Automatic Generation of S-LAM Descriptions from UML/MARTE for the DSE of Massively Parallel Embedded Systems

Manel Ammar, Mouna Baklouti, Maxime Pelcat, Karol Desnos and Mohamed Abid

Abstract Massively Parallel Multi-Processors System-on-Chip (MP2SoC) architectures require efficient programming models and tools to deal with the massive parallelism present within the architecture. In this paper, we propose a tool which automates the generation of the System-Level Architecture Model (S-LAM) from a Unified Modeling Language-based (UML) model annotated with the Modeling and Analysis of Real-Time and Embedded Systems (MARTE) profile. The S-LAM-based description of the MP2SoC architecture is conformed to the IP-XACT standard. The integration of our generator within a co-design framework provides the specification of the whole MP2SoC system using UML and MARTE. Then, gradual refinements allow the execution of a rapid prototyping process.

1 Introduction

Recent trends in High-Performance Computing (HPC) architectures show that, due to the end of processor frequency scaling, performance increases are mostly gained by employing more processor cores [1]. This trend draws attention to the effectiveness of Massively Parallel Multi-Processors System-on-Chip (MP2SoC) architectures in the HPC domain. Designers of high performance MP2SoC are facing many critical design challenges including:

M. Ammar (✉) · M. Baklouti · M. Abid
CES Laboratory, National Engineering School of Sfax, Sfax, Tunisia
e-mail: manel.ammar@ceslab.org

M. Pelcat · K. Desnos
IETR, INSA Rennes, CNRS UMR 6164, UEB, Rennes, France
e-mail: mpelcat@insa-rennes.fr

R. Lee (ed.), *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2015*, Studies in Computational Intelligence 612,
DOI 10.1007/978-3-319-23509-7_14

1.1 Raising the Level of Abstraction of the Specification

The raising complexity of embedded systems creates a need for intensive specification task. In the history of design flows, changes in design productivity were always related to raising the level of abstraction in design entry. In the 1970s, the highest level of abstraction was a transistor schematic. 10 years later, design entry had moved up from transistors to gates. Then, with the appearance of Hardware Description Languages (HDL) other levels of abstraction were proposed including the Register-Transfer Level (RTL) and the behavioral level. In the beginning of the 2000s, and with the emergence of new languages (mainly SystemC) for the description of systems, a higher level of abstraction was created named the system-level. Current research targeting the Model Driven Engineering (MDE) methodology [2] shows the effectiveness of this methodology in the domain of System-on-Chip (SoC) design. Describing complex systems using models, which is the primary issue of MDE, leads to the creation of a higher level-of-abstraction: the model level. This level is mainly based on the Unified Modeling Language (UML) [3] and a domain-specific profile dealing with a specific type of systems: embedded systems.

1.2 Reusing IP Blocks

Historically, design reuse has proven its utility in the SoC design field as system complexity continuously increases [4]. However, there is one important challenge in adopting this methodology: the lack of formal characterization of platforms. As a result, platforms should be formally defined in terms of semantics to facilitate verification, automatic design, reuse and interoperability between Electronic Design Automation (EDA) tools. IP-XACT [5] was created to face this challenge. It describes electronic components and their designs in an Extensible Markup Language (XML) format that facilitates exchanging IPs between different EDA tools for complex SoC design. IP-XACT was standardized by the SPIRIT Consortium.

1.3 Building Well Structured Methodologies

Methods and tools used in the specification and design space exploration of HPC architectures aim at managing the increasing complexity of hardware architectures specification task while promoting IP reuse through the IP-XACT standard. Current hardware specification efforts within the MDE community can be summarized in two key points:

- Modeling IP-XACT designs in UML and annotating models with IP-XACT specific stereotypes
- Applying UML as high-level specification methodology and link it with IP-XACT in a lower-level of abstraction using MDE transformation rules

The work presented in this paper is an effort towards the second key point. Actually, we propose a new approach that takes advantage from UML as high-level modeling language combined with the Modeling and Analysis of Real-Time and Embedded Systems (MARTE) profile [6] and introduces another level that facilitates IP integration, architecture generation and system analysis. This level is based on the System-Level Architecture Model (S-LAM) [7] which conforms to the IP-XACT standard. S-LAM proposes a simple description of MP2SoC architectures at system-level while reducing the architecture simulation complexity. This paper presents the MARTE to S-LAM generator, able to generate from a UML/MARTE description of the MP2SoC architecture, the corresponding S-LAM description required for running a system-level rapid prototyping process.

This paper is organized as follows: related works dedicated to hardware resource modeling and IP-XACT integration are highlighted in Sect. 2. Section 3 introduces our framework for the co-design of MP2SoC embedded systems. Section 4 details our proposed S-LAM generator including the implemented meta-models and transformation rules. Finally, Sect. 5 gives some experimental results.

