Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

The AXIOM software layers

Carlos Álvarez[a,*], Eduard Ayguadé[a], Jaume Bosch[a], Javier Bueno[a], Artem Cherkashin[a], Antonio Filgueras[a], Daniel Jiménez-González[a], Xavier Martorell[a], Nacho Navarro[a], Miquel Vidal[a], Dimitris Theodoropoulos[b], Dionisios N. Pnevmatikatos[b], Davide Catani[c], David Oro[d], Carles Fernández[d], Carlos Segura[d], Javier Rodríguez[d], Javier Hernando[e], Claudio Scordino[f], Paolo Gai[f], Pierluigi Passera[g], Alberto Pomella[g], Nicola Bettin[g], Antonio Rizzo[h], Roberto Giorgi[h]

[a] *Barcelona Supercomputing Center and Computer Architecture Dept., Universitat Politecnica de Catalunya, Barcelona, Spain*
[b] *FORTH-ICS, Greece*
[c] *SECO, Arezzo, Italy*
[d] *Herta Security, Barcelona, Spain*
[e] *Universitat Politecnica de Catalunya, Barcelona, Spain*
[f] *Evidence Srl, Pisa, Italy*
[g] *VIMAR SpA, Marostica, Italy*
[h] *University of Siena, Siena, Italy*

ARTICLE INFO

Article history:
Received 18 January 2016
Revised 1 June 2016
Accepted 7 July 2016
Available online 9 July 2016

Keywords:
Cyber-physical systems
Ompss
Cluster programming
FPGA Programming
Distributed shared memory
Smart home
Smart video-surveillance

ABSTRACT

People and objects will soon share the same digital network for information exchange in a world named as the age of the cyber-physical systems. The general expectation is that people and systems will interact in real-time. This poses pressure onto systems design to support increasing demands on computational power, while keeping a low power envelop. Additionally, modular scaling and easy programmability are also important to ensure these systems to become widespread. The whole set of expectations impose scientific and technological challenges that need to be properly addressed.

The AXIOM project (Agile, eXtensible, fast I/O Module) will research new hardware/software architectures for cyber-physical systems to meet such expectations. The technical approach aims at solving fundamental problems to enable easy programmability of heterogeneous multi-core multi-board systems. AXIOM proposes the use of the task-based OmpSs programming model, leveraging low-level communication interfaces provided by the hardware. Modular scalability will be possible thanks to a fast interconnect embedded into each module. To this aim, an innovative ARM and FPGA-based board will be designed, with enhanced capabilities for interfacing with the physical world. Its effectiveness will be demonstrated with key scenarios such as Smart Video-Surveillance and Smart Living/Home (domotics).

1. Introduction

We are entering the Cyber-Physical age, in which both objects and people will become nodes of the same digital network for exchanging information. Therefore, the expectation is that "things" or systems will become somewhat smart as people, having to permit a rapid and close interaction not only human-human and system-system, but also human-system, and system-human. More scientifically, we expect that such Cyber-Physical Systems (CPS) will at

* Corresponding author.
E-mail addresses: carlos.alvarez@bsc.es (C. Álvarez), eduard.ayguade@bsc.es (E. Ayguadé), jaume.bosch@bsc.es (J. Bosch), javier.bueno@bsc.es (J. Bueno), artem.cherkashin@bsc.es (A. Cherkashin), antonio.filgueras@bsc.es (A. Filgueras), daniel.jimenez@bsc.es (D. Jiménez-González), xavier.martorell@bsc.es (X. Martorell), nacho.navarro@bsc.es (N. Navarro), miquel.vidal@bsc.es (M. Vidal), dtheodor@ics.forth.gr (D. Theodoropoulos), pnevmati@ics.forth.gr (D.N. Pnevmatikatos), davide.catani@seco.com (D. Catani), david.oro@hertasecurity.com (D. Oro), carles.fernandez@hertasecurity.com (C. Fernández), cseguramail@gmail.com (C. Segura), javier.rodriguez@hertasecurity.com (J. Rodríguez), javier.hernando@upc.edu (J. Hernando), claudio@evidence.eu.com (C. Scordino), pj@evidence.eu.com (P. Gai), pierluigi.passera@vimar.com (P. Passera), alberto.pomella@vimar.com (A. Pomella), nicola.bettin@vimar.com (N. Bettin), antonioriz@gmail.com (A. Rizzo), giorgi@dii.unisi.it (R. Giorgi).

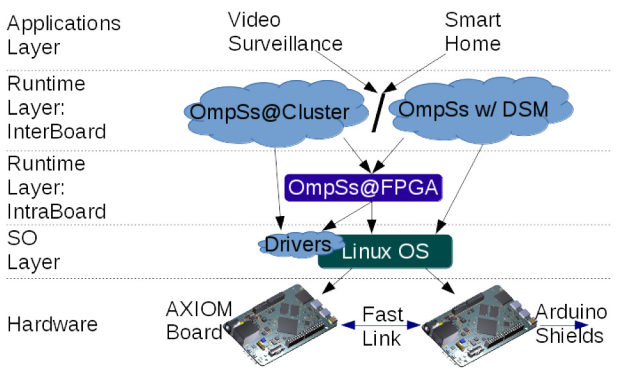


Fig. 1. The AXIOM Software Layers.

least react in real time, provide enough computational power for the assigned tasks, consume the least possible energy for such tasks (energy efficiency), allow for an easy programmability, scaling through modularity and exploit at best existing standards at minimal costs.

The AXIOM project (Agile, eXtensible, fast I/O Module) aims at researching new hardware/software architectures for CPSs in which the above expectations are realized. The project, started on February 2015, will span over 3 years. The coordination of the project is carried out by the University of Siena (UNISI). UNISI also takes the evaluation part of the project. Foundation for Research and Technology - Hellas (FORTH) develops the interconnection between boards. Barcelona Supercomputing Center (BSC) is responsible of the OmpSs (OpenMP+StarSs) programming model and software toolchain. Partner EVIDENCE takes the lead on the development of the runtime systems. Partner SECO designs and builds the prototype board. Partner HERTA Security provides a video-surveillance use case. Partner VIMAR provides a smart-building use case.

Fig. 1 shows the software layers used in this project. As it can be seen the project addresses all the levels of the system, from the application level, that includes two key application domains, to the hardware level. That includes developing a specific runtime software manager (OmpSs@FPGA), a fast interconnection link (Fast Link) and even the AXIOM board itself. As can also be seen in Fig. 1 the project aims to develop a board that can work well both alone or as part of a larger system (i.e. a group of boards interconnected by the AXIOM link). This modular capabilities are addressed from both the hardware side (the implementation of the AXIOM link) and the software side (the development of inter-node execution capabilities using the OmpSs programming model). From the hardware point of view is one of the aims of the project to make the board accessible in terms of cost (as cheap as possible, even around one hundred euros) while making it powerful enough to deal with the envisioned use cases. This holistic development is what we call the AXIOM platform.

The specific objectives of the AXIOM project are:

- Realizing a small board that is flexible (suitable for a wide range of applications), energy efficient and modularly scalable (AXIOM Board in Fig. 1). We will use an ARM- and FPGA-based chip with custom high-speed interconnects to build the AXIOM prototype board.
- Easy programmability of multi-core, multi-board, FPGA node, with the OmpSs programming model (OmpSs@Cluster/OmpSs over DSM, and OmpSs@FPGA in Fig. 1), and improved thread management and real-time support from the operating system. The software will be Open-Source.
- Easy interfacing with the Cyber-Physical world, based on the Arduino shields [1,2], pluggable onto the board. This shields are going allow the developed board to be extended with sensors (e.g. a camera). They will provide new functionalities to the developed board to widen the scope of its applications.
- Contribute to standards, in the context of the Standardization Group for Embedded Systems (SGET) and OpenMP.

The rest of the paper is organized as follows. Section 2 explains the AXIOM software layers. Section 3 explains the AXIOM link development. Section 4 explains the applications evaluated and the expected scenarios. Section 5 explains the experimental setup followed by Section 6 that presents the first results obtained by the project. Section 7 explains the related work. Finally, Section 8 summarizes the conclusions and the envisioned future work.

2. The AXIOM software

One of the problems when building a complex ecosystem like the one described in Fig. 1 is how to easily program applications that should take advantage at the same time of both on-chip resources (i.e. the FPGA and the multiple cores) and multiple board resources (through fast link multiple board connection).

Several solutions have been proposed during the last decades to parallelize computations on multi-core systems. However, no unanimous consensus on the best solution has been achieved. On one hand, some solutions are based on message-passing mechanisms (e.g., MPI), which are usually considered too difficult to use for developers not accustomed to parallel programming. For example, parallelizing existing legacy serial codes, like face detection, audio processing or search algorithms, with MPI need a large code rewriting to add the communication primitives and synchronization needed. Usually this means to rewrite the full application at once to take advantage of the cluster. Instead, models targeting SMPs, are usually based on code annotations, that allow introducing less changes in the original code, and also incrementally work on the different parts of the applications, that can be tested much earlier than when using message passing.

Another possibility that is going to be explored in this project is the use of a DSM system. Distributed shared memory (DSM) is a form of memory architecture where actually physically separate memories can be addressed as one logically shared address space. The main advantage of this memory organization is that it can be easily programmed as the program can access all the available memory despite its real physical location being the DSM support (probably integrated with the OS) the one responsible of managing the communication. On the other hand, this management when not properly handled can lead to unnecessary or inefficient communication patterns.

AXIOM will leverage OmpSs, a task dataflow programming model that includes heterogeneous execution support as well as data and task dependency management [3] and has significantly influenced the recently appeared OpenMP 4.0 specification.

2.1. The OmpSs programming model

In OmpSs, tasks are generated in the context of a team of threads that run in parallel. OmpSs provides an initial team of threads as specified by the user upon starting the application.

