MODEL-DRIVEN APPROACH FOR AUTOMATIC DYNAMIC PARTIALLY RECONFIGURABLE IP CUSTOMIZATION.

Gilberto Ochoa-Ruiz[1], Ouassila Labbani-Narsis[1], El-Bay Bourennane[1], Phillipe Soulard[2]
[1]LE2I Laboratory - Burgundy University - 21000 DIJON - FRANCE
[2]SODIUS – 6 rue de Cornouaille – F-44300 NANTES - FRANCE
Corresponding author: gilberto_ochoa-ruiz@etu.u-bourgogne.fr

Abstract – **This paper presents a framework which automates the generation of DPR capable IP cores. The approach is based in an MDE methodology, which exploits two widely used standards for Systems-on-Chip specification, UML/MARTE and IP-XACT. The approach aims at generating IPs which incorporate different functionalities by using code templates. The templates correspond to IP-XACT components that represent VHDL modules to be implemented in the IP. The IP-XACT sub-system description is generated from the MARTE description, effectively diminishing the complexity of creating this kind of systems by increasing the level of abstraction. We present the MARTE modeling concepts and how these models are mapped to IP-XACT objects; the emphasis is given to the generation of IP cores that can be used in the Xilinx EDK environment, since we aim to develop a complete flow around their Dynamic Partial Reconfiguration design flow. A model for the DPR IP is presented and a case study for a simple IP is presented. The use of our MDE approach is introduced to demonstrate how the generation from MARTE to EDK systems is performed.**

Index Terms—Dynamic Partial Reconfiguration, UML MARTE, MDE, IP-XACT, ESL Design, EDA.

1. INTRODUCTION

Run-time reconfiguration (RTR) has been introduced in recent years as a means of virtualizing hardware tasks in FPGA systems [1]. However, it wasn't until the introduction of Dynamic Partial Reconfiguration (DPR) technologies by Xilinx that these systems became a reality. In DPR systems, parts of the system can be reconfigured on run-time while the other functionalities in the FPGA remain operational [2].This capability can potentially provide enormous benefits to the systems designers, such as power and resources reduction utilization, amongst others.

However, despite the efforts by Xilinx and many industrial and academic endeavours, using DPR in very complex systems remains a very daunting task. This is due, in the first place, to the complexity of the design flow [3], which requires and in-depth knowledge of many low level aspects of the FPGA technology. Secondly, efforts in the academia to extend the capabilities of DPR design flow have further increased the complexity of DPR SoC designs. Some of these capabilities are, among others, RTOS support, context switching functionalities, and protocol adaptation. These features impact the design both at the system and at the IP level. Extra functionalities, in the form of static and dynamic wrappers, are added to the core [4], and automatic composition could help to facilitate this task.

In this paper, we propose a methodology to leverage the conception of DPR IP cores. It is based in a Model Driven Engineering (MDE) [4] approach in tandem with a component based approach. Therefore, the seamless integration and interoperability of the used IP is a necessity.

Several approaches have made use the UML profile and extensions to support embedded hardware resources modelling. Many of them made use of the UML profile for "Modelling and Analysis of Real Time and Embedded Systems" (MARTE) [4], which is a proposal for OMG standard profile. However, the mechanisms to move from UML specifications to enriched levels have not been standardized, and every approach manages this issue by defining their own transformation rules and information repository (using custom XMI data-books).

The SPIRIT consortium has developed the IP-XACT specification [5] that describes a standard way documenting IP meta-data for SoC integration. Several industrial cases studies have demonstrated that the adoption of IP-XACT facilitates the configuration, integration, and verification in multi-vendor SoC design flows [6], [7]. Additionally to the IP packaging and integration, IP-XACT also provides ways to automate the design flows where different tools are used.

The standard has generated enormous interest in the industrial and scientific communities as a means to overcome the complexity of system integration. Several research efforts have been carried out to integrate IP-XACT into MDE flows. Initial attempts at bridging the gap between UML/MARTE and IP-XACT have been presented in [14]. In [15], authors presented a methodology to perform this mapping, by means of models transformations; however, they target SystemC code generation. However, none of these approaches targets specifically DPR systems and IP customization.

