











Article

Performance and Energy Footprint Assessment of FPGAs and GPUs on HPC Systems Using Astrophysics Application

David Goz ^{1,*}, Georgios Ieronymakis ^{2,*}, Vassilis Papaefstathiou ², Nikolaos Dimou ², Sara Bertocco ¹, Francesco Simula ³, Antonio Ragagnin ¹, Luca Tornatore ¹, Igor Coretti ¹ and Giuliano Taffoni ¹

¹ INAF-Osservatorio Astronomico di Trieste, 35122 Padova, Italy; sara.bertocco@inaf.it (S.B.); antonio.ragagnin@inaf.it (A.R.); luca.tornatore@inaf.it (L.T.); igor.coretti@inaf.it (I.C.); giuliano.taffoni@inaf.it (G.T.)

² ICS-FORTH, GR-700 13 Heraklion, Crete, Greece; papaef@ics.forth.gr (V.P.); ndimou@ics.forth.gr (N.D.)

³ INFN-Sezione di Roma, 00185 Rome, Italy; francesco.simula@roma1.infn.it

* Correspondence: david.goz@inaf.it (D.G.); ieronym@ics.forth.gr (G.I.)

† These authors contributed equally to this work.

Received: 6 March 2020; Accepted: 11 April 2020; Published: 17 April 2020



Abstract: New challenges in Astronomy and Astrophysics (AA) are urging the need for many exceptionally computationally intensive simulations. “Exascale” (and beyond) computational facilities are mandatory to address the size of theoretical problems and data coming from the new generation of observational facilities in AA. Currently, the High-Performance Computing (HPC) sector is undergoing a profound phase of innovation, in which the primary challenge to the achievement of the “Exascale” is the power consumption. The goal of this work is to give some insights about performance and energy footprint of contemporary architectures for a real astrophysical application in an HPC context. We use a state-of-the-art N-body application that we re-engineered and optimized to exploit the heterogeneous underlying hardware fully. We quantitatively evaluate the impact of computation on energy consumption when running on four different platforms. Two of them represent the current HPC systems (Intel-based and equipped with NVIDIA GPUs), one is a micro-cluster based on ARM-MPSoC, and one is a “prototype towards Exascale” equipped with ARM-MPSoCs tightly coupled with FPGAs. We investigate the behavior of the different devices where the high-end GPUs excel in terms of time-to-solution while MPSoC-FPGA systems outperform GPUs in power consumption. Our experience reveals that considering FPGAs for computationally intensive application seems very promising, as their performance is improving to meet the requirements of scientific applications. This work can be a reference for future platform development for astrophysics applications where computationally intensive calculations are required.

Keywords: astrophysics; HPC; N-body; ARM-MPSoC; GPUs; FPGAs; hardware acceleration; acceleration architectures; Exascale; Energy Delay Product

1. Introduction and Motivation

In the last decade, energy efficiency has become the primary concern in the High-Performance Computing (HPC) sector. HPC systems constructed from conventional multicore Central Processing Units (CPUs) have to face, on one side, the reduction in year-on-year performance gain for CPUs and on the other side, the increasing cost of cooling and power supply as HPC clusters grow larger.

Some technological solutions have already been identified to address the energy issue in HPC [1]; one of them is the use of power-efficient Multiprocessor Systems-on-Chip (MPSoC) [2–8].

These hardware platforms are integrated circuits composed of multicore CPUs combined with accelerators like Graphic-Processing-Units (GPUs) and/or Field-Programmable-Gate-Arrays (FPGAs). Such hardware accelerators can offer higher throughput and energy efficiency compared to traditional multicore CPUs. The main drawback of those platforms is the complexity of their programming model, requiring a new set of skills for software developers and hardware design concepts, leading to increased development time for accelerated applications.

The Astronomy and Astrophysics (AA) sector is one of the research areas in physics that requires more and higher performing software, as well as the necessity of Exascale supercomputers (and beyond) [9]. In AA, HPC numerical simulations are the most effective instruments to model complex dynamic systems, to interpret observations and to make theoretical predictions, advancing scientific knowledge. They are mandatory to help capture and analyze the torrent of complex observational data that the new generation of observatories produce, providing new insights into astronomical phenomena, the formation and evolution of the universe, and the fundamental laws of physics.

The research presented in this paper arises in the framework of European Exascale System Interconnect and Storage (EuroExa) European funded project (EuroExa: <https://euroexa.eu/>) aiming at the design and development of a prototype of an Exascale HPC machine. EuroExa is achieving that using low-power ARM processors accelerated by tightly coupled FPGAs.

Building an Exascale platform, i.e., a supercomputer able to reach a peak performance greater than 10^{18} Floating point Operations Per Second (FLOPS), is a clear priority that is spurring worldwide initiatives. All the different proposed approaches imply the re-design of the underlying technologies (i.e., processors, interconnect, storage, and accelerators) both to improve their performance and to reduce their energy requirements by about one order of magnitude. On the other side, Exascale platforms are composed by an extremely large number of nodes (e.g., in the case of EuroExa platform we expect 10^6 nodes) “glued” together by a low latency and high throughput interconnect. This implies that a large effort also on usability (e.g., system software), reliability and resiliency must be conceived. Efficiently running scientific applications therein requires the exploitation not only of CPUs and accelerators, but also of the complex memory hierarchy supported by the multi-hierarchy interconnect available on the platforms.

Focusing on performance and energy efficiency, in this work we exploit four platforms:

- (I-II) two Linux x86 HPC clusters that represent the state-of-the-art of HPC architectures (Intel-based and equipped with NVIDIA GPUs);
- (III) a Multiprocessor SoC micro-cluster that represents a low purchase-cost and low-power approach to HPC;
- (IV) an Exascale prototype that represents a possible future for supercomputers. This prototype was developed by the ExaNeSt European project (ExaNeSt: <https://exanest.eu/>) [10–12] and customized by the EuroExa project.

