A model-driven based framework for rapid parallel SoC FPGA prototyping

Mouna Baklouti†*, Manel Ammar†, Philippe Marquet*, Mohamed Abid† and Jean-Luc Dekeyser*
*LIFL, Univ. Lille 1, INRIA Lille Nord Europe
UMR 8022, CNRS, F-59650, Villeneuve d'ascq, France
Email:{mouna.baklouti,philippe.marquet,jean-luc.dekeyser}@lifl.fr
†CES Laboratory, Univ. Sfax, ENIS School
BP 1173, Sfax 3038, Tunisia
Email:manel.ammar@ceslab.org, mohamed.abid@enis.rnu.tn

***Abstract*—Model-Driven Engineering (MDE) based approaches have been proposed as a solution to cope with the inefficiency of current design methods. In this context, this paper presents an MDE-based framework for rapid SIMD (Single Instruction Multiple Data) parametric parallel SoC (System-on-Chip) prototyping to deal with the ever-growing complexity of such embedded systems design process. The design flow covers the design phases from system-level modeling to FPGA prototyping. The proposed framework allows the designer to easily and automatically generate a VHDL parallel SoC configuration from a high-level system specification model using the MARTE (Modeling and Analysis of Real-Time and Embedded systems) standard profile. It is based on an IP (Intellectual Property) library and a basic parallel SoC model. The generated parallel configuration can be adapted to the data-parallel application requirements. In an experimental setting, four steps are needed to generate a parallel SoC: data-parallel programming, SoC modeling, deployment and generation process. Experimental results for a video application validate the approach and demonstrate that the proposed framework facilitates the parallel SoC exploration.**

I. Introduction

With the rising complexity of multimedia and radar/sonar signal processing applications, parallel programming techniques and multi-core Systems-on-Chip (SoC) are more and more used. Single Instruction Multiple Data (SIMD) systems have shown to be powerful executers for data-intensive applications [1], especially in pixel processing domain [2]. Many SIMD on-chip architectures, in particular based on FPGA (Field Programmable Gate Arrays) devices, have emerged to accelerate specific applications [3]–[6]. Compared to ASIC (Application Specific Integrated Circuit), FPGA devices are characterized by an increased capacity, smaller non-recurring engineering costs, and programmability [7]. Dealing with the ever-growing challenge of parallel SoC design, most of the proposed SIMD solutions are application-specific SoC which lack flexibility: changing a SoC configuration may necessitate extensive redesign. While these specific systems provide good performances, they require long design cycles. The size of a parallel SoC and the complexity involved in its design are continuously outpacing the designer productivity. An important challenge is to find adequate design methodologies that efficiently address the issues about large and complex SoC.

Nowadays, Computer-Aided Design tools are imperative to automate complex SoC design and reduce the time-to-market. Two approaches have been proposed to cope with this problem. Firstly, IP (Intellectual Property) reuse and platform-based design [8] are used to maximize the reuse of pre-designed components and to allow the customization of the system according to system requirements. Secondly, Model-Driven Engineering (MDE) [9] approach has been introduced to raise the design abstraction level and to reduce design complexity. It stresses the use of models in the embedded systems development life cycle and argues automation via model transformation and code generation techniques. Complex systems can be easily understood thanks to such abstract and simplified representations. Approaches based on MDE have been proposed as an efficient methodology for embedded systems design [10], [11]. An interesting model specification language is UML (Unified Modeling Language) [12], which proposes general concepts allowing expressing both behavioral and structural aspects of a system. The latest release of UML (2.0) has support for profiles that enable the language to be applied on particular application and platform domains with sophisticated extension mechanisms. As an example, the MARTE (Modeling and Analysis of Real-Time and Embedded systems) standard profile [13] is proposed by the OMG to add capabilities to UML for model-driven development of real-time and embedded systems. The MARTE profile enhances possibility to model SW, HW and relations between them.

Using the proposed framework, the designer focuses on modeling his needed SIMD configuration and not on how implementing it, since the system modeling is independent of any implementation detail. Specifying a model is written based on unified language. The presented design flow is a library-based method that hides unnecessary details from high-level design phases and provides an automated path from UML design entry to FPGA prototyping. So, it can be easily used by non-HW experts of on-chip systems implementation. This makes our approach better than using some clever VHDL coding.

System concerns are represented in separated dimensions:

978-1-4577-0660-8/11/$26.00

data-parallel coding, SoC modeling, IP selection and implementation. The implementation is performed via the generation tool based on a model-to-text transformation using Acceleo [14]. The framework uses an IP library with various components (processors, memories, interconnection networks...) that can be selected in the deployment process to generate the needed SIMD configuration. The modeled SoC has to be conform to a basic parallel SoC model which is parametric, flexible and programmable, proposed in previous work [15].