The contributions of this paper relate to presenting an MDE approach that uses the UML MARTE profile that enables moving from high level models to HDL code generation for the implementation of the IP core sub-system description. IP-XACT is used as an intermediate model, used to configure the deployed IPs in the platform and to automate the parameterisation and customization of DPR IPs. The parameterised system and IPs are then used to generate the necessary inputs to the DPR design flow. Our approach simplifies the conception of FPGA-based SoCs, and it greatly facilitates the composition of DPR designs.

The rest of this paper is organized as follows: in Section II, we explain which role the different IEEE 1685 standard concepts play in the proposed methodology. An important aspect of any MDE methodology is the passage from the high-level specification to the intermediate representation; thus, in section III we present the mapping between MARTE and IP-XACT. In section IV, we present a model of DPR IP, and explain how the IP-XACT concepts are used to customize this template-based architecture. Section V presents a case study in which our framework is deployed for implementing a simple image processing architecture. Finally, in section VII we conclude, outlining future works.

2. IP-XACT CONCEPTS USED IN OUR METHODOLOGY.

The IP-XACT standard defines a set of XML data-books for IP and system description. This specification is meant to be used by IP providers and system integrators and all major EDA vendors as way to standardize the access to IP related information, hence promoting IP-reuse among EDA tools. The meta-data is used to configure, integrate and verify IP in advanced SoC design environments. Furthermore, it enables the creation of vendor-independent tools for automated IP parameterisation, integration, and as in the case of our approach, system customization.

The standard defines four central object descriptions, which are bus definition, abstract definition, component, and design descriptions. These four elements are sufficient for structurally describing a system and the IP cores the compose it. The use of these concepts at the system level has already been presented in [18]. We make use of these concepts to describe customizable DPR cores, both at the system level as at the subsystem level, as it will be explained in more detail in the next sub-sections.

2.1. Bus and abstraction definition.

These two concepts are used in tandem to describe standard buses, which comprise the different interfaces of an IP core. The bus definition describes high-level aspects of a bus. It basically serves as a reference to a more detailed description provided by the abstraction definition. The latter provides a complete description of the set of interfaces of a bus, detailing its ports (width, direction, and specific constraints) and other characteristics.

In our approach, we have defined bus and abstraction definitions not only for bus standards, but for the interfaces comprising the IP at the subsystem level, in order to facilitate the customization of the IP. This enables sub-component stitching of only those sub-elements that have been defined at the higher level of abstraction.

2.2. Component Description.

A component description packages the information related to an IP core, as depicted in Figure 1. In our approach, we have defined three types of IP cores: fixed IPs (non-parametrisable), soft IPs (typically parametrisable components, which low level implementation is hidden to the user), and customizable IPs. A special case of the customizable IPs is the aforementioned DPR core, whose functionalities are established at a high-level of abstraction, and added in the form of templates. Each type of IP requires a different set of IP-XACT elements at the component level, which will be described as follows.

Firstly, a component description contains interfaces linked to bus definitions, which abstracts the low level implementation details. This element is common to all types of IPs, since is used for top level stitching.

The Parameters/choices sub-elements permit the configuration of an IP core, and enable to automate the parameterization of an IP within a tool flow. Obviously, these elements are used both in soft and customizable IPs, but their use depends on the intended design flow. In our methodology, parameters/choices are set from the high-level MARTE description. Models and views are used for describing different implementations of the same component (e.g. different levels of abstraction) or for defining several hierarchical sub-systems, each of them implementing different functionalities for a component with the same interfaces.

