ASAM : Automatic Architecture Synthesis and Application Mapping

ASAM: Automatic Architecture Synthesis and Application Mapping

Lech JOZWIAK[1], Menno LINDWER[2], Rosilde CORVINO[1], Paolo MELONI[3], Laura MICCONI[4], Jan MADSEN[4], Erkan DIKEN[1], Deepak GANGADHARAN[4], Roel JORDANS[1], Sebastiano POMATA[3], Paul POP[4], Giuseppe TUVERI[3], Luigi RAFFO[3]

[1] Technische Universiteit Eindhoven; [2] Intel; [3] Università degli Studi di Cagliari; [4] Danmarks Tekniske Universitet

***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 (Architecture Synthesis and Application Mapping) of the ARTEMIS program. The paper briefly presents the results of our analysis of the main problems to be solved 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 challenges. Finally, it introduces and briefly discusses the design-flow and its main stages proposed by the ASAM project consortium to enable an effective and efficient solution of these problems.**

***Index Terms*— embedded systems, heterogeneous multi-processor system-on-chip (MPSoC), customizable ASIPs, architecture synthesis, MPSoC and ASIP design automation;**

I. INTRODUCTION

The recent spectacular progress in semiconductor technologies has enabled implementation of increasingly complex multi-processor systems on single chips (MPSoCs), and has facilitated rapid progress in mobile and autonomous computing, as well as, wire-less and wired communication. New important opportunities have been created: the traditional applications can now be served much better, and numerous new sorts of systems became technologically feasible and economically justified. A big stimulus has been created towards development of various kinds of embedded systems. Examples of the new systems include various measurement, monitoring, control, multi-media and communication systems that can be put on or embedded in mobile, remote, poorly accessible or dangerous objects, installations, machines or devices, in home, office or hospital equipment, or even implanted in human or animal body.

However, in parallel to creation of unusual new opportunities, the spectacular advances in microelectronics and information technology introduced unusual silicon and system complexity. This complexity results in a number of new difficult issues to be addressed such as:

- ensuring high-quality and validation of the highly complex systems;
- adequately addressing the MPSoC energy crisis, and increased leakage power;
- resolving the interconnect scalability problems and on-chip communication problems;
- adequately accounting for the dominating influence of interconnects and communication on major physical system characteristics (area, speed, energy consumption);
- substantially decreasing the high system development and production costs, and long development times.

Due to the progress in semiconductor technologies and processor architectures, requirements of many applications, demanding sophisticated hardware solutions in the past, can be satisfied by software implementations executed on modern micro-, signal-, graphic- and other processors. There are however many new highly-demanding embedded applications in several fields (e.g. multimedia and entertainment, communications and networking, consumer electronics, medical and other instrumentation, monitoring and control systems, advanced machinery, automotive, military, etc.) for which the straightforward software solutions are not satisfactory. For these highly-demanding applications increasingly complex and sophisticated MPSoCs are required to perform real-time computations to extremely tight schedules, with high demands regarding energy, area, costs and development efficiency. Moreover, many of the MPSoCs are required to be highly flexible. The modern complex embedded applications typically include many various parts and numerous different algorithms involving various kinds of information processing. They are from their very nature heterogeneous. Consequently, to implement these complex and demanding heterogeneous applications effectively and efficiently, the heterogeneous application-specific multi-processor system approach should be used.

This paper focuses on mastering the 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 that have to be solved 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 challenges. Finally, it introduces and briefly discusses the design-flow and its main stages proposed by the ASAM project consortium to enable effective and efficient solution of these problems.

II. ASIP-BASED MPSOC TECHNOLOGY

The **architecture platform** targeted in the ASAM project is a ***configurable and extensible for specific applications heterogeneous multi-ASIP platform***. In particular, the project targets the MPSoC platforms of its industrial partner Intel/SiliconHiv (SH) involving generic customizable ASIPs that can be optimized for various application fields. Each ASIP is composed of an actual processor core (*core*) and core I/O (*coreio*) that together form a VLIW machine capable of executing parallel software with a single thread of control (see Fig.1). An ASIP includes a VLIW datapath controlled by a sequencer that uses status and control registers and executes programs from the local program memory. The data path contains functional units organized in several parallel scalar and/or vector issue slots 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 to the devices in coreio via master/slave interfaces. The local memories collaboration with particular issue slots enables scalar access for the scalar slots and vector or block access for the vector slots. Both SIMD and MIMD processing can be realized. The ASIPs are configurable and extensible. The numbers, kinds and parameter settings of functional units, issue slots, register files, memories, interfaces, etc. can be freely selected. Moreover, new functional units, issue slots, etc., specific for a particular application, can be developed and added.

The parameters to be explored and set in the system-level design of an ASIP-based MPSoC include: the number and types of ASIPs, the number, type and size of global memories, the scheduling and mapping of the application parts to the selected ASIPs and their data to the selected memories, and the system-level architecture and parameters of the global communication structure. The parameters to be explored and set to create an optimized ASIP architecture 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 and data width, and local memories, the architecture and the parameters of the local communication structure, etc.