The platforms are probed using a direct N -body solver for astrophysical simulations, widely used for scientific production in AA, e.g., for simulations of star clusters up to ~ 8 million bodies [13,14]. Even if, the role of the interconnect is one of the core issues in designing an Exascale platform, the application studied in this paper is a CPU bound code, so different interconnect technologies implemented by the 4 platforms do not affect the results discussed.

The goal of this paper is to investigate the performance-consumption plane, namely the parameter space where time-to-solution and energy-to-solution are combined, exploiting the different devices hosted on the platforms. We include the comparison among high-end CPUs, GPUs and MPSoCs tightly coupled with FPGAs systems. To the best of our knowledge, this paper provides one of the first comprehensive evaluations of a real AA application on an Exascale prototype, comparing the results with today’s HPC hardware.

The paper is organized as follows. In Section 2 we describe the computing platforms used for the analysis. In Sections 3 and 4 we discuss the methodology employed to make the performance and energy measurement experiments, including considerations on the usage of the different platforms

and the configuration of the parallel runs. Section 5 is devoted to present the scientific application used to benchmark the platforms. Our results are presented in Section 6. The final Section 7 is devoted to the conclusions and the perspectives for future work.

2. Computing Platforms

In this section, we describe the four platforms used in our tests. In Table 1, we list the devices, and we highlight in bold the ones exploited in this paper.

Table 1. The computing node and the associated devices. The devices exploited are highlighted in bold.

Node	CPU	GPU	FPGA
mC	4 × (ARM A53) + 2 × (ARM A72)	ARM Mali-T864	None
IC	40 × (Xeon Haswell E5-4627v3)	None	None
ExaBed	16 × (ARM A53) + 8 × (ARM R5)	4 × (ARM Mali-400)	4 × (Zynq-US+)
GPUC	32 × (Xeon Gold 6130)	8 × (Tesla-V100-SXM2)	None

2.1. Intel Cluster

Each node of the Intel cluster (hereafter IC) is equipped with 4 sockets INTEL Haswell E5-4627v3 at 2.60 GHz with 10 cores each and 256 GB (6 GB per core). The theoretical peak performance of each core is 83.2/41.6 GFLOPS for FP32/FP64, respectively. The interconnect is the Infiniband ConnectX-3 Pro Dual QSFP+ (54Gbs), and the storage system is a BeeGFS parallel file system, with 4 IO servers offering 350TB of disk space [15,16]. Each computing node is equipped with an iLO4 management controller that can be used to measure the node instantaneous power consumption (1 sample every second).

2.2. GPU Cluster

Each node of the GPU cluster (hereafter GPUC) is equipped with 2 sockets INTEL Xeon Gold 6130 at 2.10 GHz with 16 cores each along with 8 NVIDIA Tesla-V100-SXM2. The theoretical peak performance of each GPU is 15.67/7.8 TFLOPS for FP32/FP64, respectively. The GPUs are hosted by a SuperServer 4029GP-TVRT system by SuperMicro®, which integrates a Baseboard Management Controller (BMC) that through Intelligent Platform Management Interface (IPMI) provides out-of-band access to the sensors embedded into the system. Among the physical parameters that these sensors are able to measure (temperature, cooling fans speed, chassis intrusion, etc.), this system is also able to continuously monitor amperage and voltage of the different rails within the redundant power supply units, in order to give at least a ballpark figure of its wattage. Right after booting and with all GPUs in idle, the system wattage is given at about 440 Watts. To get full throttle GPU's power measurements we rely on the built-in sensors queried by NVIDIA nvidia-smi tool (The readings are accurate to within +/− 5 Watts, as stated by NVIDIA documentation. That accuracy does not affect our results.).

2.3. ARM-Micro-Cluster

We design our ARM-Micro-Cluster (hereafter mC) starting from the OpenSource MPSoC Firefly-RK3399 [17]. This single-board is equipped with the big.LITTLE architecture: 4 × (Cortex-A53) cores with 32 kB L1 cache and 512 kB L2 cache, and a cluster of 2 × (Cortex-A72) high-performance cores with 32 kB L1 cache and 1M L2 cache. Each cluster operates at independent frequencies, ranging from 200 MHz up to 1.4 GHz for the LITTLE and up to 1.8 GHz for the big. The MPSoC contains 4 GB DDR3-1333 MHz RAM. The MPSoC also features the OpenCL-compliant Mali-T864 embedded GPU that operates at 800 MHz. The theoretical peak performance of each A53 core is 11.2/2.8 GFLOPS for FP32/FP64, and of each A72 core is 14.4/3.6 GFLOPS for FP32/FP64, respectively. The theoretical peak performance of the T-864 GPU is 109/32 GFLOPS for FP32/FP64, respectively. The ARM-Micro-Cluster, composed by 8 Firefly-RK3399 single-boards, is based on Ubuntu 18.04 Linux and scheduled using

SLURM [18]. The interconnect is based on Gigabit Ethernet, and the storage system is a device shared via NFS.

2.4. ExaNest HPC Testbed Prototype

The ExaNest HPC testbed prototype [10] (hereafter ExaBed) is a liquid-cooled cluster composed of the proprietary Quad-FPGA daughterboard (QFDB) [19] computing nodes, interconnected with a custom network and equipped with a BeeGFS parallel filesystem. In Figure 1, we present a block diagram of the computing node of the platform.

