Microprocessors and Microsystems 37 (2013) 1002–1019

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

ASAM: Automatic architecture synthesis and application mapping

Lech Jozwiak [a], Menno Lindwer [b], Rosilde Corvino [a,*], Paolo Meloni [c], Laura Micconi [d], Jan Madsen [d], Erkan Diken [a], Deepak Gangadharan [d], Roel Jordans [a], Sebastiano Pomata [c], Paul Pop [d], Giuseppe Tuveri [c], Luigi Raffo [c], Giuseppe Notarangelo [e]

[a] Technische Universiteit Eindhoven, The Netherlands
[b] Intel, The Netherlands
[c] Università degli Studi di Cagliari, Italy
[d] Danmarks Tekniske Universitet, Denmark
[e] STMicroelectronics, Catania, Italy

ARTICLE INFO

Article history:
Available online 11 September 2013

Keywords:
Embedded systems
Heterogeneous multi-processor system-on-chip (MPSoC)
Customizable ASIPs
Architecture synthesis
MPSoC and ASIP design automation

ABSTRACT

This paper focuses on mastering the automatic architecture synthesis and application mapping for heterogeneous massively-parallel MPSoCs based on customizable application-specific instruction-set processors (ASIPs). It presents an overview of the research being currently performed in the scope of the European project ASAM of the ARTEMIS program. The paper briefly presents the results of our analysis of the main challenges to be faced in the design of such heterogeneous MPSoCs. It explains which system, design, and electronic design automation (EDA) concepts seem to be adequate to address the challenges and solve the problems. Finally, it discusses the ASAM design-flow, its main stages and tools and their application to a real-life case study.

1. Introduction

The recent spectacular progress in semiconductor technologies has enabled the implementation of increasingly proficient multi-processor systems on chip (MPSoC). New important opportunities have been created: the traditional applications can now be served better, and numerous sorts of new systems became technologically feasible and economically justified.

A big stimulus has been created towards the development of innovative embedded systems. Examples of the new systems include various measurement, monitoring, control, multi-media and communication systems that can be embedded in machines or devices, or even implanted in human or animal bodies. However, these new opportunities come with a price. On the one hand, unusual silicon and system complexity has been introduced. This complexity results in a number of difficult design issues, such as:

- ensuring high-quality complex systems and their validation;
- adequately addressing the need of energy reduction;
- resolving the interconnect scalability problems;
- adequately accounting for the dominating influence of interconnects and communication on major physical system characteristics;
- decreasing the high system development and production costs, and long development times.

On the other hand, new highly-demanding embedded applications appear in several fields (e.g. consumer electronics, medical, monitoring and control systems, etc.) for which the straightforward software solutions are not satisfactory. These complex embedded applications typically include various parts, implementing different algorithms and kinds of processing. They are from their nature heterogeneous and highly demanding. Consequently, they require application-specific heterogeneous MPSoCs to perform real-time computations with extremely tight schedules, energy, area and costs requirements. Moreover, due to the rapid evolution of the embedded applications towards newer improved versions and due to the high cost of application specific circuit realization, a flexible hardware solution is needed, as provided by ASIP technology.

This paper focuses on mastering the automatic MPSoC architecture design for such highly-demanding embedded applications. It presents an overview of the research being currently performed in the scope of the European project ASAM (Architecture Synthesis and Application Mapping for heterogeneous MPSoCs based on adaptable ASIPs) of the ARTEMIS program. The paper briefly presents the results of our analysis of the main problems and challenges to be faced in the design of such heterogeneous MPSoCs. It explains which system, design, and electronic design automation (EDA) concepts seem to be adequate to resolve the problems and address the

* Corresponding author. Tel.: +31 0402474497.
E-mail address: r.corvino@tue.nl (R. Corvino).

0141-9331/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2013.08.006

challenges. Finally, it introduces and discusses the design-flow, its main stages and the tools proposed by the ASAM project consortium to enable an effective and efficient solution of these problems. It also shows the application of the ASAM tools to a real-life case study.

2. ASIP-based MPSoC technology

The **architecture platform** targeted in the ASAM project is a ***heterogeneous multi-ASIP platform*** of Intel (previously SiliconHive – SH), which can be ***configured and extended for specific applications***. Each ASIP of the platform forms a VLIW machine capable of executing parallel software with a single thread of control. An ASIP (see Fig. 1) includes a processor core (*core*) performing the actual data processing and core I/O (*coreio*) ensuing the communication of the ASIP with the rest of the system. The ASIP *core* includes a VLIW datapath controlled by a sequencer that uses status and control registers and executes programs from the local program memory. The datapath contains scalar and/or vector functional units organized in several parallel issue slots. The issue slots are connected via programmable input and output interconnections to several registers organized in register files. The functional units perform computations on intermediate data stored in the register files. The *coreio* provides the access to the local memory and I/O subsystem enabling an easy integration of the ASIP in any larger system, which can access the devices in *coreio* via master/slave interfaces. Both SIMD and MIMD processing can be realized.

The ASIPs are configurable and extensible. The parameters to be explored and set to create a new ASIP configuration include: the number and type of issue slots and (scalar or vector) instructions inside the issue slots, the number and type of issue slot clusters to optimize parallelism exploitation and communication between the issue slots, the number and size of register files, the type, data width, and size of local memories, the architecture and the parameters of the local communication structure, etc.

Several different ASIPs, each customized for a particular part of a complex application, can be interconnected via a bus or a Network-on-Chip (NoC) including shared memories and DMAs. The parameters to be explored and set at the system-level include: the number and types of ASIPs; the number, type and size of shared memories; the scheduling and mapping of the application parts onto the ASIPs and their data onto the memories; and the architecture and parameters of the global communication structure.

Several ASIPs with approximately 100 issue slots in total, each for 64-way vector processing, can be placed on a single chip implemented in 22 nm CMOS technology. When operated at 400–600 MHz, these ASIPs can deliver more than 1 Tops/s, with power consumption far below the upper limit of mobile devices. Such ASIP-based heterogeneous MPSoC platforms enable efficient exploitation of various kinds of parallelism: the multiple ASIPs enable the coarse-grain parallelism at the task level, while the ASIP's parallel issue slots, and vector instructions enable the fine-grained data and instruction-level parallelism.

This adaptable ASIP-based MPSoC technology addresses several fundamental challenges for the development of highly-demanding embedded applications:

- it is able to deliver high performance, high flexibility and low energy consumption at the same time;
- it is relevant for a very broad range of application domains;
- it is applicable to several implementation technologies, e.g.: SOC or ASIC, structured ASIC, and FPGA.

Provided that an effective and efficient highly automated customization technology will become available, it will become possible to build adaptable ASIP-based MPSoCs at substantially lower costs and with shorter time to market than the hardwired ASICs. This is the primary target of the ASAM project.

3. Issues and challenges of the ASIP-based MPSoC development

The realization of complex and highly demanding applications, for which the presented ASIP technology is suitable, requires

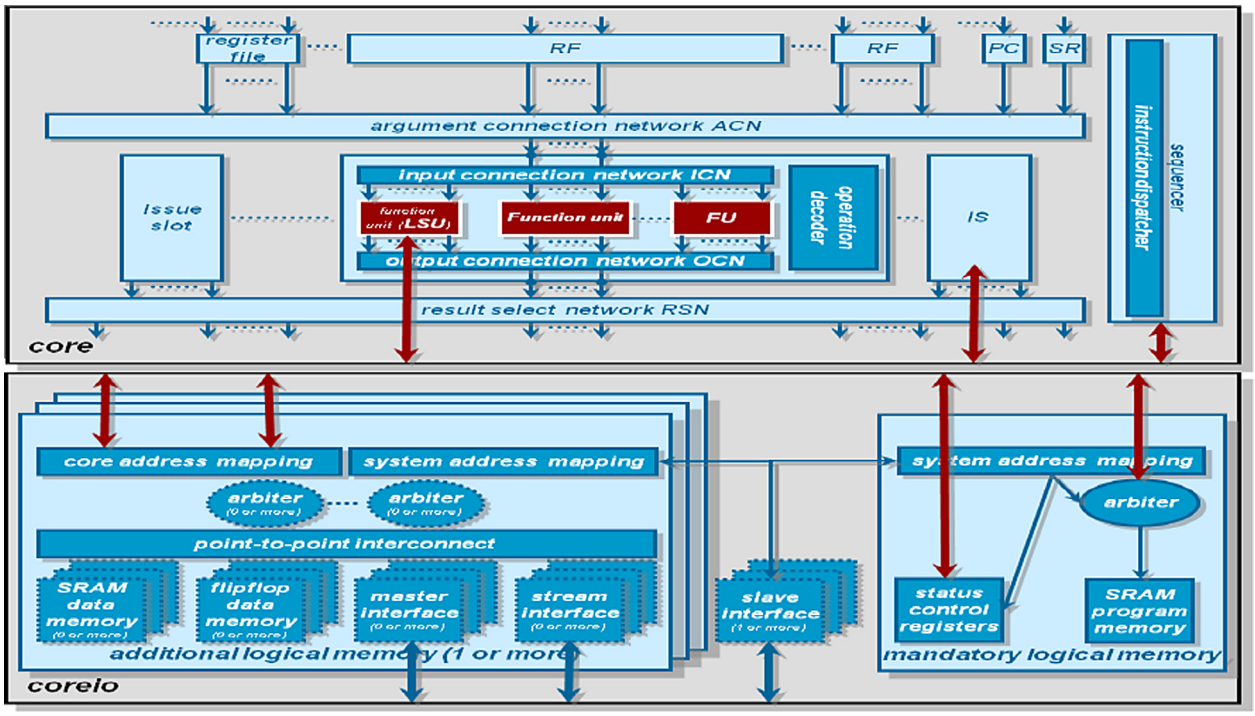


Fig. 1. Generic ASIP architecture of the targeted MPSoC platform.

performance and energy usage comparable to those of hardwired ASICs. It also requires programmability, area and cost efficiency. Satisfaction of these stringent and often conflicting requirements fosters the construction of highly-optimized hardware architectures and of their embedded software. The optimization of such ASIP construction can be achieved through an efficient exploitation of the application parallelism and an efficient exploration of the trade-off between the hardware solutions design characteristics, which are considered at different design levels and concern different system parts.

Typically, the development of an embedded MPSoC for a highly-demanding application involves many stages as: application analysis and characterization, application parallelization and partitioning, system macro-architecture design, processor selection or design, application scheduling and mapping, hardware generation and software compilation.

Unfortunately, in the traditional approaches for embedded system design, these major stages are largely disjoint. They are performed by different teams and with different supporting tools. This leads to inefficiencies, errors, and costly reiterations in the design process.

In particular, the traditional algorithm and software development approaches require an existing and stable computation platform, while for the modern MPSoCs based on adaptable ASIPs, the quality of hardware and software architectures can be substantially improved through a process of HW/SW co-tuning. Indeed, on the one hand, based on the results of the application software analysis and parallelization, optimized hardware architectures can be proposed.

On the other hand, the optimized parallel software structures have to be restructured to optimally map the proposed hardware structures. So, the designs of parallel software and hardware architectures influence each other and should be decided at the same time. Unfortunately, the efficiency of the required combined HW and SW development is still too low with the currently available development technology. This is due to lack of a holistic automated method and to the weak interoperability of the HW/SW architecture design, and hardware synthesis tools. The identified inefficiencies can substantially lower the attainable quality of the resulting systems and can also increase the necessary development time and the development costs.

Although many application analysis, restructuring, compilation tools and ASIP configuration frameworks exist (see Section 4), the automated customization decision making is missing, and the collaboration among the different tools is not yet automated. Also, most of the ASIP customization tools are devoted to a single ASIP customization. While, as mentioned above, current heterogeneous applications require multi-ASIP systems. Consequently, also the customization tools should be able to handle the design of multiple ASIPs at the same time. The various ASIPs in a system have to be customized together, and in a strict relation with system level concerns as the selection of the number of ASIPs and inter ASIPs communication sub-system. The MPSoC macro-architecture and the ASIP micro-architectures are strictly interrelated. Many trade-offs have to be resolved regarding the granularity of individual processing cells, and between the amount of parallelism realized at the system and processor levels.

The two architecture levels are strongly interwoven also through their relationships with the memory and communication structures. Each micro-/macro-architecture combination, with a different parallel computation structure, requires different compatible memory and communication architectures. For instance, exploitation of more data parallelism in a computing unit micro-architecture requires a simultaneous access to memories in which the data reside (with e.g. vector, multi-bank or multi-port memories) and a simultaneous transmission of the data (with e.g. multiple interconnects). The requirement of simultaneous access and transmission radically increases the memory and communication hardware. Additionally, many applications require implementation of algorithms that involve complex interrelationships between the data and computing operations. For applications of this kind, the main design problems are related to an adequate resolution of memory and communication bottlenecks and to decreasing the memory and communication hardware complexity. The memory and communication structure design, and the architecture design for computing units cannot be performed independently, because they substantially influence each other.

