Project: **AXIOM - Agile, eXtensible, fast I/O Module for the cyber-physical era**
Grant Agreement Number: **645496**
Call: **ICT-01-2014: Smart Cyber-Physical Systems**

FRAMEWORK PROGRAMME
ICT-01-2014: Smart Cyber-Physical Systems

PROJECT NUMBER: 645496

Agile, eXtensible, fast I/O Module for the cyber-physical era

D5.3 – Parallel Programming Library and documentation

Due date of deliverable: 31st January 2017
Actual Submission: 7th February 2017 (agreed extended date)

Start date of the project: 1st February 2015 Duration: 36 months

Lead contractor for the deliverable: EVI

Revision: See file name in document footer.

Project co-founded by the European Commission within the HORIZON FRAMEWORK PROGRAMME (2020)	
Dissemination Level: PU	
PU	Public
PP	Restricted to other programs participant (including the Commission Services)
RE	Restricted to a group specified by the consortium (including the Commission Services)
CO	Confidential, only for members of the consortium (including the Commission Services)

Change Control

Version#	**Date**	**Author**	**Organization**	**Change History**
0.1	25.01.2017	Paolo Gai, Stefano Garzarella	EVI	Initial version taken from the Google doc
0.2	01.02.2017	Paolo Gai, Davide Catani	EVI/SECO	First revision and typos.
0.3	02.02.2017	Xavier Martorell	BSC	Second revision.
0.4	04.02.2017	Paolo Gai	EVI	Final version for last revision.
0.5	05.02.2017	Paolo Gai	EVI	Added executive summary.

Release Approval

Name	**Role**	**Date**
Paolo Gai	WP Leader	05.02.2017
Roberto Giorgi	Project Coordinator for formal deliverable	06.02.2017

The following list of authors will be updated to reflect the list of contributors to the document.

Paolo Gai, Stefano Garzarella, Bruno Morelli
R&D Department
Evidence

Carlos Álvarez, Daniel Jiménez, Xavier Martorell
CS Departament
Barcelona Supercomputing Center (BSC) - AXIOM
Universitat Politècnica de Catalunya - AXIOM

TABLE OF CONTENTS

TABLE OF FIGURES

GLOSSARY

ARM – Instruction set architecture developed by ARM Holdings Ltd.
ASIC – Application-specific integrated circuit
ATLAS – Automatically tuned lineal algebra software
BLAS – Basic linear algebra subprograms
CNN – Convolutional neural network
FC – Fully-connected layer
GLFW – Open-source multiplatform library for OpenGL, OpenGL ES and Vulkan
IP – Intellectual property
LibAv – Open-source libraries derived from the FFmpeg project to handle multimedia data
LBP – Local binary pattern
LRN – Local response normalization
Mali – A GPU microarchitecture developed by ARM Holdings Ltd.
Mercurium – OmpSs compiler
MKL-DNN – Intel Corporation math kernel libraries for deep neural networks
Nanos++ – OmpSs runtime
NEON – SIMD extensions for the ARM instruction set
NDA – Non-disclosure agreement
NIC – Network interface
OpenGL ES – Reduced specification of the OpenGL standard that targets embedded devices
PL – Programmable logic
PReLU – Parametric rectified linear unit
Qt – Cross-platform application framework developed by The Qt Company
ROC – Receive operating characteristic
RTL – Register transfer language
RTSP – Real-time streaming protocol
SDSM – Software Distributed Shared Memory
SIMD – Single instruction, multiple data
SMT – Simultaneous multithreading
SoC – System on chip
SGEMM – Single-precision floating-point general matrix multiply
STD – Standard deviation
SVS – Smart surveillance scenario for the AXIOM board
SHL – Smart home living scenario developed for the AXIOM project

1 Executive summary

This document is a short guide through the software release that is part of Task T5.3. The software release includes a complete software stack starting from the AXIOM Linux drivers and arriving to the OmpSs@ cluster programming library.

