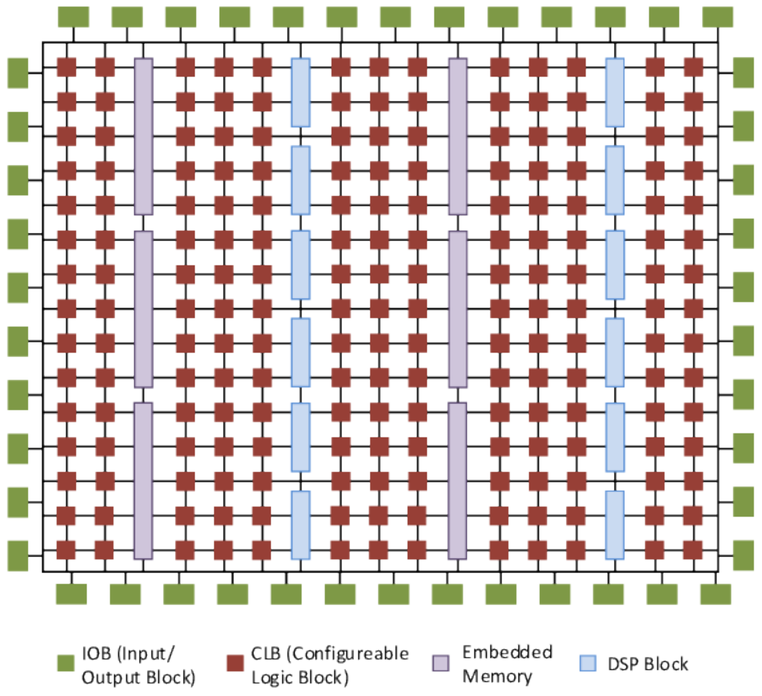
- Task parallelism: different people are performing different tasks at the same time.
  Work suitable for **CPU.**



- Data parallelism: different people are performing the same task at the same time, but on different equivalent and independent data.
  Work suitable for **GPU.**



What is a Field Programmable Gate Array (FPGA)?



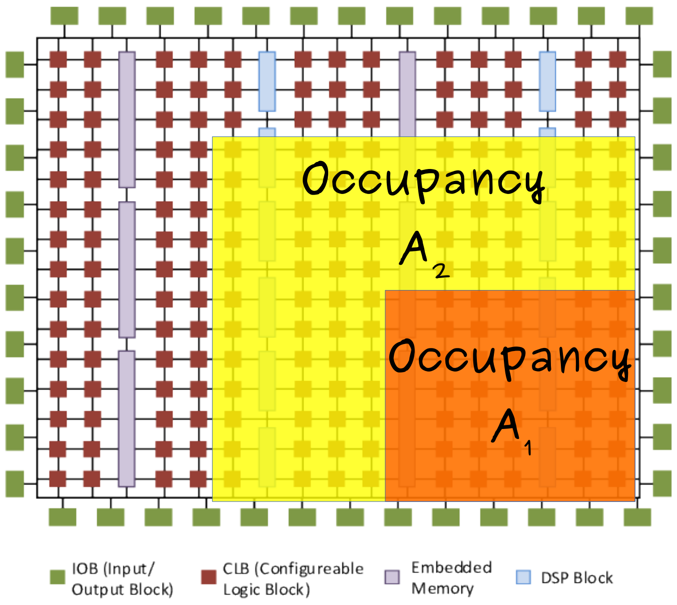
FPGAs are semiconductor devices that can be programmed (i.e. no fixed architecture):

- desired functionality of the FPGA can be (re)-programmed by downloading a configuration into the device;
- highly parallel customizable architecture (both data and task parallel computation);
- optimal power efficiency (3-4x than of GPU).