2 Related Work

In recent years, there has been an extensive interest in merging MDE-based frameworks and metadata IP reuse approaches. Initial efforts targeting to combine UML design entries with IP-XACT have been gaining traction [8–10]. These efforts aim to choose the adequate profile that covers the specification of complex hardware platforms on the one hand, and to implement the adequate mapping that generates the required IP-XACT description of the architecture on the other hand.

2.1 Using UML Profiles for HW Resource Modeling

UML is a general language but its extensibility, introduced with UML 2.0 via the notion of profiles, extends the language to domain-specific problems. More precisely, UML started to be adopted as a standard in the domain of real-time and embedded systems during the past years. Several profiling mechanisms aiming to use UML in SoC design and especially in hardware specification have been proposed including UML for SoC [11] and Omega-RT [12] profiles. With the ever increasing demand and complexity of embedded systems, a new profile has emerged. This standardized profile, named MARTE [6], is structured around two central concerns, modeling

the characteristics of embedded systems and annotating the models to support the analysis of the system features. Defining accurate semantics for time and Hw/Sw resource modeling and supporting real-time and embedded systems co-design flows are the major goals of the MARTE profile. These two goals can be achieved using the MDE foundations when defining embedded system design flows. This explains the use of MARTE and MDE in the proposed co-design flow. In one hand, MDE facilitates automatic transformations from one abstraction level to a lower one, for simulation or implementation purposes. In the other hand, it promotes the integration of different tools thanks to transformation techniques. As a result, analysis tools, verification tools and modeling tools can be coupled in a single co-design flow.

2.2 Merging UML and IP-XACT in MDE-based Design Flows

Several works have shown the importance of integrating IP-XACT while taking advantage from MDE principles in their design flows. In [8] a MARTE-based methodology that exploits IP-XACT to specify and automatically generate Dynamic Partial Reconfiguration (DPR) SoC designs was proposed. MARTE models of the platform are parsed executing a chain of model transformations to obtain an IP-XACT description of the system that can be used in the Xilinx EDK (Embedded Design Kit) environment. In the COMPLEX framework [9], the IP-XACT description of the architecture can be automatically generated from the UML/MARTE model using the MARTE to IP-XACT (MARTIX) code generator [10]. Then, an executable model can be built from the IP-XACT platform description for functional validation and performance estimation. In another work [13], IP-XACT was used as input point in an MDE-based approach aiming to generate SystemC code. The authors propose a multi-level design flow that integrates extensions of the IP-XACT standard and different meta-models. Comparing these related works with our approach, we can observe that none of them uses IP-XACT for the high-level design space exploration of MP2SoC systems. Moreover, these works try to exploit the whole IP-XACT metadata targeting low-level simulations. On the contrary, our approach is based on a simplified sub-set of IP-XACT, named S-LAM, for the high-level analysis of MP2SoC.

3 A Co-Design Framework Integrating the S-LAM Generator

Our proposed approach, depicted in Fig. 1, is a complete EDA tool for the co-specification, design space exploration and code generation of MP2SoC systems that relies on Object Management Group (OMG) standards and MDE techniques. Being based on the Eclipse framework, front-end, transformation engine and back-end tools are grouped together in a fully-integrated flow.