Several such different ASIPs, each customized for a particular part of a complex application, can be interconnected with global memories and possible hardware accelerators and other digital or analog sub-systems using a configurable heterogeneous interconnection network, and implemented on one chip. Several such powerful 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 400-600MHz, these ASIPs can deliver more than 1 Tops/s, with power consumption far below the upper limit of mobile devices. The ASIP-based customizable for specific applications heterogeneous MPSoC platform enables efficient application-specific exploitation of various

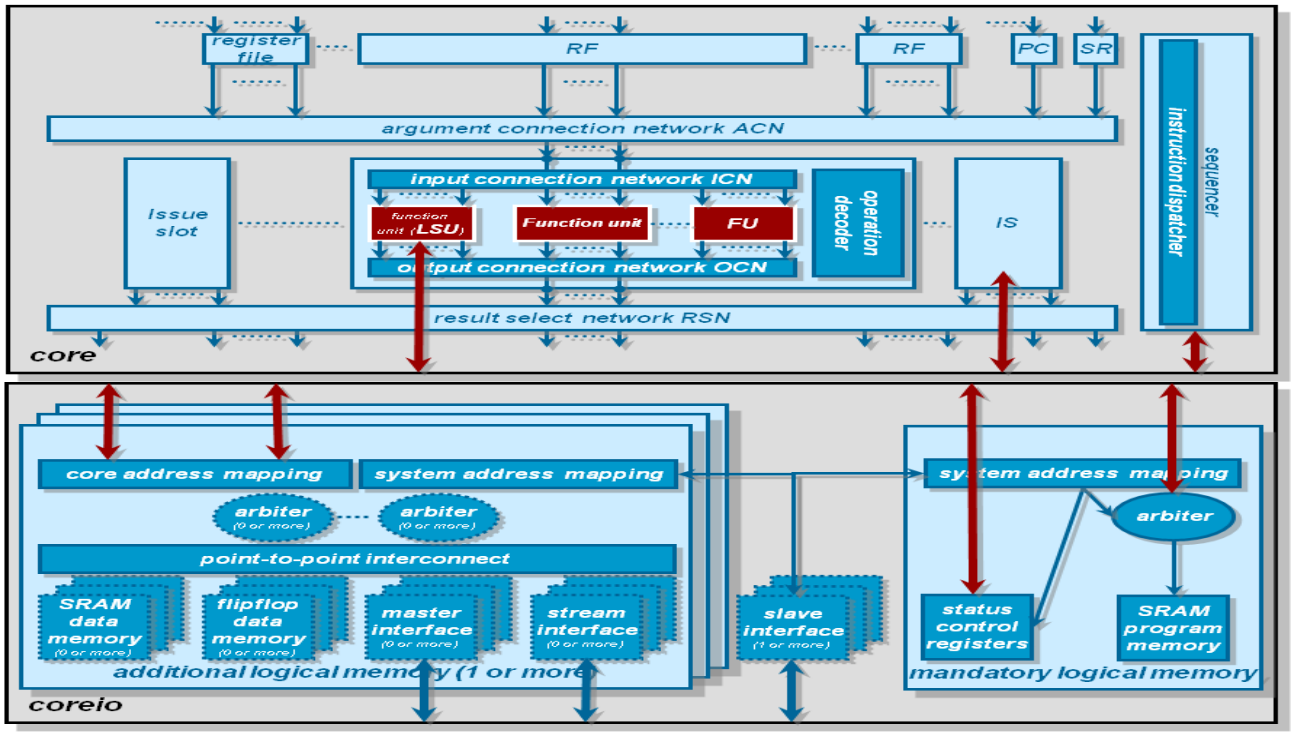

Figure 1 Generic ASIP architecture of the targeted MPSoC platform.

kinds of parallelism: the multiple ASIPs serve the coarse-grain parallelism exploitation at the task level, while the ASIP's parallel issue slots, custom instructions and their hardware serve the fine-grained acceleration.

The adaptable ASIP-based MPSoC technology addresses several fundamental development challenges of electronic systems for modern highly-demanding embedded applications:
- it is able to deliver a 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.

It is especially suitable for complex applications involving different kinds of processing with various kinds of parallelism and being highly demanding regarding physical and economic characteristics. Provided an effective and efficient highly automated customization technology will become available (what is targeted by the ASAM project), it will become possible to build the adaptable ASIP-based MPSoCs at substantially lower costs and with shorter times to market than the hardwired ASICs or programmable MPSoCs based on custom processors built from scratch.

III. Issues and Challenges of the ASIP-based MPSoC Development

The systemic realizations of complex and highly demanding applications, for which the customizable multi-ASIP MPSoC technology is especially suitable, demand the performance and energy usage levels comparable to those of ASICs, while remaining small-size and cost-effective. Satisfaction of these stringent and often conflicting application demands requires construction of highly-optimized hardware architectures and corresponding software structures mapped on the architectures. This can be achieved through an efficient exploitation of different kinds of parallelism involved in these applications, implementation of critical parts of their information processing in application-specific hardware, as well as, efficient trade-off exploitation among various design characteristics, and between solutions considered at different design levels and for different system parts.

Unfortunately, in the traditional embedded system development approaches, the major development activities are usually largely disjoint and performed by different teams using different supporting tools. This leads to inefficiencies, errors, and costly reiterations in the design process. Moreover, the traditional algorithm and software development approaches require an existing and stable computation platform (HW platform, compilers etc.), while for the modern embedded systems as MPSoCs based on adaptable ASIPs the hardware and software architectures have to be application-specific, and must be developed largely in parallel. Unfortunately, the efficiency of the required parallel HW and SW development is much too low with the currently available development technology due to lack of effective automated methods of industrial strength for many MPSoC design problems, and weak interoperability of the HW/SW architecture design, and hardware synthesis tools. The inefficiencies identified above result in a substantially lower than attainable quality of the resulting systems, much longer than necessary development time, and much higher development costs.

