The next Generation of Exascale-class Systems: the ExaNeSt Project

R. Ammendola[†‖], A. Biagioni[*], P. Cretaro[*], O. Frezza[*], F. Lo Cicero[*], A. Lonardo[*], M. Martinelli[*], P. S. Paolucci[*], E. Pastorelli[*], F. Simula[*], P. Vicini[*], G. Taffoni[¶], John Goodacree[‡], Mikel Lujn[‡], J. Navaridas[‡], and J. P. Saiz[‡], N. Chrysos[§], and M. Katevenis[§]
for the ExaNeSt team

***Abstract*—The ExaNeSt project started on December 2015 and is funded by EU H2020 research framework (call H2020-FETHPC-2014, n. 671553) to study the adoption of low-cost, Linux-based power-efficient 64-bit ARM processors clusters for Exascale-class systems. The ExaNeSt consortium pools partners with industrial and academic research expertise in storage, interconnects and applications that share a vision of an European Exascale-class supercomputer. Their goal is designing and implementing a physical rack prototype together with its cooling system, the storage non-volatile memory (NVM) architecture and a low-latency interconnect able to test different options for interconnection and storage. Furthermore, the consortium is to provide real HPC applications to validate the system. Herein we provide a status report of the project initial developments.**

I. Introduction

The ExaNeSt European project (call H2020-FETHPC-2014, n. 671553) aims to develop both the system-level interconnect and a fully-distributed NVM (Non-Volatile Memory) storage together with the cooling infrastructure for an European Exascale-class supercomputer based on low cost, low-power many ARM-cores plus computing accelerators (FPGAs).

This approach is shared with other European projects with synergic goals: ExaNoDe [1] and ECOSCALE [2].

The Consortium brings together industrial and academic research expertise to develop all the different components of the system:

- the packaging and advanced cooling mechanism: a key component of a system aiming to be dense and power efficient;
- a novel distributed storage architecture based on local storage devices attached to the computing nodes enabling near-data computation and reducing the energy and latency of I/O traffic. In addition we exploit the high bandwidth of the NVM-e devices by using a custom parallel file system (BeeGFS [3]);
- low latency, high throughput unified RDMA-based interconnections (UniMem [4], APEnet [5]...) enable the scalability of both storage and I/O bandwidth, together with the compute capacity. The interconnection will be designed and validated using different topologies on FPGAs-based test-bed (see section II) to better exploit congestion-minimizing routing functions and network support to system resiliency;
- a set of relevant Exa-scalable scientific and big-data applications (DPSNN, GADGET-3, PINOCCHIO,...) needed to efficiently co-design the ExaNeSt architecture in design phase and to benchmark and optimize the implementation of the final prototype platform.

[*]INFN Sezione di Roma, Italy.
[†]INFN Sezione di Roma Tor Vergata, Italy.
[‡]University of Manchester, England.
[§]FORTH - Foundation For Research & Technology, Hellas.
[¶]INAF, Osservatorio Astronomico di Trieste
[‖]Electronic Engineering Dept., University of Roma "Tor Vergata", Italy
M. Martinelli is the corresponding author (email: michele.martinelli@roma1.infn.it.)

This work was carried out with support from the ExaNeSt project, funded by the European Union Horizon 2020 Research and Innovation Programme under Grant Agreement No. 671553, and from the Human Brain Project (HBP), Grant Agreement No. 720270 (HBP SGA1).

A. ExaNeSt HW description

The development of an RDMA-compliant, configurable network architecture possessing both low latency and high bandwidth that offered at the same time the power of a microprocessor while maintaining the versatility of an FPGA motivated the choice of a device with an integrated MPSoC. The device we chose in order to implement and test the ExaNeSt Network is a Xilinx Zynq Ultrascale+ FPGA. The device integrates four ARMv8 Cortex-A53 (PS) hard cores running at 1.5 GHz interfaced to the Programmable Logic (PL) through AXI Bus; this is referred as "unit". The on-chip availability of the hardened ARM cores grants low latency between PS and PL; the system is also equipped with a Memory Management Unit (MMU) with two-stage translation and 40-bit physical addresses for easy resolution of virtual addresses thus allowing user-level initiation of UNIMEM communication. Moreover, the same ARM sub-system provides cache-coherent access from the PL to the memory of remote systems.