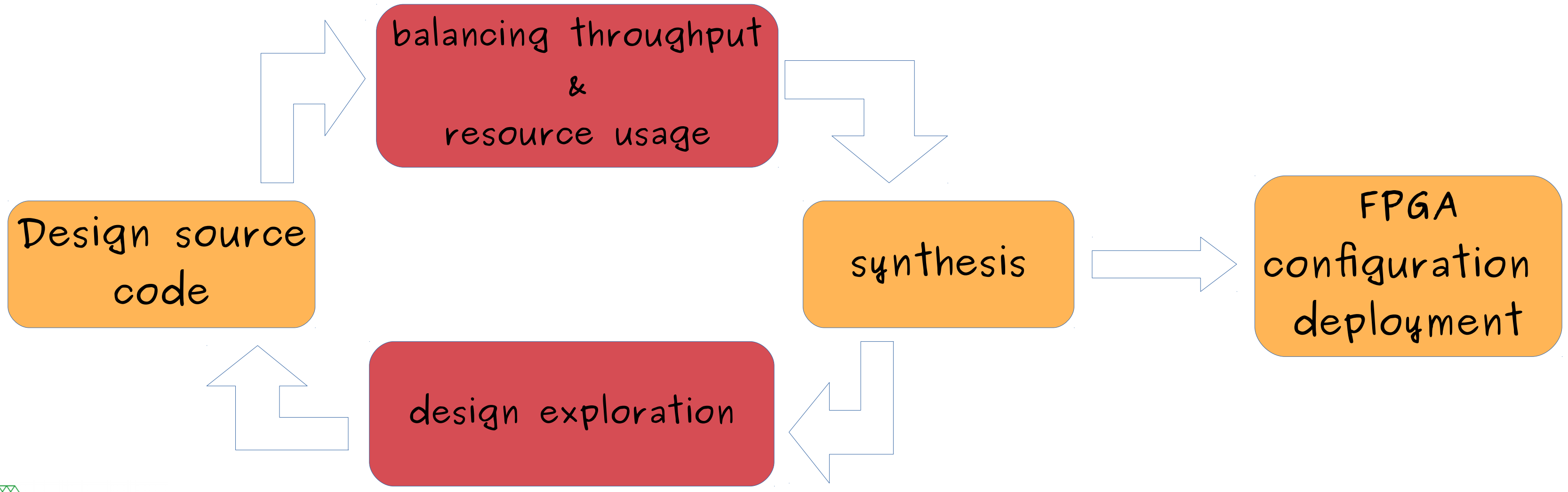
Main drawbacks:

- low level programming required for high performance;
- currently double-precision arithmetic is resource-eager and performance-poor (dramatic for scientific calculations!).