The release also includes a QEMU emulation framework that is able to emulate the AXIOM-link, and a VirtualBox image with all software preinstalled to ease the distribution and testing of the developed software. A small tutorial accompanied by a short video shows how to use the software on two example applications: a matrix multiply and an Nbody simulation.

All this software has been released on the project public website (as due) on the 31st January 2017:

http://download.axiom-project.eu

2 Introduction

The content of this document is a set of “pointers” to a large quantity of public material, repositories, documentation and videos developed in the context of the AXIOM project. It also offers a brief guide that enables external users to re-use (for research, study or further development) the AXIOM stack at the current development stage. The results of D5.3 represents also the completion of the efforts initiated in the Task T5.2 and already documented in D5.2.

During the development of the software stack, we preferred to test the functionality of it on a QEMU based platform (in line with the Xilinx recommendations and related tools).

2.1 Document structure

Section 3 provides a short description of the AXIOM architecture, useful to understand the various subsystems. Section 4 includes a description of the various subsystems, providing their location and documentation pointers. Section 5 provides a short guide on how to use the AXIOM software stack. Finally, Section 6 includes a list of all the files object of this release.

2.2 Relation to other deliverables

This document is linked to the following internal deliverables of the AXIOM project:

D5.1 Operating system and documentation

> This document describes in detail the implementation of the AXIOM Linux distribution for the AXIOM board, and the AXIOM NIC interface.

D5.2 Remote Memory Access

> This document describes the cluster setup, memory organization, memory allocator, and task synchronization. It also include details on the implementation of the Parallel Programming library.

D4.2 AXIOM Code Generation and instrumentation

> This document describes the OmpSs compilation and FPGA support as well as the instrumentation mechanism present in OmpSs

2.3 Tasks involved in this deliverable

Task 5.3 (month 6 - 24): Parallel programming library

What:

This task will consist of the porting of the Nanos++ library supporting OmpSs, on the reference platform, made by partner BSC in collaboration with partner EVI.

How:

BSC with the help from partner EVI for integration. Nanos++ will implement the runtime support for the target directive as defined in Task 4.2. The implementation consists of the necessary data movements from the ARM main memory to the FPGA and vice-versa, and the management will start and manage of the application kernels on the FPGA. Nanos++ will rely on the remote memory access mechanism developed in Task T5.2 to run tasks on remote boards. Our work will include the development of a caching mechanism across boards to reduce the need for communication, and overlap as much as possible computation and communication.

Expected output:

- The software developed in this task – running on the reference platform- released as Open-Source software, and added onto our Nanos++ distribution in the BSC website pm.bsc.es/nanox.

D5.3: Parallel programming library and documentation [24]

This deliverable will consist in an update of deliverable D5.1 with the addition of the Linux kernel driver for the highspeed interconnect developed in Task 5.1, the Remote Memory Access mechanism developed in Task 5.2 and the library for parallel programming developed in Task 5.3.

3 The AXIOM cluster and the AXIOM software stack

The AXIOM system is composed by a set of computing boards connected together to form a cluster configuration called “AXIOM-cluster”. The connection between boards is implemented through the AXIOM-link, which is a custom network interface (NIC) developed in the Xilinx Zynq FPGA as part of the AXIOM project.

For the sake of developing the software stack, we implemented an emulation layer based on the QEMU emulator also following the Xilinx recommendations for using their SoC/FPGA products. The emulation layer is able to emulate both the AXIOM network interface (NIC) registers, as well as the connections between the boards. In particular, the latest versions refer to the Zynq Ultrascale+ (64-bit) platform.