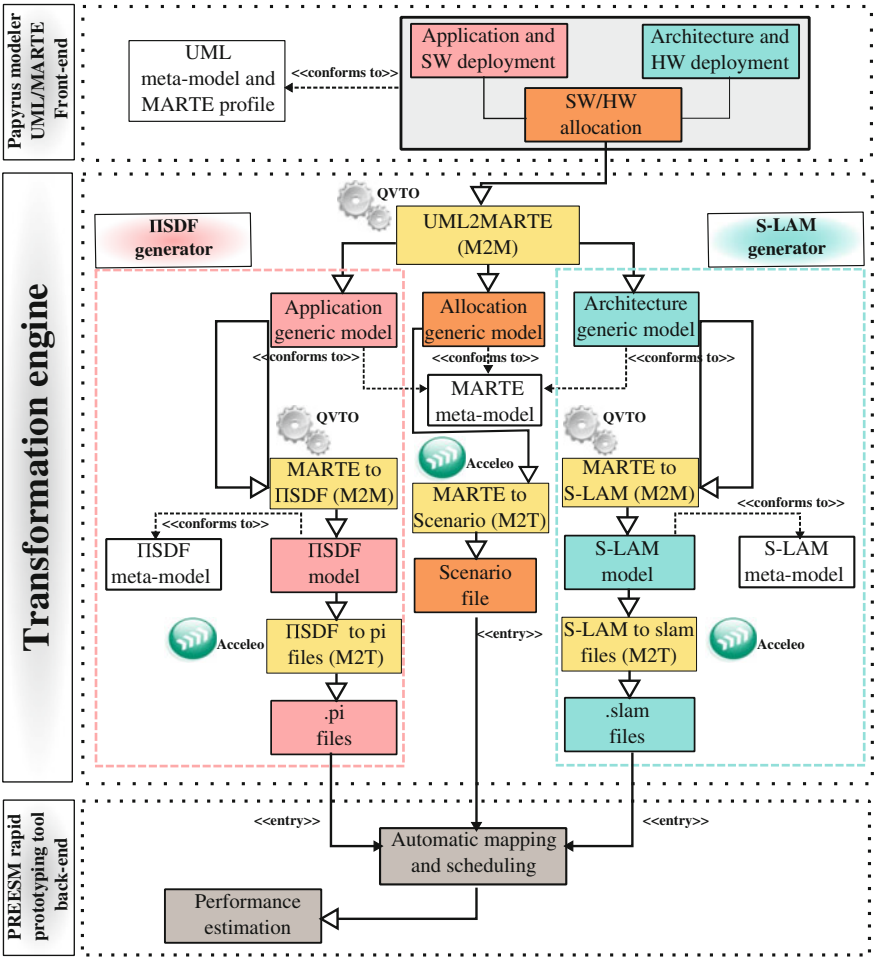

Fig. 1 The S-LAM generator in the context of the co-design flow

3.1 UML/MARTE Front-End

The proposed co-design flow uses UML/MARTE and the associated Papyrus tool [14] as modeling front-end. This high-level modeling front-end allows a user to graphically specify an embedded system conforming to the UML meta-model and the MARTE profile. Our methodology defines four sub-models to be specified and associated in a unified UML/MARTE based-model: application, architecture, allocation, and deployment sub-models.

3.1.1 Application Sub-model

Contains the structural specification of a given data-intensive application where computations are defined as a set of interconnected tasks inside a UML composite structure diagram. Application constraints and properties are defined in this sub-model including execution time value of each task using the «swSchedulableResource» stereotype from the MARTE *Software Resource Modeling* (SRM) sub-profile. The MARTE *Repetitive Structure Modeling* (RSM) sub-profile is used to model the parallel computations and the multidimensional data structures in the application. In addition, the *Generic Component Modeling* (GCM) sub-profile helps to define data flow ports and connectors.

3.1.2 Architecture Sub-model

Gathers a number of interconnected resources specifying the hardware components of an embedded system in a structural way. Therefore, the composite structure diagram is used to model the hierarchic structure of MP2SoC. Stereotypes from the MARTE *Hardware Resource Modeling* (HRM) sub-profile are exploited to indicate which kind of hardware component each UML element represents («HwProcessor» «HwMemory» «HwCommunicationResource» stereotypes). Properties of hardware processing resources, storage resources and communication resources are also specified using tagged values of these stereotypes. Multidimensional parallel resources of massively parallel MP2SoC architectures are specified using the *RSM* sub-profile. Ports and interconnections between hardware resources are annotated with stereotypes from the *GCM* sub-profile.

3.1.3 Allocation Sub-model

Defines the allocation constraints which associate tasks from the application sub-model with resources from the architecture sub-model. To allocate tasks to hardware components, the MARTE *alloc* sub-profile is used. In fact, UML dependencies between class instances of the application and the architecture are annotated with «allocate» or «distribute» stereotypes helping to map each task to a component or a repetition of a task to a group of components. The allocation is partial and defines only mapping constraints since the rapid prototyping tool automatically makes mapping decisions.

3.1.4 Deployment Sub-model

Describes the deployment of the software and the hardware components on IPs using the UML deployment diagram. The UML deployment mechanism and the MARTE profile lack aspects that allow the deployment of IPs on a component of

the SoC. For this reason, our flow proposes an additional profile, the *Deployment* profile to facilitate deploying elementary components with IPs. The proposed profile facilitates both the high-level modeling of IPs and the automatic generation of the S-LAM system description. The «HwIP» stereotype, from the *Deployment* profile, models an IP deployed on a component of the architecture facilitating the generation of S-LAM descriptions. It gathers a set of attributes used to specify a component description in the S-LAM standard.

3.2 Transformation Engine

Three transformation engines were developed inside the transformation engine:

- **The π SDF generator:** produces πSDF graphs of the data-parallel application to facilitate the analysis of modern data-intensive applications running on MP2SoC architectures. The implementation of the πSDF generator is detailed in [15].
- **The S-LAM generator:** produces an S-LAM description of the architecture (cf. Section IV).
- **The MARTE to Scenario transformation:** produces a scenario file for the rapid prototyping framework. This scenario gathers systems constraints and properties aiming to guide the rapid prototyping process.

3.3 PREESM Tool Back-End

The generated πSDF graphs of the application, S-LAM description of the architecture and scenario file can be automatically analyzed and processed using the PREESM [7] rapid prototyping tool for automatic allocation, scheduling [16], system performance estimation [7] and finally code generation.