In an experimental setting that validates our approach, we consider a video color conversion application where we explore different parallel system configurations and decide the best one to run the application. Experimental results show that the proposed framework considerably reduces design costs and facilitates modifying the system model and regenerating the implementation without relying on costly re-implementation cycles. Using the framework, we can create SIMD implementations that are fast enough to meet demanding processing requirements, are automatically generated from a high-level specification model to reach the time-to-market and can easily be updated to provide a different functionality.

The remaining of this paper is organized as follows. Section 2 discusses related work on model-based approaches to generate on-chip multi-processor or massively parallel systems. Section 3 presents the proposed MDE framework.A case study, which illustrates and validates the framework, is described in Section 4. The FPGA platform is chosen as a target platform since it is a better alternative to test and implement various parallel SoC configurations. Finally, Section 5 draws main conclusions and proposes future research directions.

II. Related work

The high-level SoC design methodology is a rapid emerging research area. There are many recent research efforts on embedded systems design using an MDE approach. In this context, different high-level synthesis approaches are currently being studied for different specification languages. For example, xtUML [11] defines an executable and translatable UML subset for embedded real time systems, allowing the simulation of UML models and the code generation for C oriented to different microcontroller platforms. In [16], 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 [17], a transformation tool, called MODCO, which takes a UML state diagram as input and generates HDL output suitable for use in FPGA circuit design, is presented. A HW/SW co-design is performed based on the MDA approach. XML is used to generate HDL from high-level UML diagrams. In these two works, only state machines HW designs are described. In [18], a UML-based multiprocessor SoC design framework, called Koski, is described. An automated architecture exploration based on the system models in UML, as well as the automatic back and forward annotation of information in the design flow could be performed. The proposed design flow provides an automated path from UML design entry to FPGA prototyping. The final implementation is application-specific. The proposed approach is based on synthesizable library components that are automatically tuned for specific application according to the results of the architecture exploration.

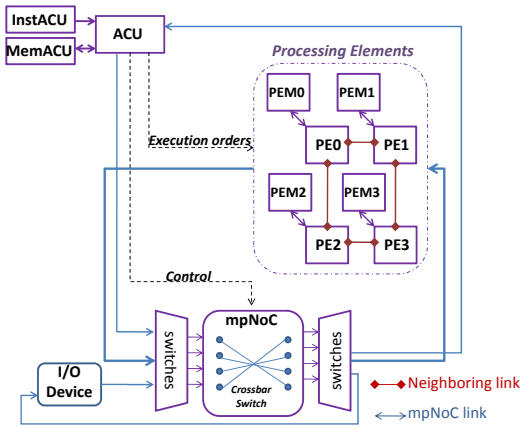

Fig. 1. Parallel SIMD SoC configuration: 4 PEs, a 2D mesh neighboring network and a crossbar based mpNoC

Our approach is related to the design of massively parallel SoC and covers the design phases from system-level modeling and parallel programming to FPGA prototyping using the notion of transformations between models. The DaRT [10] (Data Parallelism to Real Time) project also proposes an MDA-based approach for SoC design that has many similarities with our approach in terms of the use of meta-modeling concepts. The DaRT work defines MOF-based meta-models to specify application, architecture, and SW/HW association and uses transformations between models as code transformations to optimize an association model. In DaRT, no data-parallel coding is specified and the code generation for RT (Register Transfer) levels is dedicated to specific HW accelerators.

The proposed framework, presented in this paper, takes advantage of the MDE notion of transformation between models to generate a complete SIMD parallel SoC at RT level dedicated to compute data-intensive applications. Our approach is based on synthesizable library components and few model transformations to generate the synthesizable VHDL code of the modeled SIMD SoC.

III. SIMD Framework

The proposed framework is dedicated to generate different SIMD configurations derived form the based parallel SoC model [15]. These configurations can be then directly simulated using available simulation tools or prototyped on FPGA devices using appropriate synthesis tools. Figure 1 illustrates a SIMD parallel SoC configuration composed of four Processing Elements (PE) connected in a 2D mesh topology. To handle parallel I/O transfers and point-to-point communications, a crossbar based mpNoC (massively parallel Network on Chip) [19] is integrated. To accelerate and facilitate a SIMD configuration design, a model-driven framework is proposed. The framework allows the designer to model his needed configuration derived from the basic provided SIMD SoC model. He