Finally, the existing methods and tools for custom instruction-set construction or extension are devoted to a single processor; usually they extend a simple RISC processor with large hardware accelerators. In our method, we target the extension of VLIW processors, with a set of re-usable and efficient custom functional units.

The optimization of the performance/resources trade-off required by a given highly demanding application can only be achieved in a holistic approach performing an actual HW/SW co-design; a combined synthesis of processing, memory and communication sub-systems; and a combined macro- and micro-architecture synthesis.

4. Contribution and related works

The general aim of the ASAM project is to enhance the design efficiency of the ASIP-based MPSoCs for highly demanding applications, while improving the result quality. This aim is being realized through the development of a coherent system-level design-space exploration and synthesis flow including automatic analysis, synthesis and rapid prototyping environment. The flow and its implementation have to provide efficient exploration of the architecture and application design alternatives and tradeoffs.

Based on the analysis of the application, computing platform and parametric requirements, the ASAM flow will efficiently partition a given complex application and select the most appropriate set of ASIPs to create the MPSoC macro- and micro-architecture. It will reuse, instantiate, and extend the ASIPs with new application-specific hardware, developing this way the ASIP micro-architecture. Moreover, in correspondence with the macro- and micro-architecture design, it will restructure the application's software and implement the software on the so constructed application-specific multi-processor platform. Finally, it will analyze and validate the design through a rapid prototyping.

The research of the ASAM project builds on the methodology of quality-driven model-based system design proposed in [1].

The ASAM project builds also on the platform-based design of heterogeneous multi-processor embedded systems [1,2], ASIP design methods [3–8], hardware compilation techniques [2], and software analysis, re-structuring and compilation techniques [2,9].

With respect to the MPSoC macro-architecture synthesis, the project exploits the quality-driven model-based design exploration and architecture synthesis approach [2,10], and modeling, emulation, estimation and design exploration concepts developed in [2,10–12].

The new macro-architecture design space exploration (DSE) methodology proposes enhancements in reuse of generic architecture platforms, modeling of the platform in the form of an abstract architecture template, generic architecture template instantiation, abstract behavioral and parametric requirement modeling, and application scheduling and mapping.

The ASIP micro-architecture exploration and synthesis proposes enhancements in application analysis and parallelization, automatic customization of ASIP architecture in terms of storage mechanism, parallel computing and instruction set. It will be able to re-use and customize available ASIPs and also to construct new

ASIPs from scratch. The existing commercial and academic developments in this field do not provide adequate support for this critical correlated parts of ASIP design (see e.g. [2]).

As explained in the previous section, there are very strong interrelations between the macro- and micro-architecture syntheses. Therefore, ASAM architecture synthesis method considers the macro-architecture and micro-architecture synthesis as one coherent complex system architecture synthesis task, and not two separate tasks, as in the state-of-the-art methods. There are common aims and a strong consistent collaboration between the two subtasks. The macro-architecture synthesis proposes a certain number of customizable ASIPs of several types with a part of the application assigned to each of the proposed ASIPs. The micro-architecture synthesis customizes each of the ASIPs, together with its local memories, communication and other blocks, and correspondingly restructures its software to implement the assigned application part as effective and efficient as possible. Subsequently, the restructured application part software is compiled, and the RTL-level HDL descriptions of the customized ASIPs are automatically generated and synthesized to an actual hardware design. From several stages of its application restructuring and ASIP design, including the actual HW/SW implementation, the micro-architecture synthesis provides feedback to the macro-architecture synthesis on the physical characteristics of each particular sub-system implemented with each ASIP core. This way the micro-/macro-architecture trade-off exploitation is enabled. After several iterations of the combined macro/micro-architecture exploration and synthesis an optimized MPSoC architecture is constructed.

A complete MPSoC architecture involves of course the adequately instantiated ASIPs together with their local memories and communication, as well as, adequate global memory and communication structures. As explained in the previous section an effective and efficient design of the memory and communication structures is especially important for many modern applications that involve massive parallelism and algorithms with complex interrelationships between the data and computing operations. While current research in this area is mainly focused on the separate design of memory and interconnection systems, ASAM project considers the mutual relationships between interconnections, memories and processors. The memory and communication structures are optimized in an iterative refinement process, when accounting for the application-specific memory-processor communication and the technology related memory and communication features, such as power dissipation or area. Regarding the global memory and communication structures the project builds on recent results of some of the project partners [13–15].

From the above it should be clear that the ASAM design flow and its tools will implement an actual coherent HW/SW co-design process through performing a quality-driven simultaneous co-tuning of the application software and processing platform architecture to produce HW/SW systems highly optimized for a specific application. The ASAM flow and its tools consider the macro-architecture and micro-architecture synthesis as one coherent complex task, and perform the application-specific synthesis of processor, memory and communication architectures in a strict collaboration to ensure their compatibility and effective trade-off exploration among the different design aspects. As a consequence, ASAM design methods and tools have to deal with decisions regarding a huge number of architectural aspects and values of customization parameters.

To effectively and efficiently cope with such a massive combination of design choices, ASAM exploits the abstraction, separation of concerns and quality-driven design decision making principles through introducing several abstraction levels in the design flow, decomposing complex design problems into hierarchical networks of simpler sub-problems (issues), ordering the consideration of the sub-problems, and using various abstract and partial models when solving particular sub-problems [1]. The methods and tools used for each level/issue deal with a sub-set of correlated design concerns. They collaborate with each other in a well-defined coherent way to together deliver a high-quality application-specific HW/SW system design.

To the best of our knowledge, the ASIP-based MPSoC design problem as formulated above is not yet explored in any of the previously performed and published works. The related research in the MPSoC, ASIP, application analysis and restructuring, and other areas considers only some of the sub-problems of the adaptable ASIP-based MPSoC design in isolation. In result, the proposed partial solutions are usually not directly useful in the much more complex actual context.

As stated in [16], ASIP auto-customization methods can be subdivided into configuration-based and specification-based. The configuration-based methods use well defined processors optimized for an application field. Only a few parameters are left to enable customization of the processor to requirements of a specific application. This simplifies the DSE, but reduces the possibilities of tuning. The configuration-based approach is exploited by Cadence (previously Tensilica) Extensa Configurable Core [7], ARC Configurable Cores [6], etc. The specification-based methods provide the possibility to entirely describe a new processor based on an abstract model that only defines the general design rules, when using an Architecture Description Languages (ADLs) that allow describing the relevant (application-specific) aspects of the ASIP architecture at several abstraction levels. Specification-based methods and corresponding ADLs include EXPRESSION [5] and its EXPRESSION ADL [17], CoWare Processor Designer [4] using LISA ADL [18], Target Compiler Technology [3] using nML ADL [19]. Most of these methods and related tools target the design of systems involving only a single ASIP. Intel SH uses TIM language for the ASIP architecture description and HSD language for the MPSoC system-level architecture description (e.g. global communication, processors synchronization, etc.). There are several differences between TIM/HSD and the other ADL languages. TIM is a high-level HDL (it does not describe an instruction set); HSD describes multi-core systems.

Although the existing ASIP customization frameworks usually involve tools for application analysis, application re-targetable compilation, as well as, single ASIP architecture configuration, and automatic HDL generation, they usually lack any automated architecture design-space exploration and decision support, even for a single ASIP. On contrary, ASAM project aims at developing such an effective and efficient highly automated DSE and decision support, additionally not limited to a single ASIP, but for the multi-ASIP systems.

Other related works focus on the application code transformations [20] to improve the application software mapping onto a fixed architecture optimized to a broader application area, e.g. DSP or GPU. Their results are not directly applicable to the combined software and hardware structuring of the adaptable ASIP-based systems. Yet other works target the processor Instruction Set Extension (ISE) [21] and related hardware extension. Some of them try to explore and exploit the effect of loop transformations [22], e.g. unrolling, on the ISE generation. They are however devoted to a single processor, usually a simple RISC processor with one issue slot and not to a complex VLIW processor with several different issue slots. Moreover, the proxy formulations of their custom instruction set construction problems and their suggested solutions usually do not reflect well the actual problems to be solved and their required solutions.

Many published research results [1,10,23] and system design frameworks, e.g. Metropolis [24], Daedalus [39], etc., target heterogeneous MPSoC design, but they do not addresses adaptable ASIP-based MPSoCs being the target of ASAM. Nevertheless, some of the

valuable ideas and general methodologies developed in these research works will be reused for ASAM purposes, as for instance, the formalization of DSE methodologies as surveyed in [25], the methodology of quality-driven model-based system design as in [1], the DSE methodologies for memory management as in [26,27] and the DSE methodology for system design as in [28].

Most importantly however, to our knowledge, none of the published methods, tools or frameworks implements an actual coherent HW/SW co-design process through a combined simultaneous structuring of the application software and processing platform architecture. Also, most of the published research works focus on the processing unit design and application mapping, but underestimate the importance of the memory and communication architecture design. Although [16] proposes to use profiling techniques to customize memory hierarchy and infer Instruction Set Architecture design (i.e. instruction opcodes, instruction encoding, memory/register addressing modes and data types), ASAM explores the effect of loop transformations on the Data Transfer and Storage Mechanisms [20] in order to benefit from previous advances in design automation for ASICs. To our knowledge, no former research addressed the problem of the combined concurrent processor, memory and communication architecture exploration and synthesis, except for a recent work of the ASAM partners [16], but in [16] it was done for a different design target. Moreover, ASAM project proposes a novel rapid prototyping for ASIPs and ASIP-based MPSoCs. Although this platform is based on the well-known FPGA emulation, it differs in several aspects from similar existing platforms. It enables emulation of complete HW/SW designs and their characterization regarding performance and power. It also enables prototyping of several architectures concurrently. Further extensive discussion of related research can be found in the overview papers [2,23].

5. Design flow

An overview of the **ASAM design flow** is presented in Fig. 2. The flow involves **four main stages**, which are organized in three abstraction levels and correspond to the main design issues:

- System DSE,
- ASIP DSE,
- Global communication and memory (GC&M) DSE, and
- HW/SW synthesis and rapid prototyping.

These stages communicate and collaborate with each other directly or through the system DSE. All together, the stages realize a quality-driven evolutionary design flow. They transform stepwise an initial high-level application specification into an ASIP-based MPSoC. In this process, they use generic ASIP-based platform models.

The input high-level application specification includes the application's C code, the design parametric requirements and representative input stimuli.

The multi-ASIP SoC design is compliant with a generic SH/Intel hardware template, which includes a library of hierarchical customizable components and their rules of composition.

The ASAM design flow is based on the concept of **service-oriented EDA system**. Each of its main stages can be requested for and provides services for the others. The collaboration among the stages of the flow or some of their parts follows a request-response protocol. This enables clear organization of the stages collaboration and results in a high flexibility. Indeed, the flow can be further extended with additional features and the modifications of one of its services are independent from the others. The stages of the ASAM design flow use services of the underlying SH/Intel design flow, which provides a library of Intellectual Properties (IP), HW generation, SW compilation and simulation environments. In particular, each stage of the ASAM flow can get an IP from the SH/Intel library, use it as is (e.g. for simulation or emulation) or customize it and insert into the IP library anew. The flow execution is originally determined by its primary inputs and the user control inputs. However, the progress of the flow execution is more and more influenced by the results of previous explorations, in order to ensure its convergence.

System DSE takes as inputs: an application C-code, parametric and structural requirements, and representative stimuli. It is responsible for the entire design of the multi-ASIP SoC. It defines its structure composed of several ASIPs communicating though a network of distributed shared memories. While performing this task, System DSE asks for specific services from the middle abstraction level which includes the stages ASIP DSE and the GC&M DSE. These services can range from a coarse estimation of design parameters to the optimized synthesis of ASIPs and GC&M subsystems. System DSE can also ask for services from the lower stage of "rapid prototyping" (the communication is represented by light color arrows in Fig. 2). This request aims at performing simulation or emulation of (parts of) a multi-ASIP SoC. When asking for a service, the system DSE specifies to a given stage what is the requested services, the C code of the application part to be analyzed and the related design requirements and input stimuli. Orchestrating the analysis and synthesis services of the different stages and combining their results, the system DSE produces a final optimized ASIP-based MPSoC design.

ASIP DSE aims at the design of a single ASIP and its associated software for the execution (of a part of) the whole application. From the System DSE, it takes as inputs a service request, the C code of the application part on the target, the partial design requirements, and the related input stimuli. ASIP DSE consists of a simultaneous co-tuning of the ASIP architectures and their embedded software. It integrates HW/SW co-design techniques from high-level synthesis for the architectural decision-making and parallelization techniques from software compilation for the code optimization. In performing design and parallelization tasks, the ASIP DSE can ask for specific services from the HW/SW synthesis and rapid prototyping, as well as, from the SH/Intel design flow.