Although some application analysis, restructuring and compilation tools and ASIP configuration frameworks exist (see Section IV), most of them miss the design decision layer. The ASIP configuration semantics provide some IP libraries, tools to generate and analyze architecture, and tools to generate corresponding hardware descriptions, but assume customization decisions taken by a human designer and denoted in an appropriate way. However, they miss the tools for automated customization decision making, and the collaboration among the different tools. Also, most of them are devoted to a single ASIP customization. However, many modern applications include many various parts and numerous different algorithms involving various kinds of information processing with various kinds of parallelism (task-level, loop-level, operation/instruction-level, and data parallelism). They are from their very nature complex and heterogeneous and require heterogeneous application-specific multi-processor system approach. In a customizable multi-ASIP MPSoC its various ASIPs have to be customized together, and in a strict relation to the selection of the number of ASIPs, as well as to the scheduling and mapping of the application's required computations on the particular ASIPs. The MPSoC macro- and the micro-architectures, at multi-ASIP system level and at the single ASIP processor level, are strictly interrelated. Important trade-offs have to be resolved regarding the granularity of individual processing cells, and between the amount of parallelism and resources at each of the two architecture levels. Moreover, at both architecture levels, the optimized parallel software structures have to be implemented on the corresponding optimized for them parallel hardware structures.

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 organization 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 modern applications (e.g. various communication, multimedia, networking or encryption applications) require hardware implementation of algorithms that involve complex interrelationships between the data and computing operations and require complex memory accesses and complex communication between the memories and computing units in the related hardware. 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, which has to be achieved through an adequate memory and communication structure design. Moreover, for applications of this kind, the memory and communication structure design, and the micro-architecture design for computing units cannot be performed independently, because they substantially influence each other.

Finally, the existing methods and tools of custom instruction-set construction or extension are devoted to a single processor, usually a simple RISC processor with one issue slot,

and not to several VLIW processors, each with several issue slots.

In consequence, optimization of the performance/resources trade-off required by a particular application can only be achieved through a careful construction of an adequate application-specific macro-/micro-architecture combination. The aim here is thus to find an adequate balance between the number of parallel processors, the complexity of the inter-processor communication, and the intra-processor parallelism and complexity. To achieve this aim, based on the application analysis and restructuring several promising macro-architecture/micro-architecture combinations have to be automatically constructed and evaluated in an iterative process, and finally, the best of them has to be selected for an actual realization.

IV. CONTRIBUTION AND RELATED WORKS

The main aim of the ASAM project is to considerably enhance the design efficiency of the ASIP-based MPSoCs for highly demanding applications, while substantially improving the result quality. This aim is being realized through development of a coherent system-level design-space exploration and synthesis flow, and automatic analysis, synthesis and rapid prototyping environment implementing the flow that will provide efficient exploration of the architecture and application mapping alternatives and tradeoffs. Based on the application, computing platform and parametric requirement analysis, the ASAM flow will efficiently partition a given complex application and select the most appropriate ASIP types for different application parts, creating this way the MPSoC macro-architecture; subsequently, 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.

System design is actually the definition of the required system quality, in the sense of providing a satisfactory answer to the questions: "What new (or modified) quality is required?" and "How can it be achieved?". Therefore the research of the ASAM project builds on the methodology of quality-driven model-based system design proposed in [1].
Moreover, the ASAM project builds on the platform-based design of heterogeneous multi-processor embedded systems [1], [2], ASIP design methods [3–9], hardware compilation techniques [2], and software analysis, re-structuring and compilation techniques [2], [10].

With respect to the MPSoC macro-architecture synthesis, the project exploits the quality-driven model-based system-level design exploration and architecture synthesis approach [2], [11], and modeling, emulation, estimation and design exploration concepts [2], [11–13], earlier developed by some of the project partners. The new macro-architecture DSE methodology and supporting tools will enable an effective and efficient reuse of a generic architecture platform, modeling of platform in the form of an abstract architecture template, generic architecture template instantiation, abstract requirement modeling, and application process scheduling and mapping on the generic architecture template instance when observing the constraints objectives and tradeoffs of the requirement model.

The ASIP micro-architecture exploration and synthesis will perform the actual ASIP construction for a given application part, when re-using the existing application-class specific generic customizable ASIP IP cores. It is perhaps the most difficult task in the design of a system based on customizable ASIPs. It has to reveal the parallel computation and data structures of a given application part that can be executed/processed concurrently on separated hardware clusters and local register files, and construct the corresponding instruction set hardware clusters, register files, memories, and other processor parts, when adequately accounting for hardware reuse and satisfying the application's constraints, design objectives and trade-offs. The existing commercial and academic developments in this field do not provide adequate support for this critical part of the designers' work (see e.g. [2]). Therefore, new methods and prototype tools are being developed in the scope of the project to much better reflect the actual problems to be solved and better address their required solutions.

As explained in the previous section, there are very strong interrelations between the macro- and micro-architecture synthesis. 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 sub-tasks. 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 execute 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 subsystem implemented with each ASIP core. This way the micro-/macro-architecture trade-off exploitation is enabled, and after several iterations an optimized MPSoC architecture is constructed.

A complete MPSoC architecture involves of course the adequately selected and instantiated ASIPs, 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 will be 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 [14–16].

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. They will also implement 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 effectively exploit the trade-offs among the different design aspects. In 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 the complex design problem into a hierarchical network 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, and collaborate with each other in a well-defined coherent way to together deliver a high-quality application-specific HW/SW system design.

To 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 partial problems considered by related research and their proposed solutions are much simpler, and the proposed partial solutions are usually not directly useful in the much more complex context of ASAM.

As stated in [17], ASIP auto-customization design methods can be sub-divided 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 Tensilica Extensa Configurable Core [8], ARC Configurable Cores [7], etc. The specification-based methods provide the possibility to entirely describe a processor based on an abstract model that only defines the general design rules, when using an Architecture Description Languages (ADLs) that allows describing the relevant (application-specific) aspects of the ASIP architecture at several abstraction levels. Specification-based methods and corresponding ADLs include EXPRESSION [6] and its EXPRESSION ADL [18], CoWare Processor Designer [5] using LISA ADL [19], Target Compiler Technology [4] using nML ADL [20] and Intel/Silicon Hive [3] using TIM language for the ASIP architecture description and HSD language for the MPSoC system-level architecture description (e.g. global communication, processors synchronization, etc.). Most of these methods and related tools target the design of systems involving only a single ASIP.