Tasks are defined as portions of code enclosed in the *task directive*, or as user-defined functions, also annotated as tasks, as follows:

```
#pragma omp task [clause − list]
  {structured − work|
  function − declaration|
  function − definition}
```

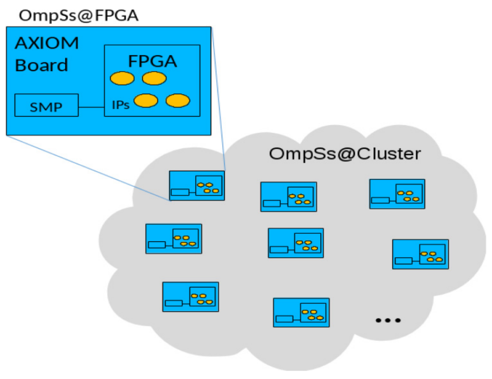


Fig. 2. General view of OmpSs@FPGA and OmpSs@Cluster execution context.

A task is created when the code reaches the task construct, or a call is made to a function annotated as a task. The task construct allows to specify, among others, the clauses *in, out* and *inout*. Their syntax is:

$$\texttt{in}\ (\texttt{data}-\texttt{reference}-\texttt{list})$$
$$\texttt{out}\ (\texttt{data}-\texttt{reference}-\texttt{list})$$
$$\texttt{inout}\ (\texttt{data}-\texttt{reference}-\texttt{list})$$

The information provided is used to derive dependencies among tasks at runtime, and schedule/fire a task. Tasks are fired when their inputs are ready and their outputs can be generated.

Dependencies are expressed by means of data-reference-lists. A data-reference in such a list can contain either a single variable identifier, or also references to subobjects. References to subobjects include array element references (e.g., *a[4]*), array sections (*a[3:6]*), field references (*a.b*), and elaborated shaping expressions (*[10][20] p*). The latter means the rectangular area starting at address *p*, with a shape of 10 rows and 20 columns.

OmpSs is based on two main components: i) The Mercurium compiler gets C/C++ and FORTRAN code, annotated with the task directives presented above, and transforms the sequential code into parallel code with calls to the Nanos++ runtime system; and ii) The Nanos++ runtime system gets the information generated by the compiler about the parallel tasks to be run, manages the task dependences and schedules them on the available resources, when those tasks are ready. Nanos++ supports the execution of tasks in remote nodes, and heterogeneous accelerators.

At the lower level, the AXIOM project will investigate and implement the OmpSs programming model on top of the following intra- and inter-node technologies:

- **Intra-node:** The most important target here is FPGA programmability support.
  - OmpSs@FPGA, for easy exploiting of the FPGA acceleration;
- **Inter-node:** In this case two different approaches can be addressed based on the performance requirement, although they can be integrated in the same scenarios, to work with different memory address spaces.
  - OmpSs@cluster, for efficient parallel programming hiding message-passing complexities;
  - OmpSs on a DSM-like paradigm, for easy parallelization of legacy code.

Fig. 2 shows the overall view of OmpSs@FPGA and OmpSs@cluster execution context in a multi-board system. Each FPGA-based node will be addressed by the OmpSs@FPGA support meanwhile the OmpSs@cluster will help to transparently program all the multi-node system.

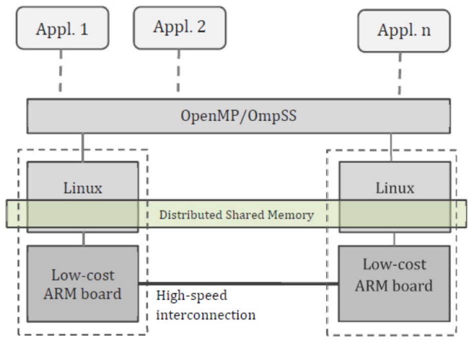


Fig. 3. General view of OmpSs over a DSM system.

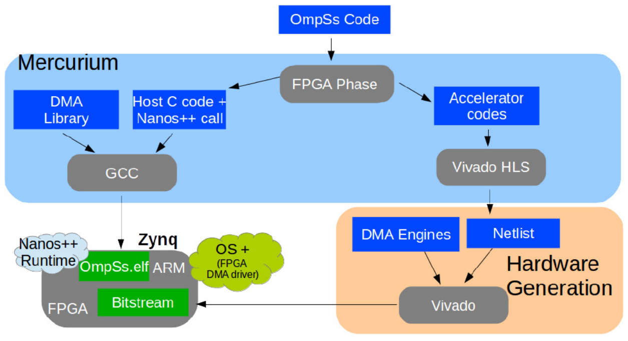


Fig. 4. OmpSs@FPGA ecosystem compilation flow.

Fig. 3 shows the overall view of a DSM system where OmpSs@FPGA would have the same intra-node influence and OmpSs@cluster will appear like a single intra-node OmpSs running over a transparent DSM system.

2.2. OmpSs@FPGA

The OmpSs@FPGA ecosystem consists of the infrastructure for compilation instrumentation and execution from source code written in C/C++ to ARM binary and FPGA bitstream for Zynq. The compilation infrastructure provides support to (1) generate ARM binary code from OmpSs code, that can run in the ARM-based SMP of the Zynq SoC, (2) extract the kernel of the part of the application to be accelerated into the FPGA and (3) automatically generate a bitstream that includes the IP cores of the accelerator(s), the DMA engine IPs, and the necessary interconnection. In addition, the ARM binary can be instrumented to generate traces to be analyzed offline with the Paraver tool [4].

The runtime infrastructure should allow heterogeneous tasking on any combination of SMPs and accelerators, depending on the availability of the resources and the target devices.

Fig. 4 shows the high level compilation flow using our OmpSs@FPGA ecosystem. The OmpSs code is passed through the source-to-source compiler Mercurium [5], that includes a specialized `FPGA` compilation phase to process annotated `FPGA` tasks. For each of those tasks, it generates two C codes. One of them is a `Vivado HLS` (source to HDL Xilinx tool) annotated code for the bitstream generation branch ("accelerator codes" box in Fig. 4). The other is an intermediate host source code with OmpSs runtime (Nanos++) calls that is generated for the software generation branch ("Host C code + Nanos++ runtime call" box in Fig. 4). Both the hardware and the software generation branches are transparent to the programmer.

Fig. 5 shows a matrix multiply example that has been annotated with OmpSs directives. This code shows a parallel tiled matrix multiply where each of the tiles is a task. Each of those tasks

```
#pragma omp target device(fpga, smp) copy_deps
#pragma omp task in(a[0:BS*BS-1], b[0:BS*BS-1]) \
                 inout(c[0:BS*BS-1])
void matrix_multiply(int BS, float a[BS][BS],
                     float b[BS][BS],
                     float c[BS][BS]) {
    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
            float sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
        }
}

int main( int argc, char * argv[] ){
 int BS = ...
 ...
 for (i=0; i < NB; i++)
   for (j=0; j < NB; j++)
     for (k=0; k < NB; k++)
       matrix_multiply(BS, A[i][k], B[k][j], C[i][j]);
 #pragma omp taskwait
 ...
}
```

Fig. 5. OmpSs directives on matrix multiplication.

has two input dependences and an inout dependence that will be managed at runtime by Nanos++. Those tasks will be able to be scheduled/fired to an SMP or FPGA, as it is annontated in the target device directive, depending on the resource availability. The `copy_deps` clause associated to the `target` directive hints the Nanos++ runtime to copy the data associated with the input and output dependences to/from the device when necessary.

2.3. OmpSs@cluster

OmpSs@cluster is the OmpSs flavor that provides support for a single address space over a cluster of SMP nodes with accelerators. In this environment, the Nanos++ runtime system supports a master-worker execution scheme. One of the nodes of the cluster acts as the master node, where the application starts. In the rest of nodes where the application is executed, worker processes just wait for work to be provided by the master.

In this environment, the data copies generated either by the `in`, `out`, `inout` task clauses are executed over the network connection across nodes, to bring data to the appropriated node where the tasks are to be executed.

Following the Nanos++ design, *cluster threads* are the components that allow the execution of tasks on worker nodes. These threads do not execute tasks themselves. They are in charge of sending work descriptors to their associated nodes and notifying when these have completed their execution. One cluster thread can take care of providing work to several worker nodes. In the current implementation, cluster threads are created only on the master node of the execution. Slave nodes cannot issue tasks for remote execution and thus they do not need to spawn cluster threads.

In Nanos++, the device specific code has to provide specific methods to be able to transfer data from the host address space to the device address space, and the other way around. The memory coherence model required by OmpSs is implemented by two generic subsystems, the *data directory* and the *data cache*, explained below.

Fig. 6 shows how the different Nanos++ subsystems are organized to manage the memory of the whole cluster. The master node is the responsible for keeping the memory coherent with the OmpSs memory coherence model, and also for offering the OmpSs single address space view. The master node memory is what OmpSs considers the *host memory* or *host address space*, and

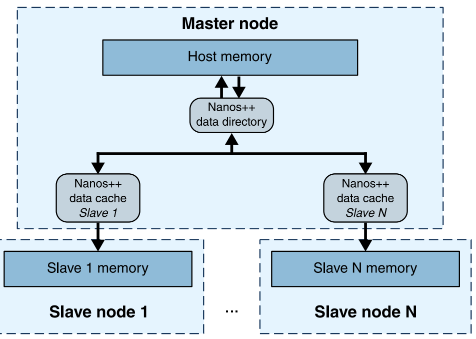


Fig. 6. Nanos++ distributed memory management organization.

it is the only address space exposed to the application. The memory of each worker node is treated as a private device memory and is managed by the master node.