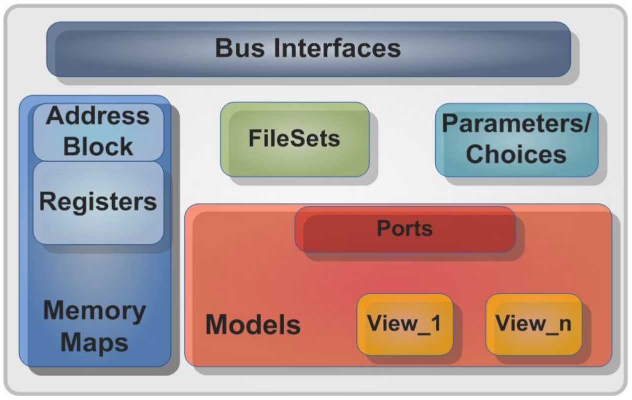


Figure 1. IP-XACT concepts for a component description.

2.3. Design Description.

A design, as depicted in Figure 2, describes an actual top level design as a set of component instances, which can be configured through configurable elements. The sub-elements in a design are connected between bus interfaces (that conform to predefined bus definitions). There are three kinds of connections, named interconnections in IP-XACT: interconnections, ad-hoc connections and hierarchical connections.

We make use of design descriptions for two purposes: for describing the top level architecture, as described in [16], but also to describe the implementation of customizable and DPR IPs. In IP-XACT, hierarchical components can be an instance of a given top-level design, but make reference to reference another design, describing its internal implementation sub-components.

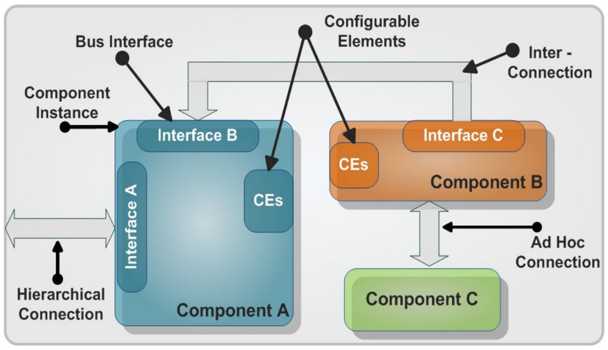


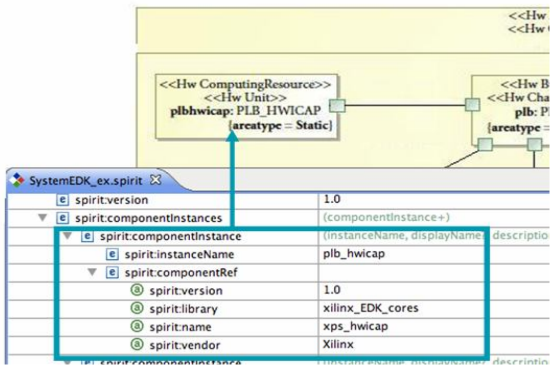
Figure 2. IP-XACT concepts for a design description.

The user can select which features are to be added to the IP. These features include protocol adaptation logic, and DPR services (e.g. context saving support) in the form of code templates related to IP-XACT component instances. This customization will be further details in Section 5.

3. MAPPING BETWEEN MARTE AND IP-XACT.

In this section, we briefly discuss how the mapping from MARTE to IP-XACT is used in our methodology. This is an important step, since facilitates the passage from a high-level specification to an enriched model (IP-XACT). As mentioned before, the aspects of deployment phase discussed in this work are only concerned with structural information of the system. UML for SoC has been used for describing the architectural information of a platform. IP-XACT is introduced in our methodology as an intermediate representation that contains rich information about IPs and their configuration.

Therefore, IP-XACT decouples the MARTE modeling from the targeted design flows. Moreover, IP-XACT has been especially developed with the IP and SoC industries in mind. This fact guarantees that both MARTE models and IP-XACT descriptions remain interoperable; if the targeted tool and design flow changes, it suffices to change the way the meta-data is used in the flow. Figure 5 shows an example of a MARTE to IP-XACT mapping.