Although the existing ASIP customization frameworks involve a rich set of 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. That is why such an effective and efficient highly automated DSE and decision support is targeted by the ASAM project, additionally not limited to a single ASIP, but for the multi-ASIP systems.

Other related works focus on the application code transformations [21] 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) [22] and related hardware extension. Some of them try to explore and exploit the effect of loop transformations [23], 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, most of them are very simplistic. The construction or extension is based on some proxy attributes and optimization objectives, on a kind of simplified application analysis, and performed without actually accounting for the related data-path and control-path implementation. For instance, it is usually performed without accounting for the effects and trade-offs of hardware sharing by various instructions. In result, the proxy formulations of the custom instruction set construction problems and their suggested solutions do not reflect well the actual problems to be solved and their required solutions.

Many published research results [1], [11], [24] and system design frameworks, e.g. Metropolis [25], Daedalus etc., target the heterogeneous MPSoC design, but none of them addresses the adaptable ASIP-based MPSoCs design being the target of ASAM, with all its complexity and difficult to solve issues as described in this paper. Nevertheless, some of the valuable ideas and general methodologies developed in this research can and will be reused for ASAM purposes. In particular, ASAM project builds on the methodology of quality-driven model-based system design proposed by Jóźwiak [1].

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 them focus on the processing unit design and application mapping, and underestimate the importance of the memory and communication architecture design for the demanding modern application implementation. Although [17] 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), in ASAM on the top of this we explore the effect of loop

transformations on the Data Transfer and Storage Mechanism [21] 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.

Further extensive discussion of related research can be found in the overview papers [2] and [24].

V. Design Flow

A simplified view of the **ASAM design flow** is presented in Figure 1. The flow involves **four main stages** corresponding to main design issues and abstraction layers:

- system DSE (macro-architecture),
- ASIP DSE (micro-architecture),
- GC&M DSE, and
- HW/SW synthesis and rapid prototyping.

Each of these stages communicates and collaborates with the remaining stages directly or through the system DSE. All together they realize the quality-driven evolutionary system engineering process briefly described above, stepwise transforming the initial high-level application specification (given by application C code, parametric requirements and representative input stimuli) and generic platform specification (representing structural constraints and given by templates of generic ASIP-based MPSoC and its modules) into a HW/SW ASIP-based MPSoC prototype or the final ASIP-based MPSoC design.

The ASAM architecture design flow implements the concept of **service-oriented EDA system**. Each of its main stages and their parts can be requested for and provides some services for some other stages or parts. The whole collaboration among the stages and their parts is organized as requests and answers to requests for specific services. This enables clear organization of the system, and specifically, of the collaboration among its stages and parts, as well as, results in a high flexibility in relation to the EDA system extension or modification through adding new or modifying services.

The stages of the ASAM design flow also communicate and collaborate with the basic SH design flow which provides several services to them, such as IP library, HW and SW generation and compilation, HW and HW/SW simulation etc. In particular, each stage of the flow can get from the IP library an existing IP, use it (e.g. for simulation or emulation) or customize it according to requirements and insert the customized IP into the IP library. The flow execution is originally determined by its primary inputs (i.e. application behavior specification, parametric and structural requirements, and representative input stimuli) and the user control inputs (not represented in