A Quad-FPGA Daughter-Board (QFDB, described in Fig. 1), containing four Zynq Ultrascale+ FPGAs, 64 GB of DRAM and SSD storage, is the so called ExaNeSt "node". The QFDB FPGAs are specialized for different tasks: two of them are pure computing nodes, the third one (the "Storage FPGA") additionally manages an SSD interface while the "Network FPGA" is the QFDB network peer.

The topology of Intra-QFDB connectivity is *all-to-all* and it is implemented using two different concurrent technologies: a) the LVDS parallel bus, allowing for short messages low latency communication, and b) multiple high speed serial links (GTH) allowing for high bandwidth, high throughput packet-based communications. Inter-QFDBs connectivity is built on

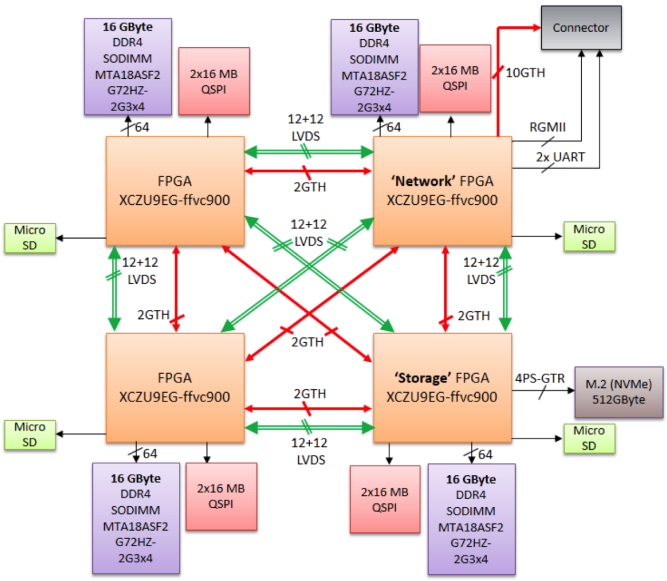

Fig. 1. Schematic of the QFDB module

top of a set of multiple, independent GTH channels integrated in the QFDB Network FPGA. In the first ExaNeSt system prototype release, four out of ten channels are used to connect neighbouring QFDBs hosted on the so called "mezzanine" (or "blade") while remaining channels are routed to standard SFP+ connectors, placed on the edge of the mezzanine and allowing blade-to-blade connectivity with configurable topology (more details are reported in section II).

A single "Chassis" hosts 9 blades; a "Rack" is an assembly of multiple (up to 8) Chassis and a "Top-of-the-rack" (TOR) switch for inter-racks connectivity. The design of a TOR based on advanced fully optical technology will be explored to sustain the huge bandwidth traffic generated by the inter-racks connectivity.

The system mechanic components (blades, chassis and racks) are designed ad-hoc and leverage on a new liquid-cooled technology (developed by consortium partner ICEOTOPE) allowing for high density and power-dissipation effective system integration.

II. Interconnection

ExaNeSt explores two kinds of networks: direct blade-to-blade (Inter-Mezzanine Channels) and indirect blade-switch-blade (Intra-Mezzanine Channels). In the former case, the switching and routing functions towards units are integrated close to computing elements; in the latter the network is tested by connecting the blades to commercially available components, based on ASICs or FPGAs. In order to test several direct network topologies, 24 SFP+ connectors on each mezzanine can be used — with an aggregated bandwidth of 240 Gbps — to connect each mezzanine within the same Chassis.

A. Unimem-based

The UNIMEM memory model, originally developed in the context of the EuroServer [6] project, offers a global address space that allows direct access to a remote node memory and it will be tested in large-scale ARM-based systems. With UNIMEM, in order to enable a global address space, each "coherence island" (the four A53 processors inside an FPGA) has to allocate disjoint regions of physical memory to local memory and peripherals, and to the external world. This exposed memory is in the form of an address "window" which enables direct memory and I/O accesses to remote islands. Various communication protocols (AXI, Ethernet, etc.) can be used to address this remote world, while a translation mechanism provides a mapping (either static or dynamic) between the island physical address space and the global address space. In the ExaNeSt prototype, the Zynq Ultrascale+ MPSoC will support a 448 GB memory window, *i.e.* more than 16 GB of DRAM memory for 16 "coherence islands".

