Modeling reconfigurable Systems-on-Chips with UML MARTE profile: an exploratory analysis

Sana Cherif, Imran Rafiq Quadri, Samy Meftali, Jean-Luc Dekeyser

Modeling reconfigurable Systems-on-Chips with UML MARTE profile: an exploratory analysis

Sana Cherif*, Imran Rafiq Quadri*, Samy Meftali* and Jean-Luc Dekeyser*
*INRIA Lille Nord Europe - LIFL - USTL - CNRS, 40 avenue Halley, 59650 Villeneuve d'Ascq, FRANCE
Email:{Firstname.Lastname}@inria.fr

***Abstract*—Reconfigurable FPGA based Systems-on-Chip (SoC) architectures are increasingly becoming the preferred solution for implementing modern embedded systems, due to their flexible nature. However due to the tremendous amount of hardware resources available in these systems, new design methodologies and tools are required to reduce their design complexity. In this paper we present an exploratory analysis for specification of these systems, while utilizing the UML MARTE (Modeling and Analysis of Real-time and Embedded Systems) profile. Our contributions permit us to model fine grain reconfigurable FPGA based SoC architectures while extending the profile to integrate new features such as Partial Dynamic Reconfiguration supported by these modern systems. Finally we present the current limitations of the MARTE profile and ask some open questions regarding how these high level models can be effectively used as input for commercial FPGA simulation and synthesis tools. Solutions to these questions can help in creating a design flow from high level models to synthesis, placement and execution of these reconfigurable SoCs.**

I. Introduction

Modern SoCs are considered as an integral solution for designing embedded systems. From avionics, transport, defense, medical and telecommunication systems to general commercial appliances such as smart phones, high definition TVs, gaming consoles; SoCs are now omnipresent, and it is difficult to find a domain where these miniaturized systems have not made their mark. In a SoC, the computing units such as programmable processors, memories, interconnection networks and I/O devices, are all integrated into a single silicon chip. A large number of these SoCs are generally dedicated to application functionalities that require intensive data-parallel computations: such as software-defined radio, radar/sonar detection systems and multimedia video codecs. As compared to general parallel applications that focus on code parallelization, the targeted SoC data-parallel applications concentrate on regular data partitioning, distribution and their access to data.

Reconfiguration can be seen as an integral feature of modern SoCs based embedded systems, in an increasingly evolving market space. A reconfigurable SoC offers increased functional extensibility in return for lower performance. These systems can be reconfigured an arbitrary number of times and offer designers the means to add new functionalities and make system modifications after the fabrication of a SoC. *Dynamic reconfiguration*, which is a special type of reconfiguration [1], enables system modification at run-time, introducing the concept of *virtual hardware*. Thus designers can change the executing applications on these systems, depending upon Quality-of-Service (QoS) criteria related to the environment or the platform: such as consumed surface area, energy consumption levels, etc. Currently, *Field Programmable Gate Array* (FPGA) based SoCs offer an ideal solution for implementing dynamic reconfiguration. Moreover, SoC application functionalities can be easily implemented as hardware designs on these reconfigurable SoCs. As compared to traditional SoCs, these dynamically reconfigurable SoCs offer advantages such as low energy consumption, increased flexibility; with the compromise of additional costs per unit.

In the wake of the continuous hardware/software evolution related to SoCs and the addition of features such as dynamic reconfiguration, the complexity of design and development of SoC has escalated to new heights in an exponential manner. If more hardware components are integrated, or an application is deemed to provide more features, development costs and time to market shoot up proportionally. Without the usage of effective design tools and methodologies, large complex SoCs are becoming increasingly difficult to manage, resulting in a *productivity gap*. The design space, representing all technical decisions that need to be elaborated by the SoC design team is therefore, becoming difficult to explore. Similarly, manipulation of these systems at low implementation levels such as *Register Transfer Level* (RTL) can be hindered by human interventions and the subsequent errors.

Currently, we are therefore faced with a need to design more effective SoCs. Various methodologies and propositions have been proposed to reduce SoC design complexity. A *Platform* or *component* based approach is widely accepted in the SoC industry, permitting system conception and eventual design in a compositional manner. The hierarchy related to the SoC is visible quite clearly, and designers are capable to reutilize components that have been either developed internally or by third parties. Other methodologies make use of high abstraction levels at different design levels, in order to elevate the low level technical details.

Unified design approach is an emerging research topic for addressing the various issues related to SoC Co-Design. High level SoC co-modeling design approaches have been developed such as Model-Driven Engineering or MDE [2]. MDE enables high level system modeling of both software and hardware, with the possibility of integrating heterogeneous components into the system. *Model transformations* [3] can then be carried out to generate executable models from the

high level models. MDE is supported by several standards and tools.

The UML MARTE (Modeling and Analysis of Real-Time and Embedded Systems) profile is an upcoming industry standard of Object Management Group (OMG) [4], that is dedicated to model-driven development of embedded systems. MARTE extends UML, in order to model the features of software and hardware parts of a real-time embedded system and their relations, along with added extensions (for e.g. performance and scheduling analysis). Although rich in concepts, MARTE lacks concepts for the complete specification of reconfigurable SoCs.