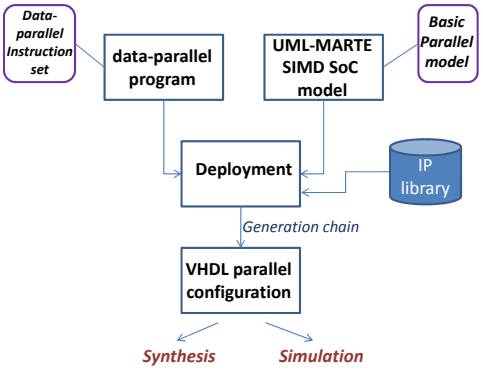

Fig. 2. Framework concepts

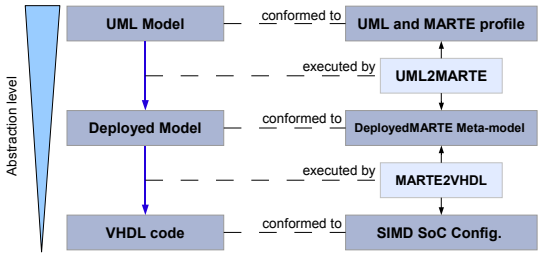

Fig. 3. MDE-based design flow

has to specify the system's parameters (number of PE, memory size, neighboring topology) and the different components that will be integrated (mpNoC, neighborhood network, devices). The designer has also to code his data-parallel program using the specified data-parallel instruction set depending on the chosen processor IP. A help manual is in fact provided to the designer to facilitate the parallel programming and describe the different instructions to use according to the chosen processor.

The framework, in particular the deployment phase, is based on an IP library which contains dedicated IP that can be directly integrated in the system. Providing an extensive library requires a significant effort. Currently, the IP library contains processors (MIPS, OpenRisc, NIOS II), networks (crossbar, shared bus and multi-stage networks), memories and some devices. To add new IP resources, the IP provider must adapt the IP to the architecture dedicated specific interface (described in the help manual). Thus, a new component can be put into the library by following the requirements for interfaces formats. To assemble processors in the SIMD design, we distinguish two methodologies: reduction and replication. The reduction consists on reducing an available processor in order to build a PE with a small reduced size that can be fitted in large quantities into an FPGA device. The replication consists on implementing the ACU as well as the PE by the same processor IP so that the design process is faster. We clearly notice that there is a compromise between the design time and the number of integrated PEs in the SIMD configuration depending on the applied design methodology. The designer can select the suitable methodology according to his application constraints. The three processors of the IP library are provided with the two methodologies.

At this step, the designer can generate different implementations while integrating different IPs. The deployment is also responsible of loading the binary data-parallel program in the ACU instruction memory. The SIMD generation approach is depicted in Figure 2. This approach allows a flexible and rapid platform development and platform end-user productivity.

To generate SIMD configuration at RT level, an MDE based design flow, presented in Figure 3, is developed. The proposed flow uses two meta-models: the MARTE meta-model and the Deployed meta-model. All meta-model concepts are specified as UML classes and then converted into Eclipse/EMF models [20]. The generation process is based on model transformations implemented as QVT (Query, Views, Transformations) resources, standardized by OMG.

The designer can generate a SIMD massively parallel SoC configuration in four steps: data-parallel programming, SoC modeling, deployment and then implementation generation.

A. Data-parallel programming

The designer has to write his data-parallel program using the provided data-parallel instruction set. Based on available processor compilers (miniMIPS, OpenRisc 1200 and NIOS II) in the IP library and the developed special parallel instructions, the designer can generate his parallel program's binary. For the miniMIPS processor, an extended parallel MIPS assembler language [21] is developed. For the OpenRisc and NIOS processors, high-level asm macros are defined and they can be used in any C program for control and communication instructions. The NIOSII IDE (Integrated Development Environment) and the OR1Ksim [22] tools are used respectively with the NIOS and OpenRisc processors. The developed SW chain is a multi-compiler chain that is responsible of generating the SW code depending on the specified target processor.

Some particular instructions are specified to be used in the programs as delimiters for parallel and sequential code. Table I shows three examples of instructions from the provided data-parallel instruction set. It is clearly that these instructions depend on the processor instruction set. At this step, a SW library is provided. It includes pre-implemented application algorithms such as matrices multiplication, FIR (Finite Impulse Response) filter, reduction algorithm, image rotation, color conversion (RGB to YIQ, RGB to CMYK), etc.

After generating the executable SW, the second step consists on modeling the HW system.

