A Model Driven Engineering design approach to generate VHDL for MPPSoC

M. Ammar[1], M. Baklouti[1,2], Ph. Marquet[2], M. Abid[1] and JL. Dekeyser[2]
[1]*CES, National Engineering School of Sfax, Sfax, Tunisia*
[2]*Univ. Lille, F-59044, Villeneuve d'ascq, France*
LIFL, Univ. Lille1, F-59650, Villeneuve d'ascq, France
INRIA Lille Nord Europe, F-59650, Villeneuve d'ascq, France
UMR 8022, CNRS, F-59650, Villeneuve d'ascq, France
manel.ammar@ceslab.org, mouna.baklouti@ieee.org

***Abstract*— Massively Parallel Processing System on Chip (MPPSoC) provides an interesting solution when high performance is needed for embedded parallel applications. The increasing amount of hardware resources in MPPSoC calls for efficient design methodologies and tools to reduce its development complexity. This paper presents an MPPSoC design flow, which uses the MARTE (Modeling and Analysis of Real-Time and Embedded systems) standard profile for high-level system specification. This flow is based on a Model-Driven Engineering approach. It promotes separation of concerns, reusability and automatic model refinement from higher abstraction levels to executable VHDL description facilitating the design space exploration.**

I. Introduction

Massively data-parallel applications are predominant in several application domains such as mobile multimedia processing, high-definition TV and radar/sonar signal processing. They play an increasingly important role in embedded systems. Parallel massive data processing is a key feature in these applications. When dealing with massive computation and data intensive processing, the use of massively parallel architectures is very useful. An MPPSoC system is a generic massively parallel embedded architecture designed for data-parallel applications. MPPSoC has a SIMD (Single Instruction Multiple Data) [1] parallel architecture that results from an assembly of different components and may be implemented on a single chip. In addition, MPPSoC proves very fruitful in massively parallel applications domain. However, the design and implementation of such systems become critical due to their long design and development cycles. In fact, the MPPSoC's design is facing today a strong pressure on reducing time-to-market while the complexity of this system has been increasing. Changing a SoC configuration may also necessitate extensive redesign. Design abstraction offers a possible solution to address the above issues concerning the time-to-market and complexity dilemma. It is in the context of improving the primary productivity of MPPSoC, that our work finds its proper place. This work is part of the MPPSoC project which consists in defining and designing a programmable and flexible SIMD SoC, called Massively Parallel Processing System-on-Chip, that can be simulated and prototyped on FPGA (Field Programmable Gate Arrays) devices. To facilitate and accelerate the design of an MPPSoC configuration dedicated to a given parallel application, our contribution consists in proposing an MPPSoC framework. This framework uses the MARTE [2] profile for high level specifications of SoC applications and architectures and it is based on Model Driven Engineering [3]. This methodology is based on two concepts: model and transformation. Data and their structures are represented in models, while the computation is done by transformations and enables to target different execution platforms for automatic generation of the respective code. A model basically highlights the intention of a system without describing the implementation details. UML is a model specification language [6], which proposes general concepts allowing one to express both behavioral and structural aspects of a system. The UML/MDE-based systems' design is a very young discipline [10]. In fact, UML and MDE have been adopted in co-design methods [11], [12], [13], [14] in the last years with success. Abstract models favor an efficient design reuse, typically through different refinements from higher level models to lower level models. Different approaches to high-level synthesis are currently being studied for different specification languages. Among the proposed approaches, we can mention a transformation tool, called MODCO [16], which takes a UML state diagram as input and generates HDL output suitable for use in FPGA circuit design. A HW/SW co-design is performed based on the MDA approach. XML is used to generate HDL from high-level UML diagrams. In [15], an approach using VHDL synthesis from UML behavioral models is presented. The UML models are first translated into textual code in a language called SMDL. This latter can be then compiled into a target language as VHDL. The translation from UML models to SMDL is performed using the aUML toolkit. In the preceding works, only state machines HW designs are described. In [17], the UML based design and implementation of an H.264 video decoder core is presented. The FalconML tool is used to directly generate the System C and VHDL code targeting ASIC technology. According to these various research works, there is no focus on designing parallel processing systems and proposing dedicated frameworks able to accelerate their design.