The contributions of this paper relate to presenting an overview of the current MARTE concepts for the specification of reconfigurable SoCs. Our exploratory analysis illustrates that the current version of the profile is inadequate for the detailed description of these systems. We then present some extensions in the profile to solve this issue and then ask some open questions on how we can create a link between the high level UML models of reconfigurable SoCs and the commercial tools that are eventually used by a SoC designer to implement and execute these systems.

The plan of the paper is as follows. Section 2 describes an overview of reconfigurable FPGA based SoCs, followed by an overview of the MARTE profile and the concepts for specification of reconfigurable architectures. Section 4 describes the limitations in MARTE and illustrates our initial attempts to model SoCs using the MARTE profile. Section 5 gives the related works followed by a conclusion.

II. Reconfigurable Systems-on-Chips

The main difference between a classical SoC architecture and a reconfigurable SoC is the presence of reconfigurable areas: i.e., if FPGAs or *Complex programmable logic devices* (CPLDs) are part of the Systems-on-Chip device. Others may be design specific only: these types of SoCs are thus implemented as *Application Specific Integrated Circuits* (ASICs). Classical ASIC based SoCs are normally designed to execute only one application with very tight performance constraints (latency, area, power consumption, throughput). Whereas reconfigurable SoCs are designed to execute different applications relying on same hardware capabilities. They thus introduce the notion of *virtual hardware*. Similarly in terms of fabrication, ASIC based solutions produce an extremely costly SoC with a long time-to-market requiring intervention by different teams and multiple designers, resulting in introduction of errors in the design cycle. The alternative solution is the use of FPGAs for construction of the reconfigurable SoCs.

Figure 1 shows an example of a complex reconfigurable SoC used in the European *MORPHEUS* (Multi-purpOse dynamically Reconfigurable Platform for Intensive HEterogeneoUS processing) project. This 100 mm^2 90 nanometer RSoC is composed of 97 million transistors along with an ARM9 microprocessor and three reconfigurable architectures (DREAM PiCoCA, Abound Logic eFPGA, Pact XPP matrix), memories, buses, Network-on-Chips etc. The complexity of such a large complex SoC architecture necessitates the utilization of an effective and efficient SoC design methodology.

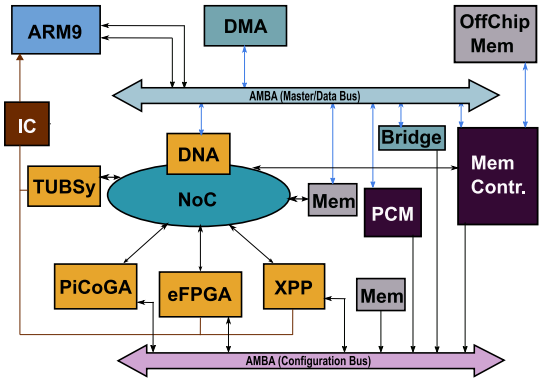

Figure.1: Morpheus: a reconfigurable SoC

A. Advantages and disadvantages

Reconfigurable FPGA based SoC designs have two main features that distinguish them from traditional SoC designs. The first is that the hardware functionality can be switched by modifying the corresponding executing configuration in the FPGA. So, a SoC can contain a digital-to-analog converter for one application, reconfigured for a analog-to-digital converter for another application; or even a completely different peripheral such as a network device.

The second advantage is an offshoot of the first. Some elements of the reconfiguration can be performed at run-time once the initial configuration has been loaded allowing to handle issues related to fault tolerance and system performance. Also, reconfiguration can be carried out an arbitrary number of times after loading of the initial FPGA configuration.

FPGA based reconfigurable SoCs have minimal upfront costs as compared to custom designs implemented on ASICs. Design costs are reduced because changes can be immediately made to the chip during development phases before final fabrication. They are thus a popular choice for prototyping as designers can initially implement, and afterwards, reconfigure a complete SoC for the required customized solution. Thus these prototypes offer a path for final customized ASIC or SoC implementation. Similarly, chip simulation becomes less tiresome as the real hardware is available immediately. Other advantages include task swapping depending upon application needs, overcoming hardware limitations and Quality-of-Service (QoS) requirements fulfillment (power consumption, performance, execution time etc.). In [5], researchers found that moving critical software loops to reconfigurable hardware resulted in average energy savings of 35% to 70% along with a speedup of 3 to 7 times.

In terms of reconfigurable architectures, FPGAs are given preference over CPLDs due to the presence of higher level embedded functions such as adders, multipliers and embedded memories. They are also more flexible due to the dominance of configurable interconnects, but with a cost of increased design complexity. FPGAs find their use in any area or domain

where massive parallelism is a requirement. High performance applications also exploit FPGAs as key computational kernels for executing operations, such as FFTs and convolution.

The main downside of using a standard reconfigurable SoC is the cost compared to a custom SoC. The trade-off is related to the number of chips that will be shipped and any advantage for getting the product to market sooner. While custom designs normally have large up-front development costs, they offer low individual chip costs. Reconfigurable SoCs, on the other hand, have a comparatively small up-front cost but usually they are more expensive per single unit or chip.