GC&M DSE aims at the exploration and optimization of the global communication and memory structures for a multi-ASIP system. This is performed through an iterative construction and refinement of interconnect and memory structures driven by the constraints and objectives decided by the System DSE. In particular, the System DSE mapping of the application tasks is used to produce a communication graph, which is the input of the GC&M DSE. GC&M DSE iteratively proposes and evaluates candidate interconnect and memory architectural configurations, characterizes them using simulation-based methods and, when needed, accesses the lower-level prototyping infrastructure to endorse the simulation-based characterization. A Pareto front of configurations compliant with the input mapping and the input constraints is selected and serves as a feedback to the System DSE. **HW/SW synthesis** accepts as input service requests from the System DSE, ASIP DSE and GC&M DSE. It also takes as input the abstract architecture description (in TIM and HSD languages) of the designed MPSoCs or their parts, and the corresponding restructured application C-code. From the TIM/HSD descriptions the HW synthesis automatically generates the RTL hardware description and the SW synthesis generates and compiles the restructured C-code. In this way a complete HW/SW (sub-) system design is produced. Rapid prototyping accepts as its inputs the HW, SW or HW/SW designs and performs their simulation or emulation. All parts of the ASAM flow use specific services of the SH/Intel design flow.

In the next sections each of the main flow stages will be discussed more precisely.

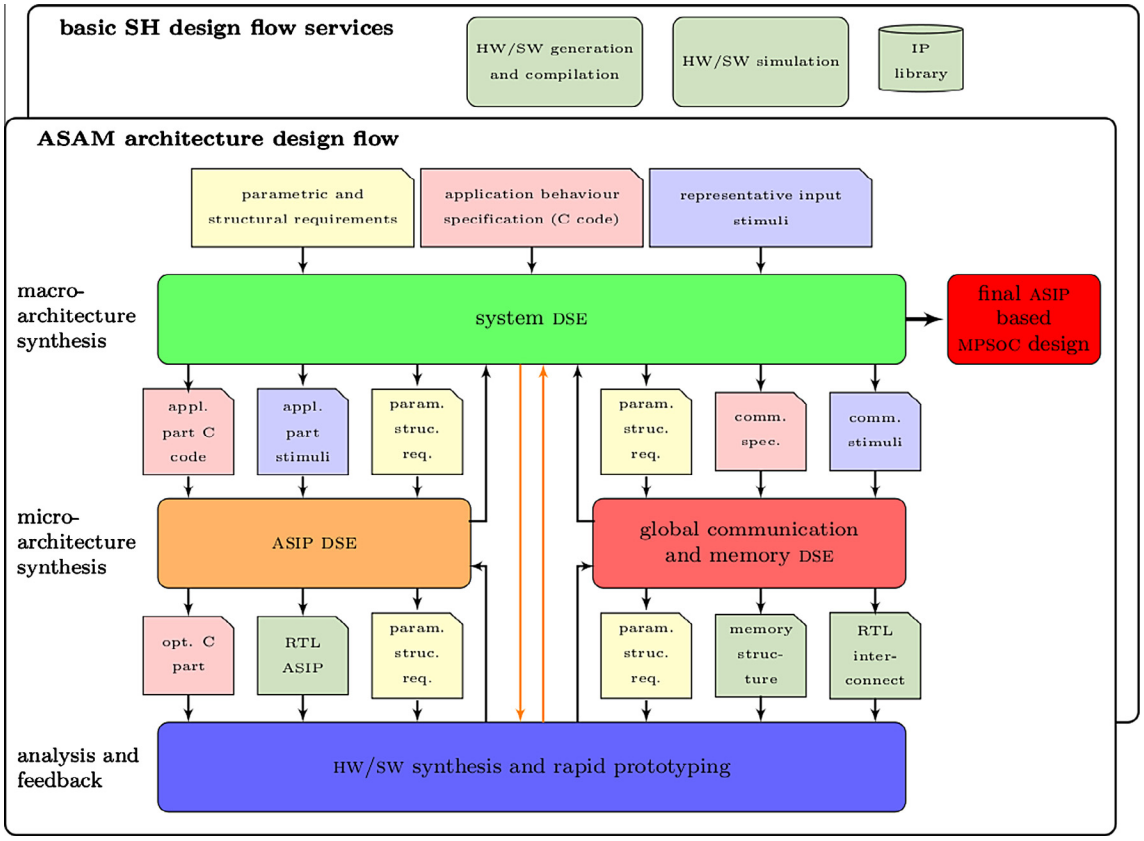


Fig. 2. ASAM design flow.

6. Main stages of the ASAM flow

6.1. System DSE

The System DSE is in charge of developing the macro-architecture of a multi-ASIP system. This involves deciding the number and type of ASIPs, the structure of global memories and interconnections, as well as, the scheduling and mapping of different application parts onto the ASIPs. The system-level design decisions are taken selecting among multiple design possibilities of the system parts, as provided by the ASIP DSE and the Communication and Memory DSE (Fig. 3).

The System DSE requires as input the C-code of one or more applications and the parametric requirements of the design, e.g. the execution deadline of each application, the area and power requirements of the whole system, etc. The original application code is modeled as a task graph: the C code is partitioned into tasks (application parts) to extract **task and inter-task (pipeline) parallelism**. The partitioned application is generated by commercial tool of one of the project partners, Compaan Design [29]. The tool identifies the tasks and their execution concurrencies. The communication between connected tasks is modeled by messages, which represent the amount of exchanged data.

The main purposes of the System DSE is to determine, from the initial task graph, the clustering/mapping and scheduling of the tasks on the ASIPs and of the messages on the interconnection resources.

An initial instance of the computing platform, obtained by assigning each task to a separate ASIP, defines the upper bound of the number of allocated ASIPs. This is decided according to the number of tasks and annotated with the information of on Worst Case Execution Time (WCET) of each task provided by ASIP DSE.

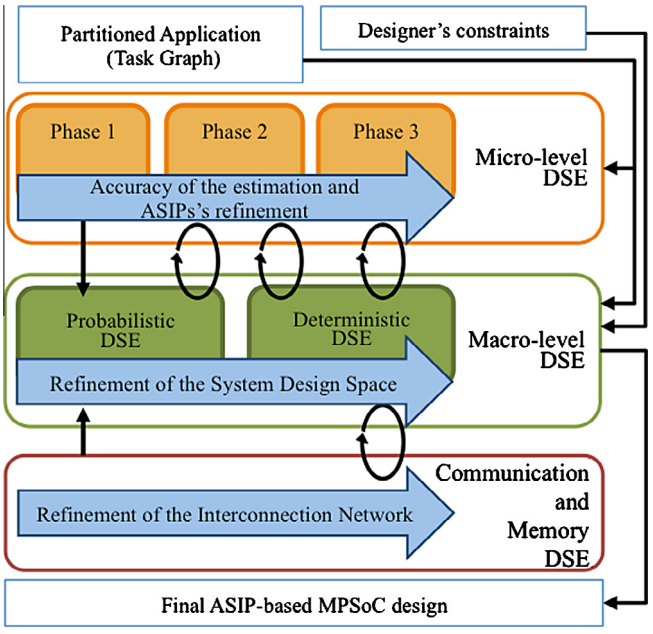


Fig. 3. System DSE.

It is also possible to define a maximum number of ASIPs as an input constraint of the analysis. The initial platform model assumes the use of a very generic Network-on-Chip (NoC), in which each ASIP is assigned to a switch and all switches are connected through point-to-point links, generating a fully connected network. This choice is justified by the need of a general initial platform allowing for a wide exploration of tasks clustering onto the different ASIPs.

A first set of possible clustering solutions is generated by System DSE at the beginning of the ASAM flow. This set is then passed to the other DSE stages, i.e. the ASIP DSE and Communication and Memory DSE, and improved through a number of exploration iterations as outlined in Fig. 3. In particular the ASIP DSE provides multiple ASIP micro-architectures characterized by the corresponding values of performance, area and power, while the Communication and Memory DSE returns an optimized interconnection network satisfying given performance constraints. The System DSE collects these partial results from the different DSE stages and combines them together to verify the performances, area and power consumption of the entire system. As depicted in Fig. 3, the System DSE exchanges information with the other DSE stages in an iterative way. This allows a gradual refinement of the design space reducing the number of system solutions towards better solutions, in order to finally converge to a single multi-ASIP system.

The System DSE is composed of two phases (Probabilistic and Deterministic DSE), both in charge of performing a DSE and evaluating the entire system. The final output of the System DSE is the description of a multi-ASIP system that meets the design requirements to a satisfactory degree. In the next two sections we present details of the Probabilistic and Deterministic DSE and details of their interaction with the other design stages.

Probabilistic DSE: This is the first and also the most challenging phase of the System-level Design. At the beginning of the ASAM flow, there is no computing platform available (the interconnections and the ASIP micro-architectures are unknown). Only the task graphs of the applications, their C-code and parametric requirements are provided. We are facing a so-called *chicken-and-egg* problem in which the tasks need to be grouped, scheduled and assigned (*clustered*) to multiple ASIPs, without any knowledge of the ASIP micro-architecture. This implies that it is not possible to evaluate and therefore compare the performances of different clustering solutions. At the same time, the ASIP DSE needs information about the proposed clustering of tasks to define the micro-architecture of a single ASIP, as each ASIP has to be tuned to efficiently execute the assigned tasks. The different micro-architectures that can be identified during the ASIP design are called ASIP configurations. The number of possible ASIP configurations is very high due to the high number of configuration parameters. For this reason it is not feasible to perform a System DSE analysis taking into account all the possible configurations of the ASIPs. In a traditional approach, the task clustering is user-defined based on designers experience and knowledge, the remaining problem is to tune the ASIP micro-architecture and the tasks code accordingly. In the ASAM flow we propose a method to automate the clustering of the applications at the system level. To make it possible, we propose a Probabilistic Estimation Method to break the circular dependency between the exploration of clustering solutions and the need of an ASIP micro-architecture.

An initial estimation of the computing performance required to execute a given task can be obtained from an analysis of the task code. This allows estimating the bounds of the performances that can be achieved for a specific task. As will be described in Section 6.2 (Phase1), we evaluate the Worst Case Execution Time (*WCET*) for two different possible executions of the same task: a sequential execution on a scalar ASIP, upper bound WCET (*uWCET*) and a parallel execution with an ASAP scheduling without architecture constraints, lower bound WCET (*lWCET*). We chose the WCET, as we need to verify if the input applications are able to meet their deadline with a non-preemptive static scheduling policy. The range identified by lower and upper bound of WCET is used to define a cumulative distribution function (CDF) that represents the performances for the execution of one task **on all the possible micro-architecture configurations of an ASIP** (whose number and individual performances are yet unknown). The WCET becomes a stochastic variable that captures all the possible variations of the ASIP micro-architecture for that specific task.

In the Probabilistic DSE, the WCET is considered to be the only optimization parameter, as there is no actual synthesis of the ASIPs or a particular definition of their micro-architectures. Therefore it is not possible to build a model for their power and area requirements. A CDF is built **for each task**. More details on the probabilistic model of the WCET are available in [30]. We assume a fully connected NoC with point-to-point connections between each pair of switches to be an initial interconnection network. Each element of the NoC (network interfaces and switches) has certain latency and throughput, which we assume to be constant. Given this assumption on the interconnections, we model the latency of message passing as the time spent for transferring the amount of data in the message. However, it is also possible to consider different types of switches and network interfaces having multiple WCETs for each message. Values of latency and throughput for the NoC elements are provided by the Communication and Memory DSE. Once all messages and tasks have been characterized, it is possible to evaluate the performances of a particular clustering solution. The tasks are assigned to the ASIPs and the messages are assigned to the interconnection elements. The WCET CDF **of each application** executing on the system is calculated based on the individual task CDFs. We take into account both task level and pipeline parallelism. Moreover, we consider the data dependences between tasks and the resource contentions. When multiple tasks/messages want to access a shared hardware resource, they are ordered according to their priority (fixed value assigned to each task/message at the beginning of the ASAM flow). Then, using the deadline constraints provided as input, we obtain the probability of meeting the deadline of each application. We calculate the mean value of these probabilities to get the final cost of a specific clustering solution. A design space exploration engine has been built to explore and evaluate different clustering solutions. The ones with the highest probability of meeting the deadline are chosen. When multiple clustering solutions have the same probability, the one that uses less hardware resources (e.g. smaller number of ASIPs) is selected. The first output of the Probabilistic DSE is therefore **a set of clustering solutions**. Every single solution is composed of multiple *clusters*; a cluster corresponds to one or more tasks grouped together to be executed on a single ASIP. Each solution in the set of clustering solutions is then separately considered. For each of them, the Probabilistic DSE is re-run multiple times as discussed below. For each solution, System DSE requests the optimization of one cluster (i.e. one ASIP) from the ASIP DSE. The ASIP DSE returns to the System DSE a subset of micro-architecture configurations for that ASIP. The result of this partial micro-architecture synthesis is used by the System DSE, which can repeat on the basis of this more precise information. The cluster of the already optimized ASIP is maintained (one of the available ASIP micro-architecture received by the ASIP DSE is selected) and the Probabilistic DSE is re-run on the other not-yet-configured ASIPs. This phase is repeated until all ASIPs have been optimized by the second phase of the ASIP DSE.