The *data cache* component manages the operations needed at the master node to transfer data to and from worker memories. There is one data cache for each address space present on the system. Operations performed in a data cache include allocating memory chunks, freeing them and transferring data from their managed address spaces to the host address space and the other way around. Data caches also keep the mapping of host memory addresses to their private memory addresses. Memory transfer operations are implemented using network transfers. Allocation and free operations are handled locally at the master node.

A memory reference may have several copies of its contents on different address spaces of the system. To maintain the coherence of the memory, the master node uses the *data directory*. It contains the information of where the last produced values of a memory reference are located. With it, the system can determine which transfer operations must perform to execute a task in any node of the system. Also, each task execution updates the information of the data directory to reflect the newly produced data.

The implementation of the network subsystem is currently based on the active messages provided by the GASNet communications library. In the context of AXIOM, we will adapt the networking on the communications library provided for the Zynq platform.

2.4. OmpSs on DSM-like systems

DSM is a well-known research topic, and it can be implemented either at software or at hardware level (with a full range of hybrid approaches).

We will work on the performance analysis of current DSM implementations. After that the project will decide upon the design and development of a proper, reliable and efficient mechanism to implement a DSM-like paradigm integrated in the Linux OS. The mechanism will run on the reference platform. It will allow to leverage the simplicity and scalability of the OmpSs framework on top of the AXIOM platform. It will be released as Open-Source software, and it is expected to bring benefits to both the ICT and the embedded industries.

2.5. Operating system support

The operating system used in the project will be Linux. One of the advantages of using a SoC like the Zynq is that Linux can be run on the ARM cores of the platform off-the-shelf. This kind of system has the advantage of the easiness to program a standard processor like the ARM along with the raw performance power of the FPGA fabric that will be used through the OmpSs programming model.

We will investigate the possibility of integrating features in the OS to load balance the work across the nodes through the high-speed interconnection. Finding an efficient solution is an aimed outcome of the project since current solutions for load balancing in distributed systems may be expensive, too specific, or difficult to program (with paradigms such as MPI).

Particular attention will be given to scalability and latency issues, by implementing lock-free data structures. Another relevant aspect will be the necessity of properly managing events in real-time.

The OS scheduler will be extended to enable it distributing threads across the different nodes. The low-level thread scheduler (LLTS [6–10], discussed in Section 7) may be accelerated in hardware, by mapping its structure in the FPGA cards composing the evaluation platform. This will avoid bottlenecks from the scheduler, thus increasing the performance of parallel applications.

3. The AXIOM link

The AXIOM platform will be built around FPGA-based SoC, as exemplified by the Zynq platform by Xilinx. Zynq devices feature a dual- or quad-core ARM Cortex A9 processor closely connected to an FPGA fabric. The closeness of the connection (and hence the low latency) and the flexibility of the reconfigurable FPGA logic make the combination very powerful in terms of customization. In addition, Zynq devices feature gigabit-rate transceivers that will be used to provide ample communication bandwidth between AXIOM nodes.

In terms of connectivity, AXIOM -besides including classical connectivity (e.g., Internet)- will also bring modularity at the next level, allowing the construction of more compute intensive and performance systems through low-cost but scalable high-speed interconnect. This interconnect, subject of research and design during the project, will utilize relatively low cost SATA connectors to interconnect multiple boards. Such connectivity will allow to build (or upgrade at a later moment) flexible and low-cost systems with simplicity by re-using the same basic (small) module without the need of costly connectors and cables.

We will provide three bi-directional links per board, so that the nodes can be connected in many different ways, ranging from ring, to the well-established 2D-mesh/torus, and up to arbitrary 3-D topologies such as mesh/torus. The AXIOM interconnect will have customizable parameters (such as packet size, formats, etc) if needed by applications, further improving the efficiency and performance.

In AXIOM we will provide a powerful network interface (NI) -implemented in the FPGA region- that will efficiently support the communication protocols needed by the applications. Besides implementing a MPI-like communication library, we will support a (distributed) Shared Memory model with support from the OmpSs programming model, the Operating System, and the Run-time. One such optimization is the efficient implementation of remote direct memory access (RDMA) and remote-write operations as basic communication primitives visible at the application level.

The AXIOM interconnection library will support two main packet types, (a) RDMA, and (b) short messages. RDMA packets will be used to (a) request large data from a remote node (RDMA requests), and (b) transmit large data (RDMA writes). Short messages will be used to exchange short data between nodes that will contain either raw data or acknowledgement packets (ACKs). Towards a balanced and efficient bandwidth network utilization, we employ a packet priority transmission scheme; ACKs, RDMA writes / messages, and RDMA requests are classified with the highest, middle and lowest transmission priority respectively.

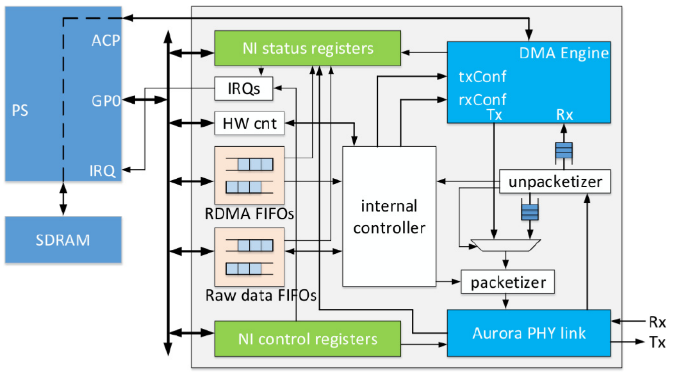


Fig. 7. The Network Interface controller structure.

Fig. 7 illustrates the NI internal structure for inter-node communication. The “RDMA FIFOs” will be used to store descriptors for sending / receiving RDMA packets to / from remote nodes. RDMA descriptors contain the local and remote node id, source and destination data address, and finally the payload size. The “Raw data FIFOs” will be used to store descriptors for exchanging either short data messages or ACKs. Such descriptors will contain the source and destination node id, and also encapsulate the payload data (raw data or an ACK).

The “NI control registers” are memory mapped registers that allow the local Zynq processing system (PS) to configure the NI PHY loopback mode or toggle local notifications when data are successfully transmitted. The “NI Status registers” are also memory mapped registers that allow the PS to monitor certain NI internal states, such as the DMA engine progress, queues status (empty, full, etc.), PHY channel and link states. In addition, when a FIFO state moves from empty to not empty the “IRQ” module raises an interrupt to inform the PS.

The hardware counters module (“HW cnt”) provides a set of counters to monitor the progress of RDMA requests and RDMA writes. Every new RDMA request / RDMA write that reads / writes a large set of data from / to a remote node, is essentially served by multiple short RDMA responses (packets that fetch subsets of the requested data). Moreover, each RDMA request / RDMA write gets a unique id that is assigned to a HW counter. The latter is set to the number of RDMA responses required to transmit all data; its value is decremented each time an RDMA response is finished. The PS can access all HW counters for debugging purposes via software.

The “DMA engine” is responsible for storing incoming payload / loading requested data to / from the required SDRAM address via the PS coherency port (ACP). The “Aurora PHY link” utilizes the Zynq MGT transceivers to serially send / receive data to / from remote nodes.

The “packetizer” assembles a complete packet that will be sent to a remote node; short messages and RDMA requests / RDMA writes are forwarded to the Aurora PHY link, while RDMA response headers are appended with the requested payload provided by the DMA engine. In contrast, the “unpacketizer” caches incoming packets. Simple messages / ACKs and RDMA response headers are eventually stored to the “Raw data FIFOs”. Trailing payload data from RDMA responses are forwarded to the DMA engine and stored to the SDRAM via the ACP, ensuring the PS data coherency.

Finally, the NI “internal controller” (NIC) orchestrates the overall module functionality.

4. Application domains and examples of use

AXIOM will be applied in two real life application domains: Video-surveillance and Smart-home. They will operate as benchmarks for assessing the potentialities and the limits of the proposed architecture. The two application domains have been chosen for the different kind of challenges to process capabilities they pose.

4.1. Video-surveillance

Intelligent multi-camera video surveillance is a multidisciplinary field related to computer vision, pattern recognition, signal processing, communication, embedded computing and image sensors. Smart video-surveillance has a wide variety of applications both in public and private environments, such as homeland security, crime prevention, facial marketing and traffic control, among others.

These applications are generally very computationally demanding, since they require monitoring very diverse indoor and outdoor scenes including airports, hotels or shopping malls, which usually involve highly varying environments. In many cases it is also necessary to analyze multiple camera video streams, particularly when object re-identification or tracking of individuals across cameras is required. For instance, a scenario where runners may be observed and recognized with different objectives: statistics, real-time detection of people that want to be video recorded during the race, TV reportage where the TV officer only has to say the name of a runner and the corresponding camera becomes operative, etc. Another crowded scenario may be the case of a large company with hundreds of employees that work in several different places/buildings: an employee A in any room requests videoconference with a person (in any place, any building) and AXIOM, in real-time, detects where this person is and requests permission to begin videoconference room-to-room by telling that person: “A is requesting a videoconference”. Real-time recognition may also help to track emergency vehicles to skip traffic jams by analyzing the traffic camera images in real-time.

The modular approach explored by AXIOM is particularly well-suited for tackling such challenging scenarios as it addresses the issues derived from their computational complexity, distributed nature, and need for synchronization among processes. Furthermore, the AXIOM platform makes it possible to execute compute intensive tasks on ARM with FPGA processing nodes. This will enable companies such as Herta Security to deploy their real-time face recognition technology in crowded and changeable environments using multiple cameras simultaneously.

4.2. Smart-home

Smart home means buildings empowered by ICT in the context of the merging Ubiquitous Computing and the Internet of Things: the generalization in instrumenting buildings with sensors, actuators, cyber-physical systems allow to collect, filter and produce more and more information locally, to be further consolidated and managed globally according to business functions and services. A smart home is one that uses operational and IT technologies and processes to make it a better performing building - one that delivers lower operating costs, uses less energy, maximizes system and equipment lifetime value, is cyber-secured and produces measurable value for multiple stake holders.