Figure 1 presents the high-level view of the AXIOM software stack. In particular, we note in the background that all nodes are interconnected (as planned) through the AXIOM-link. Moreover, each board has an FPGA part, including the AXIOM NIC and the XSMLL Layer. The current release, described in this document, includes an emulation of the AXIOM NIC, and does not include an emulation of the XSMLL Layer (not part of this task). The software stack is then running on each board, and is composed by the following set of main components:

- A Linux distribution (not shown in the Figure). The current release includes a minimal buildroot distribution (by the end of the AXIOM project, a complete distribution based on the PetaLinux SDK with an Ubuntu 16.04LTS root filesystem will be implemented, see D5.1);
- A set of device drivers (used to provide support for the AXIOM NIC and for the AXIOM Memory Allocator);
- A set of user libraries to handle the interfacing between the applications and the kernel drivers;

- A set of utility applications, including the daemon `axiom-init` and the `axiom-run` spawner;
- The OmpSs programming libraries, including Nanos++, GASNet, and the AXIOM GASNet conduit;
- The compilation toolchain, including the GCC cross-compiler developed as part of Buildroot, and the Mercurium source-to-source compiler.

The following Section contains a list of the software packages, including a short description. In order to simplify the usage of the software stack, at this stage we provided a pre-installed VirtualBox virtual machine, with an accompanying video, in Section 5.

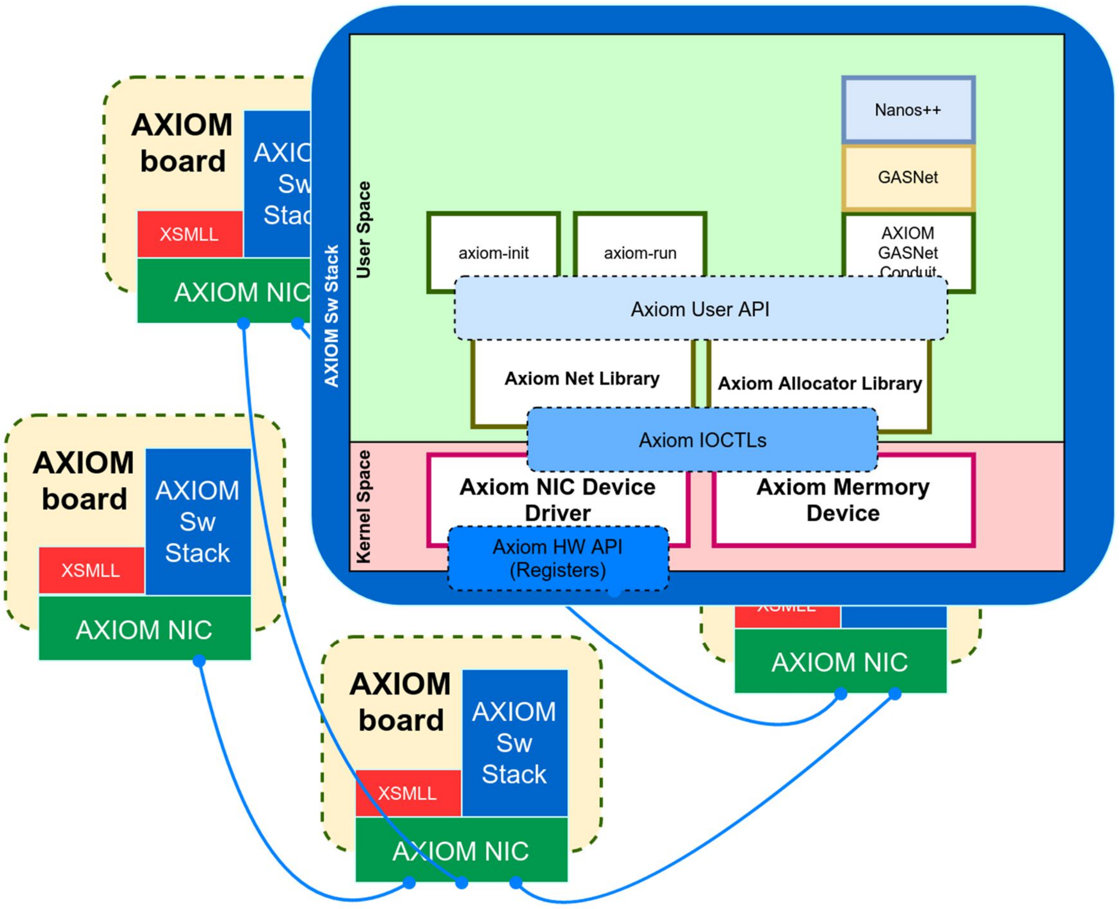

Figure 1 - The AXIOM software stack.