The second phase DSE is quite fast (its execution can be completed in a few minutes) and can be repeated multiple times if needed. Once the design of all ASIPs has been sufficiently decided, i.e. a bounded subset of the most promising specific configurations has been identified for each ASIP; the System DSE requests the Communication and Memory DSE to start the optimization of the interconnection network, given a certain clustering, as well as, throughput and latency constraints on each communication link. This is repeated for each solution in the set of clustering solutions.

Deterministic DSE: Is the second phase of the System DSE. It performs a DSE similar to the Probabilistic one, but with much more precise information on the platform and its parts. This phase

works with concrete ASIPs, memories and communication configurations and verifies the performances of the global ASIP-based system. As a consequence, in this part of the flow, it is possible to perform a multi-objective optimization taking into account also the values of the area and power consumptions obtained from the ASIP DSE.

The Deterministic DSE can interact with both the second and the third phases of the ASIP DSE. The interaction with the second phase of the ASIP DSE can be repeated multiple times, while the interaction with the third phase is more time-consuming and should be limited. Moreover the Communication and Memory DSE needs to be re-run every time the constraints on latency or throughput on the links between the processors and memories are changed due to changes in the micro-architecture configurations of each ASIP. In fact, even if the clustering solution is not modified after a second interaction with the second phase of the ASIP DSE, it will be modified after the interaction with phase 3. Indeed the third phase of the ASIP DSE is in charge of optimizing the ASIP data-path and generating custom instruction sets. This will result in changes in the performances, area and power consumption and may affect the whole system. When the design constraints are not met, the Deterministic DSE has to perform adjustment in the selected clustering solution. The number of changes in the clusters, i.e. moving one task from one processor to another, should be minimized. Moreover, these changes should take into account the characteristics of the clusters of tasks and of the ASIPs themselves. For example, it makes sense to move a task to an ASIP that is able to execute it and on which similar tasks are already running. This will limit the number of changes required in the micro-architecture of the ASIPs involved.

These DSE interactions are repeated until the design requirements are satisfied or it becomes clear that it is impossible to satisfy them given the available resources.

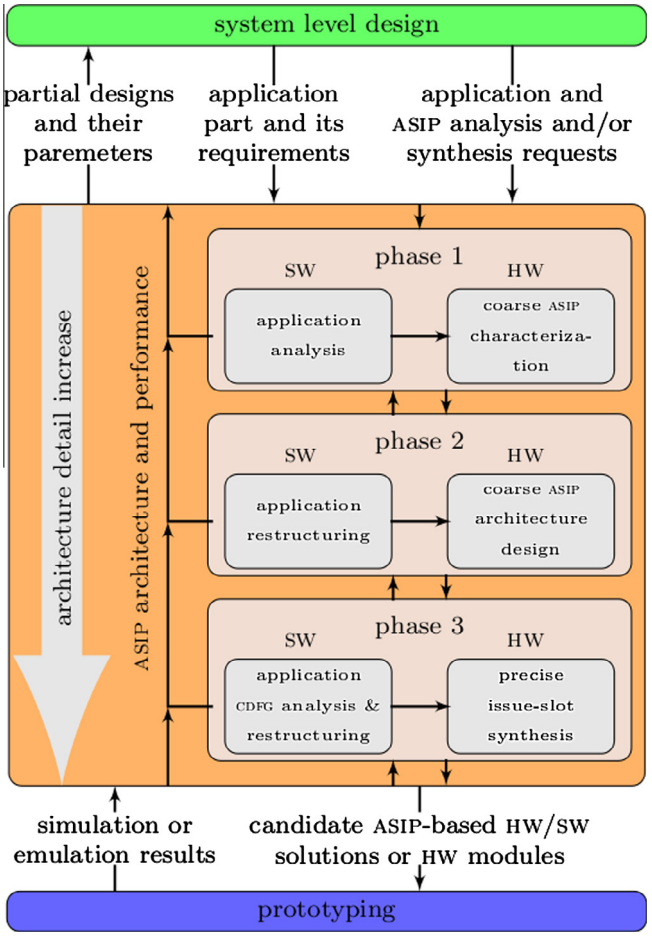


Fig. 4. ASIP level DSE.

6.2. ASIP DSE

ASIP DSE, presented in Fig. 4, aims at a blend software restructuring and ASIP architecture design. It does it for application parts assigned to a single ASIP by the System DSE.

For a given part, ASIP DSE performs application analysis and characterization, exploration and selection of possible application parallelization and ASIP designs. It can provide services related to these activities to the System DSE or directly to an end-user. Each service request has to be associated with related inputs, i.e. C code of the application parts, parametric requirements and input stimuli.

Due to the high number of possible solutions for the application restructuring and ASIP customization, a synthesis method with a reduced exploration complexity is necessary. We propose a method based on a "divide and conquer" strategy, which address different design concerns in different phases of the flow. This method is shown in Fig. 4 and involves three main phases:

Phase 1, performing application analysis and characterization,
Phase 2, performing software and hardware co-design for parallel processing, communication and storage, and
Phase 3, performing instruction set synthesis and refined application restructuring.

The services of these three phases can be provided to the System DSE or directly to an end-user. The phases can be executed separately or in combination, depending on the precision of the required analysis and synthesis. A given phase can also ask the successive phases for services.

Each phase is more precisely described below.

Phase 1 performs application analysis and characterization. It takes as input the C code of an application or an application part from the System DSE. It profiles and characterizes it in different ways.

Initially, it provides information about the upper and lower bound of the worst-case execution time (WCET) of the given C code. That is, it estimates the WCET for a sequential execution of the C code (upper bound) and for the most parallel version of the application without architectural constraints (lower bound). As explained earlier, the WCET values are used by the System DSE for the initial evaluation of the proposed application partitioning and to infer the design requirements of the application parts (cf. Section 6.1).

Subsequently, the application analysis detects and classifies the different parts of the input C code, according to the kind of processing they need. For instance, for data-intensive hot-spots, it will advocate a realization based on vector processing. As a consequence of this analysis, Phase 1 prepares the data for further analyses and calls either Phase 2 to solve parallelization issues or Phase 3 to solve synthesis issues.

Phase 2 performs data-oriented software and hardware co-design to decide the ASIP parallel processing, communication and storage architectures**.** It takes as input the parts of the C code identified by Phase 1 as data-intensive hotspots.

It explores and selects possible parallel restructuring of the software and the corresponding hardware architectures. Phase 2 is built on an internal data-oriented representation of the application, called Array-OL [31], which is used to rapidly evaluate the software restructuring and the corresponding ASIP hardware

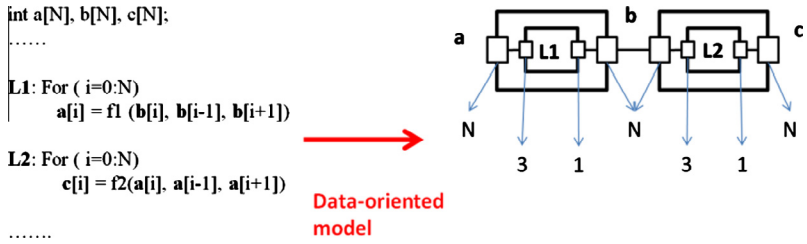


Fig. 5. Data oriented model of a loop-based C code. The terms *N*, 3 and 1 indicate the sizes of the arrays and data patterns that are consumed and produced by the two loop nests, L1 and L2.

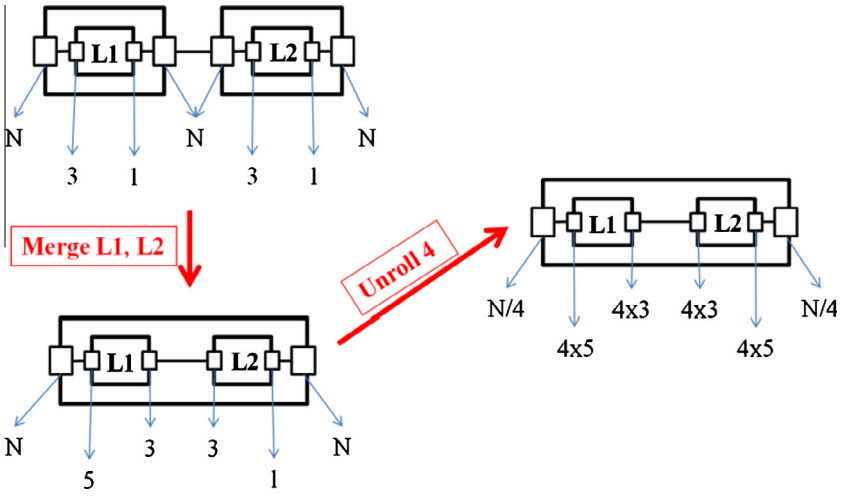


Fig. 6. Example of transformations for parallelization.

architectures. Fig. 5 shows a simplified example of the data-oriented model corresponding to the loop-based C code on the left. Such a model captures data dependencies and granularities; but, it abstracts information about the computations in the straight line code, which are a concern of Phase 3.

In order to select Pareto optimal parallel versions of the loop-based code and infer from this the corresponding ASIP architecture, Phase 2 evaluates the improvements of the code execution due to some loop transformations. This evaluation is a result of a static analysis performed on the internal data-oriented Array-OL representation. The evaluation accounts also for the performances of the allocated ASIP architecture though an analytical model of its area and execution time. A simplified example of the representation of the loop transformations through the data-oriented Array-OL model is sketched in Fig. 6, where the transformations fusion and unrolling are shown.

Using several established allocation and mapping rules, Phase 2 infers also the ASIP architecture from the internal data-oriented representation. In particular, it decides for memory hierarchy and data parallelism through vectorization or usage of multiple issue slots. An example is shown in Fig. 7, where allocation and mapping are represented. In particular the allocation of issue slots (IS1 and IS2) and the mapping of loop body (L1 and L2) on them, as well as, the allocation of register files (RF) and local memories (LM) and the mapping of data on them are represented.

The number of instantiated LMs also fixes the number of needed load and store (LS) units. To better understand the allocating and mapping rules, let us consider some example. A transformation such as loop unrolling can be used to identify possible vectorizations. Indeed, the unrolling of independent loop iterations identifies identical kernels (i.e. independent iterations of a same loop body) which can be vectorized, provided that the necessary vector instructions are allocated.

As a consequence, the architecture design should allocate a standard or custom issue slot containing vector instructions to realize the foreseen vectorization. The construction of the appropriate set of used issue slots is a concern of Phase 3. Another example of allocation and mapping rule can be given by loop fusion. Loop fusion merges two or more loops in a same iteration space, reducing the loop control to a single thread. As a consequence, it is possible to process the tasks of two or more merged loops in parallel on the same ASIP. To make this possible, the architecture design should allocate an issue slot per merged loop with the associated register files and local memories.

A simplified example of an ASIP architecture allocated to realize the example of Figs. 5 and 6, is given in Fig. 8. Such architecture includes a sequencer to manage the control thread and the issue slots associated with the loop bodies. If a loop body is unrolled and the unrolled iterations are independent, then the issue slot type is "vector", which means it includes vector instructions.

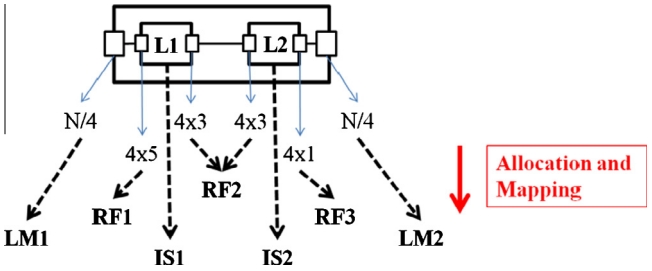


Fig. 7. Allocation and mapping rules.

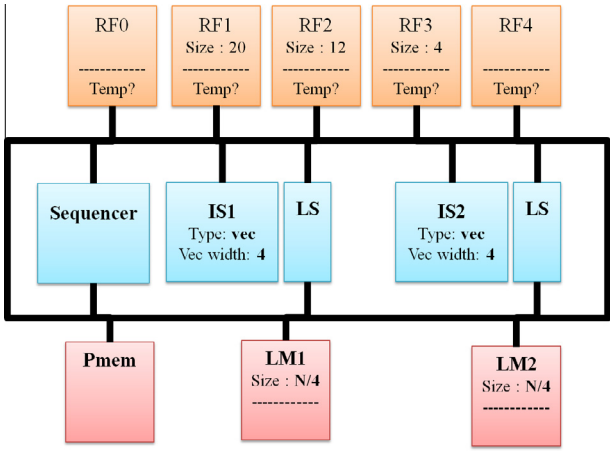


Fig. 8. Example of selected ASIP architecture. The number of allocated registers is equal to the number of the allocated issue slots. However, some register sizes are not known because their final value is computed from the subsequent phase 3.