Major challenges in such environments concern cryptography, self-testing and first of all sensor-networks management. Sensor data brings numerous computational challenges in the context of data collection, storage, and mining. In particular, learning from data produced from a sensor network poses several issues: sensors are distributed; they produce a continuous flow of data, eventually at high speeds; they act in dynamic, time-changing environments; the number of sensors can be very large and dynamic. These issues require the design of efficient solutions for processing data produced by sensor-networks.

AXIOM can help with preventive and interactive maintenance of infrastructures, climate and temperature management. This management can be remotely controlled helping to improve the energy efficiency at home, apartments and company office buildings. For instance, AXIOM may detect patterns of behavior in a company office building to adapt climate and light switching to the working way of life of the workers.

The two application domains pose also common challenges such as, board-to-board communication and easy programmability. Furthermore, the two scenarios shown can easily converge, offering opportunities for synergies and emerging services in the respective domains.

4.3. Examples of use

We are currently considering a wide range of potential uses both for Video-surveillance and for Smart-home. They range from dynamic retail demand forecasting in train/bus station to Smart Marketing in shopping malls for Video-surveillance; and from Smart home comfort to Autonomous drone for infrastructure and smart-home control. Here a taste of the scenarios, where the goals are expressed in terms of the final users of the enabling technology is showed. A discussion of another scenario, part of our scenario exploration, related to vehicle detection can also be found in another paper [11]. At the same time, these goals should match with the challenges to AXIOM processing capabilities:

- Dynamic retail demand forecasting. Due to the high fluctuation of passengers departing and arriving at train stations, demand for station retailers varies strongly over time. By forecasting such demand through video analysis, better services can be provided through more efficient staff utilization. The purpose of this scenario is to provide retailers with a real-time forecast of potential customers arriving at their outlets, to allow for better task allocation and to increase business efficiency.
- Smart marketing in shopping mall. Consumer behavior in a shopping mall can be very eclectic yet the awareness of patterns of behavior can be of help both to services providers and to clients to meet their respective goals. Demographic analyses is carried out over the captured facial snapshots, helping to identify interesting facts such as the demographic profiles of the customers, or how do they distribute into gender and age segments. The visitors are tracked from one camera to another, so as to discover the main paths they take through the mall and how long they stay at different locations. The goal is to collect statistical information about the visitors in order to define marketing strategies both for service providers and for clients.
- Smart home comfort. Comfort perception and necessities can be different in respect of time of the day/week and to the characteristics of the people actually living that space in that moment. The smart home is required to identify and manage the different situations, and to react at the people indications in an easy and smooth way. Networked sensors and actuators are distributed in each room embedded in ordinary appliances. The appliances perform their primary normal function, but also collect different kinds of information, ranging from presence detection, temperature, humidity, window and door opening, air quality, audio. The objective of the smart home comfort autopilot is to minimize power consumption and to guarantee people’s comfort and well being, without giving the impression of reducing people freedom and capacity of control.
- Autonomous rover/drone for infrastructure control. Preventive maintenance is performed on equipment to keep it running smoothly and efficiently and to help extend its life. Many types

of equipment should be put on a preventive maintenance program: HVAC systems, pumps and air compressors, air conditioning, chillers and absorption equipment, elevators, safety showers, back-flow preventers, building exteriors, roofs, windows, fire doors and generators. Autonomous rovers and drone furnished with thermo camera and ambient sensors can move inside and outside a building monitoring the energy flow, providing data for a multi-level energy flow models that can be used for preventive maintenance. The goal is maintaining building infrastructure efficient, manage operating costs, and minimizing potential downtime. It also ensures these components perform within their originally designed operating parameters, allowing data center managers the opportunity to replace components before they fail.

The software approach explored by AXIOM is particularly well-suited for tackling such challenging scenarios, as it addresses the issues derived from their computational complexity, distributed nature, and need for synchronization among processes. Moreover, we are considering some representative benchmarks to test drive the design of the software stack that two partners already explored in the ERA project [12,13].

Finally, it is worth mentioning that this project doesn't address the problem of maintaining or securing "sensible" data. In principle AXIOM is not collecting sensitive information, as per the definition of sensitive information provided by EU Directive 95/46/EC. However, according to the approved Commission Proposals on the data protection reform, biometric data has to be considered sensitive by default. This Regulation shall apply from 25 May 2018 but the project considers since the beginning that biometric data collected deserve that highest protection, at the same level of data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs. Accordingly, procedures compliant with national and EU legislation are followed to deal with data collection, storage, protection, retention and destruction and confirmation.

Regarding the software developed in the presented scenarios, it will rely on the Linux OS security layers already developed. As a full Linux compliant architecture, the AXIOM architecture supports the technical means to guarantee different privacy levels to protect the access to "sensible" plain data. Of course, it will also be archived and distributed following national and EU legislation.

5. Experimental setup

In the first year of the AXIOM project we want to properly evaluate the potential of the proposed hardware/software platform to achieve the following goals:

- Easy programmability of multi-core, multi-board, FPGA nodes using the OmpSs programming model.
- Reasonable performance and improved energy efficiency compared against state-of-the-art systems.

5.1. Benchmarks description

Three benchmarks have been used for the analysis of easy programmability when using the OmpSs@FPGA infrastructure: (1) Cholesky matrix decomposition, working on a dense matrix of 64x64 double-precision complex numbers; (2) Covariance, working on arrays of 32-bit integer complex numbers; and (3) Matrix multiplication, working on a matrix of single precision floating point values (32×32 and 64×64 sizes). On the order hand, for performance results the same matrix multiplication has been used with a larger matrix 2048×2048, and different block sizes.

Table 1
Total number of *additional* lines of code compared to a baseline C implementation.

Application	*Pthread*	*Accel*	*OmpSs*
`Cholesky`	26	71	3
`Covariance`	29	94	3
`MxM 64x64`	39	95	3
`MxM 32x32`	39	95	3

5.2. Hardware and software

To perform the FPGA experiments showed in this article we have used a Zynq 706 board. The board includes a Zynq 7045 with 2 ARM cores running at 800MHz and an FPGA that run at 200MHz and features 350K logic cells, 19.1Mb of block RAM and 900 DSP slices. The SoC was released at 2012 and used 28nm technology. Timing of the applications has been obtained by instrumenting with `gettimeofday` the part of the code that calls several times the kernel code while the power consumption was computed using the tools provided by Xilinx.

The OmpSs implementation is based on Mercurium 1.99.4 and Nanos++ 0.8. For the hardware compilation branch we have used the Xilinx ISE Design 14.7 and the Vivado HLS 2013.2 tools. The `#pragma HLS pipeline II=1` was used to parallelize the second loop of the matrix multiplication. All OmpSs codes have been compiled with the arm-xilinx-linux-gnueabi-g++ (Sourcery CodeBench Lite 2011.09–50) 4.6.1 and arm-xilinx-linux-gnueabi-gcc (Sourcery CodeBench Lite 2011.09–50) 4.6.1 compilers, with `-O3` optimization flag. OmpSs runtime used an AXIOM premilinary prototype of the NI interface. Results show the average elapsed execution time of 3 application executions.

The machine used to obtain the GPP reference results was an i5-3470 with 4 cores running at 3.20GHz. The processor was selected as it was released in Q2'12, close to the releasing time of the Zynq 7045, and uses a 22nm technology. As with the ARM codes, timing was measured with `gettimeofday` and power was measured reading directly the processor hardware registers. Codes were compiled with gcc version 5.2.1 using `-O3` optimization flag and MKL version 11.2.3.

6. Results

We have done some experiments for coding a set of benchmarks in the Zynq platform and an initial evaluation of programmability cost in terms of number of lines of code, as a measure of programmability complexity.

6.1. Programmability analysis

In order to have a good programmability analysis we have implemented four different versions of each benchmark code: sequential code, pthread code, FPGA-accelerated code and OmpSs code. All versions of the codes consider the full Matrix Multiply, the full Cholesky, and the full Covariance as tasks.

We want to remark the programmability facilities of our proposal. With this objective, Table 1 shows the total number of *additional* lines of code for each of the different versions of the applications, compared to the sequential version: a pthread version *only* running tasks in one or two ARM cores (`Pthread`), a sequential version using one or two hardware accelerators (`Accel`), and the OmpSs version (`OmpSs`).

The Pthread and Accel versions require more *additional* lines than the OmpSs version. This is especially high in the sequential versions using the hardware accelerators. For the Pthread version this is due to the additional calls to the Pthreads library, in order

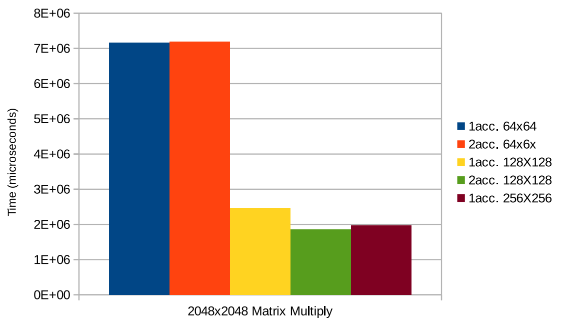


Fig. 8. Elapsed-time: 1/2 FPGA accelerators, up to 256 × 256.

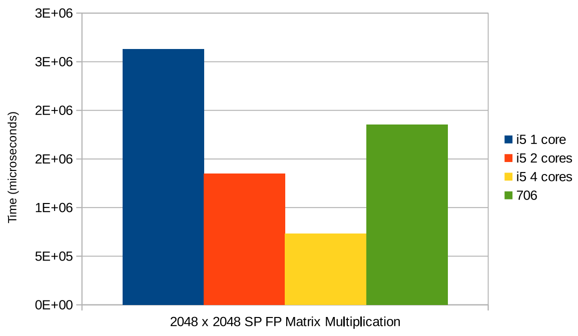


Fig. 9. Elapsed-time: FPGA MxM versus SMP MxM (MKL).