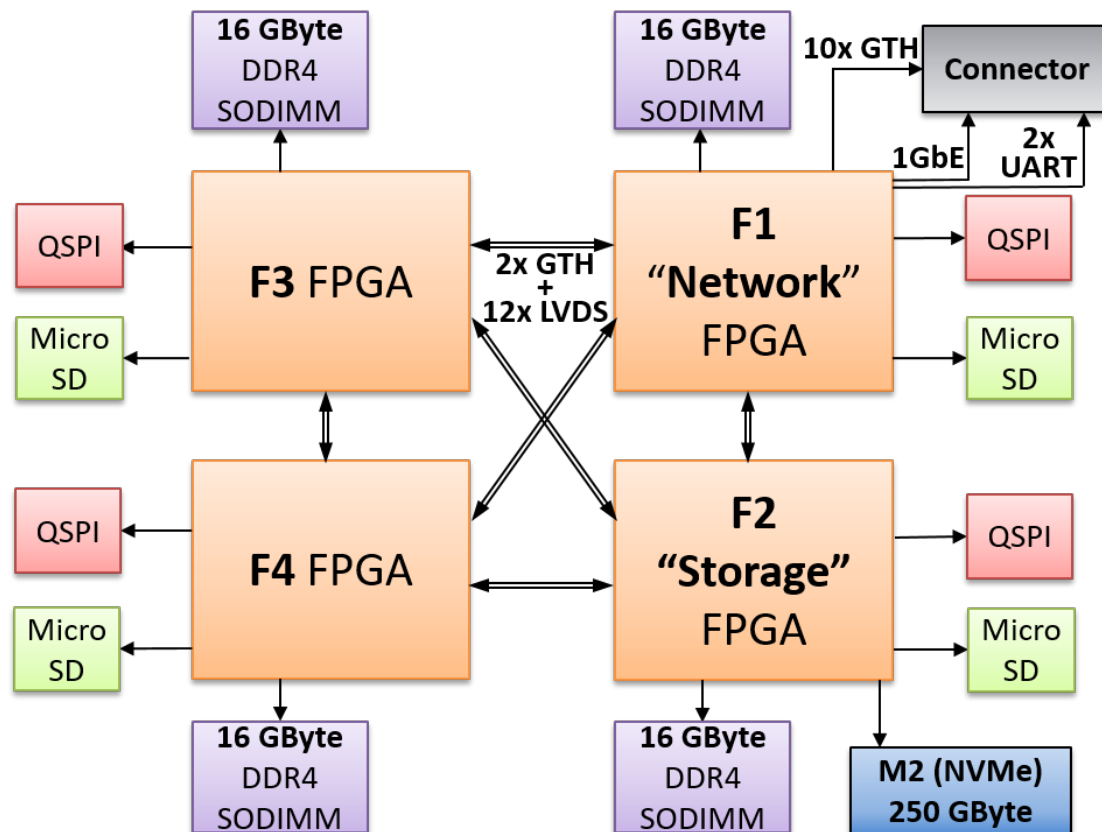


Figure 1. The Quad-FPGA daughterboard block diagram and interconnects.

The compute-node board includes 4 Xilinx Zynq Ultrascale+ MPSoC devices (ZCU9EG), each featuring 4x(ARM-A53) and 2x(ARM-R5) cores, along with a rich set of hard IPs and Reconfigurable Logic. Each Zynq device has a 16 GB DDR4 (SODIMM) attached and a 32 MB Flash (QSPI) memory. Also, as shown in Figure 1, within the QFDB each FPGA is connected to each other through 2 HSSL and 24 LVDS pairs (12 in each direction). Out of the four, only the “Network” FPGA is directly connected to the outside world, while the “Storage” FPGA has an additional 250 GB M.2 SSD attached to it. The maximum sustained power of the board is 120 Watts, while the power dissipation during normal operation is usually around to 50 Watts. Targeting a compact design, the dimension of the board is 120–130 mm while no component on top or below the printed circuit board (PCB) is taller than 10 mm.

These compute nodes are sealed within a blade enclosure, each hosting 4 QFDBs. Currently, the ExaNest prototype HPC testbed consists of 12 fully functional blades. The rack provides connectivity between the blades, while each QFDB is managed through a Manager VM and runs a customized version of Linux based on Gentoo Linux, which is called Carvoonix.

In the QFDB, the measurement of the current and power is accomplished by using a set of TI INA226 coupled with high-power shunt resistors. The INA226 minimal capture time is 140 [vs]. However,

the Linux driver default (and the power-on set-up) sets capture time to 1.1 [ms]. The Linux driver also enables averaging from 16 samples, and captures both the shunt and the bus voltages. To collect data from the sensors, each board includes 15 I2C power sensors, which allow the measurement of power consumption by major subsystems.

3. Methodology and Considerations

The platforms exhibit different behavior as concern the power policies.

In the case of ARM sockets, the frequency scaling is absent, meaning that idle and performance mode are mutually exclusive active. Our code, described in Section 5, is not able to exploit the highly heterogeneous big.LITTLE ARM socket, whose architecture couples relatively power-saving and slower processor cores (LITTLE) with relatively more powerful and power-hungry ones (big). This MPSoC is conceived to migrate more demanding threads on the more powerful cores of the big socket (A72 in the case of the mC). Hence, to disentangle the performance and the power consumption, we pin all MPI (Message Passing Interface) processes and Open Multi-Processing (OpenMP) threads to the LITTLE socket setting explicitly CPU affinity. It is worth noticing that both mC and ExaBed are equipped with A53x4/socket, letting us to run our simulations only on the former and extrapolating the results using CPUs for the latter as well. Hence, we consider it useful to focus on the performance of the FPGA in the Xilinx Zynq UltraScale+ MPSoC hosted by ExaBed, which is in turn the most important topic of this paper.

On the IC, the ExaBed, the mC and the GPUC respectively, the smallest units for which energy consumption can be measured are a 4-sockets (with 10 cores each) node, a QFDB (4 MPSoCs, with 4 cores and one FPGA each), a single-board (dual socket, with 4 and 2 cores respectively, and one GPU), and one GPU. To carry out a meaningful comparison, we decide to perform the comparison using the same amount of computational units. Given the heterogeneity of the platforms in terms of the underlying devices (Table 1 as reference), we define the computational unit as a group of four cores for CPU, and either one GPU or FPGA for accelerators. We fix at four cores the computational unit for CPU because both mC and ExaBed are equipped with A53x4 socket. Table 2 summarizes the compute units (hereafter CUs), as defined above, available for each platform.

Table 2. The compute units, as defined in Section 3, available on the platforms.