4 The S-LAM Generator

The implementation of a transformation flow in the MDE approach relies on the definition of ad-hoc meta-models for each abstraction level. For this reason, two meta-models are proposed in the context of the S-LAM generator: the MARTE meta-model and the S-LAM meta-model. In addition, model-to-model (M2M) and model-to-text (M2T) transformations were defined inside the transformation chains as depicted in Fig. 1. In our approach, M2M transformation rules are defined using the QVTO language [17] and M2T transformation rules are described using the Acceleo tool [18].

4.1 MARTE Meta-Model Relevant Parts Used in the S-LAM Generator

The input of each transformation chain in the proposed framework is a UML model compliant with the MARTE profile. Generating a MARTE model (conforming to the MARTE meta-model) from a profiled UML model (conforming to the UML meta-model) is a typical transformation in a UML/MARTE-based framework. The developed UML2MARTE transformation corresponds to a bridge connecting the specification of the system and the developed generators. This transformation is out of the scope of this paper. The open-source Ecore version of the MARTE meta-model provided with the source code of Papyrus and extended with the Deployment elements is used as the input of the S-LAM generator.

4.1.1 Conserving the Hierarchical Structure of MP2SoC with GCM Meta-Model

The *GCM* package from the MARTE profile defines a rich base of notations helping to annotate ports, interconnections, etc. However, supporting component-based models remains most important when focusing on moving up from specification purposes, where the MARTE profile is employed as a foundation, to successive transformations for DSE, where the MARTE meta-model is used as starting point. The *GCM* meta-model can preserve the hierarchical structure of a model without losing any detail since it represents an abstraction of the UML structured classes. A hierarchical component in MARTE is a *StructuredComponent* that encloses instances of other components, presented using the *AssemblyPart* element. Two assembly parts are connected via their ports (*FlowPort* element) using connectors. Connectors between two *AssemblyParts* are named *AssemblyConnectors*.

4.1.2 Capturing Repetitive Structures in the RSM Meta-Model

The *RSM* meta-model extends the basic concepts of the MARTE meta-model by providing meta-classes that capture shaped multiplicities and link topologies of intensive computation embedded systems. This meta-model proposes high-level meta-modeling mechanisms that express all the available parallelism of the hardware execution platform precisely and in a compact manner. These mechanisms are oriented toward two features: capturing the regularity of an MP2SoC system structure (composed of a repetition of structural elements) and denoting the topologies of links between hardware components of the system.

4.1.3 Capturing System Properties in the HRM Meta-Model

The *Hw_Logical* meta-model is the relevant part from the *HRM* meta-model used in the S-LAM generator as it gathers the set of hardware resources that are central to the MP2SoC platform definition. Properties of memories (size), communication networks (speedup) and processors can then be captured inside the meta-classes of this meta-model.

4.1.4 Capturing IP Properties in the Deployment Meta-Model

The Ecore version of the current MARTE meta-model was extended to enable its merging with the *Deployment* meta-model. Properties of each IP can be then deduced in the generated MARTE model from the «hwIP» stereotype and captured inside the *hwIP* meta-class.

4.2 The S-LAM Meta-Model

At high-levels of abstraction, a detailed description of each hardware resource is not necessary to succeed a rapid prototyping process. For this reason, the S-LAM meta-model does not use the entire IP-XACT meta-model, but it exploits a sub-set of concepts that capture the needed information for the exploration phase. This sub-set includes two meta-models: the *component* meta-model and the *design* meta-model.

4.2.1 The Component Meta-Model: Simplifying IP Description for DSE

A component, according to the IP-XACT standard, specifies a single hardware IP and details the required information for the integration of this IP including its interfaces and its internal structure. Assuming that a specification approach that ignores the implementation details of each component of the hardware architecture while detailing its primary properties makes the system-level exploration process faster and gives satisfactory solutions, the S-LAM *component* meta-model defines only three component types: operators, enablers and communication nodes. These components are efficient enough to specify a massively parallel embedded architecture that gathers processing elements (operators), local and shared memories (enablers) and regular and irregular communication networks (communication nodes).

4.2.2 The Design Meta-Model: Supporting Hierarchy and Composition

The S-LAM *design* meta-model, depicted in Fig. 2, describes a design as a set of component instances (*ComponentInstance* element), links (*Link* element), hierarchy