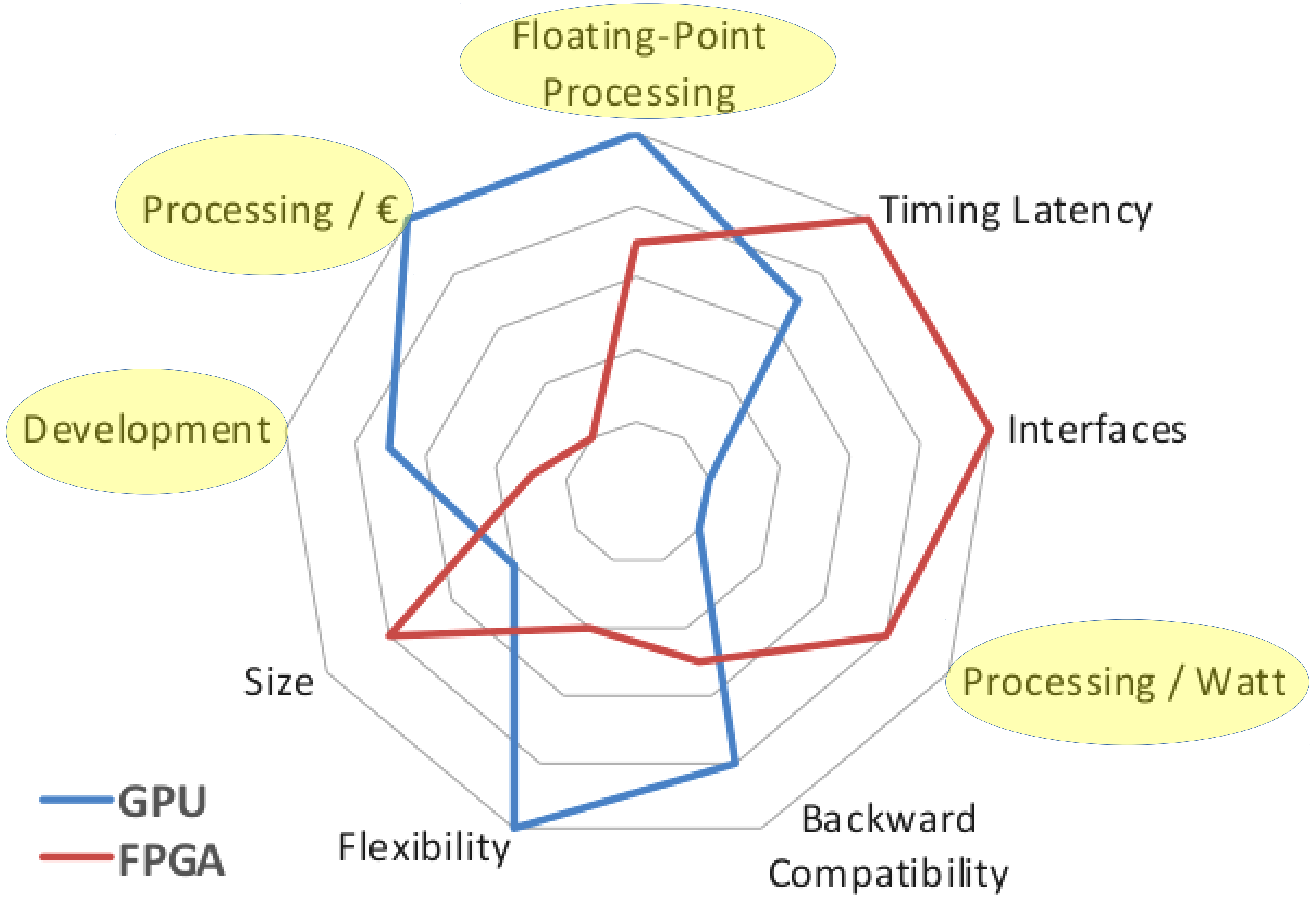
FPGA: flexible interconnect



FPGA architecture:
- millions of reconfigurable logic elements (flexible interconnect);
- multiple ARM cores;
- thousands of variable precision DSP blocks;
- thousands of memory blocks.


Design source code
balancing throughput & resource usage
synthesis
FPGA configuration deployment
design exploration