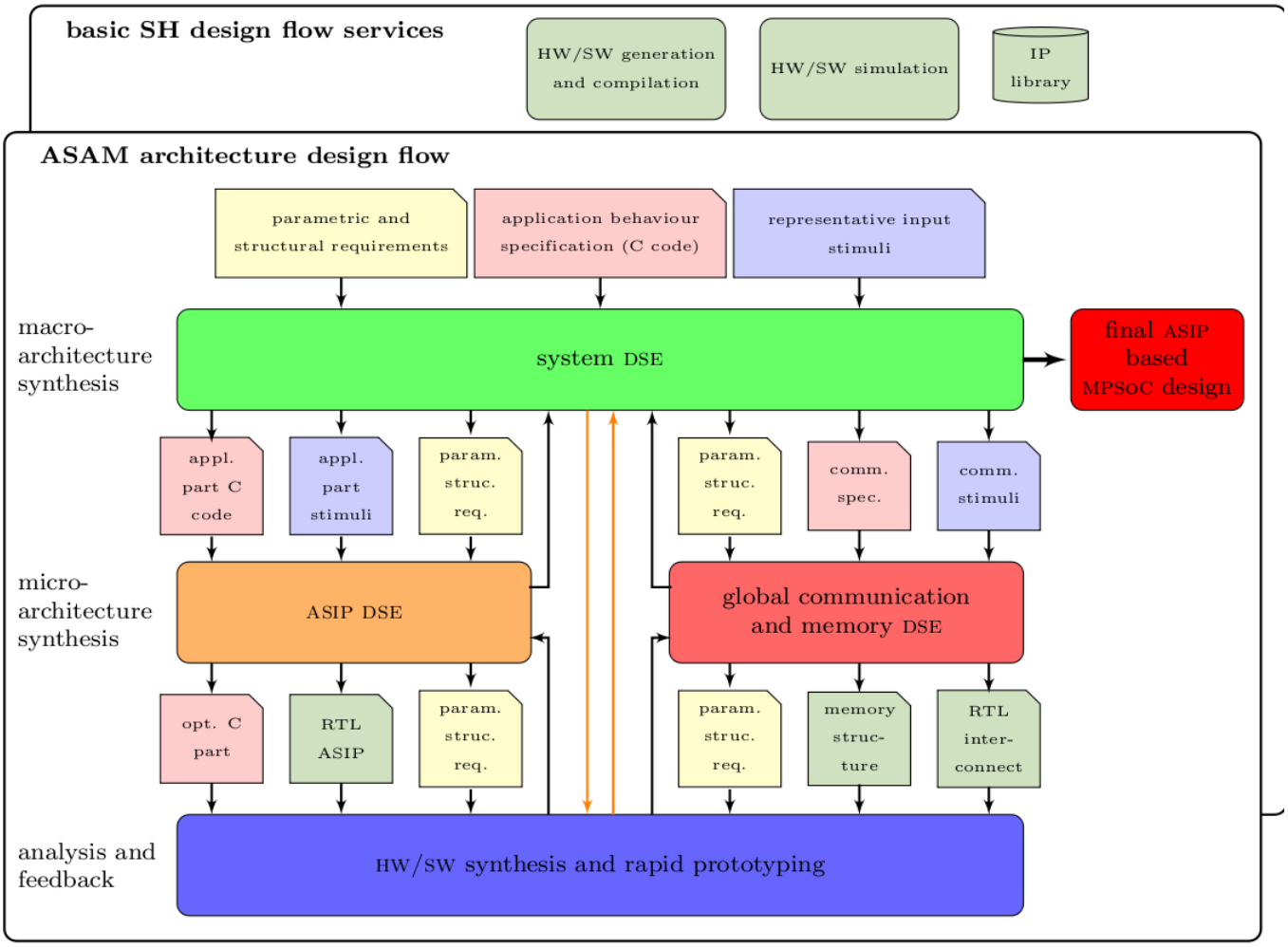

Figure 2 ASAM design flow.

Fig. 2). However with the progress of execution it is more and more influenced by the results of previous explorations. This is necessary a. o. to ensure the exploration effectiveness and efficiency, and in particular, to avoid lengthy or endless reiterations.

System DSE takes as its inputs: an application C-code, parametric and structural requirements, and representative testbenches and stimuli for the application behavior, simulation or emulation in the rapid prototyping environment. It is responsible for the total design of the multi-ASIP MPSoC. It directly performs the system-level DSE for the multi-ASIP MPSoC and defines its structure composed of several ASIPs (which include their local memories and communication), global memories and global communication among the ASIPs. In performing this task, it asks for specific services from the ASIP DSE and the GC&M DSE. These services can range from coarse parameter estimation to an optimized design of subsystem. It also asks for services from the rapid prototyping (light color arrows in Fig. 2). The request can be to perform simulation or emulation of a complete proposed HW/SW ASIP-based MPSoC or of its single parts. When asking for a service, the system DSE specifies the kind of service requested (not represented in Fig. 2), as well as, the behavioral, parametric and structural requirements, and input stimuli related to a given design part and the service requested. Through directly performing the system-level DSE and using the analysis and synthesis services of ASIP DSE, GC&M DSE, rapid prototyping and basic SH design flow, the system DSE produces as its output a final optimized ASIP-based MPSoC design.

ASIP DSE takes as its inputs the kind of service requested by the System DSE, as well as, the behavioral, parametric and structural requirements, and input stimuli related to a given design part defined by one ASIP and application part mapped on it by the System DSE. It is responsible for the total design of a single HW/SW ASIP-based part of the MPSoC. It directly performs the ASIP DSE which consists of the actual coherent HW/SW co-design through simultaneous co-tuning of the application software part assigned to a given ASIP and ASIP architecture. In performing this task, it asks for specific services from the HW and SW synthesis and rapid prototyping, as well as, from the SH design flow. In satisfying a service request it provides its outputs (ranging from a coarse parameter estimation to a design of a complete optimized ASIP-based HW/SW sub-system) to the requesting System DSE.

GC&M (GC&M) DSE accepts analogous inputs, performs analogous tasks and collaborates analogously with its surroundings as the ASIP DSE, but does it in relation to the global memories and global communication sub-system instead of ASIPs.

HW and SW synthesis and rapid prototyping accept as their input service requests from the System DSE, ASIP DSE and GC&M DSE, as well as, the architecture description language (ADL)-based specification of the designed MPSoCs or their parts, and corresponding application C-code. From the ADL descriptions the **HW synthesis** automatically generates corresponding refined hardware description language (HDL) based hardware specification and **SW synthesis** generates the corresponding software structures and compiles the 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 use specific services of the SH design flow. In answer to the service requests, the HW and SW synthesis and rapid prototyping provides the results of the HW or SW synthesis or prototyping results to the requesting flow stage.

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

VI. Main Stages of the ASAM Flow

A. System Level DSE

The System Level (SL) DSE is in charge of developing the MPSoC macro-architecture, deciding the number and kind of ASIPs, the global memories, their interconnections and mapping of different application tasks into ASIPs and of their data into memories. The original application code is modeled for the SL DSE, as a task graph (TG): the code is partitioned into tasks (*T*) for extracting the **task level and inter-task (pipeline) parallelism**. The partitioned application is generated by commercial tools [26]. The communication between the tasks is modeled through Message Tasks (*MT*), which represent the amount of data to be exchanged. The SL DSE performs the mapping and scheduling of application and data access through the mapping and scheduling of the Tasks and Message Tasks on ASIPs and interconnection resources, respectively. Moreover, the SL DSE has the role of collecting the partial results from the different DSE entities (ASIP DSE and GC&M DSE) and combining them together to verify the performances of the entire system.