This works proposes a design flow to automatically

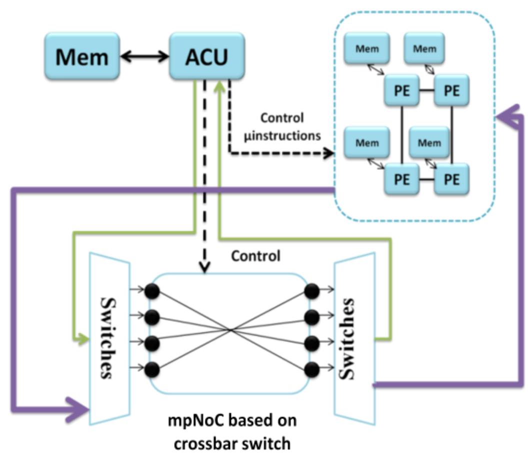

Fig. 1. MPPSoC configuration.

generate VHDL executable code for an MPPSoC configuration. The flow transforms MPPSoC meta-model to synthesizable VHDL code which can be then simulated or prototyped on any FPGA technology in order to measure performances. These steps help the designer to evaluate the generated SIMD configuration and choose the adequate one for a given application. Thus, our framework facilitates the design space exploration.

The remainder of this paper is organized as follows: Section 2 introduces the MPPSoC with focus on its flexibility and parametricity. The MPPSoC UML/MARTE model is addressed in Section 3 and the code generation approach is highlighted in Section 4. Finally, Section 5 gives concluding remarks.

II. MPPSOC ARCHITECTURE'S OVERVIEW

MPPSoC is an IP-based massively parallel architecture [4]. It (figure 1) is composed of a number of processing elements (the PEs) working in perfect synchronization. A small amount of local and private memory is attached to each PE. Every PE is potentially connected to its neighbors via a regular network. The whole system is controlled by an Array Controller Unit (ACU).

The configurable neighborhood interconnection network is implemented to assure inter-PE communications depending on its configurable topology (Mesh, Torus, Xnet, Linear array and Ring), as illustrated in the Figure 2. Furthermore, each PE is connected to an entry of mpNoC, a massively parallel Network on Chip that potentially connects each PE to one another, performing efficient irregular communications. The mpNoC is integrated to manage point to point communications through different types of connections. In fact, the mpNoC includes a configurable router which can be of different types (Shared Bus, Crossbar, Delta MIN (omega, baseline and butterfly)). The mpNoC can perform different communication modes (PE-PE, PE-ACU, PE-Device) since it contains a mode manager. This latter is implemented by two switches responsible of connecting the required sources and destinations respectively. The designer can choose to integrate none, one or both MPPSoC networks (neighborhood/mpNoC) to build a given SIMD configuration. The MPPSoC system can be customized to target diverse applications [8]. In fact, our design approach aims to define an MPPSoC configuration adapted to a given application. This customization is achieved with the parameterization as well as the extensibility and the configurability of the architecture. In fact, MPPSoC is parametric in terms of the number of PEs as well as the memories' sizes. It has three configurable aspects: processor design methodology, the integrated neighboring network's topology and the mpNoC interconnection network's type.

The processor design methodology is the manner to assemble processor IPs to build the SIMD system. We distinguish two methodologies: processor reduction and processor replication. The former consists on reducing an available open core processor in order to build a processing element with a small reduced size. The PE can be then fitted in large quantities into an FPGA device. In this case, it is only responsible of executing micro-instructions (decoded instructions) broadcasted from the ACU. This methodology allows putting a large number of PEs on a single chip; however it necessitates a long development cycle. Whereas the replication methodology consists on implementing the ACU as well as the PE by the same processor IP so that the designing process is faster. The criterion to choose this methodology on a SIMD on chip architecture is to use a smaller processor so that a big number can be put on the FPGA device. We clearly notice that there is a compromise between the development time and the number of integrated PEs in the MPPSoC configuration for the two proposed processor design methodologies. The designer can select the suitable methodology according to his application constraints.

We have briefly demonstrated that the MPPSoC is implemented as a flexible, parametric and configurable architecture. The designer can model an MPPSoC configuration suited to a given data parallel application. The MPPSoC models are explained in the following section.