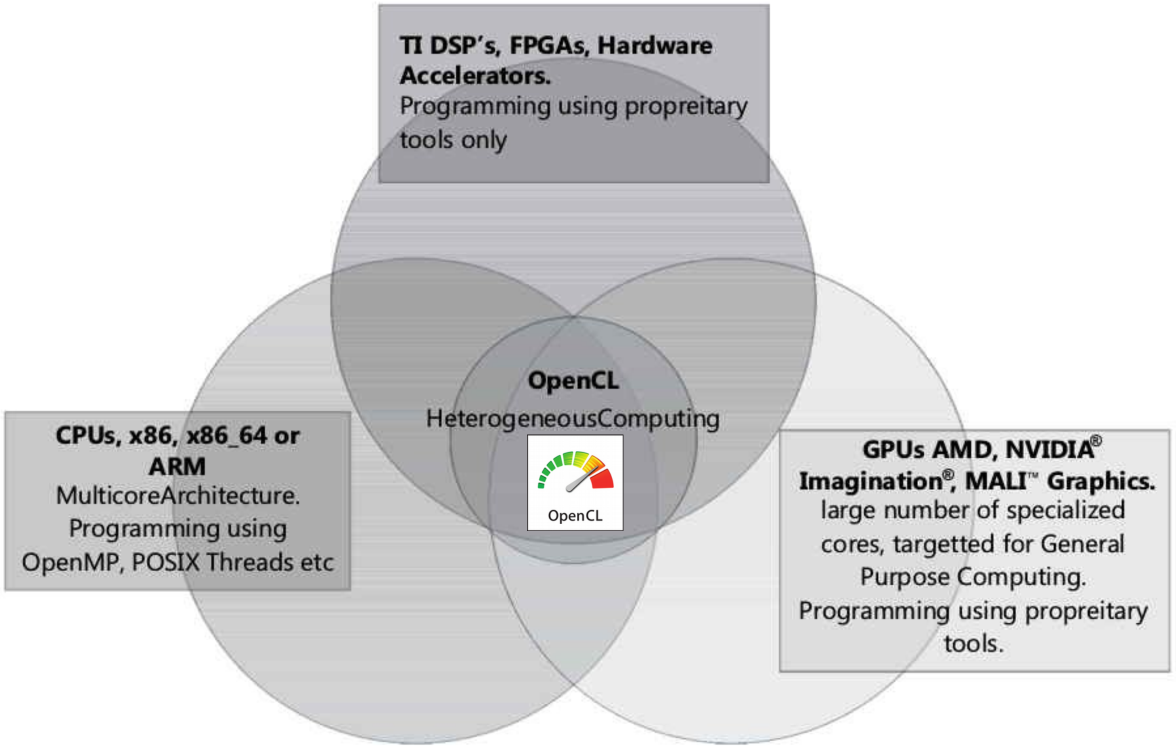
Qualitative comparison of GPU and FPGA



Credit http://www.bertendsp.com/

How to program in an heterogeneous system?

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms.



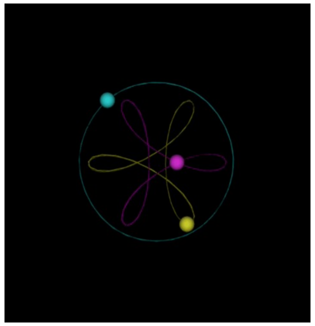
TI DSP's, FPGAs, Hardware Accelerators.
Programming using propreitary tools only
CPUs, x86, x86_64 or ARM
MulticoreArchitecture.
Programming using OpenMP, POSIX Threads etc
OpenCL
HeterogeneousComputing
OpenCL
GPUs AMD, NVIDIA® Imagination®, MALI™ Graphics.
large number of specialized cores, targetted for General Purpose Computing.
Programming using propreitary tools.

Heterogeneous computing (reproduced from "OpenCL programming by example").

- OpenCL is vendor agnostic (AMD, Intel, NVIDIA, ARM,…);
- OpenCL defines a language to write kernels running on different devices;
- OpenCL provides a standard interface for parallel computing using task- and data-based parallelism;
- kernels are portable across any OpenCL-compliant device;
- performance optimization is a huge topic!

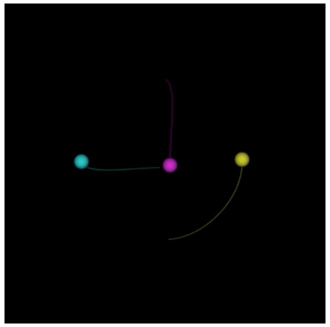
HiGPUs: N-body code in the ExaNest project

The numerical solution of the **direct** N-body problem is still considered a challenge (e.g. $O(N^2)$ computational cost).