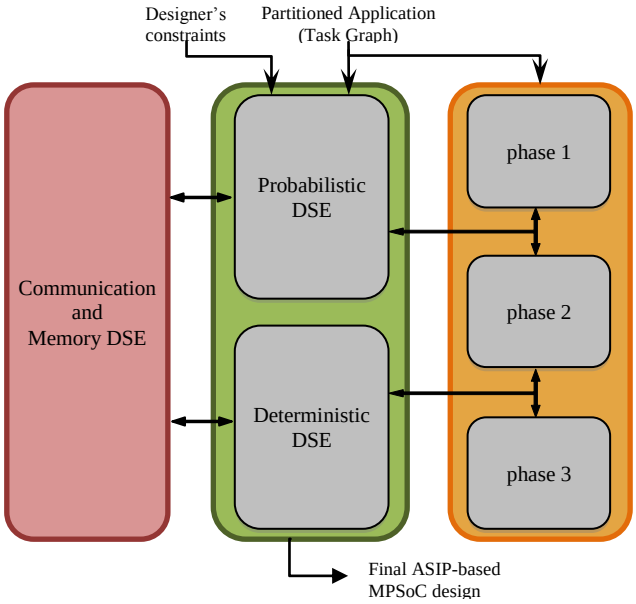

Figure 3 System Level DSE

An initial instance of the computing platform model is defined according to the total number of tasks in the application TG, which defines the upper bound for the number of ASIPs. The initial platform model assumes the use of a bus for the global communication (this limitation is later relaxed during the GC&M DSE). The SL DSE can be divided into two main phases: Probabilistic and Deterministic DSE (cf. Figure 3).

Probabilistic DSE: This is the most challenging phase of the SL DSE; it has to solve a kind of *chicken-and-egg* problem in which an adequate application mapping and scheduling need to be established without any initial knowledge of the macro and micro-architecture of the multi-ASIP platform. In fact, the ASIP-level DSE needs information about mapping and stimuli/parameters for the definition of the micro-architecture of each single ASIP. At the same time, the System-level DSE needs information on single ASIP micro-architectures to get a reasonably precise performance estimation of an application mapping/scheduling solution. The different micro-architectures that can be identified during the micro-level ASIP design are called **ASIP configurations**. The number of possible ASIP configurations is huge and challenging to identify. Therefore, it is not feasible to perform a macro-level DSE through taking into account all the possible ASIP configurations. This issue is solved through the implementation of a Probabilistic Estimation Method. The performances that can be obtained by an application part (e.g. a single task initially) executing on **different configurations of the same ASIP** are modeled through a stochastic variable and its probability distribution. Depending on the performances (e.g., execution time, energy consumption) that have to be evaluated during the DSE, corresponding stochastic variables have to be defined. For example, we use the Worst Case Execution Time (*WCET*) as one of the stochastic variables to estimate the performance. The cumulative distribution function (CDF) of the WCET is defined using two points ([*lWCET*,*uWCET*]) that represent the lower (lWCET) and upper (uWCET) boundaries of the performances for the execution of a given application part on all the possible configurations of an ASIP. These values are calculated by analyzing each task (application part): the WCET is computed for a hypothetical sequential execution on a simple sequential ASIP instance (u*WCET*, the slowest execution) and for a parallel execution on a parallel ASIP instance with the non-constrained resources (l*WCET*, the fastest execution possible). Such values are provided by the first phase of the ASIP level DSE. A CDF is drawn between such two values to model the distribution of a given performance parameter for all possible configurations (whose number and individual performances are yet unknown). A CDF is built for each application part mapped to an ASIP and for each of the performance parameters that have to be evaluated. A CDF is also built for each of the Message Tasks and their mapping on the interconnection network. Different mapping solutions can then be proposed and analyzed: the CDFs of the different tasks and message tasks are then combined together according to the mapping and execution order of the tasks, and a CDF of the total task graph is computed (for each performance parameter), taking into account both the task level and the pipeline parallelism. Each mapping solution is then evaluated according to the design constraints and objectives (e.g. execution deadline in case of the WCET). The mapping solutions with the highest probability of meeting the constraints are selected for further consideration.

The mapping results obtained in such a way are then passed to the ASIP level DSE and to the GC&M DSE that can respectively start the actual synthesis of each single ASIP, according to the mapped tasks. The (partial) micro-architecture synthesis results are then returned to the SL DSE that can repeat its DSE on the basis of this more precise information received. This phase is repeated until the task performances are modeled as stochastic variables, that is until the design of all ASIPs memories and their communications have been sufficiently decided, i.e. a bounded subset of the most promising specific configurations has been identified.

Deterministic DSE is in charge of performing a design exploration similar to the one of the previous phase, but with much more precise information on the computing platform and its parts. This phase works with concrete ASIP, memory and communication configurations and verifies the actual performances of the global ASIP-based system. The interaction between the different stages and phases of the design flow is repeated until the entire design constraints are satisfied or it becomes clear that it is impossible to satisfy them given the available resources.

B. ASIP level DSE

The micro-level ASIP DSE, presented in Figure 4, aims at exploring and selecting optimized parallel software structure and the corresponding ASIP architecture for an application part assigned to a single ASIP by the system-level DSE. Due to the high number of application restructuring and ASIP customization parameters, a synthesis method with a reduced exploration complexity is necessary. To perform an early pruning of the design space, **the ASIP DSE process is subdivided into three main phases:**

Phase1, which performs application analysis and coarse ASIP characterization,

Phase2, which performs application parallelization and mapping, as well as, design of top-level ASIP architecture involving parallel processing, communication and storage, and

Phase3, which performs instruction set synthesis and refined application restructuring.

The services of these three phases are provided to the SL DSE flow or directly to the end-user. They can be executed separately or in combination, depending on the level of analysis and synthesis details required from the ASIP DSE. A given phase can also ask the successive phases for services.

Phase1: application analysis and coarse ASIP characterization provides information on the upper bound and lower bound of the worst-case execution time (WCET) of an application part. The WCET values are used by the SL DSE for a rapid initial evaluation of the proposed application partitioning and by the following phases of the ASIP level DSE to infer the design requirements of the application sub-parts. The application analysis also detects the type of processing (vector/scalar) and the hot-spots (e.g. bottlenecks or frequently executed sub-parts) of the analyzed application part. This information can be used to decide the kind and amount of effort devoted to the optimization of the ASIP architecture for individual sub-parts of the application.