III. MPPSOC MODELS

An MDE approach to design MPPSoC is developed. This approach allows the designer to automatically select an MPPSoC configuration at a very early design stage, before system synthesis and code generation have been performed. Our MPPSoC's modeling methodology relies on the MARTE profile. MARTE has been recently standardized for the modeling of real-time embedded systems. Subsets of the

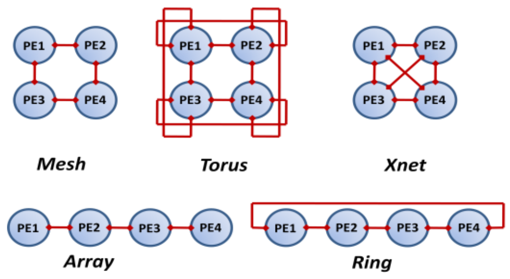

Fig. 2. Neighborhood network configurations.

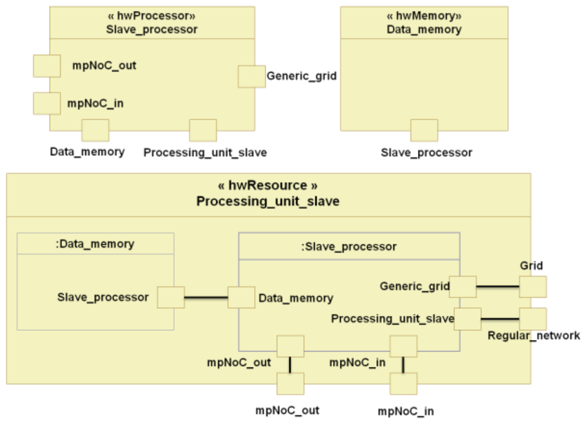

Fig. 4. Processing unit architecture modeling.

profile allow describing the MPPSoC HW components in a structural way. Our modeling methodology leverages from this profile the Hardware Resource Modeling (HRM), the Repetitive Structure Modeling (RSM) and the Generic Component Model (GCM) packages.

The designed SIMD architecture is also configurable and parametric; so that the designer can choose different configurations depending on the application requirements. UML2 templates [6] support mechanisms to express such characteristic. They are used, in this case, to easily define MPPSoC parameters.

A. *Hardware architecture modeling*

The HRM sub-package is used to specify the detailed platform architecture's elements. Its purpose is to describe HW execution supports with different details' levels and views essential to fulfill the application specification. The HRM [2] consists of two views, a logical view and a physical view. In our work we have used logical view that classifies HW resources based on functional properties. To specify the flow-oriented communication paradigm nature between MPPSoC components, we have taken advantage of the GCM package. A flow port may handle incoming, outgoing or bidirectional flows.

Figure 3 shows the ACU as component of the Reduction_processing_unit_master class depending on the reduction methodology (Replication_processing_unit_master class in the case of replication). The ACU has three ports in order to be

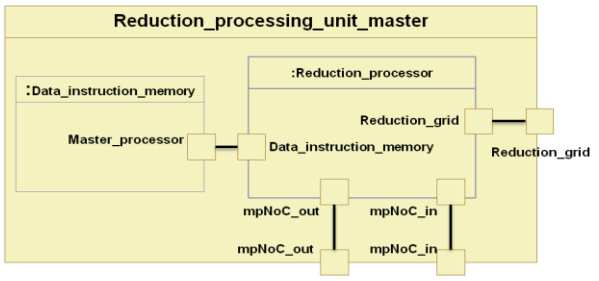

Fig. 3. The ACU architecture modeling in the case of the reduction methodology.

connected to the PEs (parallel µ-instructions (in the case of reduction) or instructions (in the case of replication)

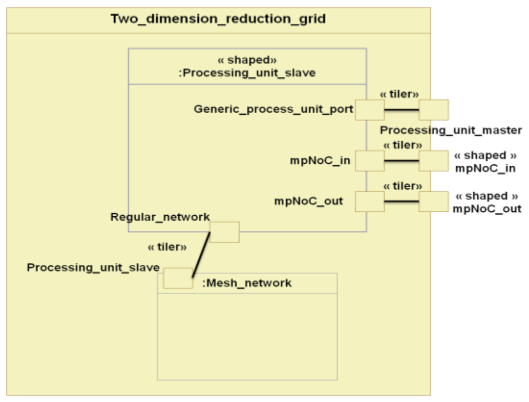

Fig. 6. Two dimension reduction grid model.