N-body orbits

(http://www.princeton.edu/~rvdb/)

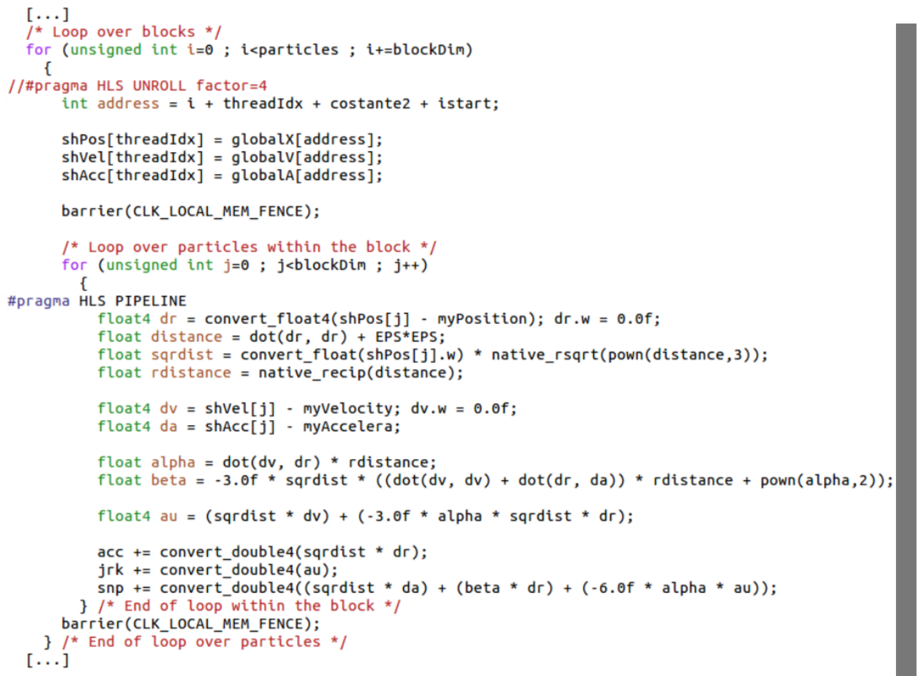


HiGPUs (R. Capuzzo-Dolcetta, M. Spera and D. Punzo, 2013) is a direct summation N-body code which employs a Hermite 6th order time integrator. Main features:

- **programming language**: C/C++, parallelized using MPI and OpenCL to exploit GPU clusters (100x gain over the CPUs);
- **parallelization scheme**: one MPI process per node;
- **precision**: double-precision (DP) in inter-particles distance and acceleration is used to minimize the round-off errors.