CUs				
Platform				
CU-Type	IC	mC	ExaBed	GPUC
CPU	10	1	None	None
GPU	None	1	None	8
FPGA	None	None	4	None

One of the aims of this work is to shed some light on the crucial comparison between the energy consumption and performance of current platforms and (possibly) Exascale-like ones.

4. Power Consumption Measurements

Since the IC, ExaBed and GPUC have built-in sensors, for those three platforms, we rely on the power measurements returned by the diagnostic infrastructure. The mC, on the contrary, does not have any sensor, so we obtain the energy consumption by measuring the actual absorption using a Yokogawa WT310E Digital Power Meter.

We assess that having different methods of energy measurements is not affecting the results. To make power measurements, we set-up simulations so that their runtime is much larger than the sampling time of on-board sensors so that fluctuations are averaged out.

For each platform, we measure both the energy consumed under no workload (E_{idle}) and the total energy consumption under 100% workload (E_{full}) using the CUs. In Table 3 the E_{idle} and E_{full} using one compute unit (of different type) for each platform are shown.

Table 3. The energy consumption of platforms. E_{idle} is the average power over 3 min in idle of the platform; E_{full} is the average energy used over 3 min of HY-NBODY continuous execution with 100% load, using one CU of type either CPU, GPU, or FPGA ($E_{full-CPU}$, $E_{full-GPU}$, and $E_{full-FPGA}$ respectively).

E[W]	Platforms			
	IC	mC	ExaBed	GPUC
E_{idle}	160	3.15	42.5	440
$E_{full-CPU}$	223	4.55	N/A	N/A
$E_{full-GPU}$	N/A	4.75	N/A	710
$E_{full-FPGA}$	N/A	N/A	53.5	N/A

In the following we report the energy-to-solution (total energy required to perform the calculation) excluding the E_{idle} , i.e., $E_{work-CU} = E_{full-CU} - E_{idle}$, in order to focus on the power consumption of different CUs. The idle energy and the energy consumed by the processing units (CPUs, GPUs, FPGAs) are distinct targets for engineering and improvement, and it seems useful to disentangle them while considering what is most promising in the Exascale perspective.

Finally, we estimate the energy impact of the application also in terms of Energy Delay Product (EDP). The EDP proposed by Cameron [20] is a “fused” metric to evaluate the trade-off between time-to-solution and energy-to-solution. It is defined as:

$$EDP_{CU} = E_{work-CU} \times T_{work-CU}^w \quad (1)$$

where $E_{work-CU}$ is the E_{work} consumed during the run by the CU, $T_{work-CU}$ is the time-to-solution of the given CU and w (usually $w = 1, 2, 3$) is a parameter to weight performance versus power. The larger is w the greater the weight we assign to its performance.

5. Astrophysical Code

As aforementioned, we compare both time-to-solution and energy-to-solution performance using a real scientific application coming from the astrophysical domain: the HY-NBODY code [21,22].

In Astrophysics the N -body problem consists of predicting the individual motion of celestial bodies interacting purely gravitationally. Since everybody interacts with all the others, the computational cost scales as $O(N^2)$, where N is the number of bodies. HY-NBODY is a modified version of a GPU-based N -body code [23–25], it has been developed in the framework of the ExaNeSt project [10], and it is currently optimized for Exascale-like machines within the FET-HPC H2020 EuroExa project. The code relies on the 6th order Hermite integration schema [26], which consists of three stages: a predictor step that predicts particle’s positions and velocities; an evaluation step to evaluate new accelerations, their first order (*jerk*), second order (*snap*), and third order derivatives (*crackle*); a corrector step that corrects the predicted positions and velocities using the results of the previous steps.

Code profiling shows the Hermite schema spends more than 90% of time calculating the evaluation step, characterized by having an arithmetic intensity $I \simeq 10^4$ [FLOPs/byte] (ratio of FLOPS to the memory traffic) using 32^3 particles. In the following, time-to-solution and energy-to-solution measurements refer to that compute-bound kernel.

Three version of the code are available:

- (i) Standard C code: cache-aware designed for CPUs and parallelized with hybrid MPI+OpenMP programming;
- (ii) OpenCL code: conceived to target accelerators like GPGPUs or embedded GPUs. All the stages of the Hermite integrator are performed on the OpenCL-compliant device(s).

The kernel implementation exploits *local* memory (OpenCL terminology) of device(s), which is generally accepted as the best method to reduce global memory latency in discrete GPUs. However, on ARM embedded GPUs, the *global* and *local* OpenCL address spaces are mapped to main host memory (as reported by the ARM developer guide (http://infocenter.arm.com/help/topic/com.arm.doc.100614_0312_00_en/arm_mali_midgard_opengl_developer_guide_100614_0312_00_en.pdf)). Therefore, a specific ARM-GPU-optimized version of all kernels of Hy-Nbody, in which *local* memory is not used, has been implemented and used in the results shown in the paper. The impact of such an optimization is shown in [21].

Regarding the host parallelization schema, a one-to-one correspondence between MPI processes and computational nodes is established and each MPI process manages all the OpenCL-compliant devices available per node (the number of such devices is user defined). Inside each share-memory computational node the parallelization is achieved by means of OpenMP. Such an implementation requires that particle data be communicated between the host and the device at each time-step, which gives rise to synchronization points between host and device(s). Accelerations and time-step computed by the device(s) are retrieved by the host on every computational node, reduced and then sent back again to the device(s);

- (iii) Standard C targeting HLS tool: Xilinx Vivado High-Level Synthesis tool (<https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html>) was used to develop a highly optimized hardware accelerator for QFDB's FPGAs. The kernel was designed to be parameterizable, to experiment with different area vs performance implementations and to provide the capability of deploying it to any Xilinx FPGA with any amount of reconfigurable resources.