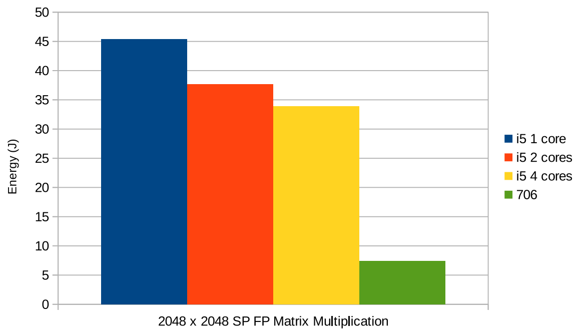


Fig. 10. Energy consumption: FPGA MxM versus SMP MxM (MKL).

to create, manage and join the pthreads. For the Accel version, this is because the application needs to call the low-level infrastructure to setup the communications layer with the FPGA and perform the actual data transfers back and forth to the FPGA hardware.

On the other hand, in the case of the OmpSs version, the thread management, the setup of the communications and data transfers to and from the FPGA are all done internally by the Nanos++ runtime. This way, the programmer does not need to write any line of code related to low level management, but only the directives triggering the communications.

Indeed, the current compilation and runtime infrastructure of the OmpSs programming model allows to exploit the heterogeneous characteristics of the Zynq All-Programmable SoC with the only effort of two directive lines. Note however that Table 1 indicates that the OmpSs version needs an additional third line. This line is a `taskwait` before the program ends, as it can be observed in Fig. 5. Actually, the code showed in Fig. 5 is used to generate both the 32 × 32 and the 64 × 64 versions of the matrix multiplication, using all the available resources (ARM cores and FPGA), simply by redefining the BS variable as 32 or 64 elements.

For the Pthreads and Accel versions however, different block sizes need new scheduling schemes, adding more complexity to the transformation of the code. Indeed, implementing a fourth version of the code managing heterogeneous executions would require more development time and additional lines that the ones showed in Table 1.

6.2. Performance results

In order to study the suitability of our approach to the High Performance Computing (HPC) environment, it is necessary to demonstrate that our systems is not only able to be easily programmed but also that it can achieve a reasonable performance when compared to other current state-of-the-art approaches.

First, an evaluation of the best accelerator size for the selected FPGA was performed. Fig. 8 shows the elapsed time for a 2048 × 2048 matrix multiplication using 1/2 accelerators of sizes (blocks) 64 × 64, 128 × 128 and 256 × 256. Results show that using 1/2 64 × 64 accelerators are the worst choice. This accelerator size is too small for the problem since the data transfer to/from the FPGA overcomes the computational benefits of using the FPGA. Indeed, as the communication channel is shared, using two accelerators does not improve the performance that is bounded by the DMA. On the other hand, there is a significant improvement when moving to 128 × 128 accelerators. Those bigger accelerators compute blocks of four times the size of 64 × 64 accelerators and consequently the data movements are divided by four. Therefore, 128 × 128 accelerators are also 8 times more time consuming than 64 × 64 accelerators since doubling the block size means eight times more multiplications, and then, using two accelerators can help to improve the performance. However, due to FPGA limited resources, the compiler is not able to make the two accelerators, sharing resources, as fast as only one, using all the resources. This limit explains why the largest accelerator (blocks of 256 × 256) is not able to be as fast as two 128 × 128 ones. Although the data transfers are again divided by four the accelerator is six times slower than one 128 × 128 due to the limited resources and this results in a worse overall performance. One not so obious, but nevertheless important result of Fig. 8 is that all the accelerators were compiled and executed using the same source code (listed in Fig. 5) changing only the block size (BS). Both the compiler and the runtime take care of all the details.

Fig. 9 shows the time in microseconds that takes to compute a 2048 × 2048 matrix multiplication, using the best block size, in two different systems. Columns named i5 show the time used by the Corei5 machine described in Section 5.2 when using 1, 2 and 4 cores respectively with the `sgemm` function of the MKL library. Column 706 shows the time used by the same computation when it is performed in a Zynq 706 board using the code showed in Fig. 5 and the OmpSs compilation and execution framework. As it can be seen the FPGA board is competitive with the conventional SMP with a result between 1 and 2 Corei5 cores in performance terms. Fig. 10 shows the energy consumption of the same computation in the same machines. As it shows the FPGA system clearly outperforms the conventional SMPs in terms of energy consumption which shows that our approach is promising for future computing systems.

What is more important from the results showed in Figs. 9 and 10 is not that an FPGA of an older technology process can clearly outperform in terms of energy a conventional processor but the fact that writing the code for the FPGA was actually simpler than writing the code for the SMP. Indeed, as mentioned above the FPGA code was directly the one showed in Fig. 5 while the SMP code was changed to call the parallel `sgemm` version of the MKL library instead of the original matrix multiply function. So, OmpSs was not used for the Core i5 version. Arguably, the change was not cumbersome neither extensive but the fact is that the naive original version of the MxM code, although compiled with the `-O3` optimization flag, performed much (36 × slowdown) worse than the MKL `sgemm` implementation forcing us to change the code to provide a fair evaluation. In our opinion this highlights the potential of using the OmpSs programming model for heterogeneous systems.

7. Related work

The AXIOM project will exploit the OmpSs dataflow features in the AXIOM heterogeneous architecture. OmpSs is the result of the integration of StarSs [14] and OpenMP.

In this section we discuss some work that has been fundamental for the development of this project and provided the necessary inspiration and vision to develop some basic concepts related to the dataflow execution model. Dataflow execution model had been studied since long time ago [15] as they provide a simple an elegant way to efficiently move data from one computational thread to another one [16,17]. In the context of the TERAFLUX project [9,18] such dataflow model had been extended to multiple nodes executing seamlessly thanks to the support of an appropriate memory model [7,10]. In such memory model a combination of consumer-producer patterns [8,19] and transactional memory [20,21] permits a novel combination of dataflow concepts and transactions in order to address the consistency across nodes, where each node is assumed to be cache-coherent, i.e., like in a classical multi-core. Dataflow models also allow the system to take care in a distributed way of faults that may compromise a node [22,23].

In order to integrate heterogeneous execution of the same applications over processors and FPGA fabric, OmpSs@FPGA is a key point in the project. Although to the best of our knowledge OmpSs@FPGA [24, 25] is the first successful attempt to implement hardware accelerators from high-level directives in a total transparent way, other approaches have been used in the past. Some tools try to reduce the FPGA programmability problem by offering the possibility of generating HDL code from C or C-like languages like ROCCC [26,27] or generating systems with an embedded soft processor connected to the generated hardware accelerators like LegUp [28] and C2H tool [29]. However, with the new SMP/FPGA SoCs, new strategies are required in order to exploit those current heterogeneous *and* parallel platforms. Our ecosystem also covers runtime support for parallel execution of heterogeneous tasks on those SoCs, unlike other.

PGI [30] and HMPP [31] programming models are two other approaches quite related to OmpSs. PGI uses compiler technology to offload the execution of loops to the accelerators. HMPP also annotates functions as tasks to be offloaded to the accelerators. We think that OmpSs has higher potential in that it shifts part of the intelligence that HMPP and PGI delegate in the compiler to the OmpSs runtime system. Although these alternatives do support a fair amount of asynchronous computations expressed as futures or continuations, the level of lookahead they support is limited in practice.

To execute over several nodes, OmpSs@cluster [32] is one of the alternatives explored in the project. As alternatives, Partitioned Global Address Space (PGAS) programming models expose an abstracted shared address space to the programmer simplifying its task, while data and thread locality awareness is kept to enhance performance. Representative PGAS languages are UPC [33], and X10 [34]; and Chapel [35], which implement Asynchronous PGAS model, offering asynchronous parallelism. An alternative way to provide asynchronous parallelism on clusters is a hybrid programming model that composes SMPSs [36], that inspired OmpSs, with MPI. The main idea is to encapsulate the communications in tasks so they are executed when the data is ready. This technique achieves an asynchronous dataflow execution of both communication and computation.

OpenCL [37] attempts to unify the programming models for general-purpose multi-core architectures and the different types of hardware accelerators (Cell B.E., GPUs, FPGAs, DSPs, etc.). The participation of silicon vendors (e.g., Intel, IBM, NVIDIA, and AMD) in the definition of this open standard ensures portability, low-level access to the hardware, and supposedly high performance. We believe, however, that OpenCL still exposes much of the low-level details (i.e. explicit platform and context management, kernel special intrinsic functions, explicit program, kernel and data transfer management, etc.), making it cumbersome to use by non-experts.

Another alternative for mutli-node programming is DSM. DSM is a recently revived topic [38]. Some attempts for creating Software DSM implementations for Linux have been carried out during the last decades. Examples are Treadmarks (TMK), JIAJIA [39], Omni/SCASH [40,41], Jump [42,43], Parade [44,45], NanosDSM [46]. Some of these projects only supported very specific hardware, and none of them has been maintained during the last decade.

Regarding applications, state-of-the-art implementations of video-surveillance or voice-identification scenarios currently rely on machine learning techniques based on deep neural networks (DNNs). As recent studies have pointed out, DNNs are particularly good for addressing computer vision image classification and recognition problems exhibiting highly non-linear properties. Applications ranging from face recognition [47] and age/gender estimation [48] to pedestrian detection [49] have experienced dramatic improvements in terms of accuracy just by training DNNs with huge amounts of data. Due to the architectural properties of these models and the advances in HPC, it is now cost-effective to massively scale the infrastructure to train such networks with millions of sample images that have been previously manually tagged by humans on the widely-available social networking services.