4 Components of the software stack

The following is a list of the main components of the AXIOM software stack. It is meant as a reference that will help to navigate the various packages and directories.

4.1 AXIOM Drivers, User libraries and applications

4.1.1 AXIOM NIC driver and libraries source code

- Short Description: Implementation of AXIOM NIC device driver, User Space libraries and documentation.
- Repository: https://git.axiom-project.eu/axiom-evi-nic
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-nic`
- Doxygen Documentation: in `axiom-v0.11-doxygen` PDF or HTML, `files/axiom-evi-nic`

4.1.2 AXIOM allocator source code

- Short Description: Implementation of the three level AXIOM allocator.
- Repository: https://git.axiom-project.eu/axiom-allocator
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-allocator`
- Doxygen Documentation: in `axiom-v0.11-doxygen` PDF or HTML, `files/axiom-allocator`

4.1.3 AXIOM memory driver source code

- Short Description: Implementation of the memory device driver to handle virtual to physical memory mapping.
- Repository: https://git.axiom-project.eu/axiom-evi-allocator-drv
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-allocator-drv`
- Doxygen Documentation: in `axiom-v0.11-doxygen` PDF or HTML, `files/axiom-evi-allocator-drv`

4.1.4 AXIOM memory library source code

- Short Description: Implementation of 3 level software allocator based on LMM.
- Repository: https://git.axiom-project.eu/axiom-evi-allocator-lib
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-allocator-lib`
- Doxygen Documentation: in `axiom-v0.11-doxygen` PDF or HTML, `files/axiom-evi-allocator-drv`

4.1.5 AXIOM application source code

- Short Description: Implementation of AXIOM support application and daemons (`axiom-init`, `axiom-run`, etc.)
- Repository: https://git.axiom-project.eu/axiom-evi-apps
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-apps`
- Doxygen Documentation: in `axiom-v0.11-doxygen` PDF or HTML, `files/axiom-evi-apps`

4.1.6 AXIOM scripts

- Short Description: Makefile and scripts to compile the all components of the AXIOM software stack.
    - Repository: `https://git.axiom-project.eu/axiom-evi/tree/master/scripts/`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/scripts`

4.1.7 AXIOM tests

- Short Description: Regression and benchmark tests for AXIOM NIC, GASNet and OmpSS.
    - Repository: `https://git.axiom-project.eu/axiom-evi/tree/master/tests/`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/tests`

4.2 GASNet conduit

- Short Description: Modified version of GASNet library that includes the new AXIOM conduit.
- Repository: `https://git.axiom-project.eu/axiom-evi-gasnet`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-gasnet`

4.3 OmpSs framework

4.3.1 Extrae

- Short Description: Modified version of Extrae to support the trace of IOCTLs and AXIOM API.
- Repository: `https://git.axiom-project.eu/axiom-evi-extrae`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-extrae`
- Documentation: `https://tools.bsc.es/extrae`

4.3.2 Mercurium

- Short Description: Modified version of mcxx to support AXIOM GASNet conduit and cross-compilation.
- Repository: `https://git.axiom-project.eu/axiom-evi-mcxx`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-mcxx`
- Documentation: `https://pm.bsc.es/mcxx`

4.3.3 Nanos++

- Short Description: Modified version of nanox to support AXIOM GASNet conduit and cross-compilation.
- Repository: `https://git.axiom-project.eu/axiom-evi-nanox`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-nanox`
- Documentation: https://pm.bsc.es/nanox
- OmpSs User’s Guide: https://pm.bsc.es/ompss-docs/user-guide/