Vivado HLS provides a directive-oriented style of programming where the tool transforms the high-level code (C, C++, SystemC, OpenCL) to a Hardware Description Language (HDL) according to the directives provided by the programmer. Some of the optimizations performed in this kernel are described below:

- calculation in chunks: Given the finite resources of the FPGA and the need to accelerate the Hermite algorithm in large arrays that exceed the amount of internal memory inside the FPGA (Block RAM), we followed a tiled approach where the kernel loops over the corresponding tiles of the original arrays and the core Hermite algorithm is performed in chunks of data stored internally. Block RAM is a low latency memory cell that can be configured in various widths and depths to store data inside the FPGA fabric, but their capacity is limited (32.1 MB in our device). Thus, at the start of each computation the kernel fetches the data for the corresponding tile from main memory and stores it into Block RAM. During computation, the partial results are also kept internally and as soon as the kernel finishes working with a tile, it writes the results back to the main memory and fetches the data for the next tile. Hence, the kernel has immediate access to the data it needs and communicates with the higher-latency DRAM only at the beginning and end of processing each tile, resulting in higher computational efficiency.
- burst memory mode: As defined in the AXI4 protocol (which is used by the kernel to communicate with the DRAM) a "beat" is an individual transfer of a single data word, while a "burst" is a transaction in which multiple sequential data are transferred based upon a single address request. Since the data to be processed are stored sequentially in large arrays inside the DRAM, the kernel was implemented to request and fetch the data in bursts, and the burst size selected was the maximum burst size allowed by the AXI4, which is 4 kB. This results in higher hardware complexity and resource use inside the FPGA fabric, but provides higher memory bandwidth and more efficient communication with the device's memory controller.

- loop pipeline, loop unroll, array partitioning: Core Hermite algorithm has been pipelined achieving an initiation interval of 1 clock cycle. To achieve this, we increased the amount of the kernel's AXI4 read/write interfaces that communicate with the DRAM to fetch data from multiple arrays simultaneously. Also, by applying the loop unrolling directive we allowed the algorithm to be performed on more particles per cycle, with the corresponding increase of the FPGA resources needed due to the demand of more computational units. To achieve pipelining, the arrays stored internally were partitioned in multiple Block RAMs, since each BRAM has 2 ports for reading/writing and the kernel needs to access the data of many particles per cycle. These modifications minimized the idle time, considering the kernel is able to perform calculations on many particles in each individual cycle and remains idle only at the beginning and at the end of the processing of each tile when it communicates with the DRAM.

In our previous work [22] we demonstrated a kernel showing a single QFDB's FPGA full potential. Due to its extra connectivity capabilities, the "Network" FPGA results in a higher reconfigurable resource congestion to operate. Thus, the above kernel's high demand of resources made it unfeasible to deploy it to the "Network" FPGA, so in order to demonstrate the application running in many FPGAs and split the computation load evenly inside the QFDB we chose a different size for this work's kernel. This kernel has 75% throughput of the previous one and operates on a slightly higher frequency (320 MHz compared to 300 MHz).

Floating Point Arithmetic Considerations

Arithmetic precision plays a key role during the integration of the equations of motion of an N -body system. Generally, Hermite integration schema requires double-precision arithmetic to minimize the accumulation of the round-off errors, preserving both the total energy and the angular momentum during the simulation. We have already demonstrated that extended-precision arithmetic [27] can speed up the calculation on GPUs, while is performance-poor on both CPUs and FPGAs [22], due to its higher arithmetic intensity compared to the double-precision algorithm (additional accumulations etc.).

Given that, we obtain the results shown in Section 6 using double-precision arithmetic to exploit both CPUs and FPGAs, while extended-precision arithmetic is employed to exploit GPUs.

6. Computational Performance and Energy Consumption

In all simulations, in the case of CPUs, the cores composing the CUs are exploited by means of OpenMP threads, and the multi-CUs by means of MPI; for GPUs, instead, we use a fixed number of 64 for the work-group-size (we have already shown that the performance on ARM embedded GPUs is not driven by any specific work-group-size, regardless the usage of the local memory [21]).

We investigate the time-to-solution of HY-NBODY running two different test series. First, keeping the number of CUs constant, we increase the number of particles. We run three simulations with 32^3 , 64^3 and 128^3 particles. Then, keeping the number of particles constant, we vary the number of CUs used, from 1 to 4.

In Figure 2, we report the speedup (the ratio of the execution time using one CU to the time using multiple CUs) using 2 and 4 CUs. Different symbols refer to a different number of CUs (square for 2 CUs and pentagon for 4 CUs).

As expected, for almost all types of CUs, a linear speedup time execution reduction is achieved. We observe a speedup time execution reduction using 4 CUs for 32^3 particles only in the case of the mC_{T864}. For the IC, on the contrary, a super linear speedup time execution is achieved using 2 and 4 CUs for 128^3 particles.

Figure 3 shows the performance-consumption plane (energy-to-solution, E_{work} , vs. time-to-solution) using 64^3 particles and varying the number of CUs.

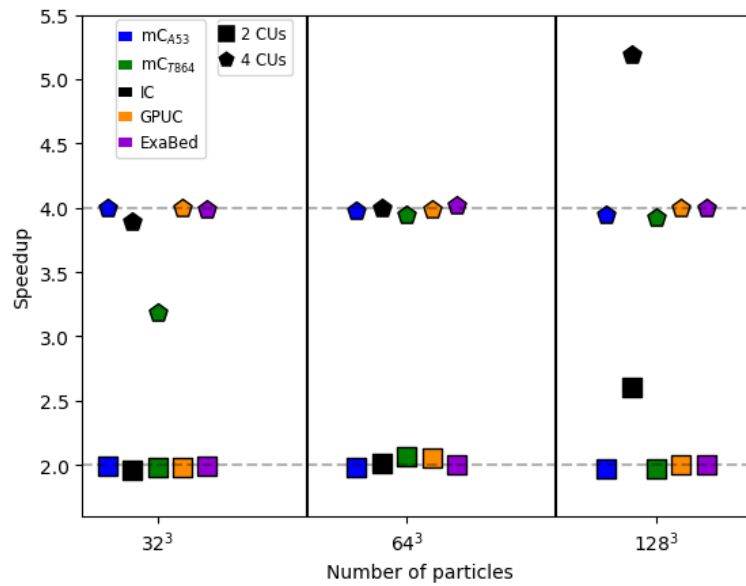


Figure 2. Speedup as a function of the number of particles using 2 and 4 CUs.