B. Automatic generation of system-level topologies

The use of FPGA-based switches provides high flexibility in order to select the design to be employed for the interconnect. The aim of ExaNeSt is also to compare our custom-made interconnect with other proprietary solutions to select one that provides us the best performance/efficiency ratio. This flexibility offers the chance of favouring topologies that prioritize one (or several) of the following objectives: reach specific performance levels (*i.e.* applications with known communication patterns), achieve fault tolerance requirements or maintain certain cost-efficiency levels. However, the large size of the system together with its reconfiguration capability lead to an extremely large design space that is unfeasible to be manually explored. In order to maintain system flexibility, we prefer to avoid restricting our design to standard topologies only (*e.g.* fat-trees, dragonflies or tori) and thus we decided to automate the exploration of the search space by developing a framework to automatically generate and evaluate large-scale inter-rack topologies [7]. This framework guides the selection of network topologies that favour some specific characteristics. In the case of ExaNeSt, the main design aims are to maximize performance and fault tolerance, whilst keeping the cost of connectivity as moderate as possible. With this in mind, we have developed objective functions that maximize the bisection bandwidth and the path-diversity while minimizing the number of physical connections.

The generated inter-rack topologies were shown to reduce communication times by up to 60% when compared with an irregular jellyfish network. Future work will expand this framework to support other forms of topologies and will be compared with tori and fat-trees.

The evaluation of inter-rack topologies for ExaNeSt imposes a big challenge due to the size of the fully populated system. For this reason, we are developing two different simulators: INSEE [8] and INRFlow. The strengths of each simulator are different: while INSEE is a flit-level simulator that uses a very detailed model of the interconnect, INRFlow is a flow-level simulator, less detailed than INSEE. This difference determines the size of the networks that they are able to simulate. INSEE can deal with networks in the order of thousands of nodes whilst INRFlow can scale the simulations

up to millions of nodes. Both simulators implement several types of networks, including trees and tori, as well as some hybridizations.

Both simulators are capable of evaluating with realistic traffic by maintaining traffic causality, either at packet- or flow-level, including traces from the ExaNeSt applications. In addition we have implemented a range of custom-made traffic generators modelling HPC applications, such as stencil-based, collectives or multi-step, as well as specific Data-driven patterns, such as *mapreduce* or graph-based applications.

III. Storage

Traditional multi-computer systems use an independent network to carry out storage traffic. Opposite to this approach, ExaNeSt uses a novel architecture based on local storage devices attached to the computing nodes. This enables near-data computation and reduces the energy and latency of I/O traffic. In addition, we take advantage of the high bandwidth of the NVM-e devices which will be used as a cache layer to store data and will be accessed by the applications using a custom parallel file system (BeeGFS [3]). However, this storage model imposes new challenges. The performance could be severely affected by the use of an unified, low latency interconnect for both types of traffic due to their specific characteristics: while application-oriented traffic is more sensitive to latency, the storage traffic is more sensitive to bandwidth. For this reason, avoiding traffic interference is crucial in ExaNeSt and so we are exploring the possibility of enhancing schedulers with specific allocation policies that exploit temporal and spatial locality to minimize these interferences. Preliminary results show a huge potential for improving the performance of I/O-intensive applications, up to one order of magnitude in many configurations but still positive even in pathological low-locality configurations. Furthermore, we are also exploring traffic prioritization mechanisms to dynamically change the priority of the flows, based on their type to deal with latency and throughput. Both techniques combined will minimize the interference between both intra-application traffic (application- and storage-oriented) and inter-application (generated by different applications sharing the network).

IV. Applications

A. DPSNN

The Distributed and Polychronous Spiking Neural Network (DPSNN) simulation engine is a natively distributed mini-application benchmark representative of plastic spiking neural network simulators, coded as a network of C/C++ processes interfaced using a pure MPI implementation.

Each process describes the synapses in input to a cluster of neurons with an irregular interconnection topology, with complex inter-process traffic patterns broadly varying in time and per process. It has been designed to be natively distributed and parallel, with full parallelism exploited also during the creation and initialization of the network.

DPSNN is based on a mixed time- and event-driven engine, dedicated to the simulation of the dynamics of networks of point-like neurons and synapses, optionally including synaptic spike-timing dependent plasticity (STDP). During each millisecond of simulation, the network evolution is the result of the dynamic of each single neuron, computed at sub-millisecond time resolution. Such a simulated neural activity generates spikes with target synapses distributed both in the same process and in other processes. The exchange of spike messages (*i.e.* the "axonal-spikes" composed of the identity of the source neuron and the original time of emission), constitutes the inter-process traffic payload.