HiGPUs: N-body code in the ExaNest project

Aim: porting the most computationally demanding HiGPUs' kernel on FPGA



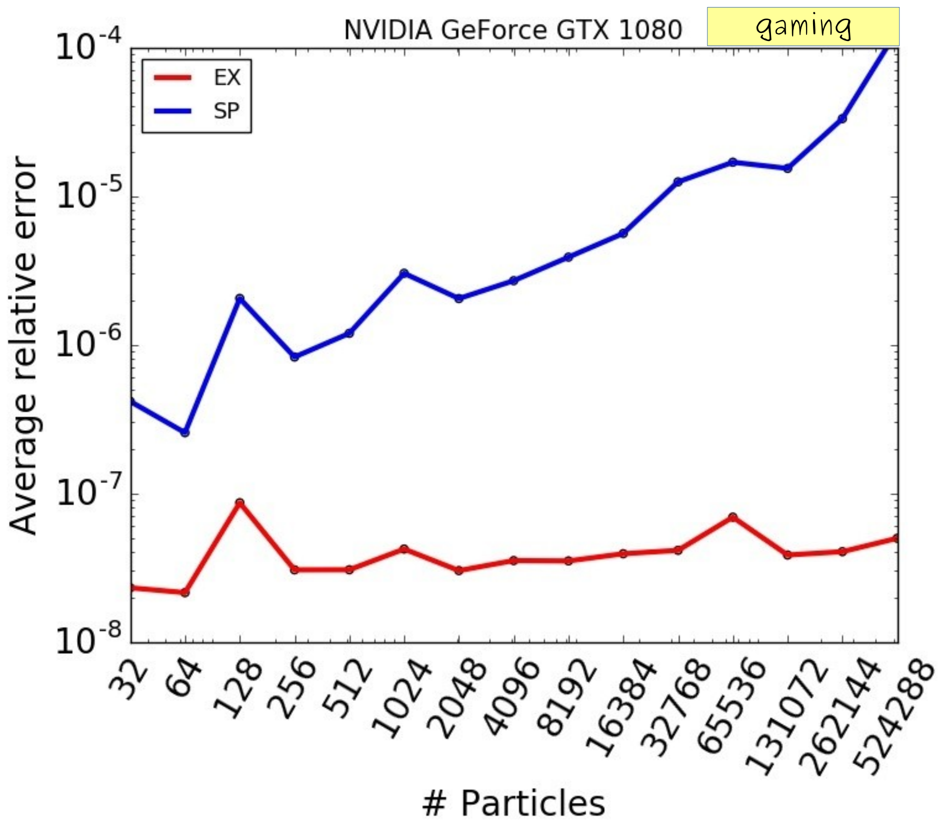
```
  [...]
  /* Loop over blocks */
  for (unsigned int i=0 ; i<particles ; i+=blockDim)
    {
//#pragma HLS UNROLL factor=4
      int address = i + threadIdx + costante2 + istart;

      shPos[threadIdx] = globalX[address];
      shVel[threadIdx] = globalV[address];
      shAcc[threadIdx] = globalA[address];

      barrier(CLK_LOCAL_MEM_FENCE);

      /* Loop over particles within the block */
      for (unsigned int j=0 ; j<blockDim ; j++)
        {
#pragma HLS PIPELINE
          float4 dr = convert_float4(shPos[j] - myPosition); dr.w = 0.0f;
          float distance = dot(dr, dr) + EPS*EPS;
          float sqrdist = convert_float(shPos[j].w) * native_rsqrt(pown(distance,3));
          float rdistance = native_recip(distance);

          float4 dv = shVel[j] - myVelocity; dv.w = 0.0f;
          float4 da = shAcc[j] - myAccelera;

          float alpha = dot(dv, dr) * rdistance;
          float beta = -3.0f * sqrdist * ((dot(dv, dv) + dot(dr, da)) * rdistance + pown(alpha,2));

          float4 au = (sqrdist * dv) + (-3.0f * alpha * sqrdist * dr);

          acc += convert_double4(sqrdist * dr);
          jrk += convert_double4(au);
          snp += convert_double4((sqrdist * da) + (beta * dr) + (-6.0f * alpha * au));
        } /* End of loop within the block */
      barrier(CLK_LOCAL_MEM_FENCE);
    } /* End of loop over particles */
  [...]
```

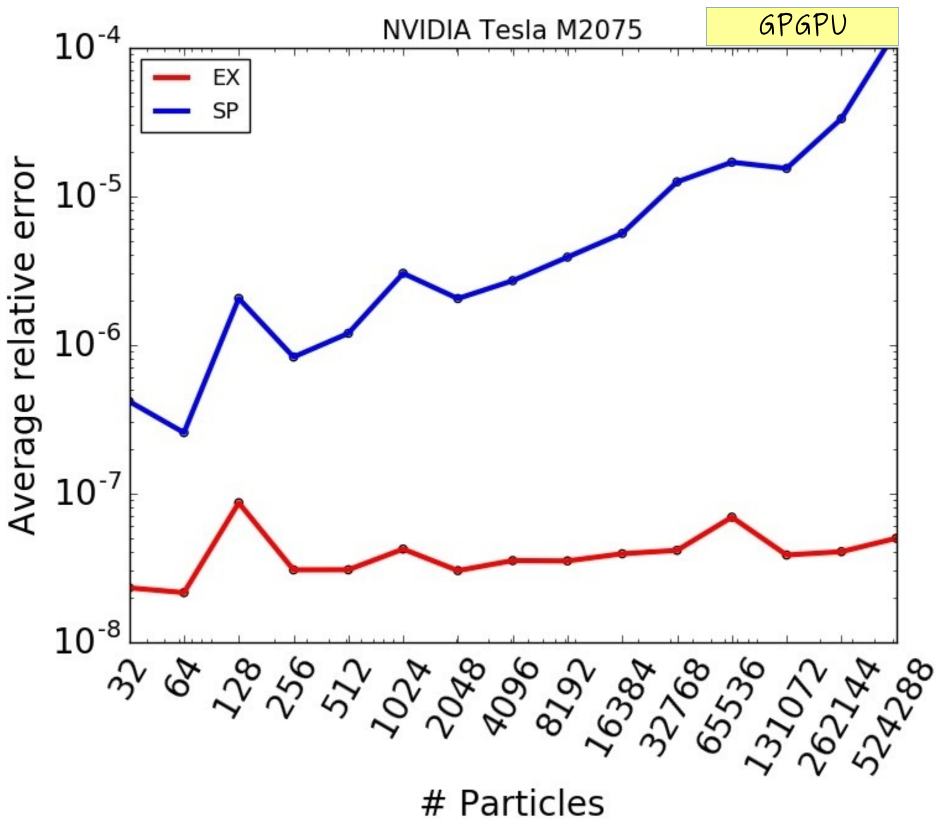
Kernel extracted from HiGPUs and then re-designed for FPGA.