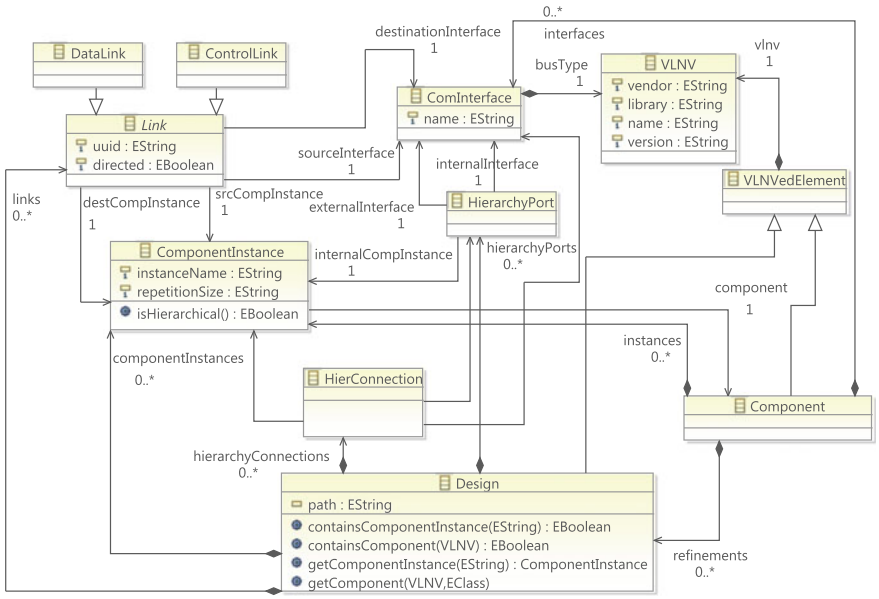

Fig. 2 S-LAM design meta-model

ports (*HierarchyPort* element) and hierarchy connections (*HierConnection* element). Both *Design* and *Component* elements are identified using their *VLNV* which specifies the vendor, the containing library, the element name, and the version number of a given IP. Each component instance in the design refers to the initial component description. These component instances can be connected using two types of connection elements: *Link* and *HierConnection*. While links are point-to-point connections between communication interfaces (*ComInterface* element) of the component instances, hierarchy connections connect sub-designs or components from different hierarchical levels using hierarchy ports. The original Ecore version of the S-LAM meta-model [7] was extended to allow the specification of a repetition of the same IP. The *repetitionShape* attribute was added to the *ComponentInstance* meta-class allowing to specify the repetition shape of a given component instance.

4.3 M2M Mapping Rules: From MARTE Model to S-LAM Model

The basic UML to MARTE and MARTE to S-LAM implemented QVTO mappings are sketched in Fig. 3.

UML meta-model	MARTE meta-model	S-LAM meta-model
Design		
UML::Class (with parts)	GCM:: StructuredComponent	slam::Design
UML::Property (part inside a hierachical class)	GCM:: AssemblyPart	slam:: ComponentInstance
UML::Port (port inside a hierachical class)	GCM::FlowPort (of a StructuredComponent)	slam:: HierarchyPort
UML::Connector (between two parts)	GCM::AssemblyConnector (between AssemblyParts)	slam::DataLink or slam::ControlLink
Component		
UML::Property (part inside a hierachical class)	GCM:: AssemblyPart	slam::Component
UML::Port (port inside a part)	GCM::FlowPort (of an AssemblyPart)	slam::ComInterface
UML::Property (stereotyped HwProcessor)	GCM::AssemblyPart (with a HwProcessor classifierTypeExtension	slam::Compoonent::Operator

Fig. 3 Mappings between UML, MARTE and S-LAM meta-models

4.3.1 Building the Hierarchical Structure of S-LAM

The S-LAM generator navigates the MARTE-compliant model and produces one S-LAM model. This model is produced if and only if the S-LAM generator finds at least one *StructuredComponent* in the MARTE model. Then, the hierarchical structure of the S-LAM model is created based on the *Design* meta-model. First, each *StructuredComponent* is transformed into a *Design*. Each *AssemblyPart* within the *StructuredComponent* becomes a *ComponentInstanse* inside the *Design* element. Moreover, if the shape of the *AssemblyPart* is superior to one, the *repetitionSize* attribute of the *ComponentInstance* will take the value of the *shape* element, indicating a repetition of a hardware component instance. Examining each *StructuredComponent*, the S-LAM generator looks for the *AssemblyConnectors* which associate *AssemblyParts*, and produces *DataLinks* or *ControlLinks* depending on the *AssemblyParts* type (HwProcessor, HwMemory, etc.). In addition, *AssemblyConnectors* linking an *AssemblyPart* with the *StructuredComponent* itself are transformed into *HierarchicalConnections*. For the production of *HierarchyPorts*, the generator explores the ports set of a given *StructuredComponent*, and transforms each *FlowPort* into a *HierarchyPort*.

4.3.2 Generating the Interface Set of Each Component Instance and Deducing its Type

For each *AssemblyPart* of the *StructuredComponent*, the S-LAM generator simultaneously produces a *ComponentInstance* and a *Component*. The implemented transformation automates the generation of the corresponding *ComInterfaces* of each *Component*. In fact, *FlowPorts* of each *AssemblyPart* are converted into *ComInterfaces* when mapping the corresponding *AssemblyPart* into *Component*. Furthermore, the generator is able to produce the right type of *Component* once it checks the *classifierTypeExtension* element attached to the *AssemblyPart*. In fact, if the *AssemblyPart* is not hierarchic, it will be transformed into an *Operator*, a *Mem* or a *ComNode* depending on its *classifierTypeExtension* (HW_Processor, HW_Memory, HW_CommunicationResource, HW_Bus). A hierarchical *AssemblyPart* is an instance of a *StructuredComponent* which was a hierarchical class stereotyped «HwResource» in the UML model. It is transformed into an *Operator* if it contains in its internal structure a processor.