Speed-up measures and strong and weak scaling analysis have been performed to demonstrate the scalability of simulations run over MPI processes ranging from 1 to 1024. To address this issue, we ran a number of simulations of spiking neuron networks organized in bi-dimensional grids of modules, each module aiming at modelling a cortical column, including up to 50 G synapses connecting 46 M point-like neurons (Leaky Integrate and Fire with Spike Frequency Adaptation) distributed over a large set of MPI processes. The execution platform was a server composed of up to 64 dual-socket nodes, each socket was equipped with Intel Xeon Haswell E5-2630 v3 processors (8 cores @ 2.40 GHz clock).

B. GADGET-3 and PINOCCHIO

In Astrophysics, the scientific problem under investigation often involves complex physics, a very large dynamical range, or both. This kind of computations requires high resolution, that translates in a very large number of computational elements — usually particles.

In the ExaNeSt project we use the TreePM+SPH code GADGET-3, an evolution of the public code GADGET-2 (GAlaxies with Dark matter and Gas intEracT) [9], and PINOCCHIO semi-analytical code [10]. In simulations used in ExaNeSt, besides gravity and hydrodynamics, the following processes are modeled: radiative cooling of the gas, star formation, chemical evolution, energy feedback from exploding stars, uniform time-dependent UV background, evolution of black holes and their energy outputs. These numerical computation are dynamical in space and time: the computation evolves from an initially homogeneous, easy to balance configuration, to an extremely non-homogeneous one. Furthermore, galaxies move with respect to each other, possibly colliding and clustering together. This behaviour makes load-balancing a severe issue and those codes an excellent and challenging candidate to design and optimize the interconnect of a supercomputer.

To test the environment and to verify our porting, we are using two different cosmological simulations. The first one is a portion of the Universe, a cube having a side of 25 Mpc, with relatively low resolution (details in [11]). This simulation does not resolve the internal structure of galaxies and is thus relatively easy to balance in all of its phases. The second one follows the birth and the evolution of a single galaxy. Here, the resolution is higher and the dishomogeneity towards the end of the simulation is larger (details in [12]). In neither case we run the full simulation, which would be very time-consuming.

V. Status of test-bed and preliminary performance

Being the early stages of the ExaNeSt system prototype so that the node is still under development, the Trenz TEBF0808 [13] system has been used since it features the same Xilinx Ultrascale+ MPSoC FPGA family chosen for the final prototype (XCZU9EG). Preliminary tests were performed to validate the network, connecting up to four boards through the two SFP+ connectors available on each system.

A. KARMA

King ARM Architecture (KARMA) is a software-oriented test-bed for the network subsystem. On the hardware side, the switch FIFOs are directly connected to the ARM HPM AXI port through an adapter IP, whose only purpose is the conversion between streaming and memory-mapped protocols. Moreover, a set of configuration/status registers is accessible on the same AXI bus through the "Target Controller" IP, which allows configuration of the router (*e.g.* setting coordinates and lattice size) and probing FIFOs and link status.

On the software side, we created a first test in user-space by simply writing commands and data to the hardware (using the `/dev/mem` to access the memory-mapped hardware). In this phase we have no interrupts, no system-wide locking and no easy virtual-to-physical address translation.

Then, we developed a kernel-space device driver to create a "proc" file-system entry to output debug and status information, together with the output of the internal configuration/status registers.

The module also parses the device tree to find the IRQ number associated to the "NIC" and then assigns a callback function to handle the interrupt request generated by the arrival of new data.

Interrupts are handled both in the "top half" by a kernel module callback function, and in the "bottom half" by using a work-queue which is in charge of asynchronously copying the freshly landed data into the pre-registered user buffer.

In the sending phase, the kernel-space module copies data from the user buffer to a bounce buffer, then prepares header and footer and writes them onto the corresponding FIFOs.

The receiving phase is just the opposite: arriving data is copied in a bounce buffer (waiting for the user-space process to request them) while header and footer are "consumed".

Round-trip latencies between two boards have been measured at different sizes, up to 4 KB. Because of the non-optimal bounce-buffering mechanism and the notoriously slow interrupt handling by GNU/Linux, in Fig. 2 we compared these results with a test where the kernel driver is bypassed by a user-space ping-pong application, again exploiting `/dev/mem` to directly access the memory-mapped hardware.