- kernel extracted from HiGPUs;
- kernel re-designed for FPGA:
  - vectorization (each work-item does 4x much work);
  - geometric functions added;
  - usage of the local memory of the FPGA;
  - extended-precision (EX) arithmetic adopted (EX number provides 48 bits of mantissa at SP exponent range);
- working with ECOSCALE to download a configuration into the FPGA (difficult task).

Can EX numeric type represent a trade-off in FPGA?

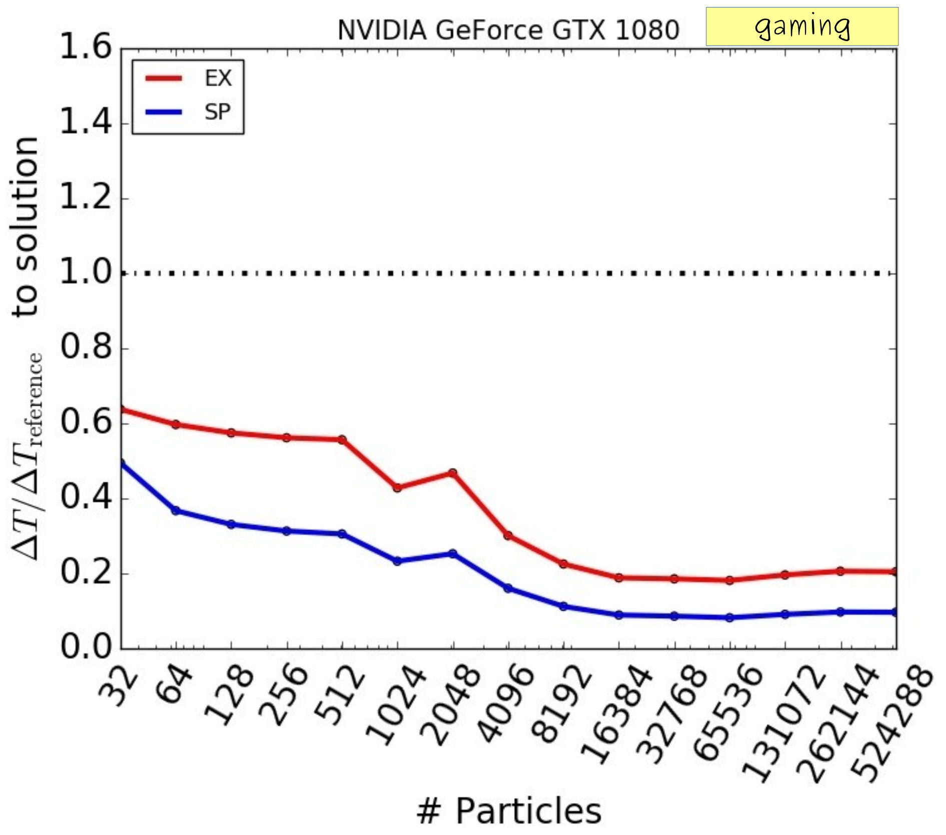
HiGPUs: errors with different precision arithmetic