The vector width is equal to the number of the loop body unrolling factor. The architecture also contains local memories (LM) and register files (RF) as specified by the allocation and mapping rules (Fig. 7). Each LM needs to be associated with a load/store unit to be accessed without conflict. Finally, in order to maximally benefit from the parallel issue slots of the architecture, Phase 2 instantiates a number of RFs (or RF ports) equal to the number of used issue slots.

In order to rapidly evaluate and select the solutions of this exploration, Phase 2 uses an analytic model. This technique avoids the error-prone and time-consuming process that actually constructs all the considered architectures. In particular, the analytic model estimates the execution time of the proposed software structure onto the proposed architecture; it estimates the area of the proposed solution; and, in a future version, it will estimate the power consumption. An exploration process based on the Opt4J genetic algorithm framework [32] creates and evaluates a large number of possibilities. Few solutions with the best area and execution time trade-off are selected and can be communicated either to the System DSE for an improvement of the currently considered application partitioning or to the rapid prototyping environment for a more accurate evaluation of the found solutions. This part of Phase 2 is based on previous works further explained in [27,33–35]. For each selected Pareto solution, Phase 2 generates two outputs.

- An AOL file containing the restructured data-oriented model. This file suggests the loop transformations that should be applied to the C code of the application part.
- A XML file describing how to compose the ASIP architecture out of the elements of SH/Intel library.

Using the AOL file together with the services of the software restructuring and compilation flow of SH/Intel environment, Phase 2 constructs the restructured C code. Using the XML file and the SH/Intel library, Phase 2 constructs the TIM code of the ASIP. The ASIP system defined by the pair of restructured C code and TIM code, can be used for simulation in the SH/Intel simulation environment or in the rapid prototyping stage.

Phase 3 performs instruction architecture synthesis and related application restructuring. It mainly involves the identification and selection of an application specific instruction set for each ASIP design. Phase 3 takes as architectural constraints of the instruction architecture synthesis, the number of parallel load and store (LS) operations that can be executed per cycle and memory mapping as computed by Phase 2. The selection in Phase 3 of an appropriate instruction set goes through several stages. First, the number of needed issue slots is computed in order to execute the input straight-line code according to required scheduling constraints. Then, the required hardware operators are identified and distributed over the allocated issue slots so that the scheduling requirements are met. The selection of an instruction set may concern individual basic blocks (e.g. loop bodies). In this case, the result of Phase 3 is used to update the library of issue slots used by Phase 2. But, the instruction set selection, may also concern multiple basic blocks executed in parallel. In this case, Phase 3 provides a refinement of the design performed in Phase 2 and accounts for possible hardware sharing among the instructions of different basic blocks. The selection of the best instruction set is performed with respect to three design optimization criteria such as area occupancy, execution time or throughput, and energy consumption. If after the initial selection the instruction set does not meet the design requirements, custom operations can be proposed. In this extension process, the original DFG are identified, realized in hardware and included in the instruction library, before performing instruction set selection anew. Phase 3 also updates the size of the register files to store the local variables alive during the basic blocks execution.

After finishing the synthesis of a single ASIP based sub-system, the generation of the TIM and/or HDL code describing the ASIP and of the associated optimized C-code for the embedded software is performed.

6.3. Global memory and communication DSE

The main aim of this stage is the exploration and optimization of the global communication and memory structures for a multi-ASIP system. This is performed through an iterative construction and refinement of the interconnect and memory structures driven by the constraints and objectives decided by the macro-architectural level. In particular, the mapping of the application tasks at the macro-architecture level is used to produce a communication graph. Using this graph as input, the micro-architectural memory and communication optimization iteratively proposes and evaluates candidate interconnect and memory architectural configurations, characterizes them using simulation-based methods and, when needed, accesses the lower-level prototyping infrastructure to endorse the simulation-based characterization. This way, a Pareto front of configurations compliant with the input mapping and the input constraints is selected and serves as a feedback to the system level, as depicted in Fig. 5. The selected design points are characterized in terms of timing, area and energy consumption, when exploiting the support for technology awareness described in Section 6.4. In this way, the macro-architectural layer gets information for a multi-objective architectural optimization.

GM&C configurations are compositions of different functional blocks, such as:

- FIFO-based point-to-point connections.
- Single-layer shared buses.
- Multi-layer bus subsystems.
- NoC modules.
- Shared Parallel memory modules.

In this way architectural configurations compliant with a wide variety of system-level designs (various number and kind of processors, communication structures and global memories) can be composed.

For simulative evaluation of design points during the GM&C DSE, the Sesame MPSoC simulation framework [36] is exploited.

Sesame is a modeling and simulation environment for an efficient design space exploration of heterogeneous embedded systems. According to the Y chart design approach, it recognizes separate application and architecture models within a system simulation. An application model describes the behavior of a (set of) concurrent application(s). An architecture model defines architecture resources and captures their performance characteristics. Subsequently, using a mapping model, the application model is explicitly mapped onto the architecture model (i.e. the mapping specifies which application tasks and communications are performed by which architectural resources in an MPSoC).

The upper layers of the design flow provide details about the application, that are fed into the application model. More in detail, the aforementioned task graph is evaluated to obtain information about the computational latency associated with each task and about the communication channels between tasks (number and size of the token exchanged through the channels).

When the application model is executed, each process records its computational and communication actions, and generates a trace of application events. These application events are an abstract representation of the application behavior and are necessary for driving the architecture model.

Typical examples of application events are:

- read (channel id, pixel block) that represents a communication event, in this case a data from an input channel;
- execute (DCT) that represents an atomic computation event, for example the execution of a DCT kernel.

The architecture model simulates the performance consequences of such computation and communication events generated by the application model. It is parameterized with an event table (calibration table hereafter), that contains latency values associated for the events to the architectural components. More in detail, for example, it includes the latency associated to the NoC router components (input and output buffers, arbiter, crossbar), the stalls due to congestion, the network interfaces latencies, related to buffering and packeting operations and the latency associated with accesses to memory modules. Typically, the content of the calibration table can be obtained by means of the FPGA based prototyping environment.

As mentioned before, within the ASAM flow, the mapping information is also provided in input to the GM&C optimization phase by the System DSE. According to the mapping, the application and architecture models are co-simulated to qualitatively and quantitatively study the performance.

6.4. Rapid prototyping platform

The main role of the prototyping infrastructure in the ASAM flow is to provide an accurate executable FPGA-based model, capable of reducing the gap between the estimation of the performances considered during the early steps of the design flow and those really measurable after the implementation. Furthermore, the designed FPGA prototyping platform is enhanced with particular techniques that allow obtaining a speedup in the exploration phase, especially useful when dealing with an exploration of many different architectural configurations.

The prototyping platform is composed of several ASAM tools that interact with each other and cooperate with several commercial tools. The platform takes different inputs, namely micro- and macro-architectural specifications that drive the HDL generation step and the FPGA prototyping phase Fig. 10. The macro-architectural description is provided as input to describe the top-level view of the system, by means of a proprietary HSD format. The micro-architectural description is related to both the topology of interconnect and memories, and the processor architectural configuration of ASIPs.

The ASIP architecture is described using a proprietary language (TIM). The interconnect topology description is provided using an in-house developed format that can be translated by a dedicated utility in a completely configured HDL description. The interconnect structure, being based on a Network-on-Chip fabric, is customizable to a great extent in terms of a number of switches, connections among them, network interfaces and other parameters.

The interconnect topology can be instantiated as a black-box in the HSD system-level description, so that its HDL code can be comfortably linked to the main system (containing ASIP processor(s)) for synthesis. The HDL generation step envisions the code instrumentation for performances evaluation. The industrial tool-chain for ASIP design has been customized in order to allow for the automatic instantiation of hardware counters inside the processor/system RTL code, to compute the event counts, i.e. counting the number of accesses to each functional block in the ASIP processor, memory or interconnect structure, and cycle counts, e.g. the number of clock cycles needed to execute a given application on a given architecture. Analytic models have been developed within the project (through performing training sets of experiments for a given technology) in order to translate the counts obtained from the FPGA prototyping hardware implementation into the technology development energy and execution time figures.

Adequate custom Tcl scripts have been developed to enable, without further user intervention or adaptation, the correct hardware structures to be created and memory-mapped to be accessed during software execution. By means of these extensions, the synthesized hardware is automatically equipped with the correct number of performance counter registers, placed in relevant parts of ASIP processors/interconnect structure, ready to be fetched by the simulation environment and coupled with appropriate area/energy/frequency models.

Furthermore, a retargetable compilation tool-chain with complete awareness of the ASIP processors/system specifications, and thus able to efficiently schedule the application tasks on the hardware resources, has been deployed. The RTL description of a system, optimized for the FPGA implementation, can be synthesized and implemented on FPGA exploiting commercial tools. At this point, the target application is compiled by means of the ASIP software compilation tool-chain, retargeted according to the ASIP processors/system specifications, and executed on the FPGA platform, to collect the relevant metrics at the end of the execution.

For the sake of manageability and data exchange between the different phases of the flow, a co-simulation approach exploiting the SysGen toolbox by Xilinx and MATLAB has been adopted. The collaborative use of these tools allows for executing the software on the FPGA board and collecting the evaluation data from a host workstation, while accessing dedicated shared-memory structures that are instantiated in the MATLAB workspace Fig. 9.

Through the SysGen toolbox, one can instantiate the FPGA hardware as a black box inside the environment, being able to fetch/send data to the memories instantiated inside the design, read counter values, initialize data structures, etc. Once done, the FPGA device is programmed as usual, and the execution is controlled directly inside the MATLAB environment.

In [37], we find an example of a typical setup where the FPGA prototyping platform is used for a characterization of a single-ASIP architectural configuration: an ASIP specification is instantiated, capable of emulating many different single-core configurations; a host processor is also present, in charge of uploading program binaries to the ASIP core; a NoC-based interconnect and a shared memory complete the diagram. At the end of the execution, from MATLAB we are able to fetch relevant activity figures from the

shared counters that can be used to further enhance the analysis, enabling a pre-estimation of the quality-of-results achievable with a prospective ASIP implementation.

In the prototyping tool-flow this objective is achieved with the usage of accurate area and energy models. The models consist in a set of analytic expressions, able to calculate the area occupation and the energy consumption of each functional block inside a processor as a function of the architectural parameters and of the activity rate. The expressions have been defined, within the ASAM project, studying the dependency of area and energy on the mentioned variables, for every ASIP functional block.

The models have to be calibrated for each target technology considered for the ASIP implementation. The expressions have to be characterized over a training set of processor configurations.

The ASIPs in the training set must be implemented at layout-level and the area occupation and energy consumption must be analyzed to build a table of coefficients to be filled in the expressions. The coefficients typically represent area and energy values "normalized" to the design parameters and to the activity rates. Obviously, the area model is activity-independent, while the energy dissipation depends on the functional block activity, i.e. on the number of cycles during which the considered functional block was enabled or accessed (in case of memory and register files).

Once the models have been preliminary calibrated for a given technology library, they can be used to back-annotate values obtained from the previously mentioned hardware counters, adequately connected to the relevant signals in the FPGA prototype. In this way, it is possible to perform a detailed technology-aware evaluation of every ASIP configuration under prototyping. Moreover, the prototyping tool-flow has been enriched with a novel support for the multi-design point characterization, to enable the extensive exploitation of FPGAs within the micro- and macro-architectural DSE.

In order to push the hardware prototyping one step further, within the framework we envision the synthesis of an emulation platform equipped with the hardware resources necessary for the evaluation of different system configuration types, to be reused for different explorations employing a software-based reconfiguration mechanism.

The general idea is to overcome the important time limitation in FPGA-based design due to the time-consuming synthesis phase. With this aim an over-dimensioned architectural description is defined covering a number of different configurations to be explored, and implemented on the FPGA.

The same kind of approach has been applied at the interconnection and processor-level, to enable rapid NoC topology selection and evaluation for multiple ASIP configurations.

Consequently, through the usage of custom-developed tools, the prototyping flow is capable of executing application binaries compiled for all the candidate architectural configurations on the same over-dimensioned FPGA prototype, leading to significant time savings in the architectural configurations exploration and selection.

7. Preliminary result

This section presents several experiments and preliminary results obtained while using main stages of the ASAM design flow with representative applications and designs.

7.1. System and ASIP DSE

To demonstrate our hierarchical design method, involving a tight cooperation between System and ASIP DSE, we use a motion JPEG (MJPEG) encoder application provided with the Compaan tool. This application is sufficiently complex to benefit from a mapping on a multi ASIP platform but also sufficiently simple to be used as an explanatory example. The ASAM flow can address more complex applications.