B. Types of reconfigurations

As research related to reconfigurable FPGAs is quite wide, a three-axis classification scheme is used to classify a reconfigurable approach by the community: mainly *where*, *when* and *how* the reconfiguration takes place. We briefly describe each of these points:

- ***Where***: Reconfiguration can either be *exo-reconfigurable* (external) or *endo-reconfigurable* (internal) in nature. In exo-reconfiguration, reconfiguration is initiated and controlled by an external source. An example is an FPGA co-processor on a PCI bus. Where as in an endo-reconfiguration, the FPGA itself loads the configuration and reconfigures itself. In this case, usually an embedded controller such as a hard/soft processor manages the reconfiguration.
- ***When***: The reconfiguration can be either *static or dynamic*. Static configuration requires the FPGA to be inactive, while dynamic reconfiguration is carried out on the fly when the FPGA is active and running. Dynamic reconfiguration can be directed by the application and offers flexibility advantages for systems where static reconfiguration is not possible, such as satellites.
- ***How***: The reconfiguration can be either *full* or *partial*. Full reconfiguration completely reconfigures the whole area of an FPGA, while partial reconfiguration concerns only a region or regions of FPGA while the remaining portions continue their normal execution. In a full reconfiguration, a full device bitstream is transmitted over a communication channel even in the case of minute changes, resulting in needless high data transfers. This is detrimental in bandwidth limited applications such as satellite payloads.

Figure 2 shows an overview of system reconfiguration. It should be noted that external reconfiguration introduces additional latency, which can prohibit or complicate dynamic reconfiguration. In contrast, internal reconfiguration is more suitable for rapidly evolving systems such as SoCs. *Internal Dynamic Partial Reconfiguration* or *Partial Dynamic Reconfiguration* (PDR) has shown tremendous advantages over other forms of reconfiguration. PDR or *self reconfiguration* [6] as it is sometimes called, has been explored only recently. Currently only Xilinx FPGAs support partial dynamic reconfiguration. Xilinx provides a comprehensive tool support for PDR, as compared to other vendors that have chosen not to incorporate this feature due to reliability and economical issues.

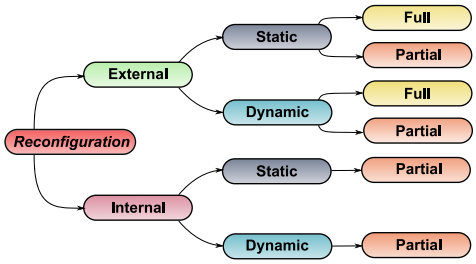

Figure.2: An overview of different types of reconfiguration

III. MARTE PROFILE FOR REAL-TIME EMBEDDED SYSTEMS

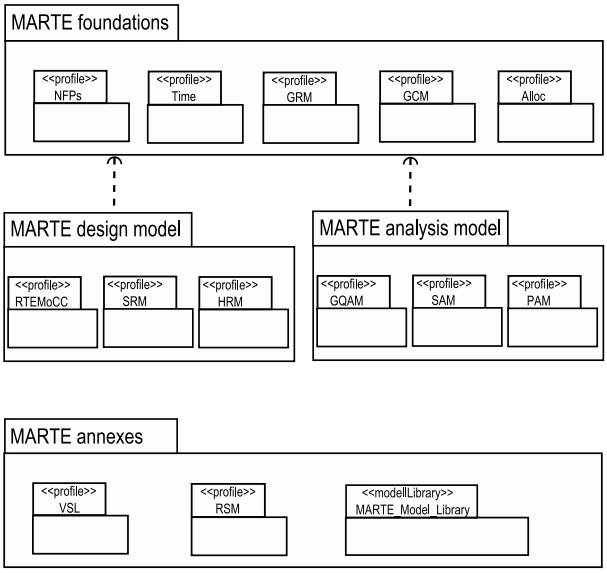

Figure.3: Global architecture of the MARTE profile

MARTE [4] (Modeling and Analysis of Real-Time and Embedded Systems) is an upcoming industry standard UML profile of OMG, for model-driven development of embedded systems. The profile is structured in two distinct directions: first, the modeling of concepts of real-time and embedded systems and secondly, the annotation of the models for supporting analyses of the system properties. The organization of the profile reflects this structure, by its separation into two packages, the *MARTE design model* and the *MARTE analysis model* respectively.

These two major parts share common concepts, grouped in the *MARTE foundations* package: for expressing non-functional properties (*NFPs*), timing notions (*Time*), resource modeling (*GRM*), components (*GCM*) and allocation concepts (*Alloc*). An additional package contains the annexes defined in the MARTE profile along with predefined libraries. Due to space limitations, we do not describe the different packages

in the profile. Detailed information about these packages can be found in [4].

A. MARTE concepts for modeling reconfigurable architectures

In order to describe the MARTE concepts for modeling reconfigurable SoCs, we first provide an overview of the hardware aspects in MARTE. The hardware concepts in MARTE are grouped in the *Hardware Resource Model (HRM)* package. HRM consists of several views, a functional view (*HwLogical* sub-package), a physical view (*HwPhysical* sub-package) or a merge of the two. The two sub-packages derive certain concepts from the *HwGeneral* root package in which *HwResource* is a core concept that defines a generic hardware entity. A HwResource can be composed of other HwResource(s) (for example a processor containing an ALU). This concept is then further expanded according to the functional or physical specifications. The functional view of HRM defines hardware resources as either *computing*, *storage*, *communication*, *timing* or *device* resources. The physical view represents hardware resources as physical components with details about their shape, size and power consumption among other attributes.