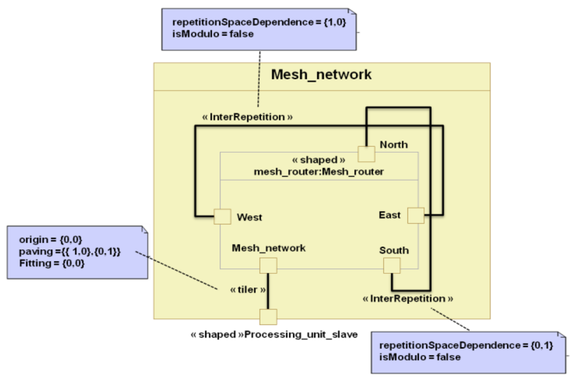

Fig. 5. The 2D mesh network architecture modeling.

broadcast) on the one hand and to the mpNoC input and output ports on the other hand.

Figure 4 depicts the processing unit component's model. It is composed of a slave processor (PE) and its data memory. The connector between the Data_memory port of the Slave_processor and the Slave_processor port of the data memory specifies how each PE communicates with its local memory. The HW architecture's modeling is also described using the repetitive concepts of the MARTE RSM package [2]. It proposes concepts to handle multidimensional structures. The considered structures are composed of repetitions of structural elements interconnected via a regular connection pattern. It provides the designer with a way to efficiently and explicitly express models with a high number of identical components. RSM is originally inspired by the Array-OL (Array Oriented Language) language [5] dedicated to intensive multidimensional signal processing.

To illustrate this package's use, we present in the figure 5 the architecture of a 2D regular communication network based on mesh routers.

The "shaped" stereotype is used to model the two dimensions grid of mesh routers. The "Shaped" stereotype's tagged values specify the number of mesh routers in each dimension. An "interRepetation" stereotype specifies dependencies between the repetitions of a given repeated structural element. This connector links a pattern of a repeated structural element with another pattern of the same repeated structural element. It is used to model the links' topology between the routers, as illustrated in the figure 5. In fact, each mesh router is connected to its neighbors in the four directions: North, South, West and East. Here, the value of the tagged value "isModulo" is equal to false because the routers in the grid's edges are not connected to each other.

The "Tiler" connector expresses how a multidimensional array is tiled by patterns. It connects an array to the patterns of a repeated structural element as illustrated in Figure 5. In this example, Processing_unit_slave is the port of the Mesh_network structural element. It represents the multidimensional array of Mesh_network. The port Mesh_network represents the pattern of the Mesh_router repeated structural element. Each mesh router is connected to one mesh network's port.

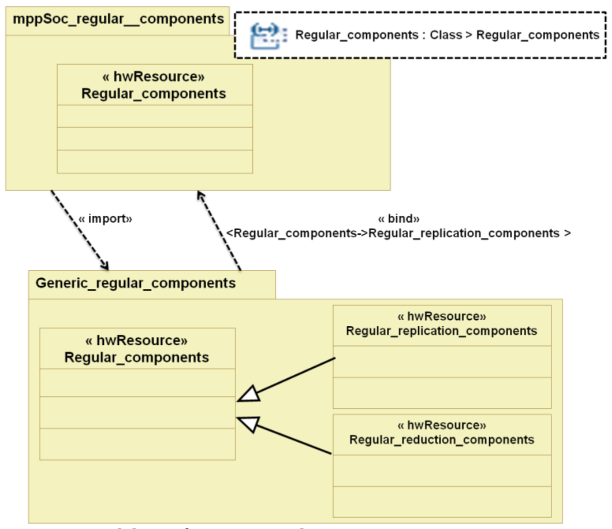
mppSoc_regular__components
Regular_components : Class > Regular_components
« hwResource»
Regular_components
« import»
« bind»
<Regular_components->Regular_replication_components >
Generic_regular_components
« hwResource»
Regular_components
« hwResource»
Regular_replication_components
« hwResource»
Regular_reduction_components

Fig. 8. Modeling of generic regular components.