The proliferation of frameworks and libraries such as Caffe [50] and cuDNN [51] have democratized the usage of parallelized DNN-based solutions on GPU architectures. However, there is a lack of ready-to-deploy implementations of DNN inference engines for embedded platforms powered by FPGA accelerators. Since DNN evaluation is highly parallel in nature, it is feasible to offload all the required SGEMM matrix multiply operations to FPGAs, and also to execute forward propagation in an efficient manner through the OmpSs programming model.

Once DNN models are trained as a result of a process that usually takes several days on a GPU cluster, it is then possible to evaluate them on the AXIOM board. With this idea in mind, we aim to produce a generic easy-to-use, low-power hardware/software stack to cheaply deploy machine learning solutions based on DNNs interacting with the Cyber-Physical world. This ecosystem powered by the AXIOM platform is expected to solve a myriad of computer vision problems, and thus dramatically improve productivity on a wide range of industries.

8. Conclusions and future work

In this paper, we have presented the software layers that we are developing on the AXIOM H2020 European Project. The main objective of the project is to bring to reality a novel small board which aims at becoming a very powerful basic brick of future interconnected and scalable embedded Cyber-physical systems, and specifically we focus on the application domains of Video-surveillance, deep learning and Smart-home. The module consists of both hardware and software that will be designed and demonstrated in the project.

On one hand, the target board architecture will be a board based on a SoC with several ARM cores and an FPGA, like the Xilinx Zynq, and with the Arduino interface to be extensible. The AX-

IOM system will comprise several of such boards linked through custom communication links, and providing application memory coherence at software level. On the other hand, we will research ways to easy programmability of the system, based on the OmpSs programming model and DSM-like techniques to achieve a global system image for applications. Currently, we are in the process of designing a high-speed communications layer between boards. These communication will be implemented using the transceivers available in the Zynq SoC. We have also started looking at the application requirements to ensure that our platform fits with their needs.

The expected impacts obtained from the AXIOM project include a platform interfacing with the physical world through compatibility with Arduino shields. This platform will be aimed to become the hardware and software platform for large scale production. In this sense we want to develop an autonomous technology that is able to break the Embedded Systems energy efficiency and programmability barriers. The same set of technologies are expected to represent the base for future European industrial exploitation in the HPC and Embedded Computing markets. Finally, it is expected to provide the basis for a new European-level research at the forefront of the development of extreme-performance system software and tools.

Our preliminary experiments have shown that the OmpSs programming model increases the expressiveness of serial or pthreads programming, thus allowing developers to focus on solving the issues related to the algorithms, instead of dealing with the low-level details of the communications among boards or data transfers between the cores and the embedded FPGA. Also we show that this easiness in programmability is joined by competitive performance and lower energy consumption when compared to standard processors.

The key features of the project presented in this paper are the possibility to modularly enhance the capabilities of the board, improve its interface with the physical world, flexibly reconfiguring it for accelerating specific functions, while providing energy efficiency and easy programmability.

Acknowledgment

We thankfully acknowledge the support of the European Union H2020 program through the AXIOM project (grant ICT-01-2014 GA 645496), the Spanish Government, through the Severo Ochoa program (grant SEV2015-0493) the Spanish Ministry of Science and Technology (TIN2015-65316-P) and the Generalitat de Catalunya (MPEXPAR, 2014-SGR-1051 and 2014-SGR-1272).

References

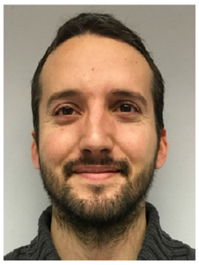
[1] A. Goransson, D.C. Ruiz, Professional Android Open Accessory Programming with Arduino, John Willey & Sons, 2013.
[2] S. Monk, Programming Arduino Next Steps: Going Further with Sketches, 1st, McGraw-Hill Professional, USA, 2013.
[3] E. Ayguadé, R.M. Badía, D. Cabrera, A. Durán, M. González, F. Igual, D. Jiménez, J. Labarta, X. Martorell, R. Mayo, J.M. Pérez, E.S. Quintana-Orti, A proposal to extend the openMP tasking model for heterogeneous architectures, in: IWOMP, 5568, 2009, pp. 154–167.
[4] V. Pillet, J. Labarta, T. Cortes, S. Girona, PARAVER: a Tool to Visualize and Analyze Parallel Code Technical Report UPC-CEPBA-95-03, European Center for Parallelism of Barcelona (CEPBA), Universitat Politècnica de Catalunya (UPC), 1995.
[5] R. Ferrer, S. Royuela, D. Caballero, A. Durán, X. Martorell, E. Ayguadé, Mercurium: design decisions for a s2s compiler, Cetus Users and Compiler Infastructure Workshop in conjunction with PACT 2011, 2011.
[6] R. Giorgi, A. Scionti, A scalable thread scheduling co-processor based on dataflow principles, ELSEVIER Future Gener. Comput. Syst. (0) (2015) 1–10, doi:10.1016/j.future.2014.12.014.
[7] R. Giorgi, iTERAFLUX: exploiting dataflow parallelism in teradevices, in: ACM Computing Frontiers, 2012, pp. 303–304, doi:10.1145/2212908.2212959.
[8] N. Ho, A. Mondelli, A. Scionti, M. Solinas, A. Portero, R. Giorgi, Enhancing an x86_64 multi-core architecture with data-flow execution support, in: ACM Proc. of Computing Frontiers, 2015, pp. 1–2, doi:10.1145/2742854.2742896.
[9] R. Giorgi, et al., TERAFLUX: harnessing dataflow in next generation teradevices, Microprocess. Microsyst. 38 (8, Part B) (2014) 976–990.
[10] R. Giorgi, P. Faraboschi, An introduction to df-threads and their execution model, in: IEEE MPP, 2014, pp. 60–65, doi:10.1109/SBAC-PADW.2014.30.
[11] G. Burresi, R. Giorgi, A field experience for a vehicle recognition system using magnetic sensors, in: IEEE MECO, 2015, pp. 1–6.
[12] N. Puzovic, S. McKee, R. Eres, A. Zaks, P. Gai, W. S., R. Giorgi, A multi-pronged approach to benchmark characterization, in: IEEE CLUSTER, 2010, pp. 1–4, doi:10.1109/CLUSTERWKSP.2010.5613090.
[13] A. Scionti, S. Kavvadias, R. Giorgi, Dynamic power reduction in self-adaptive embedded systems through benchmark analysis, in: IEEE MECO, 2014, pp. 62–65.
[14] J. Planas, R. Badía, E. Ayguadé, J. Labarta, Hierarchical task-based programming with StarSs, Int. J. High Perform. Comput. Appl. 23 (3) (2009) 284–299.
[15] F. Yazdanpanah, C. Álvarez-Martínez, D. Jiménez-González, Y. Etsion, Hybrid dataflow/von-neumann architectures, IEEE Trans. Parallel Distrib. Syst. 25 (6) (2014) 1489–1509, doi:10.1109/TPDS.2013.125.
[16] L. Verdoscia, R. Vaccaro, R. Giorgi, A clockless computing system based on the static dataflow paradigm, in: Proc. IEEE Int.l Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM-2014), 2014, pp. 30–37, doi:10.1109/DFM.2014.10.
[17] L. Verdoscia, R. Vaccaro, R. Giorgi, A matrix multiplier case study for an evaluation of a configurable dataflow-machine, in: ACM CF'15 - LP-EMS, 2015, pp. 1–6, doi:10.1145/2742854.2747287.
[18] M. Solinas, et al., The TERAFLUX project: Exploiting the dataflow paradigm in next generation teradevices, in: DSD, 2013, pp. 272–279.
[19] N. Ho, A. Portero, M. Solinas, A. Scionti, A. Mondelli, P. Faraboschi, R. Giorgi, Simulating a multi-core x86_64 architecture with hardware isa extension supporting a data-flow execution model, in: IEEE Proceedings of the AIMS-2014, Madrid, Spain, 2014, pp. 264–269, doi:10.1109/AIMS.2014.41.
[20] R. Giorgi, Accelerating haskell on a dataflow architecture: a case study including transactional memory, in: CEA, 2015a, pp. 91–100.
[21] R. Giorgi, Transactional memory on a dataflow architecture for accelerating haskell, WSEAS Trans. Comput. 14 (2015b) 794–805.
[22] S. Weis, A. Garbade, J. Wolf, B. Fechner, A. Mendelson, R. Giorgi, T. Ungerer, A fault detection and recovery architecture for a teradevice dataflow system, in: IEEE DFM), 2011, pp. 38–44.
[23] S. Weis, et al., Architectural support for fault tolerance in a teradevice dataflow system, Springer Int'l J. Parallel Program. (2014) 1–25.
[24] A. Filgueras, E. Gil, D. Jiménez-González, C. Álvarez, X. Martorell, J. Langer, J. Noguera, K. Vissers, Ompss@zynq all-programmable soc ecosystem, in: Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays, in: FPGA '14, ACM, New York, NY, USA, 2014, pp. 137–146, doi:10.1145/2554688.2554777.
[25] A. Filgueras, E. Gil, C. Álvarez, D. Jiménez-González, X. Martorell, J. Langer, J. Noguera, Heterogeneous tasking on smp/fpga socs: The case of ompss and the zynq, in: 2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC), 2013, pp. 290–291, doi:10.1109/VLSI-SoC.2013.6673293.
[26] J.R. Villarreal, A. Park, W.A. Najjar, R. Halstead, Designing modular hardware accelerators in c with roccc 2.0., in: R. Sass, R. Tessier (Eds.), FCCM, IEEE Computer Society, 2010, pp. 127–134.
[27] W.A. Najjar, J.R. Villarreal, Fpga code accelerators - the compiler perspective, in: DAC, 2013, p. 141.
[28] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J.H. Anderson, S. Brown, T. Czajkowski, Legup: High-level synthesis for fpga-based processor/accelerator systems, in: Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, in: FPGA '11, ACM, New York, NY, USA, 2011, pp. 33–36, doi:10.1145/1950413.1950423.
[29] Altera, Corp., Nios II C2H Compiler User Guide, 2009. URL: www.altera.com
[30] PGI Accelerator Programming Model for Fortran & C, The Portland Group, 2010.
[31] R. Dolbeau, S. Bihan, F. Bodin, HMPP: a hybrid multi-core parallel programming environment, First Workshop on General Purpose Processing on Graphics Processing Units, 2007.
[32] J. Bueno, L. Martinell, A. Durán, M. Farreras, X. Martorell, R.M. Badía, E. Ayguadé, J. Labarta, Productive cluster programming with ompss, in: Proceedings of the 17th International Conference on Parallel Processing - Volume Part I, Euro-Par'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 555–566.
[33] UPC Consortium, UPC Language Specifications v1.2, Report Number: LBNL-59208, 2005.
[34] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, V. Sarkar, X10: an object-oriented approach to non-uniform cluster computing, SIGPLAN Not. 40 (10) (2005) 519–538, doi:10.1145/1103845.1094852.
[35] B. Chamberlain, D. Callahan, H. Zima, Parallel programmability and the chapel language, Int. J. High Perform. Comput. Appl. 21 (3) (2007) 291–312, doi:10.1177/1094342007078442.
[36] V. Marjanovic, J. Labarta, E. Ayguadé, M. Valero, Effective communication and computation overlap with hybrid mpi/smpss, SIGPLAN Not. 45 (5) (2010) 337–338, doi:10.1145/1837853.1693502.
[37] Khronos OpenCL Working Group, The OpenCL Specification, version 1.2, 2011. URL https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf
[38] S. Kaxiras, D. Klaftenegger, M. Norgren, A. Ros, K.F. Sagonas, Turning centralized coherence and distributed critical-section execution on their head: A new approach for scalable distributed shared memory, in: Proc. of HPDC, 2015, pp. 3–14.
[39] Jiajia, http://www-users.cs.umn.edu/~tiane/paper/dist.htm.