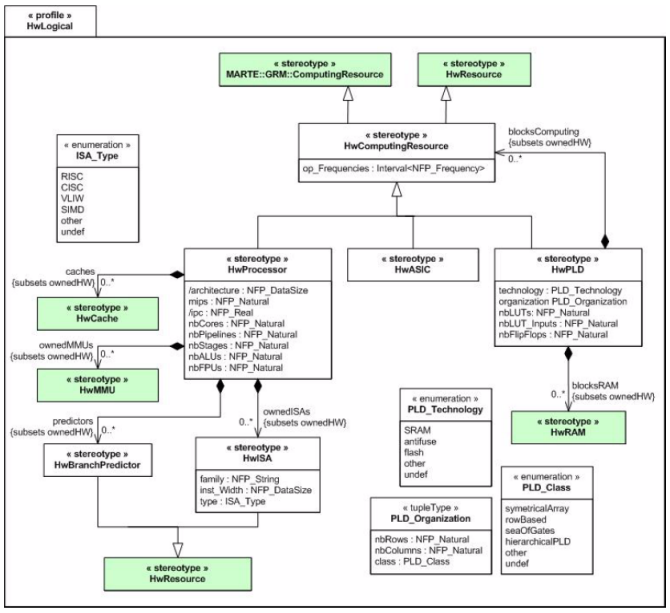

Figure.4: Concepts related to the FPGA modeling in the MARTE HRM package

Additionally, although MARTE provides adequate modeling semantics for describing computing resources such as processors and ASICs, it is not enriched enough to provide a detailed FPGA model at the high abstraction levels. Figure 4 shows the profile concepts related to computing resources as present in the MARTE *HwComputing* sub-package of the HRM package.

The *HwComputing* sub-package in the HRM functional view defines a set of active processing resources pivotal for an execution platform. A `HwComputingResource` symbolizes an active processing resource that can be specialized as either a processor (`HwProcessor`), an ASIC (`HwASIC`) or a PLD (`HwPLD`). An FPGA is represented by the `HwPLD` stereotype; it can contain a RAM memory (`HwRAM`) (as well as other `HwResources`) and is characterized by a technology (SRAM, Antifuse etc.). The cell organization of the FPGA is characterized by the number of rows and columns, but also by the type of architecture (Symmetrical array, row based etc.). A processor may contain some instruction set architectures (ISAs); and can have some `HwCaches`, `HwMMU` (main memory units) along with zero or more branch predictors (HwBranchPredictor).

IV. Analysis of the MARTE profile and proposed extensions

In this section, we carry out an exploratory analysis of the profile for modeling and specification of reconfigurable SoCs at high abstraction levels. Regarding these aspects, we first detail some of the important criteria that should be taken into account in the specification of these complex systems. While a large number of criteria can be added to the profile, in the context of this paper, we only focus on two key aspects: namely the FPGA architectural details and the energy consumption levels.

A. General FPGA architectural details

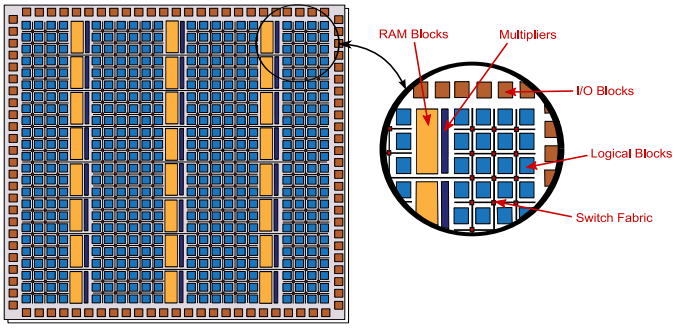

Figure.5: Configurable hardware layer in FPGAs

FPGAs usually consist of two layers. The first layer contains the reconfigurable logical blocks, i.e., *Configurable Logical Blocks* (CLBs) or *Logical Elements* (LEs) depending upon the FPGA vendor terminology. These logical blocks are present along with a hierarchy of reconfigurable interconnects allowing inter-communication between the blocks. This layer also incorporates different types of heterogeneous components such as RAM blocks, DSPs, multipliers, processors etc. All blocks of the same type (except the I/O blocks) are aligned into columns as shown in Figure 6.

The second layer consists of the FPGA configuration memory layer. The configuration memory of FPGA contains the application specific data. Writing into a configuration memory is accomplished via configuration files known as *bitstreams* (that contain packets of configuration control information as well as the configuration data). This second layer usually matches the first layer and is also organized into columns. Each column whose width depends on the covered block-columns of the first layer is further composed into sub-columns called

frames. For Virtex-II/Pro series FPGAs, a frame is the smallest unit of reconfiguration information which can be written on to the FPGA, while the more recent FPGAs such as Virtex-IV have smaller units of granularity [1]. Each frame contains fractions of configuration information required to configure the associated logical blocks assigned to a column.