4.4 Emulation and operating system

4.4.1 Buildroot

- Short Description: Modified version of Buildroot to generate the kernel, filesystem and the GCC toolchain for the AXIOM board.
- Repository: `https://git.axiom-project.eu/axiom-evi-buildroot`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-buildroot`

4.4.2 Linux

- Short Description: Modified version of Linux to support AXIOM board emulated in QEMU.
- Repository: `https://git.axiom-project.eu/axiom-evi-linux`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-linux`

4.4.3 QEMU

- Short Description: Modified version of QEMU to emulate the AXIOM-NIC device.
- Repository: `https://git.axiom-project.eu/axiom-evi-qemu`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-qemu`

4.4.4 u-boot

- Short Description: Modified version of u-boot to support AXIOM board.
- Repository: `https://git.axiom-project.eu/axiom-evi-u-boot`
- Directory in the source zip file: `axiom-evi-src-v0.11.tgz/axiom-evi-u-boot`

5 Using the AXIOM software stack

The installation procedure of the AXIOM software stack is documented in the `README` file stored inside the `axiom-evi-src-v0.11.tgz` package. The installation can be performed on a Linux host machine with a recent distribution (in our case we used Ubuntu 16.04LTS).

In order to simplify the installation procedure, we provided a pre-installed VirtualBox virtual machine, which is able to run the complete system, including QEMU. The Virtual Machine is stored in the file `Ubuntu_16_04_64bit_EVI-v0.11.vdi.7z`, and can be executed using VirtualBox (`http://www.virtualbox.org`). When running the Virtual Machine, please provide a few Gb of RAM and at least 2 VCPU. As a reference, the Youtube video below has been run with 4 GiB of RAM and 2 VCPU. This VirtualBox image also includes all the repositories that contain the code described in this document under the directory:

```
/home/axiom/axiom-git/axiom-evi
```

The remaining of this Section is a short tutorial that guides you on a first trial of the AXIOM software stack. In order to make the tutorial simpler, we recorded an execution of the described steps inside a Youtube Video, which is publicly available at the following address:

Video title: The AXIOM software stack emulated on QEMU
https://www.youtube.com/watch?v=fT7SLVXlkOU

Please note that the video is meant to be little more than the recording of the commands below, and is not meant of a general presentation of the AXIOM project.

In order to login into the Virtual Machine, please use the following credentials:

Username:	`axiom`
Password:	`axiom`

Open a terminal. Please execute the following commands:

```
cd /home/axiom/axiom-archive/
cd axiom-nic-evisim-v0.11
```

- These two commands move you to the directory where the QEMU installation is located.

```
make VMS=3 run log
```

- This command starts 3 QEMU virtual machines, with QEMU logging enabled.
- Each QEMU instance emulates a Xilinx Ultrascale+, 64 bit, with 4 cores each;
- The AXIOM-link network is configured with a ring topology.

In the QEMU console, once the QEMU guest Linux system has booted:

- on all nodes but one:
  ```
  cd /root/
  ./axiom-startup.sh
  ```
  in order to start the AXIOM subsystem (including loading kernel modules and the `axiom-init` daemon) on the node as a slave.
- on the last node:
  ```
  cd /root/
  ./axiom-startup.sh -m
  ```
  in order to start the last node as master. This will trigger also the discovery algorithm and routing table distribution, which will be printed on the console by all nodes.

On the console of one of the nodes, please try the following axiom applications (the complete list is available by typing `axiom-` on the console, and then the tab command twice):

```
axiom-whoami
```

- prints the current node number.

```
axiom-ping -d 1
```

- pings (talking to the `axiom-init` daemon running on all nodes) node 1.

```
axiom-netperf -d 1 -t long -l 2M
```

- starts a netperf to node 1 with message type long and 2 Mbytes payload.