4.4 M2T Mapping Rules: From S-LAM Model to S-LAM Files

Figure 4 shows the main Acceleo template which is the entry point of the M2T transformation. Given that this template requires an instance of the parameter *Design*, the transformation will navigate in the whole model to find all the available *Design* elements and generate one S-LAM file per Design. The produced files are named as the *Design* plus the ".slam" suffix, and encloses the «spirit:design» entry. Then, for each *ComponentInstance* element from the S-LAM model, the transformation will produce one component instance inside the «spirit:componentInstances» and «spirit:componentInstances» delimiters. At the same time, this transformation controls the *repetitionShape* value of each *ComponentInstance* in order to generate N (where N is the value defined by the *repetitionShape* attribute) component instances indicating the presence of a repetition of the same component instance in the design. The M2T transformation searches all the *DataLinks* and *ControlLinks* and produces a set of S-LAM interconnections. It also implements a similar navigation to figure out the list of hierarchical connections.

5 Case Study

To evaluate the benefits of our framework, we conduct a series of experiments on the M-JPEG encoder application. Originally developed for streaming multimedia application, the M-JPEG video compression format is now considerably exploited in video-capture devices where each video frame or video sequence is compressed separately as a JPEG image. Compared to the recently emerged video compression

```
[template public generateDesign(design : Design)]
[comment @main/]
[file (design.name.toString().concat('.slam'), false, 'UTF-8')]

<?xml version="1.0" encoding="UTF-8"?>
<spirit:design
xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4">

[design.generateVLNV()/]

     <spirit:componentInstances>
       [for (compInstance:ComponentInstance | self.componentInstances)]
           [compInstance.generateComponentInstance()/]
       [/for]
     </spirit:componentInstances>

     <spirit:interconnections>
       [for (datalink:DataLink | self.links)]
             [datalink.generateDataLink()/]
       [/for]

       [for (controllink:ControlLink | self.links)]
             [controllink.generateControlLink()/]
       [/for]
     </spirit:interconnections>

     <spirit:hierConnections>
       [for (hierconn:HierConnection | self.hierarchyConnections)]
             [hierconn.generateHierConnection()/]
       [/for]
     <spirit:hierConnections/>

<spirit:vendorExtensions>
       [design.generatevendorExtensions()/]
</spirit:vendorExtensions>

</spirit:design>
[/file]
[/template]
```

Fig. 4 Acceleo main template

standards, M-JPEG describes a relatively simple encoding workflow. But, it is a typical streaming application that contains inherent task and data parallelism the fact that provides rich experimentation opportunities when running on MP2SoC architectures. Figure 5 shows the composite structure diagram of the application. The video sequence should first be partitioned into frames (M-JPEG_encoder class). Frames are split in blocks of 8*8 pixels and processed separately as JPEG images (Encode_Frame class). We performed experiments by simulating the M-JPEG on a stream of 100 and 200 frames of QCIF format (352 × 288 pixels). For this reason, multiplicities of tasks and ports expressed via the «Shaped» stereotype were varied.