B. Specification of energy consumption levels

As the circuit integration density is evolving at a rapid pace with the evolution of SoCs, specification of system power consumption levels is an integral task that serves as a basic QoS criteria along with system performance/throughput. As all integrated components in a SoC such as processors, buses, memories contribute to either static or dynamic power consumption, it is important to estimate the power levels of each of the integrated components.

A large number of tools and researches have been developed to estimate SoC energy consumption levels. Works in [7] propose a SoC design approach uniquely on the basis of the estimation of energy consumption. The system can consist of reconfigurable modules and their interconnections. The modules are described with different parameters: such as operating frequency, number of identical modules in the circuit, the current energy state of the module, etc.; to determine the related energy consumption of each of the modules. Frameworks such as MILAN [8] also work on the same principle and define different parameters for describing the energy consumption of each system component. The framework supports other power consumption tools such as ModelSim [9].

An interesting approach is to elevate this design aspect at high abstraction levels. In [10], the authors proposed an MDE based approach to estimate the static/dynamic consumption levels of a SoC at the *Transaction Level Modeling Pattern Accurate* or TLM PA level. However, these works did not target reconfigurable SoCs and a passage to the electronic RTL level was not realized. Additionally, the UML MARTE profile was not utilized for the modeling of system power consumption levels.

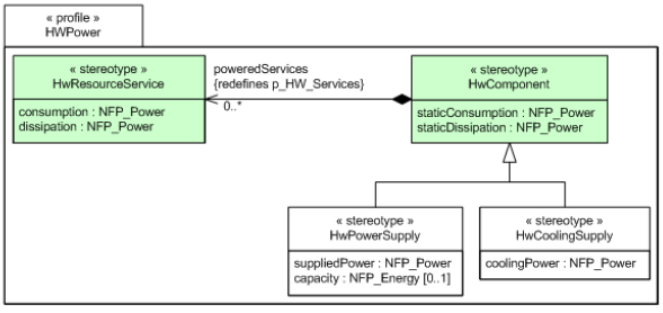

Figure.6: Concepts related to power consumption levels in MARTE profile

While the hardware concepts in the MARTE profile permit specification of energy consumption levels, these notions are not sufficient enough and do not provide a detailed description. For example, information related to the dynamic power consumption levels is not present in the profile, making it a difficult task for a SoC designer to specify these details at the system modeling level.

C. Specification of reconfigurable SoCs with MARTE

As stated before in the paper, the MARTE profile lacks several aspects for specification of reconfigurable architectures. To solve this issue, an extension of the profile is necessary. An extension at the profile level is called a *stereotype*. It specializes one or several UML classes and can contain supplementary attributes known as *tagged values*.

Firstly, the concepts related to representing a processor are not sufficient for a complex FPGA based SoC design, in which a complex processor can either be implemented as a softcore IP or integrated as a hardcore IP. Thus additional concepts are needed to address this limitation. Similarly, the concepts for `HwPLD` can be used for FPGA modeling; however, more details need to be added to this concept such as number of DSP blocks, registers, Block-RAMs etc. Another solution would be the definition of a specific *'FPGA'* metaclass as a stereotype in the profile, that inherits from the `HwPLD` concept and adds the additional attributes to that class. Similarly when communication between hardware components takes place, the latency and bandwidth needs to be treated. While HRM takes bandwidth into consideration, these issues need to be addressed as well.

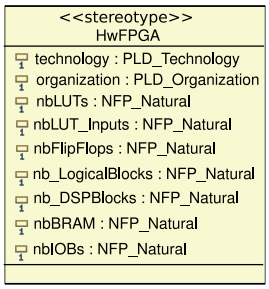

Figure.7: Addition of FPGA stereotype in MARTE profile

Thus, as our first contribution as illustrated in Figure 7, the addition of a novel `HwFPGA` stereotype is included in the profile for specifying the overall reconfigurable SoC. This stereotype contains general tagged values that allow to determine any generic FPGA characteristics such as logical blocks (Configurable Logical Blocks/CLBs or Logical Elements/LEs; depending upon the vendor terminology), I/O Blocks, DSP blocks, etc.

Additionally, we extend the current concept related to a hardware processor to take into account several aspects, as shown in Figure 8. The stereotype contains several new tagged values corresponding to the nature of the processor, as well as the energy consumption levels. The `Processor_Type` tagged value helps to determine if the processor is either hardcore or softcore in a reconfigurable SoC. Equally, the `UserDefined` type helps to specify user defined processor types such as a graphical processing unit, a micro-controller

etc. This concept increases the modeling flexibility permitting addition of user defined choices. This concept has also been used in other extended concepts to render them open to user interpretation if the need arises. Moreover, several tagged values such as `idle`, `inactive`, `logical_activity` and `arithmetic_activity` help to determine the various states of the processor.

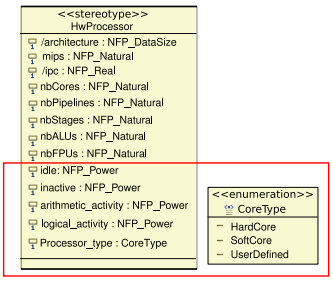

Figure.8: The extended hardware processor stereotype

While these information can be added to the physical aspect of a hardware component, we have chosen to specify them at the logical level after a long consideration. The reason is that an activity can be considered as an instruction being executed on the processor at a particular instant of time. As this aspect is functional in nature as compared to a physical one, it is added in the functional portion in MARTE hardware profile section. The tagged values `idle` and `inactive` while relate to the dynamic and static power consumption levels, depend also on the nature of the processor in question. For example, if a hardcore processor is in idle mode, it will still consume static power. However, if the processor is softcore in nature and is inactive, then it is possible to remove the processor from the system, decreasing the overall static consumption levels.