B. SoC modeling

The designer must specify the architecture models using any UML 2.0 compliant tool with applying the MARTE profile. The most important UML diagrams used in our approach to specify the system are Class, Structure composite and Deployment diagrams. The modeling of SIMD SoC configurations relies on the use of UML and the MARTE profile. Three MARTE packages are used: the Hardware Resource Modeling (HRM), the Repetitive Structure Modeling (RSM) and the Generic Component Model (GCM) packages [23]. The HRM intends to describe the HW platform by specifying its different elements. At the end, the HW modeled resources present the whole system. In our approach, only the HRM HW_Logical

TABLE I
SIMD PARALLEL MACROS

ASM Macro	Description	Coding		
		miniMIPS	OpenRisc	NIOS
P_REG_SEND (reg,dir,dis,adr)	Neighboring SEND: send data (in reg) from source to destination via the neighboring network.	p_addi r1,r0,dir p_addi r1,r1,dis p_addi r1,r1,adr p_SW reg,0(r1)	l.addi r1,r0,dir l.addi r1,r1,dis l.addi r1,r1,adr l.sw 0x0(r1),reg	IOWR (WRP_B,addr, data) Where: addr(11)='0' and addr(10:3)=dis and addr(2:0)=dir.
P_REG_REC (reg,dir,dis,adr)	Neighboring RECEIVE: receive data (in reg) from the source.	p_addi r1,r0,dir p_addi r1,r1,dis p_addi r1,r1,adr p_LW reg,0(r1)	l.addi r1,r0,dir l.addi r1,r1,dis l.addi r1,r1,adr l.lwz reg,0x0(r1)	data=IORD (WRP_B,addr) Where: addr(11)='0' and addr(10:3)=dis and addr(2:0)=dir.
P_GET_IDENT (reg)	read identity	p_lui r1,0x2 p_ori r1,r1,0 p_LW reg,0(r1)	l.movhi r1,0x2 l.lwz reg,0x0(r1)	NIOS2_READ_CPUID(id)

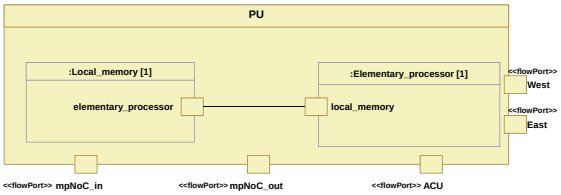

Fig. 4. PU modeling in the case of a linear configuration

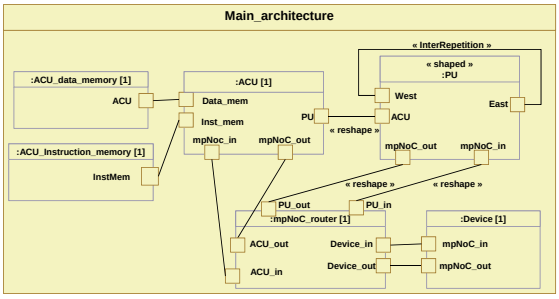

Fig. 5. 1D configuration modeling

sub-package is used. It allows to describe information about the kind of components (HwRAM, HwProcessor, HwBus, etc.), their characteristics, and how they are connected to each other. The architecture is graphically specified at a high abstraction level with HRM. Multidimensional data arrays and powerful constructs of data dependencies are managed thanks to the use of the RSM package. It defines stereotypes and notations to describe in a compact way the regularity of a system's structure or topology. The structures considered are composed of repetitions of structural elements interconnected via a regular connection pattern. It provides the designer a way to efficiently and explicitly express models with a high number of identical components. The concepts found in this package allow to concisely model large regular HW architectures as multi-processor architectures. Finally, the GCM package is used to specify the nature of flow-oriented communication paradigm between SoC components.

The modeling process is done in an incremental way. The designer begins by modeling the elementary components: PE, ACU, memories, mpNoC and I/O device. Then, the whole configuration is modeled through successive compositions. Figure 4 illustrates the elementary processing unit (PU). It is composed of a PE and its local data memory. The class named "Elementary_processor" is stereotyped *HwResource* in the case of the reduction methodology or *HwProcessor* in the case of the replication methodology. It has a bidirectional port stereotyped *FlowPort* to connect the data memory. The class "Local_memory" is stereotyped *hwMemory* with a parametric tagged value *adressSize*. In the same manner, the ACU memories have a parametric size. The PU has one port to communicate with the ACU and a number of neighboring ports equal to the number of its neighboring connections. In Figure 4, it has two neighboring ports since each PE can communicate with its neighbor in the east or west directions. If the designer chooses to integrate the mpNoC in the SIMD configuration, he must add two ports "mpNoC_in" and "mpNoC_out" to assure the communications through the mpNoC.