Phase2: application parallelization and mapping and top – level ASIP architecture synthesis aims at analyzing the application part with respect to the medium-level parallelism concerns (e.g. data dependencies, available loop-level and data parallelism, etc) and at proposing optimized parallel software structures and corresponding coarse hardware architectures for the ASIP processing issue slots, internal memories and communication. For instance, an application transformation such as loop fusion, which reduces the loop control overhead and the application run-time, corresponds to the instantiation in hardware of several parallel issue slots with their own associated register files storing locally accessed data. Several other data-oriented application transformations, mainly loop transformations as tiling, vectorization, unrolling etc., are explored in this phase. In correspondence to them, the parallelism and data related hardware design decisions are taken, in relation to the number, kind (vector/scalar) and size of ASIP issue slots, the number, kind and size of internal memories and register files, the mapping of processing to issue slots and of data to memories, as well as, the ASIP local communication structure. This is usually the most decisive step in the whole ASIP architecture design process, as the parallelism exploration, scheduling and mapping decisions influence the ASIP main parameters to a high degree. The result of this phase is a one or several promising coarse design(s) of the ASIP with a refined communication and storage micro-architecture and a general instruction set partitioned over the ASIP parallel issue slots. Phase 2 is further explained in [27–29].

Phase3: precise issue slot synthesis and application restructuring mainly involves the identification and selection of an application specific instruction set for each of the ASIP issue slots, and the related data-path and controller optimization. The identification of instruction set candidates corresponds to a partitioning of the DFG of an application basic block in several sub-graphs that cover the whole original DFG and match some instructions in an instruction library. The selection of the best candidates is performed with respect to design optimization criteria such as area occupancy, execution time or throughput, and energy consumption. If after the initial selection and related data path synthesis the selected instruction set does not meet the design requirements, an instruction set extension can be proposed. In the extension process, sub-graphs corresponding to some hotspots of the original DFG are identified, realized in hardware and included in the instruction library, before performing a new phase of the instruction set identification and selection using the extended instruction library.

After finishing the synthesis of a single ASIP based subsystem, 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.

C. Global memory and communication DSE

The main aim of this step is the exploration and optimization of the global interconnection 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 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 is selected and serves as a feedback to the upper level, as depicted in Figure 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 D. 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:

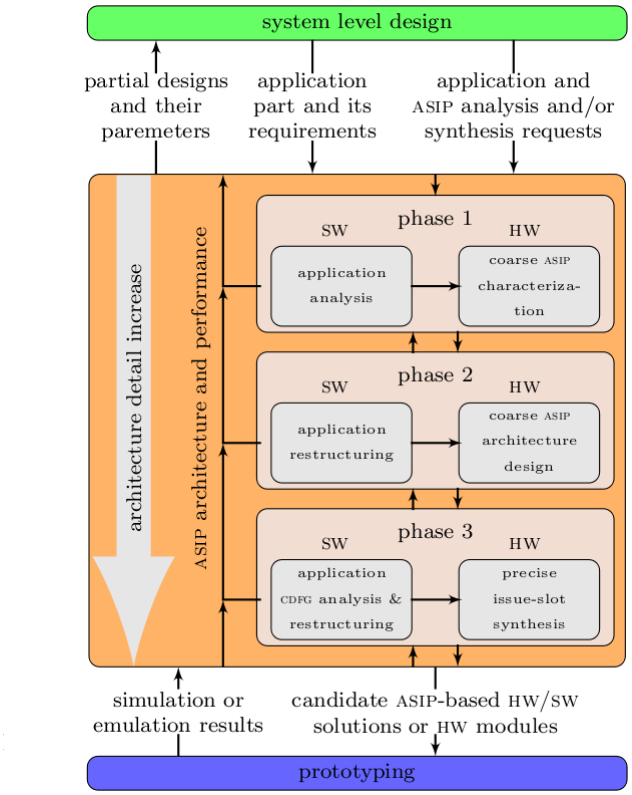

Figure 4 ASIP level DSE

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

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

D. Rapid prototyping platform

The main role of the prototyping infrastructure of ASAM 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. The developed platform is composed of several ASAM sub-tools that interact with each other and of several commercial tools. It takes different inputs being micro- and macro-architectural specifications that drive the HDL generation step. The macro-architectural description is provided as input to describe the top-level view of the system, by means of a proprietary format. The micro-architectural description is related to both the topology of interconnect and memories, and the processor architectural configuration (or several configurations, in case of multi-core heterogeneous systems). 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 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 system 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 automatic instantiation of hardware counters inside the processor/system RTL code, for computing 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. a 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.

Furthermore a retargeted compilation tool-chain with complete awareness of the ASIP processors/system specifications, and thus able to correctly exploit 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. This cooperation allows executing the software on the FPGA board, and then to collect the evaluation data from a host workstation, accessing dedicated shared-memory structures that are instantiated in the MATLAB workspace. The metrics collected from the FPGA, can be translated into technology relevant figures, enabling a pre-estimations of the quality-of-results achievable with a prospective ASIC implementation. In the prototyping tool-flow this objective is achieved with the usage of accurate area/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 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 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 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. 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 including a number of different configurations to be explored, and implemented on the FPGA. Consequently, through the usage of custom-developed tools, the tool-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. A complete and detailed description of these methodologies can be found in [30].

VII. Conclusion

In the former sections an overview has been presented of the research results of the European project. Due to a limited length of the paper, we had to focus on only the main unsolved problems and only the main solution concepts. Many more research results have been produced by the ASAM consortium than these once briefly discussed in this paper. They include results related to such important problems as energy consumption management, application analysis and restructuring or actual automatic coherent SW and HW co-development. More information on the research results from ASAM can be found in several already published papers with links to them at the ASAM home-page (http://www.asam-project.org/), a few more specific papers in this conference [27–29], [31], [32], and several papers under preparation that will be published soon.