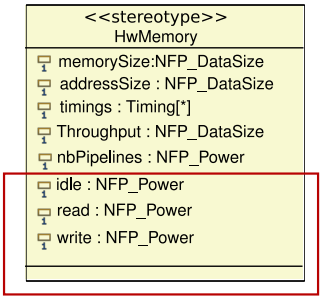

Figure.9: The extended hardware memory stereotype

In the same manner, we have extended the hardware memory concept in MARTE, as presented in Figure 9. The tagged values `read` and `write` determine the functional read/write operations of a generic hardware memory component. These operations determine the dynamic power consumption levels of this modeling concept. As also evident from this figure, while it is possible to put these aspects in the physical portion of the MARTE hardware specifications, we have chosen to integrate them in the functional portion.

Similarly, in modern SoCs, we find a large number of bus types depending upon the FPGA vendor. For this purpose, we have specified some of the most commonly used buses used in FPGA based SoCs such as the Processor Local Bus [11] and Open Peripheral Bus [12], which form part of the IBM coreconnect buses [13], using the `busType` tagged value. Also normally, while the MARTE profile is sufficient to specify details such as the bus address/data width and serial/parallel nature; details such as the number of master and slaves supported by a specific bus type are not present. Normally in case of SoCs, these buses can contain multiple number of master and slaves. For this reason, we have included the tagged values `nbSlaves` and `nbMasters` to illustrate this point .

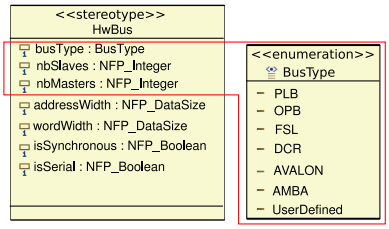

Figure.10: The extended hardware bus stereotype

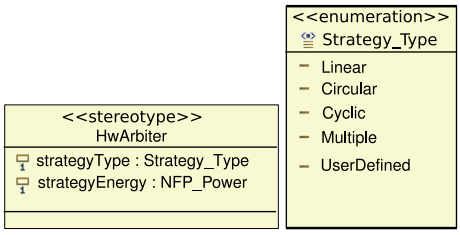

Figure.11: Enriching the HwArbiter stereotype in the current MARTE profile

In relation to this aspect, in a SoC, a hardware arbiter permits to control the bus with the different peripherals (master and slave components), which are connected to the bus. These arbiters normally provide the necessary bus control and logic required for correct functioning of the related bus. Normally these arbiters are also associated with an arbitration priority strategy, such as circular queue or round-robin. These strategies help in managing the conflicts in the communication infrastructure. For this reason, we have enriched the concept of `HwArbiter` in MARTE. The tagged value `strategyType` allows to specify some of the common arbitration strategies. However, a designer has the choice to specify other strategies not specified in the profile, using the enumerated literal `UserDefined` related to the `Strategy_Type` enumeration. Additionally, the `strategyEnergy` tagged value enables the designer to specify the power consumption levels linked with the chosen arbitration strategy.

D. Example of a modeled reconfigurable SoC and related questions

The aspects that we have integrated in the MARTE profile provide an initial attempt to model reconfigurable SoCs. This contribution is far from complete and a lot of detailed information can be added at the high abstraction levels. Unfortunately, too much information can make the models platform dependent and complex. For this reason, care should be taken to ensure that information should be added at the MARTE level which serves to permit modeling of all types of generic reconfigurable architectures, and not a specific targeted hardware platform.

Additionally, one of the biggest limitations for modeling of reconfigurable architectures such as FPGAs is the ability to bridge the gap between high level models and the commercial tools provided by FPGA vendors for carrying out simulation, placement, synthesis. Similarly new features such as partial dynamic reconfiguration should also be taken into consideration with dealing with these modern state of the art SoCs. The high level models are usually too abstract in nature and do not provide adequate mechanisms to take into account issues such as floor planning, IP-core management, notion of adapters etc. Similarly, automatic generation of processors along with their instruction set simulators (ISS) is a daunting task from high level models, via the MDE model transformations that permit automatic code generation. While a MARTE based approach has been presented regarding this aspect in [10], they are only able to produce code for the TLM PA abstraction level.