The Compaan tool analyses the C code of the application and generates the corresponding KPN in which the code is partitioned into multiple tasks (or nodes) as shown in Fig. 11 a. From the KPN, a task graph (Fig. 11 b) as the one required in input of System DSE is generated. The constraints provided as input are the application deadline, $d = 5000$ ms and the maximum area of the platform, $\alpha_p = 2 * 10\ \mu m^2$. Moreover we consider two different types of buses, b_{32}^{10} and b_{16}^{10} (32 and 16 bits width with a frequency of 10 MHz) during the hierarchical DSE. The C code of each single task τ_j is elaborated by the application analysis phase that generates the upper and lower WCETs values, respectively indicated as $WCET_j^u$ and $WCET_j^l$.

Table 1 presents the values obtained for each task and normalized with respect to the $WCET_{\tau_1}^l$ (the smallest value). For the ASIPs, we assume a working frequency of 100 MHz that is compatible with the available components of the micro-architecture library.

The probabilistic DSE is executed and, from the set of clustering solutions with the highest probability of meeting the deadline d, we select the clustering solution with the minimum number of clusters (i.e. two) and the smaller bus (b_{16}^{10}). The selected clustering solution is described in Table 2.

Given this clustering solution, Phase 2 of micro-architecture DSE generates two ASIPs customized to the specific task clusters. For each ASIP, depending on the available optimizations, multiple Pareto micro-architecture solutions are generated: in particular, 1 solution for P_1 (that allows a limited optimization as it contains only one task) and 5 solutions for P_2.

The start and end points of the tasks execution time for the Pareto solutions are estimated together with the areas values. The improvement of the error of the start and end times estimation

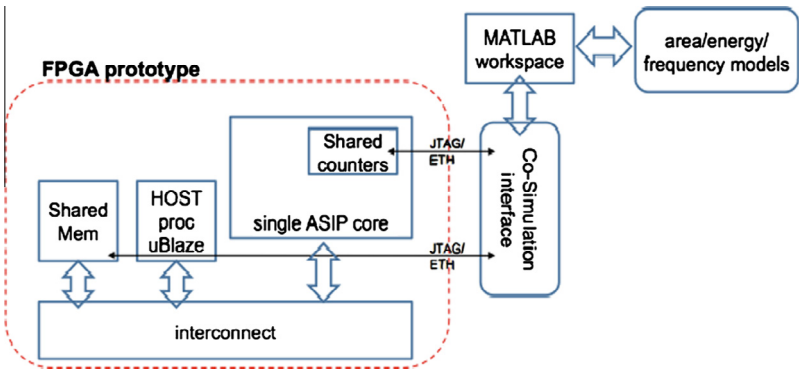


Fig. 9. Block diagram for FPGA/MATLAB SysGen interfacing.

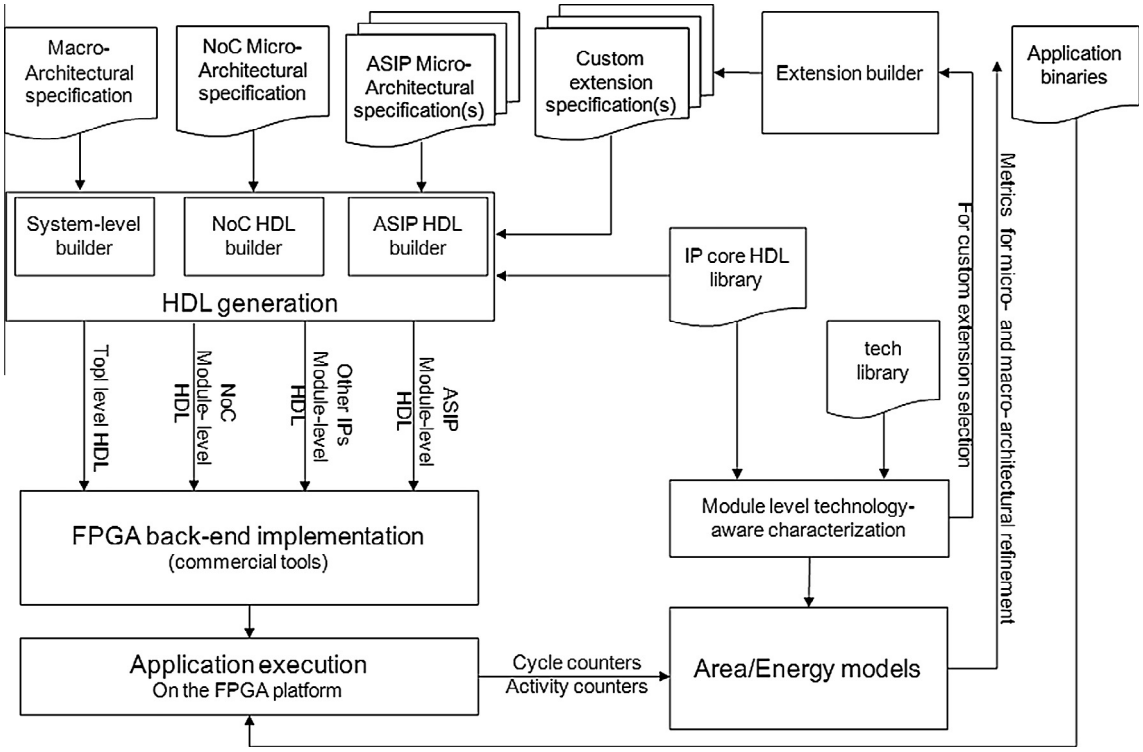


Fig. 10. FPGA prototyping.

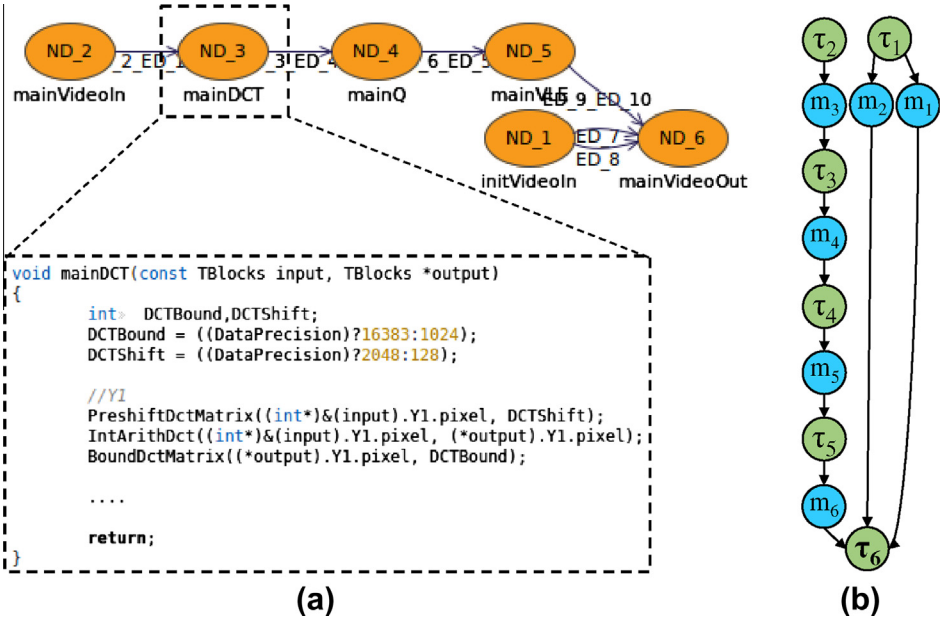


Fig. 11. KPN of the MJPEG encoder (a) and corresponding task graph (b).

Table 1
Normalized output of the application analysis.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6
$WCET_j^u$	1.4	83.5	1541	1746	790.1	26.2
$WCET_j^l$	1	45	1151	83.3	540.5	22.1

Table 2
Selected clustering solution.

P_1	P_2
τ_2	$\tau_1, \tau_3, \tau_4, \tau_5, \tau_6$

is still work in progress, but in this paper we wanted to give a first proof of concept. Then, the deterministic DSE selects between the available Pareto solutions evaluating the area and performances of the entire system. The results obtained are shown in Fig. 12. For the given input constraints (red lines in Fig. 12), there are two acceptable solutions. For the final implementation, we

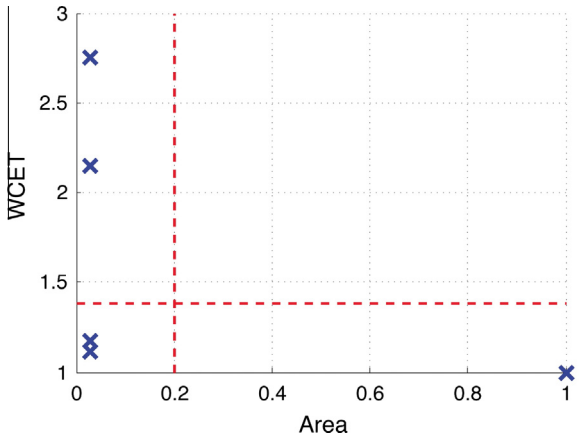


Fig. 12. Normalized output of the deterministic DSE.

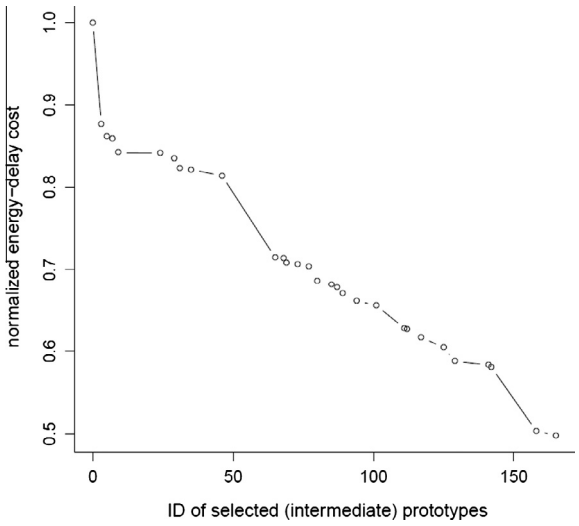


Fig. 13. Set of 31 candidate architectures for rapid prototyping, automatically selected by phase 3 after considering 186 alternative architectures in its design space.

selected the one with the smaller WCET. With this case study we demonstrate that our design flow can generate a heterogeneous bus-based multi-ASIP system targeted to a specific application, which meets the design constraints.

This section exemplifies the cooperative usage of System and ASIP DSE in a hierarchical Design Space Exploration (DSE) method to address the circular dependency problem in ASIP-based system design and to produce efficient designs of heterogeneous multi-ASIP platforms given an application task graph and design constrains as input specification. The method is demonstrated with the MJPEG case study. The single ASIP design needs to be further refined as explained in next section.

7.2. ASIP design refinement

When provided with an initial over-sized architecture by Phase 2, the third phase of the micro-level architecture exploration selects candidate architectures for the rapid prototyping environment. It explores the removal of issue-slots, function-units, and register file entries.

During this exploration, it considers the impact on the processor memory area (including program memory), application execution time, and total energy consumption of the candidate architectures.

Fig. 13 shows how a set of 31 candidate architectures for rapid prototyping were automatically selected after considering 186 alternative architectures from the design space. The Phase 3 tools provide an intelligent architecture selection algorithm which greatly reduces the number of design points that need to be considered during rapid prototyping.

7.3. Rapid prototyping

The FPGA-based prototyping platform is already fully implemented and fully operational. It has been tested on several benchmark designs and works as expected. In this section we present a use case of the previously described prototyping platform. We plot the results obtained while performing the architecture selection process over a set of 30 different ASIP configurations, which can be provided by phase 3 of the ASIP DSE or by an end-user. In the presented results, the explored design points were identified considering different permutations of the following parameter values:

- N_is (c): 2, 3, 4 or 5 (number of issue slots).
- FU_set (x, c): from 3 to 10 Functional Units (FUs) per issue slot.
- RF_size (x, c): 8, 16 or 32 entries – each 32 bits wide.
- N_mem (c): 2, 3, 4 or 5 (number of data memories).

To perform the analysis we chose an image filtering kernel, compiled for every candidate configuration. The obtained binaries were executed on the WCC prototype, after having been adequately manipulated. The host processor of the prototype reads the adapted binary from the local memory and uploads it to the ASIP core.

After the binary has been uploaded, the host processor triggers the ASIP core for its start and waits until the end of its execution, to fetch the results of the execution from the ASIP's local memory and, eventually, checks them for the presence of any errors. The adopted hardware FPGA-based platform uses the Xilinx Virtex5 XC5VLX330 device, with over 2M equivalent gates. In Fig. 14 we present the results obtained with respect to the total execution time, total latency, and total energy and power dissipation. Using the prototyping results, multi-objective optimization can be effectively performed: imposing a constraint on maximum execution time (e.g. 200K cycles), one can identify a subset of candidates that do not satisfy the constraint (gray bullets). Then, among the remaining design points, the best configuration with respect to the power or area (white bullet) can be selected.