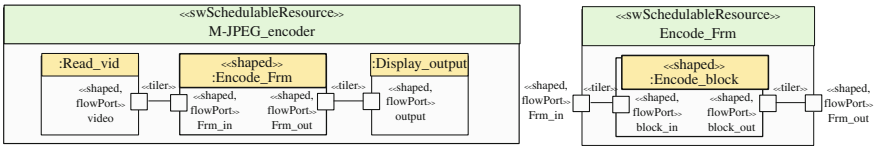

Fig. 5 UML/MARTE specification of the application

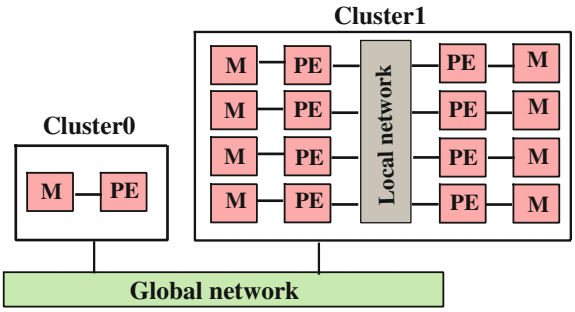

Fig. 6 MP2SoC architecture

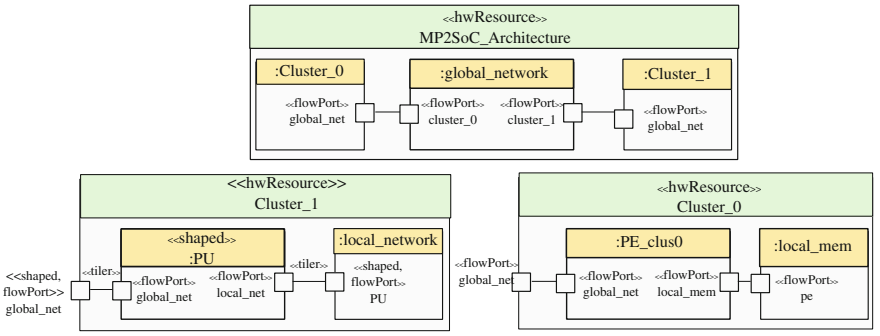

Fig. 7 UML/MARTE specification of the architecture

MP2SoC, as presented in Fig. 6, is composed of two clusters. While the first cluster contains one processing element (PE), the second cluster includes a variable number of processing elements. Processing elements inside the clusters are homogenous. Inside each cluster, each processing element is connected to its local memory and can communicate to other processors via a local network. The clusters can communicate via a global interconnection network. In order to model such complex system, a UML composite structure diagram is used as seen in Fig. 7. Each hierarchic hardware resource (MP2SoC system, clusters, processing units) is specified using a hierarchic class. For the rapid prototyping of the M-JPEG application, five configurations of MP2SoC were specified and generated varying the number of processing units (by changing the shape value of the PU class) containing 2, 4, 8, 24 and 32 processing

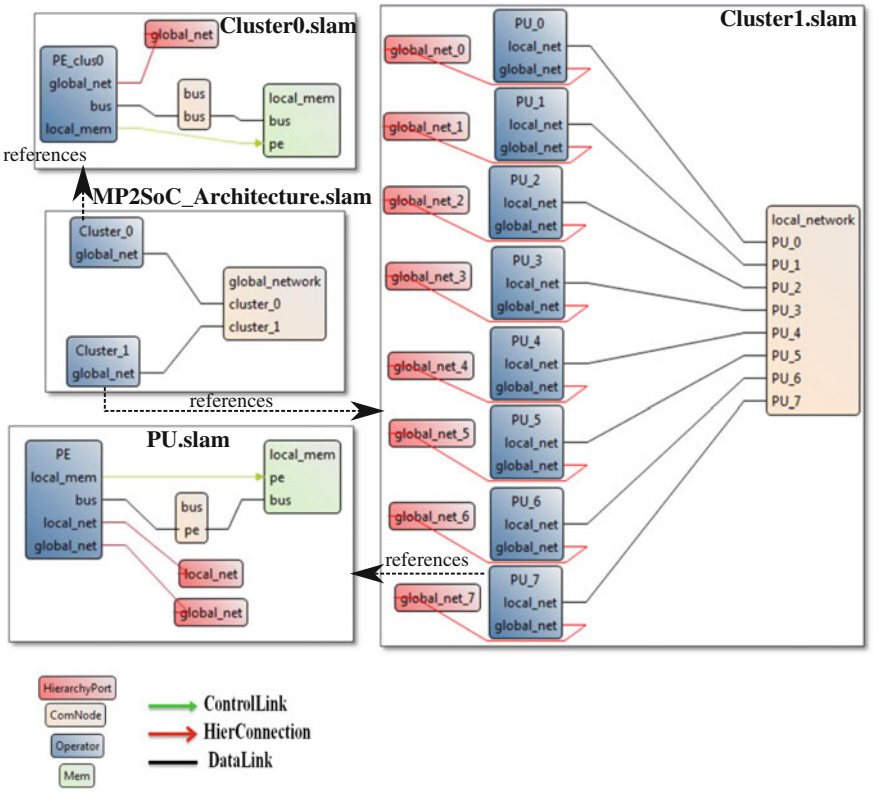

Fig. 8 Generated S-LAM files

units in Cluster1. Executing the S-LAM generator, four .slam files were created and visualized using the S-LAM editor as shown in Fig. 8. Each hierarchic class is transformed first into a Design element then into an .slam file. Class instances inside the hierarchic class are mapped into operators, memories or communication nodes. Hierarchic class instances that reference classes containing operators are transformed into operators that reference the .slam file that describes the internal structure of the classes themselves. Ports of the hierarchic classes becomes hierarchy ports. The «Shaped» annotation attached to the PU class and the port of the Cluster1 hierarchic class allows to produce eight hierarchy ports and link them with the eight operators with hierarchy connections in the MP2SoC configuration that contains eight processing units. πSDF files and the scenario file are also generated executing the two other transformation chains. The final step in the proposed approach is the rapid prototyping of the πSDF/S-LAM combination using PREESM. Figure 9 shows the average speedup of the application for two video sequence containing 100 and 200 frames respectively running on different MP2SoC configurations. We notice that for

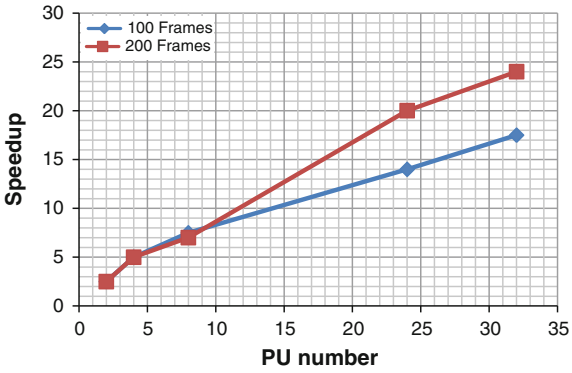

Fig. 9 Speedup results

the video sequence containing 200 frames, increasing the PU number from 2 to 32 contributes for up to 10x M-JPEG encoder speedup. This observation justifies the use of MP2SoC architectures.