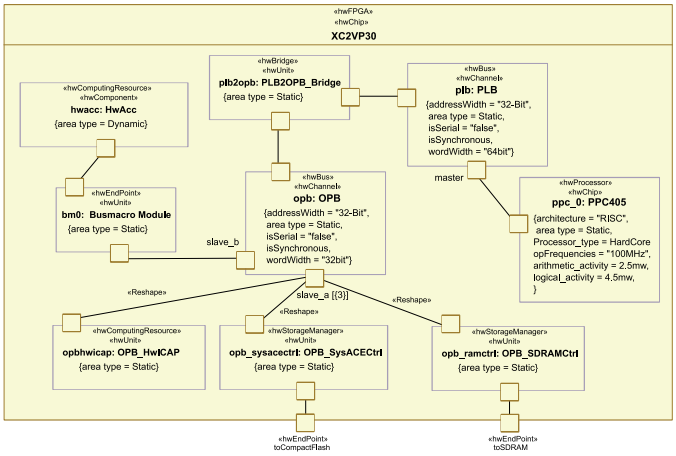

Figure.12: Modeling of an FPGA with the MARTE profile

The contributions that we have added in the MARTE profile help to model a generic reconfigurable FPGA based SoC, and permit to describe its internal structure. Figure 12 shows the modeling of a Xilinx Virtex II-Pro XC2VP30 FPGA based SoC on the XUP board [14].

By making use of a hybrid model that integrates the stereotypes of both the logical and physical MARTE hardware concepts, it is possible to carry out an initial attempt for modeling of a reconfigurable SoC. As dynamic reconfiguration is also an important aspect of these complex systems, we have also added an extension to the hardware physical portion in MARTE. Namely, `area type` tagged value has been included to the `HwComponent` concept in the profile. The latter describes in general all hardware components from a physical perspective. As in a dynamically reconfigurable system, the area type can be either static or partially dynamic in nature, this attribute helps to determine the static and run-time reconfigurable parts of the modeled SoC.

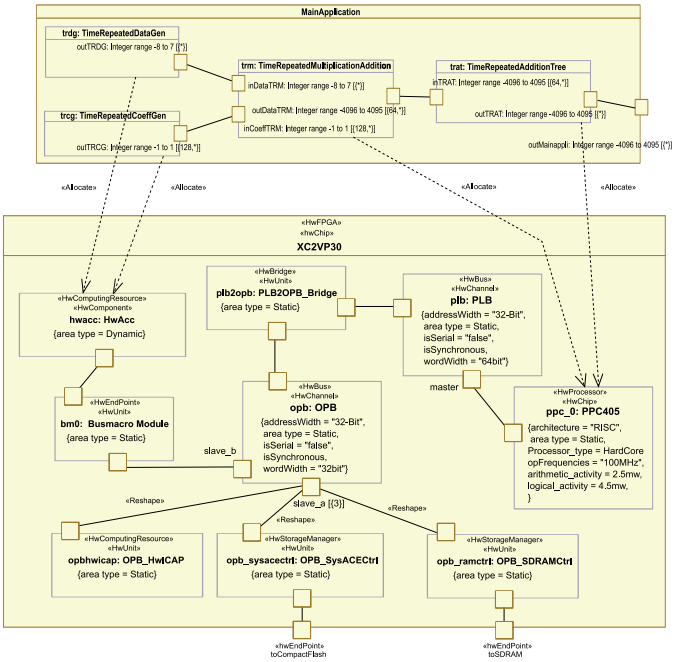

Figure.13: Allocating the application onto the hardware accelerator and processor present in the FPGA

The high level modeling helps in creating a complete system (application and architecture), while the allocation of the two, as illustrated in Figure 13 using the MARTE *Allocation* package permits to allocate the application onto the targeted architecture. This aspect can enable the designer to express different allocation choices: for example, the complete application can be placed onto a hardware accelerator, or either available hard/soft core processors present in the targeted architecture. Additionally, it is possible to separate the execution of the application: some key kernels can be executed onto hardware accelerator permitting a parallel execution, while others can be executed sequentially. However, control semantics should be introduced also at high abstraction levels to take into account issues such as data communication and synchronization.

An intermediate solution for bridging the gap between MARTE compliant UML models and low level synthesis tools can be the introduction of novel model transformations in the context of a SoC framework. The model transformations should be capable of interpreting the high level architectural models, and in turn can produce a part of input files required by the RTL tools. These files can be either in the form of some architectural description documents (such as generated by Xilinx EDK tool); for determining the structure of the

architecture, i.e., instantiation of submodules and their external interfaces/ports. The RTL tools can thus take these files and generate the corresponding HDL code and software drivers for the creation of the specified architecture. Similarly, in case of modeling user defined IP cores, the models should be able to automatically generate a wrapper in order to encapsulate the IP core functionality, in order to render it compatible with rest of the system.

Finally, our modeling methodology can also be extended by integrating the MARTE *HwPhysical* arrangement notation that provides rectangular grid-based placement mechanisms for bridging the gap between UML diagrams and actual physical layout and topology of the targeted architecture. Unfortunately, due to the current functional limitations of the UML modeling tools, it is not possible to express this view. However, evolution of these modeling tools may make it possible to take aspects such as *Floorplanning* into account. This view could be a potential additional aid to commercial PDR tools such as PlanAhead [15]. Designers can specify the FPGA layout at the MARTE specification level, helping the RTL designer in specifying an initial layout in MARTE. At the Register Transfer Level, designers can accurately estimate if the layout is feasible and determine the number of consumed FPGA resources. Finally using these simulation/synthesis results, the high-level models can be modified resulting in an effective DSE strategy for PDR-based FPGA implementation.