To further extend the analysis depth, functional unit (FU)-level detailed performance can be obtained, referring to each single FU included in the configurations under test. As an example, we show in Fig. 15 a graph reporting power consumption of each function unit in a particular configuration, during the execution of the already mentioned binaries. All the presented data are obtained after only once traversing the synthesis/implementation flow.

In order to evaluate the accuracy of the proposed approach and the achievable speed-up, we compared the emulation results obtained by means of our prototyping strategy with those obtained using a cycle-accurate software-based simulator provided with the baseline SH/Intel flow tool-chain, referring to the same image-filtering kernel. Exactly the same experimental data were obtained by means of the two methods; with respect to functional metrics such as the cycle count. But, while cycle-accurate simulation performed on a workstation (Intel Quad-Core) required few minutes (roughly five on average per configuration), onboard execution on the FPGA prototype required only few seconds (roughly two) to emulate each candidate architecture.

A synthesis/implementation flow, performed on an Intel Quad-Core machine with commercial tools, required less than half an

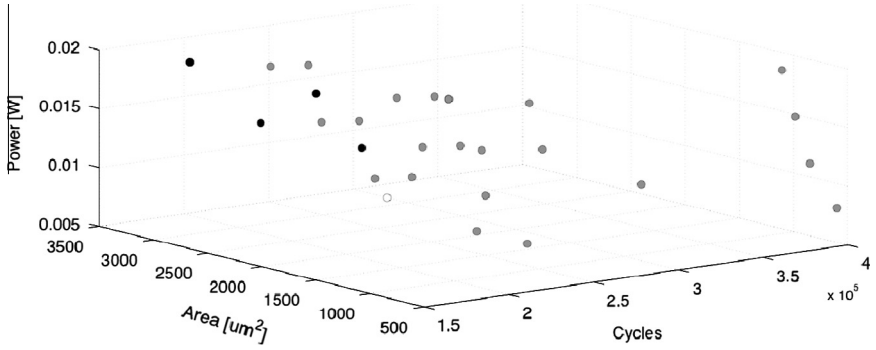


Fig. 14. Use case results. Every configuration could be represented by a different 4-tuple, whose elements represent total number of issue slots, register file capacity (in 32-bit words), number of fully-featured issue slots, number of data memories.

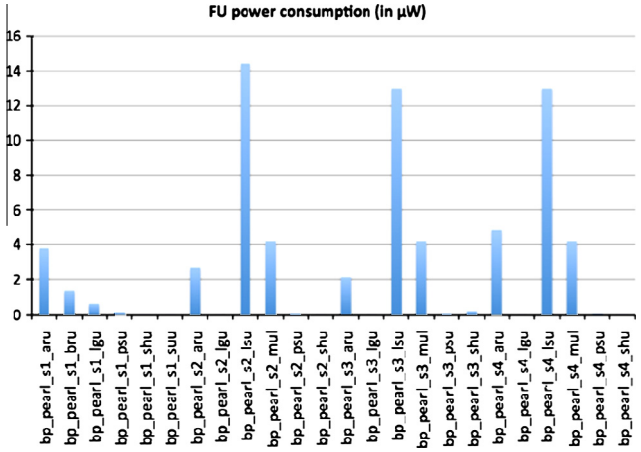


Fig. 15. Power consumption for each FU in a particular configuration, composed of four issue slots, with register files of 16 entries, reported in µW. As can be seen from graph, load-store units and arithmetical units are the power hot-spots in the design.

hour to complete. Such time obviously depends on the size of the system, but can be estimated in the order of 1 h for moderately complex systems.

Binary translation is also performed on the same machine, but the related overhead in terms of emulation time is negligible (less than a second). To provide an indication of the occupation of the commercial FPGA devices by the prototype system on, it is worth pointing out that a three ASIP processor system implementing the mentioned image kernel application occupies 19,859 slices totally on the previously cited XC5VLX330, of those 6923 are used as slice registers and 16,387 as slice LUTs.

8. Conclusion

In the former sections an overview of the research results of the European project ASAM has been presented. The overviewed results include: analysis of the main challenges to be addressed during the development of ASIP-based MPSoCs for highly-demanding applications; proposal of a multi-stage design flow that addresses and overcomes the challenges; description of the developed methods and prototype tools that implement the stages of the design flow; and discussion of the experimental results obtained when using some of the design methods and tools.

All the design methods of the ASAM flow are already developed and the related prototype EDA tools are implemented. However, the initial versions of the tools have to be extended and refined, and a few interfaces among the tools are still under development. Although the ASAM flow is not yet fully implemented, its implementation is in an advanced stage. This allowed us to perform several experiments, with representative applications and designs, and obtaining some preliminary results. These results were briefly discussed in this paper. They confirm that the proposed ASAM design flow and tools are appropriate for the automated design of ASIP-based MPSoCs. They have the potential to produce high-quality HW/SW ASIP-based systems in limited lapse of time. More information on the research results from ASAM can be found in several already published papers [30,33–35,38–45] and on the ASAM home-page (http://www.asam-project.org/).

Acknowledgements

The research reported in this paper has been performed in the scope of the ASAM project of the European ARTEMIS Research Program and has been partly supported by the ARTEMIS Joint Undertaking under Grant No. 100265.

References

[1] L. Jozwiak, Quality-driven design in the system-on-a-chip era: why and how?, Journal of Systems Architecture 47 (3–4) (2001) 201–224
[2] L. Jóźwiak, N. Nedjah, M. Figueroa, Modern development methods and tools for embedded reconfigurable systems: a survey, Integration, the VLSI Journal 43 (1) (2010) 1–33. Jan.
[3] Target, Target Compiler Technology. <http://www.retarget.com>.
[4] CoWare and Synopsys, CoWare Processor Design. <http://www.synopsys.com/Systems/BlockDesign/ProcessorDev/Pages/default.aspx>.
[5] University of California Irvine, EXPRESSION. <http://www.ics.uci.edu/~express/>.
[6] ARC-Inc. and Synopsys, ARC Configurable Cores. <http://www.synopsys.com/IP/ProcessorIP/Pages/default.aspx>.
[7] Tensilica, Tensilica – Xtensa Customizable Processors. <http://www.tensilica.com/products/xtensa-customizable>.
[8] O. Schliebusch, H. Meyr, R. Leupers, Optimized ASIP Synthesis from Architecture Description Language Models, Springer, 2010, p. 207.
[9] R. Leupers, P. Marwedel, Retargetable Compiler Technology for Embedded Systems – Tools and Applications, Springer, 2001, p. 175.
[10] L. Jozwiak, S. Ong, Quality-driven model-based architecture synthesis for real-time embedded SoCs, Journal of Systems Architecture 54 (3–4) (2008) 349–368. Mar.
[11] S. Mahadevan, F. Angiolini, J. Sparso, L. Benini, J. Madsen, A reactive and cycle-true IP emulator for MPSoC exploration, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27 (1) (2008) 109–122.
[12] A.S. Tranberg-Hansen, J. Madsen, A service based component model for composing and exploring MPSoC platforms, in: 2008 First International Symposium on Applied Sciences on Biomedical and Communication Technologies, 2008, pp. 1–5.
[13] S. Murali, D. Atienza, P. Meloni, S. Carta, L. Benini, G. De Micheli, L. Raffo, Synthesis of predictable networks-on-chip-based interconnect architectures for chip multiprocessors, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 15 (8) (2007) 869–880.
[14] F. Angiolini, P. Meloni, S.M. Carta, L. Raffo, L. Benini, A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs, IEEE Transactions

on Computer-Aided Design of Integrated Circuits and Systems 26 (3) (2007) 421–434.
[15] L. Jóźwiak, Y. Jan, Communication and memory architecture design of application-specific high-end multi-processors, VLSI Design 2012 (2012) 20.
[16] K. Karuri, R. Leupers, Application Analysis Tools for ASIP Design: Application Profiling and Instruction-set Customization, Springer, 2011, p. 232.
[17] A. Halambi, P. Grun, V. Ganesh, A. Khare, N. Dutt, A. Nicolau, EXPRESSION: a language for architecture exploration through compiler/simulator retargetability, in: Proceedings of the European Conference on Design, Automation and, Test (DATE-1999), 1999, p. 485.
[18] A. Hoffmann, H. Meyr, R. Leupers, Architecture Exploration for Embedded Processors with LISA, Springer, 2002, p. 244.
[19] A. Fauth, J. Van Praet, M. Freericks, Describing instruction set processors using nML, in: Proceedings of the European Design and Test Conference (ED&TC – 1995), 1995, pp. 503–507.
[20] F. Catthoor, K. Danckaert, S. Wuytack, N.D. Dutt, Code transformations for data transfer and storage exploration preprocessing in multimedia processors, IEEE Design & Test of Computers 18 (3) (2001) 70–82.
[21] K. Martin, C. Wolinski, K. Kuchcinski, A. Floch, F. Charot, Constraint -driven identification of application specific instructions in the DURASE system, in: Proceedings of the 9th International Workshop on Embedded Computer Systems: Architectures, Modeling, and, Simulation, 2009, pp. 194–203.
[22] A.C. Murray, R.V. Bennett, B. Franke, N. Topham, Code transformation and instruction set extension, ACM Transactions on Embedded Computing Systems 8 (4) (2009) 1–31.
[23] D. Densmore, R. Passerone, A platform-based taxonomy for ESL design, IEEE Design & Test of Computers 23 (5) (2006) 359–374.
[24] UCBerkeley, Metropolis: Design Environment for Heterogeneous Systems. <http://embedded.eecs.berkeley.edu/metropolis/platform.html>.
[25] G. Ascia, V. Catania, A.G. Di Nuovo, M. Palesi, D. Patti, Efficient design space exploration for application specific systems-on-a-chip, Journal of Systems Architecture 53 (10) (2007) 733–750.
[26] F. Balasa, P.G. Kjeldsberg, a. Vandecappelle, M. Palkovic, Q. Hu, H. Zhu, F. Catthoor, Storage estimation and design space exploration methodologies for the memory management of signal processing applications, Journal of Signal Processing Systems 53 (1–2) (2008) 51–71.
[27] R. Corvino, A. Gamatié, P. Boulet, Architecture exploration for efficient data transfer and storage in data-parallel applications, in: Euro-Par 2010-Parallel Processing, 2010, pp. 101–116.
[28] S.L. Shee, S. Parameswaran, Design methodology for pipelined heterogeneous multiprocessor system, in: Proceedings of the 44th Annual Conference on Design automation (DAC '07), 2007, p. 811.
[29] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, E. Deprettere, System design using Kahn process networks: the Compaan/Laura approach, in: Proceedings of the Conference on Design, Automation and Test in Europe (DATE '04), 2004, pp. 340–345.
[30] L. Micconi, D. Gangadharan, P. Pop, J. Madsen, Multi-ASIP platform synthesis for real-time applications, in: 8th IEEE International Symposium on Industrial Embedded Systems (SIES 2013), 2013.
[31] C. Glitia, P. Dumont, P. Boulet, Array-OL with delays, a domain specific specification language for multidimensional intensive signal processing, Multidimensional Systems and Signal Processing 21 (2) (2009) 105–131.
[32] M. Lukasiewycz, M. Glaß, Opt4J: a modular framework for meta-heuristic optimization, in: Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011), 2011, pp. 1723–1730.
[33] R. Corvino, E. Diken, A. Gamatié, L. Jozwiak, Transformation based exploration of data parallel architecture for customizable hardware: a JPEG encoder case study, in: 16th Euromicro Conference on Digital System Design (DSD 2013), 2012, pp. 774–781.
[34] R. Corvino, A. Gamatié, Abstract clocks for the DSE of data intensive applications on MPSoCs, in: Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2012), 2012, pp. 729–736.
[35] R. Corvino, A. Gamatié, M. Geilen, L. Jozwiak, Design space exploration in application-specific hardware synthesis for multiple communicating nested loops, in: Proceedings of the 9th International Conference on Embedded Computer Systems: Architectures, Modeling, and, Simulation (SAMOS-2012), 2012, pp. 1–8.
[36] A.D. Pimentel, C. Erbas, S. Polstra, A systematic approach to exploring embedded system architectures at multiple abstraction levels, IEEE Transactions on Computers 55 (2) (2006) 99–112. Feb.
[37] P. Meloni, S. Pomata, G. Tuveri, M. Lindwer, L. Raffo, Exploiting binary translation for fast ASIP design space exploration on FPGAs, in: Proceedings of the Int. Conference on Design, Automation, and Test in Europe (DATE'12), 2012, pp. 566–569.
[38] R. Jordans, R. Corvino, L. Jozwiak, Algorithm parallelism estimation for constraining instruction-set synthesis for VLIW processors, in: 16th Euromicro Conference on Digital System Design (DSD 2013), 2012.
[39] E. Diken, R. Jordans, R. Corvino, L. Jozwiak, Application analysis driven ASIP-based system synthesis for ECG", in: Embedded World Conference, 2011.
[40] R. Jordans, R. Corvino, L. Jozwiak, H. Corporaal, An efficient method for energy estimation of application specific instruction-set processors, in: 16th Euromicro Conference on Digital System Design (DSD 2013), 2013.
[41] D. Gangadharan, L. Micconi, P. Pop, J. Madsen, Multi-ASIP platform synthesis for event-triggered applications with cost/performance trade-offs", in: IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2013.
[42] E. Diken, R. Corvino, L. Jozwiak, Rapid and accurate energy estimation of vector processing in VLIW ASIPs, in: EUROMICRO/IEEE Workshop on Embedded and Cyber-Physical Systems (ECyPS 2013), 2013, pp. 33–37.
[43] R. Jordans, R. Corvino, L. Jozwiak, H. Corporaal, Instruction-set architecture exploration strategies for deeply clustered VLIW ASIPs, in: EUROMICRO/IEEE Workshop on Embedded and Cyber-Physical Systems (ECyPS 2013), 2013, pp. 38–41.
[44] L. Micconi, R. Corvino, D. Gangadharan, J. Madsen, P. Pop, L. Jozwiak, Hierarchical DSE for multi-ASIP platforms, in: EUROMICRO/IEEE Workshop on Embedded and Cyber-Physical Systems (ECyPS 2013), 2013, pp. 50–53.
[45] R. Jordans, R. Corvino, L. Jozwiak, H. Corporaal, Exploring processor parallelism: estimation methods and optimization strategies, in: 6th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2013), 2013, pp. 18–23.