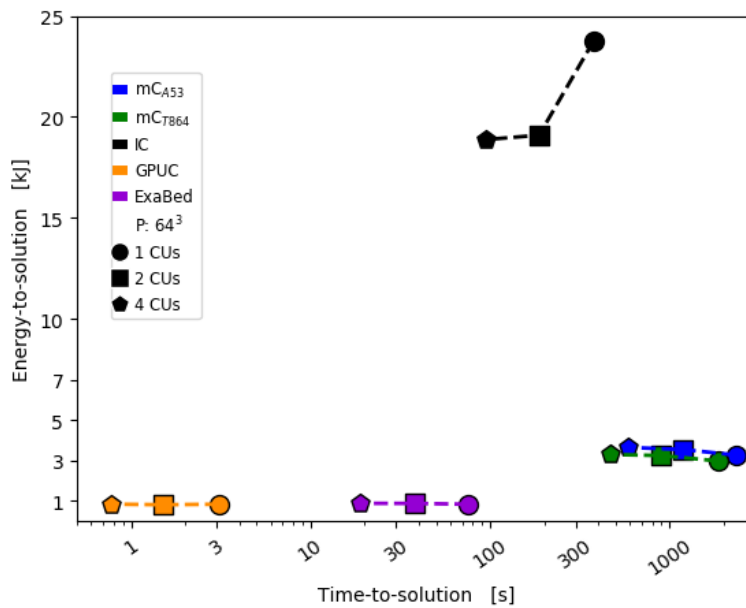


Figure 3. The performance-consumption plane for 64³ particles varying the number of CUs.

HY-NBODY is a compute-bound application, as stated in Section 5, hence, the latency of MPI communications across different CUs is negligible respect to the computational time. For this reason, in these tests we measure the computing performance of the CUs but not the network contribution.

Not surprisingly, both GPUC and ExaBed achieve the solution with less energy (E_{work}) than either IC or mC. The most interesting thing to point out is the equivalence of the energy-to-solution between GPUC and ExaBed, which indicates a definite trend toward Exascale prototype. We can also see the effect of the energy consumption overhead when a node uses only a subset of its cores or sockets. This effect is evident for IC when using 1 or 2 CUs (a subset of all the CUs available).

In Figure 4, we present the results of the EDP for $w = 1$ and 64³ particles. We note that with the same CUs, the GPUC has a better EDP than the other platforms. When comparing the ExaBed and IC for the same time-to-solution configuration, the ExaBed has a better EDP. The configuration with 1 CUs on ExaBed has the same time-to-solution of the configuration with 4 CUs of the IC (we compare the execution time of the ExaBed using one CU with the execution time of the IC using four CUs in Figure 3).

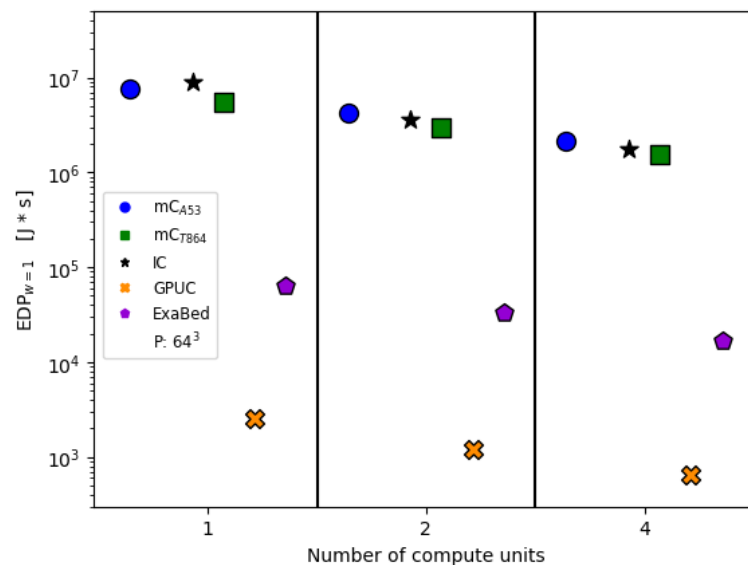


Figure 4. EDP as a function of the CUs using 64^3 particles.

7. Conclusions and Future Work

In this work, we discuss the performance evaluation of four platforms concerning both the time-to-solution and energy-to-solution for code coming from the AA sector. Two platforms that represent the current status of HPC systems, the former Intel-based (IC) and the latter equipped with NVIDIA Tesla-V100 GPUs (GPUC), an ARM-MPSoC micro-cluster (mC) that could represent a low-budget HPC solution, and the ExaNeSt Exascale prototype (ExaBed) that (possibly) represents the next generation of HPC systems.

Our analysis has been conducted using code for scientific production exploiting multi-CPU, GPUs and FPGAs of the aforementioned platforms. The compute-bound nature of our application allows us to focus on performance assessment of the computational power and energy efficiency of the devices, without dealing with the interplay of different key factors, like memory bandwidth, network latency and application execution pattern.

The overall picture, where accelerators outperform CPUs in terms of both performance and energy efficiency, is not surprising. Exploiting CPUs, when we set-up a run on the ExaBed in order to achieve the same time-to-solution with a run on the IC (ARM-A53 cores equip both the ExaBed and the mC), our results show that the former proves to be more power-efficient than the latter, which supports the Exascale perspective of having single compute units to be tailored to a better FLOP/W ratio than pure FLOPs performance.