V. Related Works

There are a few works that have carried out research using the MARTE profile for specification of reconfigurable systems. An initial approach has been presented in [16], inspired from several initial works such as illustrated in [17]. However, the authors are only able to generate a textual description related to the hardware components in the modeled FPGA, which is taken as input by the FPGA tools for eventual manual manipulation. Additionally works such as presented in [18] propose extension of the MARTE profile to introduce notion of dynamic power consumption. However, these works make use of traditional UML state machines which are already ambiguous in nature and need proper semantics [19], [20]. However, these aspects can be a future extension of our design methodology, and can be a future perspective. Additionally, while the high level models need some control semantics, specially in the case of modeling aspects of dynamic reconfiguration. These semantics should be able to respect a component based approach, in order to create a compatibility between the high level system model and the control model. While works such as [21] introduce some aspects of control at MARTE level, these are not sufficient enough to handle issues such as data management when a reconfiguration takes place.

VI. Conclusions

This paper presents an initial analysis of the MARTE profile for modeling of reconfigurable SoCs. We provide an overview of the MARTE profile and describe the limitations of the current specifications. We then provide an initial contribution to the modeling of these systems by extending the profile to incorporate significant design criteria such as power consumption. Afterwards, a complete model of a popular FPGA based SoC is presented, followed by the co-design of both the architectural and applicative aspects. Finally we provide the readers with some of the perspectives which could help in creating a complete high level methodology from MARTE models to automatic code generation.

References

[1] P. Lysaght and B. Blodget and J. Mason, "Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs," in *FPL'06*, 2006.

[2] OMG, "Portal of the Model Driven Engineering Community," 2007, http://www.planetmde.org.

[3] S. Sendall and W. Kozaczynski, "Model Transformation: The Heart and Soul of Model-Driven Software Development," *IEEE Software*, vol. 20, no. 5, pp. 42–45, 2003.

[4] OMG, "Modeling and Analysis of Real-time and Embedded systems (MARTE), Beta 3," http://www.omgwiki.org/marte-ftf2/doku.php, 2009.

[5] G. Stitt, F. Vahid, and S. Nematbakhsh, "Energy savings and speedups from partitioning critical software loops to hardware in embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 1, pp. 218–232, 2004.

[6] B. Blodget and S. McMillan and P. Lysaght, "A lightweight approach for embedded reconfiguration of FPGAs," in *Design, Automation & Test in Europe, DATE'03*, 2003.

[7] S. Choi, J. W. Wang, S. Mohanty, and V. K. Prasana, "Domain-specific modeling for rapid system level energy estimation of reconfigurable architectures," in *International Conference of Engineering of Reconfigurable Systems and Algorithms*, 2002.

[8] S. Mohanty and V.K. Prasanna and S. Neema and J. Davis, "Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation," in *LCTES/Scopes 2002*, 2002.

[9] ModelSim, "Modelsim - advanced simulation and debugging," 2010, http://model.com/.

[10] R. Ben Atitallah, "Modèle et simulation des système sur purce multiprocesseurs - estimation des performances et de la consommation d'énergie," Ph.D. dissertation, USTL, 2008.

[11] Xilinx, "Processor local bus structure," http://www.xilinx.com/products/ipcenter/PLB_Bus_Structure.htm, 2009.

[12] Xilinx, "OPB hwicap product specification," Tech. Rep., 2009.

[13] IBM Corporation, "The Coreconnect Bus Architecture, white paper," in *International Business Machines Corporation*, 2004.

[14] Xilinx, "XUP-V2Pro Board," 2009, http://www.xilinx.com/univ/xupv2p.html.

[15] N. Dorairaj and E. Shiflet and M. Goosman, "PlanAhead Software as a Platform for Partial Reconfiguration," *Xcell Journal*, vol. 55, pp. 68–71, 2005.

[16] J. Vidal and F. De Lamotte and G. Gogniat, "A co-design approach for embedded system modeling and code generation with UML and MARTE," in *Design, Automation and Test in Europe (DATE'09)*, 2009.

[17] I. R. Quadri, S. Meftali, and J.-L. Dekeyser, "A model based design flow for dynamic reconfigurable fpgas," *International Journal of Reconfigurable Computing*, 2009.

[18] T. Arpinen, E. Salminen, T. D. Hamalainen, and M. Hannikainen, "Extension to MARTE profile for Modeling Dynamic Power Management of Embedded Systems," in *1st Workshop on Model Based Engineering for Embedded Systems Design (M-BED 2010), DATE 2010*, 2010.

[19] Software Technology Laboratory, "Stl: Uml semantics project. school of computing, queen's university," http://www.cs.queensu.ca/~stl/internal/uml2/bibtex/ref_umlstatemachines.htm, 2009.

[20] H. Fecher, J. Schönborn, M. Kyas, and W. P. de Roever, "29 new unclarities in the semantics of uml 2.0 state machines," in *7th International Conf. on Formal Engineering Methods (ICFEM'05)*, 2005, pp. 52–65.

[21] I. R. Quadri, A. Gamatié, P. Boulet, and J.-L. Dekeyser, "Modeling of configurations for embedded system implementation in marte," in *1st workshop on Model Based Engineering for Embedded Systems Design (MBESD 2010), at DATE 2010*, 2010.