We distinguish between 1D and 2D mppSoC configurations. They differ in the modeling of the interconnections between PUs. In the case of 1D configuration, the number of PEs is equal to the tagged value *Shape* of the stereotype *Shaped* applied on the PU class. To model a linear neighboring network, the interconnection link between the East and West ports is stereotyped *InterRepetition*. Since the PU on the edge is not connected to the PU on the opposite edge, the tagged value *isModulo* is set to false. The *repetitionSpaceDependence* attribute is used to specify the neighbor position of the element on which the inter-repetition dependency is defined. In this case, its value is equal to $\{1\}$ since each PE[i] is connected to PE[i+1]. Figure 5 shows the mppSoC configuration modeling integrating a linear neighboring network and the mpNoC. The link connector, stereotyped *Reshape*, between the PU and the ACU shows that each PU is connected to the ACU in order to receive the execution orders. To connect PUs with the mpNoC, two *Reshape* connectors are expressed between the two ports of the PU and the corresponding ports of the mpNoC. This latter has a multiplicity equal to 1. The *repetitionSpace* tag is equal to the number of PEs. The *patternShape* tag is equal to 1 indicating that the mpNoC port is distributed among the ports of the PEs. The same modeling is followed in the case of a ring neighboring network. The only difference is the modulo tagged value which is set to true.

In the same manner, we can model a 2D SIMD configuration. We need just to know how to model the neighboring links based on the MARTE profile. Figure 6 presents a configuration

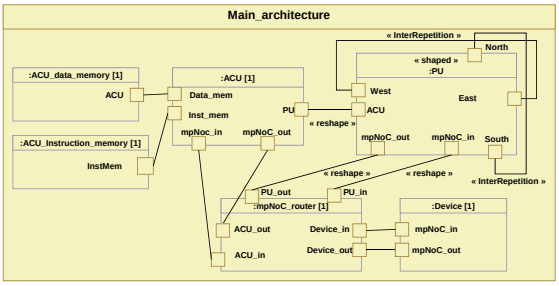

Fig. 6. 2D configuration modeling (with a mesh neighboring network)

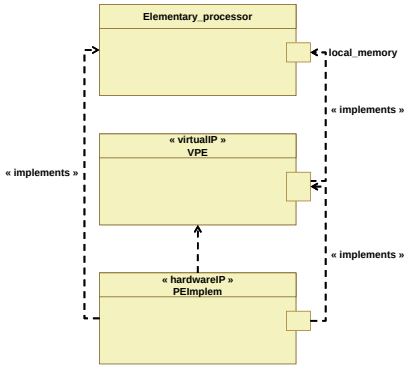

Fig. 7. Deployment of the PE

modeling integrating a 2D mesh neighboring network. We notice that the PU class is modeled with 4 ports dedicated to inter-PE communications in east, west, north and south directions. In this case, the *repetitionSpaceDependence* tagged value is equal to $\{1,0\}$ indicating that each PE[i,j] is connected to its neighbor PE[i+1,j] to assure east and west links. In addition, this tagged value is equal to $\{0,1\}$ for north and south links to assure that each PE[i,j] is connected to its neighbor PE[i,j+1]. For a mesh topology the tagged value *isModulo* is set to false since there are no connections on the edges. However, it is set to true in the case of a torus topology. The Xnet network is modeled like the 2D mesh. The designer has just to model the links on the diagonals.

C. Deployment

As described in the previous subsection, a SIMD configuration can be modeled at a high abstraction level. To generate an executable low level model, the elementary modeled components should be associated with an existing implementation based on the provided IP library. The deployment allows to move from a general platform (Platform Independent Model) to a specific platform (Platform Specific Model) according to the MDA approach. At this step, the designer can generate and evaluate different configurations. In fact, the deployment enables to precise a specific implementation for each elementary concept among a set of possibilities. It concerns the processor IP, the instruction memory, the mpNoC interconnection network and the I/O device IP if it exists. At this stage the binary data-parallel program is specified as the memory initialisation file of the main instruction memory. In fact, in our case we deal with a single data-parallel program (one of the advantages of a SIMD architecture) so no mapping of tasks needs to be performed. Thus, the mapping of the application to the hardware architecture is systematic. Figure 7 expresses the deployment of a "hardwareIP" on the "Elementary_processor". The concept of *codeFile* is used to specify the code.

A final transformation chain MARTE2VHDL is developed to generate the synthesizable VHDL implementation of the modeled SIMD configuration.