VI. Porting scientific applications kernels on test-bed

Direct porting the DPSNN application to run on the Trenz-based prototype has not required a significant effort besides sidestepping a couple of minor quirks in order to make either MPICH and OpenMPI work on an ARM boards cluster.

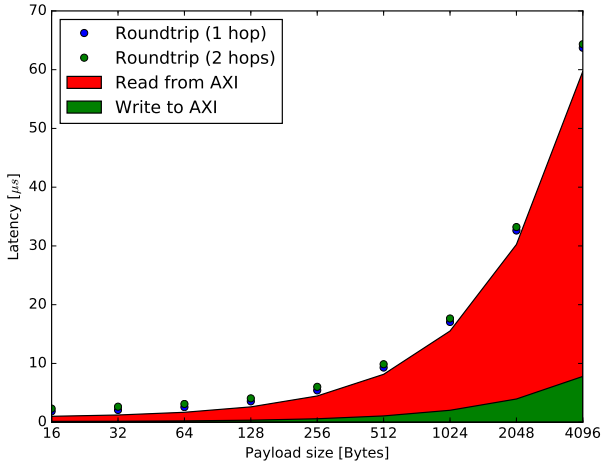

Fig. 2. Performance results in terms of total latency for a ping-pong test.

Of course, the data-set must be sized accordingly to fit the available memory; we note that at the moment inter-node communication has no better channel than 1 Gb Ethernet. In this regard, some reshuffling of allocations relieving the pressure the application put on the memory subsystem of the Trenz boards was necessary in order to make it run; moreover, message passing between processes was changed from variable payloads to a tunable fixed length exchange plus eventual remainder, which was found to work better for the range of sizes that form the bulk of the exchanges in a DPSNN run.

We are investigating another, more complex optimization: we envision a two-level hierarchy where, during collective communications, each process distinguishes between 'local' and 'distal' peers and accordingly uses different communicators. In this way, exchanges between local peers — processes belonging to the same node — can be made to employ a different channel compared to communications between distal ones — processes belonging to different nodes — which makes available different, possibly concurrent strategies for optimization. For example, inter-node collective communication can be made to exploit the broadcast/multicast capabilities of a dedicated IP on the FPGA, just like those under investigation in the APEnet+ project [14].

In table I we report the simulation run-times of a reference configuration (5000 simulated milliseconds of an 8×8 grid of cortical columns, 1250 neurons per column) for different layouts of cores and cluster nodes compared against those of a standard HPC platform (nodes are dual-socket servers populated with Intel CPUs — Xeon E5-2630 v2 clocked at 2.6 GHz) interconnected with an InfiniBand network interface.

TABLE I
DPSNN runtimes.

# Nodes ÷ # Cores	Time (ARM)	Time (Intel)
1 ÷ 1	3656.5s	632.9s
1 ÷ 2	1964.6s	336.0s
1 ÷ 4	1151.8s	181.6s
2 ÷ 8	600.5s	83.2s
4 ÷ 16	317.1s	40.7s

The porting efforts for GADGET and PINOCCHIO are aimed at redesigning the codes to profit of Exascale platforms and in particular the one designed by the project where a large number of ARM cores coupled with FPGA accelerators. Actually, we ported the code as it is on the available test-bed together with all the required libraries. Minor tests on scalability were performed together with some integration tests to verify that both libraries and codes were producing the same results as on Intel based clusters.

Moreover, to study the detailed network traffic produced by the code through MPI library calls, we instrument the code and run it for small temporal intervals at the beginning of the simulation, at intermediate times and towards the end, in order to sample the different dynamical situations, that can turn into different communication patterns and workload balances. In this way we collect statistics on network traffic and API usage.

Even if GADGET and PINOCCHIO are full-fledged high-performance codes for cosmological simulations, some architectural concerns arise at the edge of Exascale era that ponder on the necessity of redesigning the code to fully profit of the ExaNeSt platform. On such a system, it is crucial for codes to take advantage of spatial and temporal locality of data, which also requires NUMA-awareness. This concept translates into the "affinity" that a given memory segment has with a particular computing core. The GADGET memory model is not NUMA-aware and tries to exploit memory locality only with some basic stratagems. Hence, the redesign of the code must start from the redesign of the memory model in order to take into account the different affinities of different memory regions. Due to the extreme diversity of physical processes being modeled and algorithms implemented, this is a difficult task that does not have a unique solution and may require memory layout transformations in some points. Secondly, the code has been conceived as the parallel generalization of a serial code, in a pre-multithread era. Hence, the work-flow is rigidly procedural, with this meaning that all tasks perform the same operations — possibly individually using more threads in local loops — with frequent synchronization via MPI messages. In view of this fact, it is compulsory to adopt a different code design, *i.e.* to decompose the work-flow in as-small-as-possible single tasks with clear dependencies on, and conflicts between, each other from both the point of view of operations to be performed and data to be processed. In such a way, the work-flow would be translated in a queue system where idling threads perform the first available tasks on not-under-use data. Synchronization of operations should pass as much as possible through RDMA operations and the queue system itself.