Regarding accelerators, the high-end NVIDIA Tesla-V100 GPUs perform faster than Xilinx US+ FPGAs on SoC; however, the latter demonstrate superior energy efficiency (the energy-to-solution is the same). We found that FPGA programming practice continues to be challenging for HPC software developers, even using the high-level-synthesis technique, which allows the conversion of an algorithm description in high-level languages (e.g., C/C++, OpenCL) into a digital circuit. In comparison, GPU programming is pretty straightforward using the latest frameworks like CUDA, OpenCL or OpenAcc, but our great deal of effort has been devoted to optimize the kernel using extended-precision arithmetic. Therefore, at the end, we use comparable development effort in terms of design time and programmer training.

Our conclusion is that when performance alone is a priority, CPUs or embedded GPUs on MPSoC are not a valid option, albeit their power-efficiency. ARM-based Exascale prototypes may soon evolve to become a viable option for Exascale-class HPC production machines if their performance improves while still maintaining a favorable power consumption. Furthermore, to reduce the programming effort, the software environment should provide a clear, high-level, abstract interface to the programmer

to efficiently execute functionality in the coupled FPGAs, opening the path for successful and cost-effective use of such devices in HPC.

Our future activity will be aimed to exploit more computational nodes, offering a more comprehensive benchmark of both the computation power and the interconnect network of the platforms.

Author Contributions: Conceptualization: D.G. and G.I.; Software: D.G., G.I., V.P. and N.D.; Methodology: D.G., G.I., S.B., A.R., L.T. and I.C.; Investigation: D.G., G.I., S.B., F.S. and I.C.; Validation: D.G., G.I., V.P., N.D. and I.C.; Data curation: D.G., G.I., S.B., A.R. and L.T.; Formal analysis: D.G.; Writing—original draft preparation: D.G. and G.I.; Writing—review and editing: D.G., G.I., S.B., F.S. and G.T.; Supervision: D.G. and G.T.; Project administration: G.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was carried out within the EuroExa FET-HPC and ESCAPE projects (grant no. 754337 and no. 824064), funded by the European Union’s Horizon 2020 research and innovation program.

Acknowledgments: We thank the INAF Trieste Astronomical Observatory Information Technology Framework. We thank Piero Vicini and the INFN APE Roma Group for the support and for the use of INFN computational infrastructure. We also thank Giuseppe Murante and Stefano Borgani for the fruitful discussions on the energy and performance optimization of the code. This research has been made use of IPython [28], Scipy [29], Numpy [30] and Matplotlib [31].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AA	Astronomy and Astrophysics
CPU	Central Processing Unit
CU	Compute Unit
CUDA	Compute Unified Device Architecture
EDP	Energy Delay Product
ExaBed	EaxNeSt HPC testbed
ExaNeSt	European Exascale System Interconnect and Storage
FLOPS	FLoating point Operations Per Second
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
GPUC	GPU cluster
HDL	Hardware Description Language
HLS	High-Level Synthesis
HPC	High-Performance Computing
IC	Intel Cluster
ICS-FORTH	Institute of Computer Science-FORTH
INAF	Istituto Nazionale di Astrofisica
INFN	Istituto Nazionale di Fisica Nucleare
mC	ARM-Micro-Cluster
MPI	Message Passing Interface
MPSoC	Multi-Processing System-on-Chip
OpenCL	Open Computing Language
OpenMP	Open Multi-Processing
QFDB	Quad-FPGA Daughterboard

References

1. Dutot, P.; Georgiou, Y.; Glesser, D.; Lefevre, L.; Poquet, M.; Rais, I. Towards Energy Budget Control in HPC. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 381–390.
2. Cesini, D.; Corni, E.; Falabella, A.; Ferraro, A.; Morganti, L.; Calore, E.; Schifano, S.F.; Michelotto, M.; Alfieri, R.; De Pietri, R.; et al. Power-Efficient Computing: Experiences from the COSA Project. *Sci. Program.* **2017**, *2017*, 7206595. [[CrossRef](#)]