VIII. ACKNOWLEDGEMENT

The work on 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*, vol. 47, no. 3-4, pp. 201-224, Apr. 2001.

[2] L. Jóźwiak, N. Nedjah, and M. Figueroa, "Modern development methods and tools for embedded reconfigurable systems: A survey," *Integration, the VLSI Journal*, vol. 43, no. 1, pp. 1-33, Jan. 2010.

[3] SiliconHive and Intel, "Silicon Hive." [Online]. Available: http://www.siliconhive.com.

[4] Target, "Target Compiler Technology." [Online]. Available: http://www.retarget.com/index.php.

[5] CoWare and (Synopsys), "CoWare Processor Design." [Online]. Available: http://www.synopsys.com/Systems/BlockDesign/ProcessorDev/Pages/default.aspx.

[6] University of California Irvine, "EXPRESSION." [Online]. Available: http://www.ics.uci.edu/~express/.

[7] ARC-Inc. and (Synopsys), "ARC Configurable Cores." [Online]. Available: http://www.synopsys.com/IP/ProcessorIP/Pages/default.aspx.

[8] Tensilica, "Tensilica - Xtensa Customizable Processors." [Online]. Available: http://www.tensilica.com/products/xtensa-customizable.htm.

[9] O. Schliebusch, H. Meyr, and R. Leupers, *Optimized ASIP Synthesis from Architecture Description Language Models*. Springer, 2010, p. 207.

[10] R. Leupers and P. Marwedel, *Retargetable Compiler Technology for Embedded Systems - Tools and Applications*. Springer, 2001.

[11] L. Jozwiak and S. Ong, "Quality-driven model-based architecture synthesis for real-time embedded SoCs," *Journal of Systems Architecture*, vol. 54, no. 3-4, pp. 349-368, Mar. 2008.

[12] S. Mahadevan, F. Angiolini, J. Sparso, L. Benini, and J. Madsen, "A Reactive and Cycle-True IP Emulator for MPSoC Exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 109-122, Jan. 2008.

[13] A. S. Tranberg-Hansen and 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.

[14] S. Murali et al., "Synthesis of Predictable Networks-on-Chip-Based Interconnect Architectures for Chip Multiprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 8, pp. 869-880, Aug. 2007.

[15] F. Angiolini, P. Meloni, S. M. Carta, L. Raffo, and 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*, vol. 26, no. 3, pp. 421-434, Mar. 2007.

[16] L. Jóźwiak and Y. Jan, "Communication and Memory Architecture Design of Application-specific High-end Multi-processors," *VLSI Design*.

[17] K. Karuri and R. Leupers, *Application Analysis Tools for ASIP Design: Application Profiling and Instruction-set Customization*. Springer, 2011, p. 232.

[18] a. Halambi, P. Grun, V. Ganesh, A. Khare, N. Dutt, and A. Nicolau, "EXPRESSION: A language for architecture exploration through compiler/simulator retargetability," in *date*, 1999, p. 485.

[19] A. Hoffmann, H. Meyr, and R. Leupers, *Architecture Exploration for Embedded Processors with LISA*. Springer, 2002, p. 244.

[20] A. Fauth, J. Van Praet, and M. Freericks, "Describing instruction set processors using nML," in *Proceedings the European Design and Test Conference. ED&TC 1995*, pp. 503-507.

[21] F. Catthoor, K. Danckaert, S. Wuytack, and N. D. Dutt, "Code transformations for data transfer and storage exploration preprocessing in multimedia processors," *IEEE Design & Test of Computers*, vol. 18, no. 3, pp. 70-82, 2001.

[22] K. Martin, C. Wolinski, K. Kuchcinski, A. Floch, and F. Charot, "Constraint-Driven Identification of Application Specific Instructions in the DURASE System," pp. 194-203, 2009.

[23] A. C. Murray, R. V. Bennett, B. Franke, and N. Topham, "Code transformation and instruction set extension," *ACM Transactions on Embedded Computing Systems*, vol. 8, no. 4, pp. 1-31, Jul. 2009.

[24] D. Densmore and R. Passerone, "A Platform-Based Taxonomy for ESL Design," *IEEE Design & Test of Computers*, vol. 23, no. 5, pp. 359-374, May 2006.

[25] UCBerkeley, "Metropolis: Design Environment for Heterogeneous Systems." [Online]. Available: http://embedded.eecs.berkeley.edu/metropolis/platform.html.

[26] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, and E. Deprettere, "System Design Using Kahn Process Networks: The Compaan/Laura Approach," p. 10340, Feb. 2004.

[27] R. Corvino, A. Gamatié, M. Geilen, and L. Jozwiak, "Design space exploration in application-specific hardware synthesis for multiple communicating nested loops.," in *SAMOS 2012*, 2012.

[28] R. Corvino, E. Diken, A. Gamatié, and L. Jozwiak, "Transformation based exploration of data parallel architecture for customizable hardware: A JPEG encoder case study.," in *DSD 2012*, 2012.

[29] R. Corvino and A. Gamatié, "Abstract clocks for the DSE of data intensive applications on MPSoCs," in *in ISPA 2012 - 4th IEEE International Workshop on Multicore and Multithreaded Architectures and Algorithms*, 2012.

[30] P. Meloni, S. Pomata, G. Tuveri, M. Lindwer, and 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 - in press)*, 2012.

[31] R. Jordans, R. Corvino, and L. Jozwiak, "Algorithm Parallelism Estimation for Constraining Instruction-Set Synthesis for VLIW Processors," in *DSD 2012*, 2012.

[32] E. Diken, R. Jordans, R. Corvino, and L. Jozwiak, "Application Analysis Driven ASIP-based System Synthesis for ECG," in *Embedded World Conference*, 2011.