We completed a first redesign of PINOCCHIO code and our next step is to proceed with GADGET. We will start from a stripped-down version of the publicly available code, that incorporates only the gravity and hydrodynamical solvers, and the star formation plus the stellar evolution modules. We plan to separately develop mini-apps for each of the core algorithms, designed *ab-initio* as "atomic" inter-dependent tasks. In this way we will be able to develop different algorithms and strategies for each of the "pillars" detailed above in a much easier way than in a unique, monolithic code.

VII. Future work and conclusions

In this paper we discussed the general overview and preliminary results of the ExaNeSt project in terms of hardware architecture and software development. A complete ExaNeSt prototype is expected to be deployed at 4Q2018.

References

[1] "ExaNoDe," http://exanode.eu/, accessed: 2017-04-24.

[2] "ECOSCALE," http://ecoscale.eu/, accessed: 2017-04-24.

[3] "BeeGFS," https://www.beegfs.com, accessed: 2017-04-24.

[4] M. Marazakis *et al.*, "Euroserver: Share-anything scale-out microserver design," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 678–683.

[5] R. Ammendola, A. Biagioni, O. Frezza, A. Lonardo, F. Lo Cicero, M. Martinelli, P. Paolucci, E. Pastorelli, D. Rossetti, F. Simula, L. Tosoratto, and P. Vicini, "Architectural Improvements and Technological Enhancements for the APEnet+ Interconnect System," *Journal of Instrumentation*, vol. 10, no. 02, p. C02005, 2015. [Online]. Available: http://stacks.iop.org/1748-0221/10/i=02/a=C02005

[6] "EUROSERVER," http://www.euroserver-project.eu/, accessed: 2017-04-24.

[7] J. A. Pascual, J. Lant, A. Attwood, C. Concatto, J. Navaridas, M. Luján, and J. Goodacre, "Designing an exascale interconnect using multi-objective optimization," in *Proc. of IEEE Congress on Evolutionary Computation*, ser. CEC, 2017.

[8] J. Navaridas, J. Miguel-Alonso, J. A. Pascual, and F. J. Ridruejo, "Simulating and evaluating interconnection networks with insee," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 494–515, 2011.

[9] V. Springel, "The cosmological simulation code gadget-2," *Monthly Notices of the Royal Astronomical Society*, vol. 364, pp. 1105–1134, Dec. 2005.

[10] P. Monaco, T. Theuns, and G. Taffoni, "The pinocchio algorithm: pinpointing orbit-crossing collapsed hierarchical objects in a linear density field," *Monthly Notices of the Royal Astronomical Society*, vol. 331, pp. 587–608, Apr. 2002.

[11] P. Barai, P. Monaco, G. Murante, A. Ragagnin, and M. Viel, "Galactic outflow and diffuse gas properties at $z \geq 1$ using different baryonic feedback models," *Monthly Notices of the Royal Astronomical Society*, vol. 447, pp. 266–286, Feb. 2015.

[12] G. Murante, P. Monaco, S. Borgani, L. Tornatore, K. Dolag, and D. Goz, "Simulating realistic disc galaxies with a novel sub-resolution ISM model," *Monthly Notices of the Royal Astronomical Society*, vol. 447, pp. 178–201, Feb. 2015.

[13] "Trenz Board TEBF0808," wiki.trenz-electronic.de/display/PD/TEBF0808, accessed: 2017-04-24.

[14] R. Ammendola, A. Biagioni, O. Frezza, F. L. Cicero, A. Lonardo, M. Martinelli, P. S. Paolucci, E. Pastorelli, D. Rossetti, F. Simula, L. Tosoratto, and P. Vicini, "Hardware and Software Design of FPGA-based PCIe Gen3 interface for APEnet+ network interconnect system," *Journal of Physics: Conference Series*, vol. 664, no. 9, p. 092017, 2015.