D. Implementation generation

The MARTE2VHDL transformation is based on the Deployed model and the IP library to generate the corresponding synthesizable VHDL implementation depending on the modeled configuration. A model conformed to the Deployed metamodel is generated via the transformation UML2MARTE. This model is then analysed in order to deduce the specified parameters. The number of PEs, memory size, processor design methodology and the topology of the neighboring network are extracted from the UML diagrams. The other configurable components (processor IP, mpNoC interconnection network, etc.) are specified from the deployment step. The developed transformation model-to-text is based on templates. It uses the Acceleo tool [14] which is part of the Eclipse Model to Text project and provides an implementation of the MOF Model to Text OMG standard. The following code example illustrates how to deduce the type of the processor (*getPeCodeFile*) in the generation step:

[*query public getPECodeFile (m:Model):CodeFile=self.ownedElement->select (oclIsKindOf(CodeFile) and name='PEImpl_codefile')->asOrderedSet()->first()*]

Using an MDE based framework, the SIMD SoC design is accelerated. The VHDL implementation can be automatically generated based on model transformations. The SoC model is independent of any implementation detail making the design flow easy to use. The proposed framework also facilitates SoC exploration and helps the user choose the best configuration for a given application. The next section illustrates the use of this framework in a real application context.

IV. Case study

A color conversion RGB to CMYK application widely used in color printers, extracted from the EEMBC benchmark [24] has been developed based on the provided data-parallel instruction set. The program is written using high-level macros (table I). The binary is then generated depending on the used processor by selecting the corresponding compiler. The proposed framework allowed to generate different SIMD suitable configurations. An FPGA is used to do real experimentations.

A. HW platform

The used development board is the Altera D2-70 [25] equipped with a CycloneII EP2C70F896C6 FPGA which has 68416 Logic Elements (LEs) and 250 M4K RAM blocks. The used SW tools are the Quartus II V9.0 that allows synthesizing

TABLE II
SYNTHESIS RESULTS

PE	IP	Proc. design	LEs (%)	Memory ACU (bytes)	Memory PE (bytes)	Memory %
8	miniMIPS	rep.	71	4096	1024	18
32	miniMIPS	red.	93	4096	2048	66
8	OpenRisc	rep.	91	4096	1024	22
16	OpenRisc	red.	98	4096	4096	36
48	NIOS	rep.	79	8192	512	87

and prototyping the design on the FPGA, and the ModelSim-Altera v.6.4a that allows simulating and debugging the design. To test the color-conversion application, two peripherals are used: a 1M pixel camera TRDB D5M and a 800×RGB×480 pixel TRDB LTM LCD displayer. The two external SDRAM and SRAM memories are also used. In fact, the implemented VHDL camera driver directly stores the captured data to the SDRAM to be read by PEs as required. A VHDL SRAM controller is implemented. It allows to store the processed data in the SRAM and fetches it as it is required by the LCD.

B. SIMD configurations

For the tested application, only the mpNoC has been integrated in the system model (no neighboring network) since we need to assure parallel data transfers: all PEs need to read data from the SDRAM and then write data to the SRAM. In this example, each pixel processing should not exceed 10.42 Ns in order to assure real-time processing. Therefore, a 800×480 pixel frame must be processed within 4 Ms. The same system model is used for all implementation generations. It is described in composite structure diagram as illustrated in Figure 8. It models all hardware components composing the system as well as their connections. Only the deployment diagram changes from one configuration to another in order to use different processors, memories and interconnection networks. The modification from one SIMD configuration to a new one just needs few milliseconds and the re-generation process is rapidly performed. The low-level synthesizable models from the IP library are used for the final implementation. The generated configurations could be directly simulated to measure execution time and decide the performance of the SIMD modeled systems.

Table II shows the obtained synthesis results varying the SIMD parameters and components while integrating the maximum number of PEs targeting the Cyclone II FPGA. All these configurations integrate a crossbar based mpNoC since the crossbar allows fast and non-blocking parallel data transfers, necessary for real-time image processing applications. We clearly notice that the reduction methodology allows integrating a bigger number of PEs on the chip than the replication methodology. Since the miniMIPS is smaller than the OpenRisc, we can reach 32 PEs on the FPGA compared to 16 PEs when using the OpenRisc IP. The implementation results prove that the NIOS processor is optimized for the Altera FPGA. We can integrate more than 48 PEs on the chip.