[40] Omni/scash http://www.pcs.cs.tsukuba.ac.jp/omni-compiler/doc/omniscash.html.
[41] M. Hess, G. Jost, M. Müller, R. Rühle, Experiences using OpenMP based on Compiler Directed Software DSM on a PC Cluster, Workshop on OpenMP Applications and Tools (WOMPAT'02, 2002.
[42] The jump software dsm system, http://www.snrg.cs.ku.hk/srg/html/jump.htm.
[43] C.L.W.B. Cheung, K. Hwang, Jump-dp: A software dsm system with low-latency communication support, PDPTA, 2000.
[44] Parade, http://peace.snu.ac.kr/researc/parade/.
[45] Y. Kee, J. Kim, S. Ha, ParADE: an OpenMP Programming Environment for SMP Cluster Systems, Supercomputing 2003 (SC'03), 2003.
[46] J.J. Costa, T. Cortes, X. Martorell, E. Ayguadé, J. Labarta, Paper running openmp applications efficiently on an everything-shared sdsm, (JPDC) 6 (5) (2006) 647—658.
[47] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: Closing the gap to human-level performance in face verification, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2014, pp. 1701–1708.
[48] I. Huerta, C. Fernández, C. Segura, J. Hernando, A. Prati, A deep analysis on age estimation, Pattern Recognit. Lett. 68 (2015) 239–249.
[49] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, D. Ferguson, Real-time pedestrian detection with deep network cascades, in: Proceedings of BMVC 2015, 2015.
[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the ACM International Conference on Multimedia, ACM, 2014, pp. 675–678.
[51] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, cuDNN: efficient primitives for deep learning, arXiv preprint arXiv:1410.0759, 2014. URL http://arxiv.org/abs/1410.0759

Carlos Álvarez received the M.S. and Ph.D. degrees in Computer Science from the Technical University of Catalunya (UPC) in 1998 and 2007, respectively. He currently holds a position as Tenured Assistant Professor in the Computer Architecture Department at UPC, BarcelonaTech, and is a associated researcher at the Computer Sciences -Programming Models Department at BSC-CNS. His research interests cover the areas of parallel architectures, runtime systems and reconfigurable solutions for high-performance multiprocessor systems. He has co-authored more than 40 publications in international journals and conferences. He is currently advising 1 PhD student and has co-advised 2 PhD theses. He has been participating in the HiPEAC Network of Excellence and in the TERAFLUX and AXIOM European projects.

Eduard Ayguadé received the Engineering degree in Telecommunications in 1986 and the Ph.D. degree in Computer Science in 1989, both from the Universitat Politècnica de Catalunya (UPC), Spain. Since 1987 Prof. Ayguad has been lecturing at the Computer Science School (FIB) and Telecommunications Engineering (ETSETB) both in Barcelona. Currently, and since 1997, he is full professor of the Computer Architecture Department at UPC. Prof. Ayguad has lectured a number of (undergraduate and graduate) courses related with computer organization and architecture, parallel programming models and their implementation. Prof. Ayguad is also involved in the Computer Architecture and Technology PhD Program at UPC, where he has (co-)advised more than 20 PhD thesis, in topics related with his research interests: multicore architectures, parallel programming models and their architectural support and compilers for HPC architectures. In these research topics, Prof. Ayguad has published more than 300 papers and participated in several research projects in the framework of the European Union and research collaborations with companies related with HPC technologies (IBM, Intel, Nvidia, Microsoft and Samsung). Currently Prof. Ayguad is associated director for research on the Computer Sciences Department at the Barcelona Supercomputing Center (BSC-CNS), the National Center for Supercomputing in Spain located in Barcelona.

Jaume Bosch completed engineers degree in computer science at the Barcelona School of Informatics (FIB) of Universitat Politècnica de Catalunya - BarcelonaTech (UPC) in 2015 and he is studding a Master in High Performance Computing from the same School. Currently, he is working at the Programming Models Group of Barcelona Supercomputing Center - Centro Nacional de Supercomputacin (BSC-CNS).



Javier Bueno Hedo holds a PhD. degree in Computer Science from the Technical University of Catalonia (UPC). He became involved in research in 2004, when he started as a part-time student in the European Center of Parallelism of Barcelona (CEPBA) working with Software Distributed Memory Systems. In 2006 he became a full-time junior researcher at the Barcelona Supercomputing Center (BSC) and continued his work on distributed systems. From 2010 to 2015 he worked on his thesis, which provided the OmpSs programming model with support for clusters of multi-cores and clusters of GPUs. This work has also been applied to different research projects such as the Mont-Blanc2 project. His current research aims to produce new programming models and tools to ease the complexity of developing applications for modern HPC systems.

Artem Cherkashin completed engineers degree in computer science at the Barcelona School of Informatics (FIB) of Universitat Politècnica de Catalunya - BarcelonaTech (UPC) in 2015. Currently, he is studying a Master degree in High Performance Computing at the same school, while is working at the Programming Models Group of Barcelona Supercomputing Center - Centro Nacional de Supercomputacin (BSC-CNS).

Antonio Filgueras received a degree in computer science at Universitat Politècnica de Catalunya - BarcelonaTech (UPC) in 2012. Currently working at the Programming Models group of Barcelona Supercomputing Center and particimating in the AXIOM European project. His research interests are focused on heterogeneous and reconfigurable solutions for high performance computing and programmability of those.

Daniel Jiménez-González received the M.S. and Ph.D. degrees in Computer Science from the Technical University of Catalunya (UPC) in 1997 and 2004, respectively. He currently holds a position as Tenured Assistant Professor in the Computer Architecture Department at UPC, BarcelonaTech, and is a associated researcher at the Computer Sciences-Programming Models Department at BSC-CNS. His research interests cover the areas of parallel architectures, runtime systems, compilers and reconfigurable solutions for high-performance multiprocessor systems. Dr. Jimenez-Gonzalez has coauthored more than 40 publications in international journals and conferences. He is currently co-advising 1 PhD students and has co-advised 2 PhD student. He has been participating in the HiPEAC Network of Excellence and in the SARC, ACOTES, TERAFLUX, AXIOM and PRACE European projects.

Xavier Martorell received the M.S. and Ph.D. degrees in Computer Science from the Technical University of Catalunya (UPC) in 1991 and 1999, respectively. Since 1992 he has lectured on operating systems, parallel runtime systems and OS administration. He has been an associate professor in the Computer Architecture Department at UPC since 2001. His research interests cover the areas of operating systems, runtime systems, compilers and applications for high-performance multiprocessor systems. Dr. Martorell has participated in several long-term research projects with other universities and industries, primarily in the framework of the European Union ESPRIT, IST and FET programs. He spent one year working with the BG/L team in the IBM Watson Research Center. He has coauthored more than 60 publications in international journals and conferences. He has co-advised three Ph.D. theses and he is currently advising 3 PhD students. He is currently the Manager of the Parallel Programming Models team at the Barcelona Supercomputing Center. He has been participating in the HiPEAC Network of Excellence and in the SARC, ACOTES, and Intone, POP, ENCORE, MontBlanc (I and II), DEEP/DEEP-ER and the AXIOM European projects.

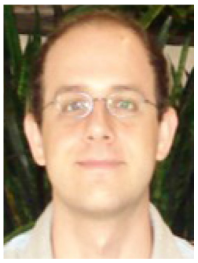
Nacho Navarro (1958–2016, in memoriam) is Associate Professor at the Universitat Politecnica de Catalunya (UPC), Barcelona, Spain, and Senior Researcher at the Barcelona Supercomputing Center (BSC), serving as manager of the Accelerators for High Performance Computing group. He holds a Ph.D. degree in Computer Science from UPC. His current interests include: GPGPU computing, multi-core computer architectures, hardware accelerators, dynamic reconfigurable logic support, memory management and runtime optimizations. He is also doing research on massively parallel accelerators like GPUs in collaboration with the University of Illinois (IMPACT Research Group). Prof. Navarro is a member of IEEE, the IEEE Computer Society, the ACM and the HiPEAC NoE.