Figure 6 delineates a reduction methodology's model using stereotypes that are previously described. The architecture consists of a mesh network (figure 5) and a repetition of the processing unit. The interconnection topology is modeled thanks to four "Tiler" connectors. One connector specifies that each potential instance of the Processing_unit_slave is connected to one Mesh_network component's port. The other three connectors stipulate how the Processing_unit_slave is connected to the reduction grid. The Processing_unit_slave [i,j] is connected to three ports: Processing_unit_master, mpNoc_in [i,j] and mpNoc_out [i,j].

The replication methodology has the same components shown in the Figure 6 except the type of the port named Processing_unit_master. In fact, this port communicates instructions, instead of µ-instructions between the replication grid and the processing unit master.

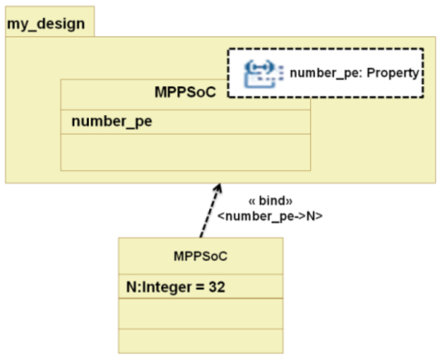

Fig. 9. Parametric PE number selection.

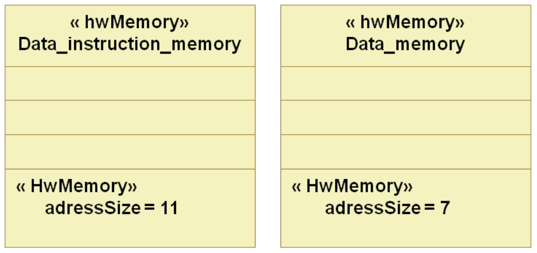

Fig. 10. Parametric memory size selection.

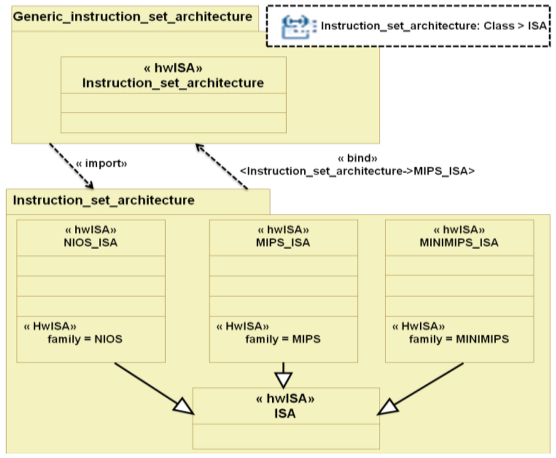
Generic_instruction_set_architecture
Instruction_set_architecture: Class > ISA
« hwISA»
Instruction_set_architecture
« import»
« bind»
<Instruction_set_architecture->MIPS_ISA>
Instruction_set_architecture
« hwISA»
NIOS_ISA
« HwISA»
family = NIOS
« hwISA»
MIPS_ISA
« HwISA»
family = MIPS
« hwISA»
MINIMIPS_ISA
« HwISA»
family = MINIMIPS
« hwISA»
ISA

Fig. 7. Processor IP selection.

B. *UML2 templates to express MPPSoC parameterization*

A template is a model element parameterized by other model elements. These elements can be classifiers, packages or operations. Classifier and package template elements are respectively called Classifier Templates and Package Templates. For parameterization specification, a template element owns a Template Signature relating to a list of formal Template Parameters. In this list, each parameter chooses an element that is part of the template. Using the template, binding relationship links a "bound" element to the signature of a target template. This causes a set of template parameter substitution in which formal template parameters are replaced by actual parameters. The MPPSoC generic characteristic is easily defined using templates. This methodology is used to define all MPPSoC configurable components.