Figure 9 shows the execution time results obtained when prototyping the generated configurations on the CycloneII

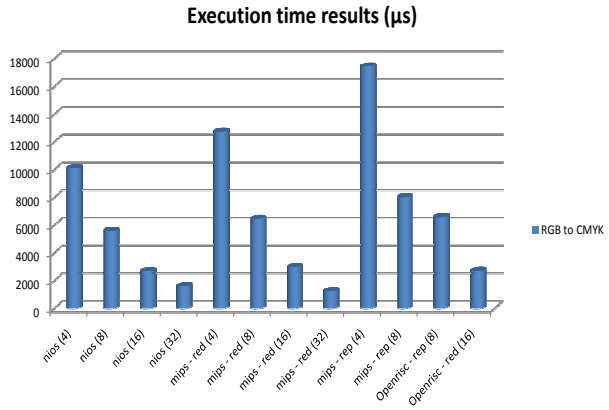

Fig. 9. Execution times for different SIMD configurations

TABLE III
DIFFERENCES BETWEEN TWO DESIGN SOLUTIONS

SIMD config.	Generic implem. with a reduced processor	Generic implem. with a replicated processor
Design time using the framework	15 minutes	40 seconds
Design time without using the framework	1 month	7 days

FPGA. So, these times are measured running the color-conversion application on parallel FPGA based configurations. The different SIMD SoC configurations perform good results while increasing the number of PEs working in parallel. The performance of the system is also closely related to the processor type and the design methodology. The experimental results show that a SIMD configuration composed of more than 8 PEs is needed to assure real-time processing. According to these results, we can choose the best configuration. The proposed approach easily allows exploration of several platform architecture alternatives.

In order to illustrate the efficiency of the model-based framework, Table III compares the implementation design time using the framework with results obtained from a conventional manual implementation method done by the same designer without using any framework. The measured design time for the second configuration (using replication methodology) is just the time needed to modify the first configuration (with reduction methodology). The results in Table III show that the proposed framework is a better solution to accelerate the design of specific SIMD parallel SoC according to the estimated design time compared to a manual design. Two months were necessary to reduce an open-source processor to obtain a small PE (with only execution units) [21]. Observing the results, we can conclude that the model-based design framework allows a very fast SIMD implementation.

This case study illustrates a design framework which facilitates SIMD SoC implementation to run data-parallel applications. Through the Model-Driven Engineering approach for parallel SoC design presented in this work, a designer can specify the needed SIMD configuration using UML models and the MARTE profile at high abstraction level and automatically generate its implementation at RT level. The designer can easily and rapidly generate different SoC configurations

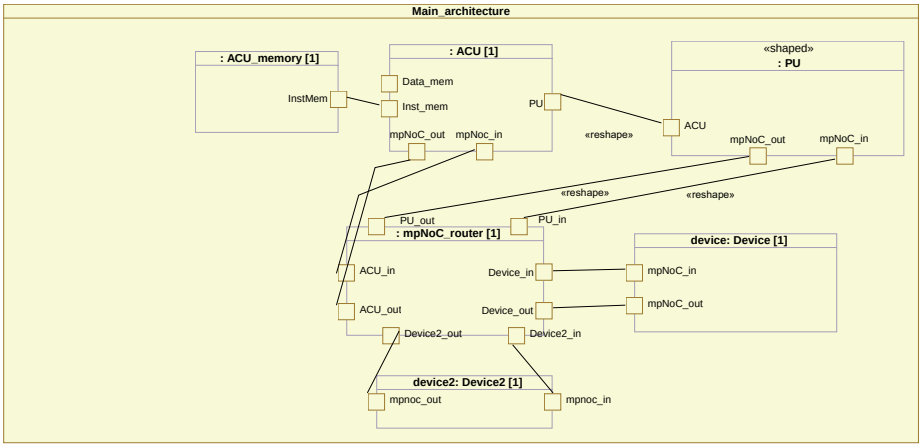

Fig. 8. SIMD configuration composite structure diagram

to look for the best alternative for a given application.

V. Conclusions and Future Work

A Model-Driven Engineering (MDE) approach for SIMD SoC design was presented. The proposed flow design is composed of four steps: application programming, system modeling, deployment and then implementation generation. The MDE fundamental notion of transformation between models is used to generate a SIMD configuration at register transfer level from its model at a high abstraction level. The framework facilitates the exploration by rapidly generating different SoC configurations in order to choose the most adequate one that better fulfills the application requirements. Experimental results show that the proposed framework strongly contributes to the increase of the designer's productivity. The case study with a video processing application proved that the presented design flow can facilitate the design of parallel SIMD SoC systems. The design flow allows reducing implementation costs. Besides, the use of UML and MDE promotes the reusability of application and system high-level models.

One of the future directions to be considered is the modeling of a data-parallel application. We also intend to develop a high-level exploration step to automatically generate the most suitable application-specific SIMD SoC configuration.