Miquel Vidal received the B.S. degree in Computer Science in 2015 from the Technical University of Catalunya (UPC). Currently he is studying a M.S. in High-Performance Computing while working at the Programming Models group at Barcelona Supercomputing Center (BSC-CNS) within the AXIOM European project. His research interests are focused on parallel architectures, multiprocessor systems, and heterogeneous and reconfigurable solutions for high-performance computing; as well as their use on bioinformatics applications.

Dimitris Theodoropoulos obtained his Diploma (5-year degree) and M.Sc degree respectively from the Electronic and Computer Engineering department at the Technical University of Crete, Greece. In 2007, he joined the Computer Engineering department of the Delft University of Technology, the Netherlands, where he received his PhD. In 2011, he joined the Computer Architecture and VLSI Systems group at the Foundation for Research and Technology - Hellas (FORTH) in Greece, where he is working as a post-doc researcher for national and international research projects. His research interests are in the domains of Embedded Systems, Computer Architecture, and Reconfigurable computing.

Dionisios Pnevmatikatos is a Professor and former Chair of the Electronic and Computing Engineering Department, Technical University of Crete and a Researcher at the Computer Architecture and VLSI Systems (CARV) Laboratory of the Institute of Computer Science, FORTH in Greece. He received his B.Sc. degree in Computer Science from the Department of Computer Science, University of Crete in 1989 and M.Sc. and Ph.D. degrees in Computer Science from the Department of Computer Science, University of Wisconsin-Madison in 1991 and 1995 respectively. His research interests are in the broader area of Computer Architecture, where he investigates the Design and Implementation of High-Performance and Cost-Effective Systems, Reliable System Design, and Reconfigurable Computing.

Davide Catani is R&D manager for ARM-based platforms at SECO. He graduated in electronic engineering at University of Florence in 2006 with a graduation thesis developed at Cesvit Microelettronica focused on the implementation of an USB macrocell on FPGA. He joined SECO in 2006 and is developing ARM-based systems since 2010, mainly focusing on industrial applications. Davide contributed to hardware development of the systems used to build Tibidabo and Pedraforca ARM-based clusters at BSC and to CARMA and Kayla platforms aimed to develop CUDA based applications on ARM-based systems.



David Oro received the B.S, and M.S. degrees in Computer Science from Universitat Politècnica de Catalunya (UPC) in 2006, and an M.S. degree in Computer Architecture in 2011, also from UPC. He started his professional career in 2005 working as a consultant in performance monitoring solutions. In 2009, he joined the Barcelona Digital Technology Centre where he held a research position on online banking cybercrime mitigation for CaixaBank. Currently, he works for Herta Security leading the GPU parallelization of several products. He has published several papers in international peer-reviewed conferences and holds two patents. His research interests include computer architecture, GPU computing and malware analysis.

Carles Fernández received his B.S. in Telecommunication Eng. and M.S. in Language and Speech from the Technical University of Catalonia (UPC) in 2005. He received an M.S. in Computer Vision and AI from the Autonomous University of Barcelona (UAB) in 2008, where he obtained his Ph.D. cum laude in 2010, receiving the 2010 Extraordinary Ph.D. Award. He has published more than 40 scientific articles in international journals and conferences. Currently he leads the research team at Herta Security. His research interests include biometrics, computer vision, and machine learning, particularly unconstrained facial analysis in image and video.

Carlos Segura received the B.S. and M.S. degrees in Telecommunication Engineering at the Universitat Politècnica de Catalunya (UPC) in 2003, the M.S. degree at the Technical University of Berlin (TU-Berlin) in 2003 and the Ph.D. cum laude degree in Computer Science from the UPC in 2011. He worked as a research fellow at the TU-Berlin in 2003 and in UPC from 2005 to 2011. Later he joined the company Herta Security under the Torres Quevedo program as the Director of Innovation until 2015. Currently he is working in Telefnica I+D as a speech scientist. He has participated in three national research projects and three EU research projects, and has published more than twenty scientific papers in peer-reviewed international journals and international conferences. His research interests include speaker localization and tracking, multimedia signal processing, computer vision and machine learning.

Javier Rodríguez Saeta received the B.S., M.S. and Ph.D. degrees in Telecommunication Engineering from the Technical University of Catalonia, UPC, Barcelona (Spain), in 2000 and 2005, respectively. He has also received the B.A. degree in Business Administration by the Open University of Catalonia (UOC), and the MBA by ESADE Business School. In 2000 he worked for Robert Bosch, GmbH, in Hildesheim (Germany). In 2001, he joined Biometric Technologies, S.L., in Barcelona (Spain), where he was the R&D Manager. He founded Herta Security in 2009 and became the CEO of the company. He has published more than 20 papers in different magazines and workshops, and he holds three patents. His main research interests include all issues related to innovation, security and biometric systems and applications.

Javier Hernando received the M.S. and Ph.D. degrees in telecommunication engineering from the Technical University of Catalonia (UPC), Barcelona, Spain, in 1988 and 1993, respectively. Since 1988, he has been with the Department of Signal Theory and Communications, UPC, where he is a Professor and a member of the Research Center for Language and Speech (TALP). He was a Visiting Researcher at the Panasonic Speech Technology Laboratory, Santa Barbara, CA, during the academic year 20022003. His research interests include robust speech analysis, speech recognition, speaker verification and localization, oral dialogue, and multimodal interfaces. He is the author or coauthor of about two hundred publications in book chapters, review articles, and conference papers on these topics. He has led the UPC team in several European, Spanish and Catalan projects. Prof. Hernando received the 1993 Extraordinary Ph.D. Award of UPC.

Claudio Scordino received the Master Degree in Computer Engineering from the University of Pisa in 2003. In 2007 he received the PhD from the same university. His main research interests include real-time scheduling, operating systems and programming models. He has collaborated with the Linux kernel community since 2008 having several patches integrated in the official Linux kernel.

Paolo Gai, CEO, graduated (cum laude) in Computer Engineering at University of Pisa in 2000 with a graduation thesis developed at the ReTiS Laboratory of the Scuola Superiore SantAnna on the development of the modular real-time kernel SHaRK. He obtained the PhD from Scuola Superiore Sant'Anna in 2004. Since 2000, he founded the ERIKA Enterprise project, an open-source RTOS which recently reached the OSEK/VDX certification, and which is currently used by various industries and universities. Since 2002 he is CEO and founder of Evidence Srl, a SME working on operating systems and code generation for Linux- and ERIKA- based industrial products in the automotive and white goods market. Since 2011 he is President and founder of SSG Srl, providing hardware turnkey solutions for the white goods market. His research interests include development of hard real-time architectures for embedded control systems, multi-processor systems, object-oriented programming, real-time operating systems, scheduling algorithms and multimedia applications.

Pierluigi Passera, R&D Director of Vimar SPA, wiring devices, Home&Building Automation (present). EMEAS industrial Deployment within Schneider Electric (2012-2010). R&D and Production Director in various Schneider Electrics units (2010-2001). Gewiss SPA laboratory Manager (2000-1996). ABB SACE basic research department.

Alberto A. Pomella, Electronics & Software R&D Manager of Vimar S.p.A., Standalone and Home and Building Automation products (present). R&D Director at CRS (2001–2003), Home automation Products; UX and embedded PC development group at SELCA S.p.A. (1992–2001); Project Validation Group for consumer PC at ASEM (1991–1992). Degree in Electronic Engineering from Politecnico of Torino in 1990, with specialization in software development and industrial automation.

Nicola Bettin earned his B.S degree in Electronic Engineering at University of Padua and in 2011 he obtained his M.S degree in Electronic Engineering at University of Bologna. In 2012 he joined the Tecnology Transfert Team T3LAB, in Bologna, and co-founded the FPGA Group. He did research in the design of a standard HW/SW architecture for machine vision and developed commercial solutions for processing multimedia data stream in embedded systems. His main interests were FPGA solutions and heterogenic multi-core system-on-chip solutions. He joined the electronic R&D dept. at Vimar Group in 2015 and his research activity is mainly focused on human interaction with smart home systems.

Antonio Rizzo Full Professor of Interaction Design, Universit di Siena and Co-founder UDOO (Present). Director 'Academy of Digital Arts and Science' - ArsNova (2000 - 2009). Chair of the European Association of Cognitive Ergonomics (2000 - 2006). Member di WG30 NATO Human Factors and Human Reliability Group (1999 2002). Member of the Programme Incitatif de Recherche sur lEducation et la Formation (PIREF) of the French Government (2002 - 2003). Head of the Human Factor Group of the Italian National Railways (1996–1999). Liaison for Apple Inc. for the Apple Design Project (1996 - 1997).

Roberto Giorgi is an Associate Professor at Department of Information Engineering, University of Siena, Italy. He was Research Associate at the University of Alabama in Huntsville, USA. He received his PhD in Computer Engineering and his Master in Electronics Engineering, Summa cum Laude both from University of Pisa, Italy. He is the coordinator of the European Project AXIOM. He coordinated the TERAFLUX project in the area of Future and Emerging Technologies for Teradevice Computing. He is participating in the European projects HiPEAC (High Performance Embedded-system Architecture and Compiler), ERA (Embedded Reconfigurable Architectures). He contributed to SARC (Scalable ARChitectures), ChARM (performance evaluation of ARM-processor based embedded systems). His current interests include Computer Architecture themes such as Embedded Systems, Multiprocessors, Memory System Performance, Workload Characterization.