On a new terminal in the VirtualBox system, we can now setup the toolchain for the compilation with Mercurium using the following commands:

```
cd /home/axiom/axiom-archive/axiom-evi-toolchain-v0.11/
./setup_toolchain.sh
```

After that, it is possible to compile two examples included in the distribution (a matrix multiply and an Nbody simulation) with the following commands:

```
cd examples
make
```

Once compiled, it is possible to copy the binaries to the QEMU guests, using the following command:

```
./utils/host/copy_to_guests.sh -n 3 -o ./output
```

Note: `-n 3` has to match the VMS parameter above. The previous script copies the contents of `./output` directory in the `/tmp` directory of the guests.

At this point, on one of the QEMU nodes, we can start the matrix multiplication example with:

```
cd /tmp
./run_test_ompss_extrae.sh ./ompss_evimm
```

Please note that this example will launch the application with Extrae tracing enabled. To launch the demo without Extrae enabled, please use the following command:

```
./run_test_ompss.sh ./ompss_evimm
```

Once run, it is possible to collect the traces on the various nodes. In a typical OmpSs execution, the OmpSs would have collected the Extrae traces automatically from the nodes of the cluster by using `scp`. In our case, the minimal buildroot system we prepared for QEMU guests does not have the `scp` command, nor the disk space to store the merged trace. For that reason, we use a script on the VirtualBox guest console:

```
cd utils/host
./extrae_merge.sh -d /tmp -n 3
```

The command copies the trace files from the `/tmp` directory of the 3 guests and merges them..

Note: `-n 3` has to match the VMS parameter above.

As a result, the script prints the destination directory in which the Extrae merged trace has been copied. In the Video, the directory was:

```
20170126_153922_trace/output.prv
```

At this point, it is possible to launch the PARAVER viewer using the following command:

```
cd 20170126_153922_trace

wxparaver output.prv ../paraver_cfg/axiom_API.cfg
```

in order to start PARAVER with the configuration of the set of additional events created by the AXIOM process.

At the end of an execution, before running another example, please remember to clean up the Extrae trace on all QEMU nodes, running the following commands on each QEMU guest:

```
cd /tmp/
rm -rf set-0
```

To launch the Nbody test, please use:

```
./run_test_nbody_extrae.sh 256 5
```

If not present, the particles file is created on the fly.

Similar commands as in the matrix multiply example can be used to collect the traces, merge them and show them with PARAVER.

6 Archive released

The following is a list of the packages included as part of this release of the AXIOM project software stack. All files are available for public download from the following address:

https://download.axiom-project.eu/?dir=RUNTIME

- `axiom-nic-evisim-v0.11.tgz`
  - Tarball that contains the precompiled version of the AXIOM NIC simulator with QEMU (+ Axiom NIC emulation) and the buildroot filesystem with the AXIOM drivers, AXIOM libraries, OmpSS libraries and user-space applications.
- `axiom-evi-toolchain-v0.11.tgz`
  - Tarball that contains the toolchain to cross-compile applications for the guests (gcc + Mercurium). There are also some examples to cross-compile.
- `Ubuntu_16_04_64bit_EVI-v0.11.vdi.7z`
  - VirtualBox image (Ubuntu 16.04 LTS x86_64) with source and precompiled version of the AXIOM NIC simulator.
- `axiom-evi-src-v0.11.tgz`
  - Archive that contains all the source repositories. There is a README inside, which explains how to compile the AXIOM software stack.
- `axiom-v0.11-doxygen.pdf`
  - Documentation of AXIOM libraries and user-space application generated by Doxygen.
- `axiom-v0.11-doxygen-html.tgz`
  - Archive that contains the HTML documentation of AXIOM libraries and user-space application generated by Doxygen.
- `axiom_nic_datasheet-v0.11.pdf`
  - Datasheet of the AXIOM NIC interface emulated on QEMU.

7 Confirmation of DoA objectives and Conclusions

This document shortly described the open-source release of the software developed during the first two years of the AXIOM Project. The software includes a full network stack with RDMA support, emulated on a QEMU infrastructure. The software also includes the complete support for the OmpSs@cluster Parallel Programming Library, including support for the AXIOM cluster configuration and the possibility to trace the software execution using the Extrae tool.