3. Ammendola, R.; Biagioni, A.; Capuani, F.; Cretaro, P.; Bonis, G.D.; Cicero, F.L.; Lonardo, A.; Martinelli, M.; Paolucci, P.S.; Pastorelli, E.; et al. The Brain on Low Power Architectures—Efficient Simulation of Cortical Slow Waves and Asynchronous States. *arXiv* **2018**, arXiv:1804.03441.
4. Simula, F.; Pastorelli, E.; Paolucci, P.S.; Martinelli, M.; Lonardo, A.; Biagioni, A.; Capone, C.; Capuani, F.; Cretaro, P.; De Bonis, G.; et al. Real-Time Cortical Simulations: Energy and Interconnect Scaling on Distributed Systems. In Proceedings of the 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia, Italy, 13–15 February 2019; pp. 283–290.
5. Calore, E.; Schifano, S.F.; Tripiccion, R. Energy-Performance Tradeoffs for HPC Applications on Low Power Processors. In *Euro-Par 2015: Parallel Processing Workshops*; Hunold, S., Costan, A., Giménez, D., Iosup, A., Ricci, L., Gómez Requena, M.E., Scarano, V., Varbanescu, A.L., Scott, S.L., Lankes, S., et al., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 737–748.
6. Nikolskiy, V.P.; Stegailov, V.V.; Vecher, V.S. Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics. In Proceedings of the 2016 International Conference on High Performance Computing Simulation (HPCS), Innsbruck, Austria, 18–22 July 2016; pp. 682–689.
7. Morganti, L.; Cesini, D.; Ferraro, A. Evaluating Systems on Chip through HPC Bioinformatic and Astrophysic Applications. In Proceedings of the 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), Crete, Greece, 17–19 February 2016; pp. 541–544.
8. Taffoni, G.; Bertocco, S.; Coretti, I.; Goz, D.; Ragagnin, A.; Tornatore, L. Low Power High Performance Computing on Arm System-on-Chip in Astrophysics. In *Proceedings of the Future Technologies Conference (FTC) 2019*; Arai, K., Bhatia, R., Kapoor, S., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 427–446.
9. Taffoni, G.; Murante, G.; Tornatore, L.; Goz, D.; Borgani, S.; Katevenis, M.; Chrysos, N.; Marazakis, M. Shall numerical astrophysics step into the era of Exascale computing? *arXiv* **2019**, arXiv:1904.11720.
10. Katevenis, M.; Chrysos, N.; Marazakis, M.; Mavroidis, I.; Chaix, F.; Kallimanis, N.; Navaridas, J.; Goodacre, J.; Vicini, P.; Biagioni, A.; et al. The ExaNeSt Project: Interconnects, Storage, and Packaging for Exascale Systems. In Proceedings of the 19th Euromicro Conference on Digital System Design, DSD, Limassol, Cyprus, 31 August–2 September 2016; pp. 60–67.
11. Ammendola, R.; Biagioni, A.; Cretaro, P.; Frezza, O.; Cicero, F.L.; Lonardo, A.; Martinelli, M.; Paolucci, P.S.; Pastorelli, E.; Simula, F.; et al. The Next Generation of Exascale-Class Systems: The ExaNeSt Project. In Proceedings of the 2017 Euromicro Conference on Digital System Design (DSD), Vienna, Austria, 30 August–1 September 2017; pp. 510–515.
12. Katevenis, M.; Ammendola, R.; Biagioni, A.; Cretaro, P.; Frezza, O.; Cicero, F.L.; Lonardo, A.; Martinelli, M.; Paolucci, P.S.; Pastorelli, E.; et al. Next generation of Exascale-class systems: ExaNeSt project and the status of its interconnect and storage development. *Microprocess. Microsyst.* **2018**, *61*, 58–71. [[CrossRef](#)]
13. Spera, M.; Capuzzo-Dolcetta, R. Rapid mass segregation in small stellar clusters. *Astrophys. Space Sci.* **2017**, *362*. [[CrossRef](#)]
14. Spera, M.; Mapelli, M.; Bressan, A. The Mass Spectrum of Compact Remnants From the Parsec Stellar Evolution Tracks. *Mon. Not. R. Astron. Soc.* **2015**, *451*, 4086–4103. [[CrossRef](#)]
15. Bertocco, S.; Goz, D.; Tornatore, L.; Ragagnin, A.; Maggio, G.; Gasparo, F.; Vuerli, C.; Taffoni, G.; Molinaro, M. INAF Trieste Astronomical Observatory Information Technology Framework. *arXiv* **2019**, arXiv:1912.05340.
16. Taffoni, G.; Becciani, U.; Garilli, B.; Maggio, G.; Pasian, F.; Umana, G.; Smareglia, R.; Vitello, F. CHIPP: INAF pilot project for HTC, HPC and HPDA. *arXiv* **2020**, arXiv:2002.01283.
17. Bertocco, S.; Goz, D.; Tornatore, L.; Taffoni, G. *INCAS: INTensive Clustered ARM SoC—Cluster Deployment*; INAF-OATs Technical Report; INAF-ICT 2018. Available online: <https://www.ict.inaf.it/index.php/31-doi/96-2018-4> (accessed on 17 April 2020).
18. Pascual, J.A.; Navaridas, J.; Miguel-Alonso, J. Effects of Topology-Aware Allocation Policies on Scheduling Performance. In *Job Scheduling Strategies for Parallel Processing*; Frachtenberg, E., Schwiegelshohn, U., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 138–156.
19. Chaix, F.; Ioannou, A.; Kossifidis, N.; Dimou, N.; Ieronymakis, G.; Marazakis, M.; Papaefstathiou, V.; Flouris, V.; Ligerakis, M.; Ailamakis, G.; et al. Implementation and Impact of an Ultra-Compact Multi-FPGA Board for Large System Prototyping. In Proceedings of the 2019 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC), Denver, CO, USA, 17 November 2019; pp. 34–41.

20. Cameron, K.W.; Ge, R.; Feng, X. High-performance, power-aware distributed computing for scientific applications. *Computer* **2005**, *38*, 40–47. [[CrossRef](#)]
21. Goz, D.; Bertocco, S.; Tornatore, L.; Taffoni, G. Direct N-body Code on Low-Power Embedded ARM GPUs. In *Intelligent Computing*; Arai, K., Bhatia, R., Kapoor, S., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 179–193.
22. Goz, D.; Ieronymakis, G.; Papaefstathiou, V.; Dimou, N.; Bertocco, S.; Ragagnin, A.; Tornatore, L.; Taffoni, G.; Coretti, I. Direct N-body application on low-power and energy-efficient parallel architectures. *arXiv* **2019**, arXiv:1910.14496.
23. Capuzzo-Dolcetta, R.; Spera, M.; Punzo, D. A Fully Parallel, High Precision, N-Body Code Running on Hybrid Computing Platforms. *J. Comput. Phys.* **2013**, *236*, 580–593. [[CrossRef](#)]
24. Capuzzo-Dolcetta, R.; Spera, M. A performance comparison of different graphics processing units running direct N-body simulations. *Comput. Phys. Commun.* **2013**, *184*, 2528–2539. [[CrossRef](#)]
25. Spera, M. Using Graphics Processing Units to solve the classical N-body problem in physics and astrophysics. *arXiv* **2014**, arXiv:1411.5234.
26. Nitadori, K.; Makino, J. Sixth- and eighth-order Hermite integrator for N-body simulations. *New Astron.* **2008**, *13*, 498–507. [[CrossRef](#)]
27. Thall, A. *Extended-Precision Floating-Point Numbers for GPU Computation*; Association for Computing Machinery: Boston, MA, USA, 2006; p. 52.
28. Pérez, F.; Granger, B. IPython: A System for Interactive Scientific Computing. *Comput. Sci. Eng.* **2007**, *9*, 21–29. [[CrossRef](#)]
29. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. Scipy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **2020**. [[CrossRef](#)] [[PubMed](#)]
30. Van der Walt, S.; Colbert, S.; Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Comput. Sci. Eng.* **2011**, *13*, 22–30. [[CrossRef](#)]
31. Hunter, J. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).