References

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

[2] R. Kleihorst and al., "An SIMD smart camera architecture for real-time face recognition," in *Abstracts of the SAFE & ProRISC/IEEE Workshops on Semiconductors, Circuits and Systems and Signal Processing*, 2003.

[3] R. Rosas, A. de Luca, and F. Santillan, "SIMD Architecture for Image Segmentation using Sobel Operators Implemented in FPGA Technology," in *Proc. of the 2nd International Conference on Electrical and Electronics Engineering ((ICEEE'05)*, 2005.

[4] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, O. Ruch, and P. Gauget, "Definition and SIMD implementation of a multi-processing architecture approach on FPGA," in *Proc. of DATE*, 2008.

[5] F. Schurz and D. Fey, "A Programmable Parallel Processor Architecture in FPGA for Image Processing Sensors," in *Integrated Design and Process Technology, IDPT*, 2007.

[6] X. Xizhen and S. G. Ziavras, "H-SIMD machine: configurable parallel computing for matrix multiplication," in *International Conf. on Computer Design: VLSI in Computers and Processors*, 2005, pp. 671–676.

[7] P. Paulin, "DATE panel: Chips of the future: soft, crunchy or hard?" in *Proc. Design, Automation and Test in Europe*, 2004, pp. 844–849.

[8] A. Sangiovanni-Vincentelli, L. Carloni, F. D. Bernardinis, and M. Sgroi, "Benefits and challenges for platform-based design," in *Proc. DAC*, 2004, pp. 409–414.

[9] D. Schmidt, "Model-driven Engineering," *IEEE Computer*, vol. 39, no. 2, 2006.

[10] C. D. L. Bond and J.-L. Dekeyser, "Metamodels and MDA transformations for embedded systems," in *FDL04*, Lille, France, 2004.

[11] S. Mellor and M. Balcer, *Executable UML: A foundation for Model Driven Architecture*. Boston: Addison-Wesley, 2002.

[12] O. M. Group. (2004, october) Uml 2 superstructure (available specification). [Online]. Available: http://www.omg.org/cgi-bin/doc?ptc

[13] L. Rioux, T. Saunier, S. Gerard, A. Radermacher, R. de Simone, T. Gautier, Y. Sorel, J. Forget, J.-L. Dekeyser, A. Cuccuru, C. Dumoulin, and C. Andre, "MARTE: A new profile RFP for the modeling and analysis of real-time embedded systems," in *UML-SoC'05, DAC 2005 Workshop UML for SoC Design*, Anaheim, CA, June 2005.

[14] Acceleo. (2009). [Online]. Available: http://www.acceleo.org

[15] M. Bakouti, P. Marquet, M. Abid, and J.-L. Dekeyser, "IP based configurable SIMD massively parallel SoC," in *PhD Forum of* 20^{th} *International Conference on Field Programmable Logic and Applications (FPL)*, Milano, Italy, August 2010.

[16] D. Bjorklund and J. Lilius, "From UML Behavioral Models to Efficient Synthesizable VHDL," in 20^{th} *IEEE NORCHIP Conference*, Copenhagen, Denmark, November 2002.

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

[18] T. Kangas, P. Kukkala, H. Orsila, E. Salminen, M. Hannikainen, and T. Hamalainen, "UML-based multiprocessor SoC design framework," *ACM Trans. Embedded Computing Systems (TECS)*, vol. 5, no. 2, pp. 88–93, May 2006.

[19] M. Bakouti, Y. Aydi, P. Marquet, M. Abid, and J.-L. Dekeyser, "Scalable mpNoC for Massively Parallel Systems - Design and Implementation on FPGA," *Journal of Systems Architecture (JSA)*, vol. 56, pp. 278–292, 2010.

[20] EMF. Eclipse modeling framework. [Online]. Available: http://www.eclipse.org/emf

[21] M. Bakouti, P. Marquet, M. Abid, and J.-L. Dekeyser, "A design and an implementation of a parallel based SIMD architecture for SoC on FPGA," in *Conference on Design and Architectures for Signal and Image Processing DASIP'08*, Bruxelles, Belgium, November 2008.

[22] OpenCores. Or1200 openrisc processor. [Online]. Available: http://opencores.org/openrisc,or1200

[23] O. M. Group. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Sys- tems, version 1.0. [Online]. Available: http://www.omg.org/spec/MARTE/1.0/PDF/.

[24] EEMBC. (2010) The Embedded Microprocessor Benchmark Consortium. [Online]. Available: http://www.eembc.org/home.php

[25] Terasic. (2010) Altera DE2-70 Board. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=226