In order to simplify the deployment and diffusion of the stack, everything has been packaged in a VirtualBox Virtual Machine, and a video has been recorded to show the main steps to do in order to run the complete system.

Other publications of the project [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] are reported in the reference list.

8 References

1. R. Giorgi, “Transactional memory on a dataflow architecture for accelerating Haskell,” WSEAS Trans. Computers, vol. 14, pp. 794–805, 2015.
2. R. Giorgi and A. Scionti, “A scalable thread scheduling co-processor based on data-flow principles,” ELSEVIER Future Generation Computer Systems, vol. 53, pp. 100–108, July 2015.
3. D. Theodoropoulos et al., “The AXIOM project (agile, extensible, fast I/O module),” in IEEE Proc. 15th Int.l Conf. on Embedded Computer Systems: Architecture, MOdeling and Simulation, July 2015.
4. R. Giorgi, "Scalable Embedded Systems: Towards the Convergence of High-Performance and Embedded Computing", Proc. 13th IEEE/IFIP Int.l Conf. on Embedded and Ubiquitous Computing (EUC 2015), Oct. 2015.
5. R. Giorgi, "Exploring Dataflow-based Thread Level Parallelism in Cyber-physical Systems", Proc. ACM Int.l Conf. on Computing Frontiers, New York, NY, USA, 2016, pp. 6.
6. A. Rizzo, G. Burresi, F. Montefoschi, M. Caporali, R. Giorgi, "Making IoT with UDOO", Interahttps://git.axiom-project.eu/axiom-evi/tree/master/tests/ction Design and Architecture(s), vol. 1, no. 30, Dec. 2016, pp. 95-112.
7. L. Verdoscia, R. Giorgi, "A Data-Flow Soft-Core Processor for Accelerating Scientific Calculation on FPGAs", Mathematical Problems in Engineering, vol. 2016, no. 1, Apr. 2016, pp. 1-21.
8. S. Mazumdar, E. Ayguade, N. Bettin, S. Bueno J. and Ermini, A. Filgueras, D. Jimenez-Gonzalez, C. Martinez, X. Martorell, F. Montefoschi, D. Oro, D. Pnevmatikatos, A. Rizzo, D. Theodoropoulos, R. Giorgi, "AXIOM: A Hardware-Software Platform for Cyber Physical Systems", 2016 Euromicro Conf. on Digital System Design (DSD), Aug 2016, pp. 539-546.
9. R. Giorgi, N. Bettin, P. Gai, X. Martorell, A. Rizzo, "AXIOM: A Flexible Platform for the Smart Home", Springer Int.l Publishing, Cham, 2016, pp. 57-74.
10. P. Burgio, C. Alvarez, E. Ayguade, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, N. Navarro, R. Giorgi, "Simulating next-generation cyber-physical computing platforms", Ada User Journal, vol. 37, no. 1, Mar. 2016, pp. 59-63.
11. Jimenez-Gonzalez, Daniel; Alvarez-Martinez, Carlos; Filgueras, Antonio; Martorell, Xavier; Langer, Jan; Noguera, Juanjo; Vissers, Kees, “Coarse-Grain Performance Estimator for Heterogeneous Parallel Computing Architectures like Zynq All-Programmable SoC” (Journal Article) Second International Workshop on FPGAs for Software Programmers FSP 2015, CoRR , 2015.
12. R. Giorgi, S. Mazumdar, S. Viola, P. Gai, S. Garzarella, B. Morelli, D. Pnevmatikatos Dionisios and Theodoropoulos, C. Alvarez, E. Ayguade, J. Bueno, D. Filgueras Antonio and Jimenez-Gonzalez, X. Martorell, "Modeling Multi-Board Communication in the AXIOM Cyber-Physical System", Ada User Journal, vol. 37, no. 4, December 2016, pp. 228-235.