The methodology followed to choose one MPPSoC configuration through the developed UML/MARTE model can be divided in different steps:

- Select the used processor: the "hwISA" stereotype is used to model the processor IP's type to be implemented,

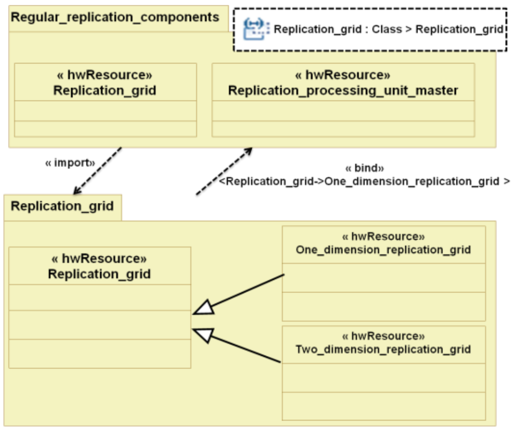
Regular_replication_components
Replication_grid : Class > Replication_grid
« hwResource» Replication_grid
« hwResource» Replication_processing_unit_master
« import»
« bind» <Replication_grid->One_dimension_replication_grid >
Replication_grid
« hwResource» One_dimension_replication_grid
« hwResource» Replication_grid
« hwResource» Two_dimension_replication_grid

Fig. 11. Processor design methodology selection.

chosen among three provided processors (see figure 7);

- Choose the processor design methodology: Targeting this configurable aspect in figure 8, we have defined the template package mppSoc_regular__components with template parameter Regular_components. This means that when an actual value for this template parameter is specified (reduction or replication), the Regular_components class acquires this value indicating the chosen design methodology;

- Fix the parametric number of PEs: introduce the value for the template parameter number_pe (see figure 9);

- Set the ACU memory address width and the PE memory address width: use the tagged value "adressSize" provided with the stereotype "hwMemory" (see figure 10);

- Choose the PE arrangement: the template parameter Replication_grid class is bound to the One_dimension_replication_grid (see figure 11) or to the Two_dimension_replication_grid if the designer choose the replication design methodology;

- Select the regular network to be integrated: as illustrated in the figure 12, the parameter exposing the One_dimension_regular_network class has been bound to the Linear_network class. As a result, a linear topology based neighborhood network is integrated in the MPPSoC configuration;

- Choose the mpNoC type: the generic package mppSoc_irregular_components (see figure 13) exposes a class as a parameter. This class presents the mpNoC router's type. The template binding relationship substitutes the parameter with the value representing one of the types illustrated in the package Generic_mpNoc.

IV. CODE GENERATOR

In this section we will give an overview of our transformation chain's implementation which is depicted in figure 14. In order to generate code, our transformation chain takes an MPPSoC configuration's model as input and produces text as output. Such transformation is called model-to-text transformation. As transformation chain's output, the user expects executable code which can be used

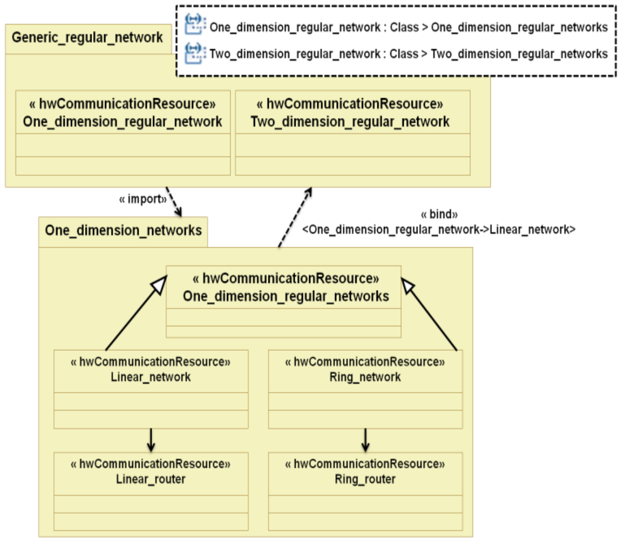
Generic_regular_network
One_dimension_regular_network : Class > One_dimension_regular_networks
Two_dimension_regular_network : Class > Two_dimension_regular_networks
« hwCommunicationResource» One_dimension_regular_network
« hwCommunicationResource» Two_dimension_regular_network
« import»
« bind» <One_dimension_regular_network->Linear_network>
One_dimension_networks
« hwCommunicationResource» One_dimension_regular_networks
« hwCommunicationResource» Linear_network
« hwCommunicationResource» Ring_network
« hwCommunicationResource» Linear_router
« hwCommunicationResource» Ring_router

Fig. 12. Regular network topology selection.

in already available tools. The target is a synthesizable VHDL description for the MPPSoC architecture. After modeling MPPSoC configurations, we have taken advantage of the various tools offered by Acceleo [7] to generate the corresponding VHDL code.