6 Conclusion

In this paper, the S-LAM generator, a tool able to generate S-LAM description of an MP2SoC architecture described in UML/MARTE under the proposed co-design flow specification methodology was presented. High-level models of the complex architecture are progressively refined enabling the production of a system-level description of the architecture for the design space exploration step, which is based on the PREESM framework. The S-LAM generator reduces the modeling effort as it starts from a co-specification of the whole MP2SoC system, including the application and the architecture parts, and captures needed information for the generation of IP-XACT compliant description of the architecture. Our next future work will be concentrated on the elaboration of a use case that takes as design entry a complex massively parallel application (An H.264 decoder for example) running on an MP2SoC architecture.

References

1. Engelmann, C., Lauer, F.: Facilitating co-design for extreme-scale systems through lightweight simulation. In IEEE International Conference on Cluster Computing Workshops and Posters, CLUSTER WORKSHOPS, 2010, pp. 1–8 September 2010
2. Lugato, D., Bruel, J-M., Ober, I.: Model-Driven Engineering for High Performance Computing Applications. In: S. Cakaj (ed.) Modeling Simulation and Optimization-Focus on Applications (2010)

3. Object Management Group. Unified Modeling Language specification, version 2.1. Available: http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF
4. Ecker, W., Müller, W., Dömer, R.: Hardware-Dependent Software, pp. 1–13. Springer, Netherlands (2009)
5. IEEE standard for IP-XACT, standard structure for packaging, integrating, and reusing IP within tools flows, IEEE Std 1685-2009, February 2010, pp. C1-360
6. Object Management Group. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, version 1.0. Available: http://www.omg.org/spec/MARTE/1.0/PDF/
7. Pelcat, M., Desnos, K., Heulot, J., Guy, C., Nezan, J.-F., Aridhi, S.: Preesm: A dataflow-based rapid prototyping framework for simplifying multicore DSP programming. In 6^{th} European Embedded Design in Education and Research Conference. EDERC 2014, pp. 36–40 (2014)
8. Ochoa-Ruiz, G., Labbani, O., Bourennane, E.-B., et al.: A high-level methodology for automatically generating dynamic partially reconfigurable systems using IP-XACT and the UML MARTE profile. Des. Autom. Embed. Syst. **16**(3), 93–128 (2012)
9. Herrera, F., Posadas, H., Villar, E., Calvo, D.: Enhanced IP-XACT platform descriptions for automatic generation from UML/MARTE of fast performance models for DSE. In 15^{th} Euromicro Conference on Digital System Design, DSD 2012, pp. 692–699, September 2012
10. Herrera, F., Villar, E.: A Framework for the Generation from UML/MARTE Models of IP-XACT HW Platform Descriptions for Multi-Level Performance Estimation. Proceedings of the Forum of Design and Specification Languages, FDL'2011, November 2011
11. Object Management Group. UML profile for System on a Chip, version 1.0. Available: http://www.omg.org/spec/SoCP/1.0/PDF/
12. Graf, S., Ober, I., Ober, I.: A real-time profile for UML. Int. J. Softw. Tools Technol. Trans. **8**(2), 113–127 (2006)
13. El Mrabti, A., Pétrot, F., Bouchhima, A.: Extending IP-XACT to support an MDE based approach for SoC design. In Design, Automation and Test in Europe Conference and Exhibition, DATE'09, pp. 586–589, April 2009
14. Papyrus, http://www.eclipse.org/papyrus/
15. Ammar, M., Baklouti, M., Pelcat, M., Desnos, K., Abid, M.: MARTE to πSDF transformation for data-intensive applications analysis. In Conference on Design and Architectures for Signal and Image Processing, DASIP, October 2014
16. Pelcat, M., Menuet, P., Aridhi, S., Nezan, J.F.: Scalable compile-time scheduler for multi-core architectures. In Proceedings of the Conference on Design, Automation and Test in Europe, DATE'09, pp. 1552–1555, April 2009
17. Guduric, P., Puder, A., Todtenhofer, R.: A comparison between relational and operational QVT mappings. In the 6^{th} International Conference on Information Technology: New Generations, ITNG '09, pp.266–271, April 2009
18. Acceleo (2015) https://www.eclipse.org/acceleo/