Dr. Lech Jóźwiak is an Associate Professor, Head of the Section of Digital Circuits and Formal Design Methods, at the Faculty of Electrical Engineering, Eindhoven University of Technology, The Netherlands. He is an author of a methodology of *quality-driven design of electronic systems*, new *information-driven approach to digital circuit synthesis*, and theories of *information relationships and measures* and *general decomposition of discrete relations* that have a considerable practical importance. He is also a creator of a number of practical products in the fields of application-specific embedded systems and EDA tools. His research interests include system, circuit, information and design theories and technologies, decision theory, artificial intelligence, embedded and high-performance systems, re-configurable and (massively) parallel computing, dependable computing, multi-objective circuit and system optimization, and system analysis and validation. He is an author of more than 170 journal and conference papers, some book chapters, and several tutorials at international conferences and summer schools. He is an Editor in Chief of the journal of "Microprocessors and Microsystems", and Editor of "Journal of Systems Architecture" and "International Journal of High Performance Systems Architecture". He is a Director of EUROMICRO; Founder and Steering Committee Chair of the EUROMICRO Symposium on Digital System Design; Advisory Committee and Organizing Committee member in the IEEE International Symposium on Quality Electronic Design; and program committee member of many other conferences. He is an advisor and consultant to the industry, Ministry of Economy and Commission of the European Communities. He recently advised the European Commission in relation to Embedded and High-performance Computing Systems for the purpose of the Framework Program 7. In 2008 he was a recipient of the Honorary Fellow Award of the International Society of Quality Electronic Design for "Outstanding Achievements and Contributions to Quality of Electronic Design". His biography is listed in "*The Roll of Honour of the Polish Science*" of the *Polish State Committee for Scientific Research* and in *Marquis* "*Who is Who in the World*" and "*Who is Who in Science and Technology*".

Menno Lindwer is Technical Program Manager at Intel IAG-VIED. He received his MSc from Twente University and his PDEng (Professional Doctorate of Engineering) from Eindhoven University in 1993. His dissertation work resulted in simulation and mapping methodology for asynchronous hardware (aka. handshake circuits or delay-insensitive circuits). He subsequently worked on database matching techniques. In 1995, he joined Philips Research, working on the design of Java and Media processors (TriMedia). In 2002, he helped starting up Silicon Hive, within the Philips Technology Incubator. He was director of product management for Silicon Hive's HiveLogic parallel processing platform. Silicon Hive was acquired by Intel's Mobile&Communications Group in February 2011. Menno's work has led to over 30 scientific articles and patent publications (seven patents currently assigned in the US). His interests include low-power application-specific processor subsystems, processor construction templates and methodologies, and spatial compilers. Menno served on the executive committee and technical program committees of the DATE conference. He co-authored several articles on Ambient Intelligence at DATE, on media processing at IEEE ISM, GSPX, on compiler technology in IJPP, and on design space exploration for multi-ASIP systems. Next to his product management role, Menno also leads Intel's activities in EU projects MADNESS and ASAM.

Rosilde Corvino is a research scientist and project manager in the Electronic Systems group at the Department of Electrical Engineering at Eindhoven Technical University, The Netherlands. She is currently a work-package leader in the European project Automatic Architecture Synthesis and Application Mapping – ASAM. In 2010, she was a post-doctoral research fellow in DaRT team at INRIA Lille Nord Europe and was involved in Gaspard2 project. She earned her PhD in 2009, from University Joseph Fourier of Grenoble, in micro- and nano-electronics. In 2005/06, she obtained a double Italian and French MSc in electronic engineer. Her research interests involve design space exploration, parallelization techniques, data transfer and storage mechanisms, high level synthesis, application specific processor design for data intensive applications. She is author of numerous research papers and a book chapter. She serves on program committees of DSD and ISQED.

Paolo Meloni is currently assistant professor at the Department of Electrical and Electronic Engineering (DIEE) in the University of Cagliari. In October 2007 he received a PhD in Electronic Engineering and Computer Science, presenting the thesis "Design and optimization techniques for VLSI network on chip architectures". His research activity is mainly focused on the development of advanced digital systems, with special emphasis on the application-driven design of multi-core on-chip architectures. He is author of a significant record of international research papers and tutor of many bachelor and master students' thesis in Electronic Engineering. He is teaching the course of Embedded Systems at University of Cagliari and is currently part of the technical board and acting as work-package leader in the research projects ASAM (www.asam-project.org) and MADNESS (www.madness-project.org).

Laura Micconi has been a PhD candidate in the Embedded System Engineering group at the Technical University of Denmark (DTU) since November 2010. Prior to this, she attended an Executive Master in Embedded System Design at the Advanced Learning and Research Institute (ALaRI) at the University of the Svizzera Italiana (USI). She got a MSc degree in Computer Science Engineering from Politecnico of Torino. Her research interests mainly lie in the field of System-level design and mapping/scheduling of application on multi-processor systems.

Jan Madsen is Professor in computer-based systems at DTU Informatics at the Technical University of Denmark. He is Deputy Head of the Department of Informatics and Head of the Section on Embedded Systems Engineering. He is senior member of IEEE and is currently serving as Vice Chair of IEEE Denmark Section. His research interests are related to design of embedded computer systems. In particular system-level modeling and analysis of multiprocessor systems, including RTOS modeling and hardware/software codesign. He is generally interested in design methodologies (including CAD tools) and implementations of embedded systems, covering areas of adaptable systems, wireless sensor networks and biochips. He has published more than 110 publications in international journals and conferences as well as co-authored 11 book chapters. He holds 1 patent and is co-founder of Biomicore. Jan Madsen is the lead delegate for Denmark in the Governing Board of the ARTEMIS Joint Undertaking, a pan-European research initiative for public–private partnership in Embedded Systems. He is on the steering committee of InfinIT, a national innovation network on ICT, where he is coordinating the strategic focus area on Embedded Systems. He is principle investigator in SYSMODEL and ASAM (both funded by ARTEMIS JU). He is participating in ProCell (NABIIT), programmable biochips, and in Wireless Sensor Network for Climate and Environmental Monitoring together with DELTA. He is participating and member of the management board for IDEA4CPS, a new Chinese–Danish Basic Research Center for theoretical foundation for Cyber-Physical Systems (Danish Basic Research Foundation). He was the leader of the Hardware Platforms and Multiprocessor System-on-Chip Cluster within the EU/IST Network-of-Excellence ArtistDesign and member of the Strategic Management Board of ArtistDesign. He is General Co-Chair of NOCS 2012, Program Chair of NORCHIP 2012, and he has been Program Chair of CODES + ISSS'11, DATE'07 and CODES'00, and General Chair of CODES'01. He is member of the Steering Committee of CODES + ISSS (ESWEEK). He is or has served on numerous program committees, including SIES, ARC, NOCS, LCTES, DAC, CODES + ISSS, ISSS, CODES, RTSS, DATE, SAC, and PARC.

Erkan Diken is a PhD student in the Electronic Systems Group of the Electrical Engineering Department at the Eindhoven University of Technology, The Netherlands. His research interests include code characterization with respect to data level parallelism (DLP) and efficient realization of DLP on hybrid VLIW/SIMD architectures. He received the MSc degree in Embedded Systems Design from the Advanced Learning and Research Institute in collaboration with ETH Zurich and Politecnico di Milano, Switzerland, in 2010, and the BSc degree in Computer Engineering from the Gebze Institute of Technology, Turkey, in 2008. He is a member of the IEEE and the HIPEAC.

Deepak Gangadharan has been a Postdoctoral Researcher at Technical University of Denmark from January 2012. He is currently working in the European project ASAM. Prior to this, he completed his PhD in Computer Science from the National University of Singapore. He has also worked in the industry developing ASIC/FPGA designs for high speed optical networks. His research interests are in Performance Analysis for System-on-Chip platforms and System Synthesis/Optimization.

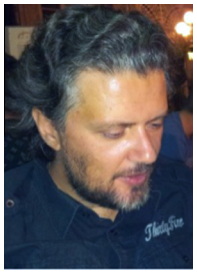
Roel Jordans is a PhD student in the Electronic Systems group at the Department of Electrical Engineering at Eindhoven University of Technology. He is currently involved in the ASAM project where his research areas include ASIP micro-architecture synthesis and instruction set synthesis. He received his Masters in Electrical Engeneering from Eindhoven University of Technology (Eindhoven, NL) in 2009.

Sebastiano Pomata joined the Department of Electrical and Electronic Engineering of University of Cagliari, as a PhD student, in March 2010. He graduated at University of Cagliari in December 2009, discussing a thesis entitled "Development of a cycle-accurate simulation infrastructure for memory hierarchies in multi-core systems". His main research interests focus on Multi-Processors computing architectures, with particular emphasis on FPGA-based technology-aware prototyping support for system-level Design Space Exploration, ASIP architectures and retargetable compilers.

Paul Pop is an associate professor at the Informatics and Mathematical Modelling Dept., Technical University of Denmark. He has received his PhD in Computer Systems from Linköping University, Sweden, in 2003. He is active in the area of analysis and design of real-time embedded systems: he is part of several national and EU projects; he has published extensively and co-authored several books. Paul Pop received the best paper award at the Design, Automation and Test in Europe Conference (DATE 2005) and at the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2011). He is the chairman of the IEEE Danish Chapter on Embedded Systems and the coordinator for the Safety-Critical Systems Interest Group in Denmark.

Giuseppe Tuveri received the BSc and MSc degrees in Electronic Engineering from University of Cagliari, Italy, in 2006 and 2009 respectively. He is part of EOLAB since March 2010, when he joined the Department of Electrical and Electronic Engineering of University of Cagliari, as a PhD student. His main research interests include embedded operating systems, system adaptivity in embedded platforms, and FPGA-based multiprocessor prototyping.

Prof. Luigi RAFFO is full professor of Electronics at the Department of Electrical and Electronic Engineering – University of Cagliari (ITALY). He received the "laurea degree" in Electronic Engineering at University of Genoa (ITALY) in 1989, the PhD degree in Electronics and Computer Science at the same university in 1994. In 1994 he joined the Department of Electrical and Electronic Engineering of University of Cagliari (ITALY) as assistant professor, in 1998 as associate professor and from 2006 as full professor of electronics. From 2006 to 2012 he was President of the Course of Studies in Biomedical Engineering. He is the Delegate for European Projects of University of Cagliari. He is coordinator of the project EU IST-248424 MADNESS – Methods for predictable Design of heterogeneous Embedded System with adaptivity and reliability Support. Current research interests are in the field of microelectronics systems for data processing in particular: VLSI design and network on chip, Microsystem for perceptual and medical applications, Biomedical equipment design. He is author of more than 100 papers in the field.



Giuseppe Notarangelo is Low Power Design and Implementation Specialist Staff Engineer at ST Microelectronics. He is HW Design Architecture Project Leader with responsibility of low-power Front-End Synthesis flow. Expert in low-power design at micro- and macro-architecture level, he is specialized in the design and implementation of power management strategies of System-on-Chip for multimedia and biomedical applications. He receives the Degree of Electronic Engineer, with Telecommunication specialization (Electromagnetic Compatibility Section) at the Polytechnic of Bari (ITALY).