To generate the VHDL MPPSoC code using the proposed

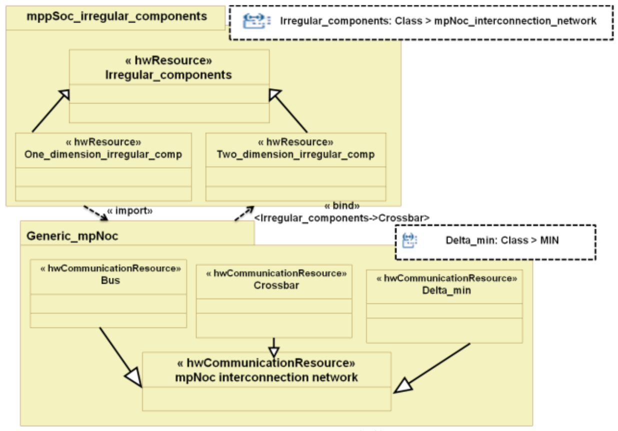
mppSoc_irregular_components
Irregular_components: Class > mpNoc_interconnection_network
« hwResource» Irregular_components
« hwResource» One_dimension_irregular_comp
« hwResource» Two_dimension_irregular_comp
« import»
« bind» <Irregular_components->Crossbar>
Generic_mpNoc
Delta_min: Class > MIN
« hwCommunicationResource» Bus
« hwCommunicationResource» Crossbar
« hwCommunicationResource» Delta_min
« hwCommunicationResource» mpNoc interconnection network

Fig. 13. Generic irregular components' modeling.

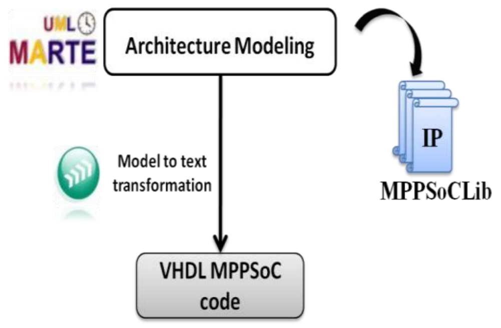

Fig. 14. MPPSoC transformation chain

framework, the designer has to choose the appropriate MPPSoC configuration based on the UML model as we have explained in section 3. Then, he can automatically generate the synthesizable code based on the modeled configuration and using our MPPSoC chain. To this end, we have realized an Acceleo module in order to browse UML diagrams and find out the MPPSoC configuration's parameters. The used processor, the processor design methodology, the number of PEs, the PE arrangement, the regular network to be integrated and the mpNoC's type are extracted from the template binding relationships. The ACU memory address width and the PE memory address width are deduced from the tagged value "adressSize" of the stereotype "hwMemory". The following code example illustrates how to deduce the data instruction memory's address size:

```
<%getStereotypeApplications().adressSize.put("MS_@_width")%>
```

According to these parameters, basics IPs will be chosen from the MPPSoCLib library and VHDL code will be automatically generated. MPPSoCLib contains different elementary IPs needed to construct an mppSoC configuration such as memory IPs, processor IPs, etc.

V. CONCLUSION

In this paper, we have presented an approach to design Massively Parallel Processing System on Chip. The proposed design flow aims at raising the specification abstraction level. This abstraction level is achieved by using MDA, which has proved its capabilities in accelerating embedded software development. The proposed approach allows to automatically generate a suitable MPPSoC code using three dedicated tools: UML2, MARTE profile and Acceleo. The MPPSoC framework offers several advantages: an efficient high level parallelism's representation, reusability and a complete generation of the hardware VHDL code. All these features strongly contribute to the increase of the designer's productivity. Thanks to the MPPSoC framework, the user is able to choose the appropriate MPPSoC configuration satisfying his needs and meeting performance requirements. In the presented work, the designer can just select the provided IPs from the MPPSoC provided HW library to generate a given MPPSoC configuration.