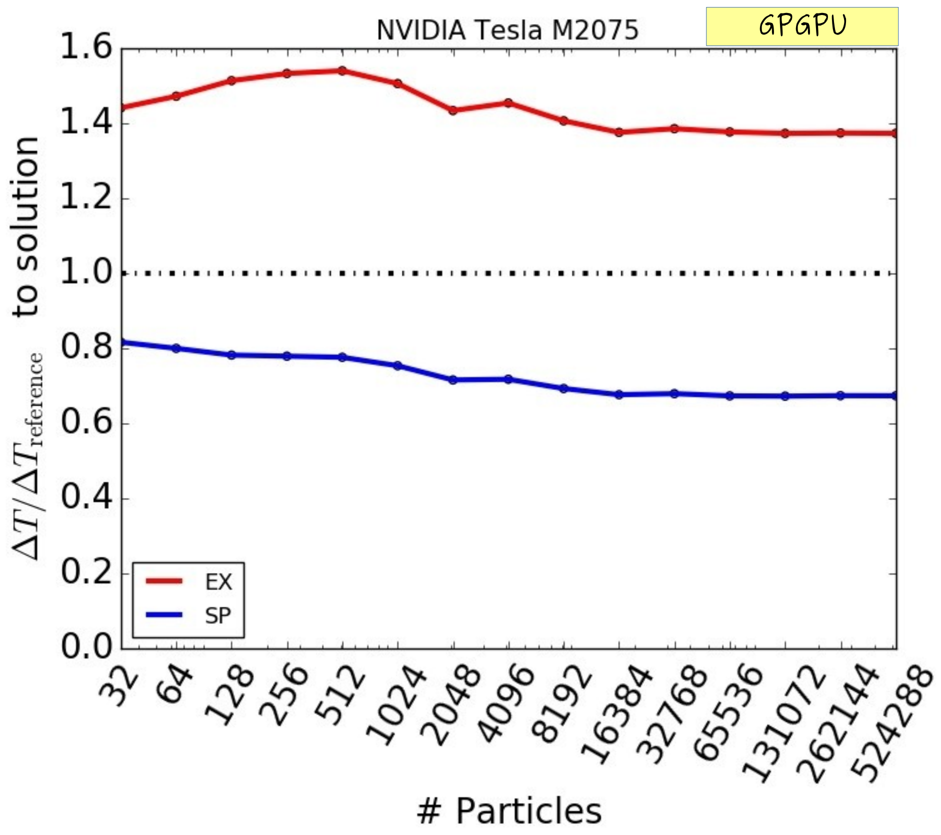




Average relative error between a given precision arithmetic and DP arithmetic as a function of the number of particles.

HiGPUs: timing profile with different arithmetics





Timing profile with different arithmetics as a function of the number of particles. Time to solution is normalized to the value obtained with DP arithmetic.

INAF – OATs people involved in ExaNeSt

Name	Topics	Email
Giuliano Taffoni	Applications and benchmarks	taffoni@oats.inaf.it
Giuseppe Murante	Codes for cosmological simulations	murante@oats.inaf.it
Luca Tornatore	Codes for cosmological simulations, leader of software developers	tornatore@oats.inaf.it
David Goz	Codes for cosmological simulations, software developer, OpenCL	goz@oats.inaf.it
Stefano Borgani	Codes for cosmological simulations	borgani@oats.inaf.it
Valentina D'Odorico	Codes for simulations and data analysis	dodorico@oats.inaf.it
Gianluigi Granato	Codes for simulations and data analysis	granato@oats.inaf.it
Elena Rasia	Codes for simulations and data analysis	rasia@oats.inaf.it