Future works aim at allowing the designer to integrate his own chosen IP. The same issue will be also handled for SW library. A high level exploration step will be integrated to help the designer directly select the most suitable application-specific MPPSoC configuration.

REFERENCES

[1] Meilander, W. C., Baker, J. W. and Jin, M. 2003. Importance of SIMD Computation Reconsidered. In International Parallel and Distributed Processing Symposium, (2003).

[2] Object Management Group, Inc. (2009). UML Profile for MARTE V 1.0 Technical Report formal/2009-11-02.

[3] Object Management Group, Inc. (2003). MDA Guide V1.0.1 Technical Report omg/03-06-01.

[4] M. Baklouti, Ph. Marquet, M. Abid and JL. Dekeyser. A design and an implementation of a parallel based SIMD architecture for SoC on FPGA. In Proc. DASIP, (2008).

[5] Boulet. P. 2007. Array-OL revisited multidimensional intensive signal processing specification. Research Report RR-6113, INRIA. (Feb. 2007).

[6] Object Management Group, Inc. (2005) UML Technical Report formal/2005-07-04.

[7] Obeo. Acceleo Generator. (2010). http://www.acceleo.org.

[8] M. Baklouti, Ph. Marquet, M. Abid and JL. Dekeyser. IP based configurable SIMD massively parallel SoC. In PhD Forum of 20th international conference on Field Programmable Logic and Applications (FPL), August (2010).

[9] R. J. Vidmar. (1992, August). On the use of atmospheric plasmas as electromagnetic reflectors. *IEEE Trans. Plasma Sci.* [Online]. *21(3).* pp. 876—880. Available: http://www.halcyon.com/pub/journals/ 21ps03-vidmar

[10] Jean-Luc Dekeyser, Abdoulaye Gamatié, Anne Etien, Rabie Ben Atitallah, and Pierre Boulet. Using the UML Profile for MARTE to MPSoC Co-Design. In *First International Conference on Embedded Systems & Critical Applications (ICESCA'08)*, Tunis, Tunisia, May 2008.

[11] T. Wang, X.-G. Zhou, B. Zhou, L. Liang, and C.-L. Peng, "A MDA based SoC Modeling Apporach using UML and SystemC," in Pro-ceedings of the sixth IEEE International Conference on Computer and Information Technology (CIT'06). IEEE Computer Society,september 2006, pp. 245–245.

[12] P. Kukkala, J. Riihimaki, M. Hannikainen, T. D. Hamalainen, and K. Kronlof, "Uml 2.0 profile for embedded system design," in DATE '05: Proceedings of the conference on Design, Automation and Testin Europe. Washington, DC, USA: IEEE Computer Society, 2005, pp. 710–715.

[13] E. Riccobene, P. Scandurra, A. Rosti, and S. Bocchio, "Designing a Unified Process for Embedded Systems," in Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, (2007). IEEE Computer Science, mars 2007, pp. 77–90.

[14] M. Rieder, R. Steiner, C. Berthouzoz, F. Corthay, and T. Sterren, "Synthesized uml, a practical approach to map uml to vhdl," in RISE, 2005, pp. 203–217.

[15] Bjorklund, D. and Lilius, J. 2002. Proc. of 20th IEEE NORCHIP Conference, (Copenhagen, Denmark, Nov. 2002) 11-12.

[16] Coyle, F. P. and Thornton, M. A. 2005. From UML to HDL: a Model Driven Architectural Approach to Hardware-Software Co-Design. Information Systems: New Generations Conference (ISNG), (Apr. 2005) 88-93.

[17] Thomson, R., Moyers, S., Mulvaney, D. and Chouliaras, V. 2008. The UML-based design of a hardware H.264/MPEG 4 AVC video decompression core. In the 5th International UML-SoC Workshop (in conjunction with 45th DAC, Anaheim Convention Center, USA, Jun. 2008).