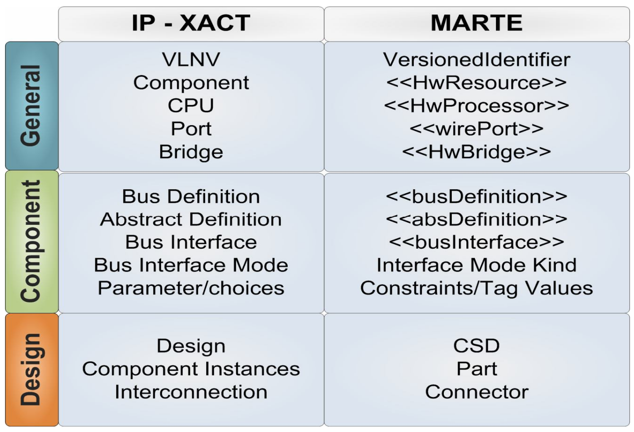


```
spirit:version                    1.0
spirit:componentInstances         (componentInstance+)
  spirit:componentInstance        (instanceName, displayName, descriptio
    spirit:instanceName           plb_hwicap
    spirit:componentRef
      spirit:version              1.0
      spirit:library              xilinx_EDK_cores
      spirit:name                 xps_hwicap
      spirit:vendor               Xilinx
```

Figure 3. An example of MARTE to IP-XACT mapping.

In order to perform the aforementioned mapping, a step must be defined in which the MARTE specification is parsed. Certain elements in the MARTE platform model will correspond to IP-XACT objects in the model library. The objective of this phase is to identify all the elements in the platform specification, generating as an output an IP-XACT design file. Components are associated to IPs in the model library by using their corresponding VLNVs, as depicted in Figure 3. Additionally, the interconnections between them must be defined, and the name is taken from the corresponding MARTE connector name and the end ports in the components the these connectors tie. A complete list of the transformation mappings is provided in Figure 4.

This step is done both at the system and at the subsystem (e.g. DPR cores) levels, producing design files for the top level implementation, and for the "top level" of each of the customizable IPs. This information is exploited, through the use of generators, to produce VHDL templates and other Xilinx files used by the Xflow of EDK, as it will be detailed in the next section.



	IP - XACT	MARTE
General	VLNV Component CPU Port Bridge	VersionedIdentifier <<HwResource>> <<HwProcessor>> <<wirePort>> <<HwBridge>>
Component	Bus Definition Abstract Definition Bus Interface Bus Interface Mode Parameter/choices	<<busDefinition>> <<absDefinition>> <<busInterface>> Interface Mode Kind Constraints/Tag Values
Design	Design Component Instances Interconnection	CSD Part Connector

Figure 4. List of MARTE to IP-XACT mappings.

4. IP CUSTOMIZATION USING IP-XACT.

In this section, we present one of the main contributions of this paper: how IP-XACT can be exploited in a component based approach, using code templates, to customize DPR IPs. First, we present a generic model for a DPR core, detailing its main components; then, we discuss how the MARTE and IP-XACT concepts presented in previous sections are exploited to customize its functionalities.

We have discussed the generation for the top-level implementation in [18]. However, the design entry phase of the DPR design flow also requires that the DPR IPs to be integrated into the top-level platform are synthesized independently, whilst maintaining the same interface for different implementations of the module. Additionally, the DPR IPs for SoC designs might have to be parameterized and customized before code generation. Furthermore, the IP must be attached to the bus, which is not always straightforward, due to bus protocol incompatibilities. Therefore, a Dapapath adaptation module must be added between the Hw Accelerator and the bus attachment, as depicted in Figure 5.

The IP contains a task interface which is entirely customizable (e.g. by using Xilinx IP core generator whose inputs we generate). Moreover, the hardware accelerator has to be wrapped as well. We refer to a DPR wrapper in the sense that a black box has to be defined which contains the static interfaces of the IP, This is a requirement of the current Xilinx DPR design flow, since the interfaces of IP in the floorplanning phase have to be frozen during the top level implementation. At the IP level, all the user logic that does not belong to the DPR wrapper is considered to be part of the static wrapper. The dynamic wrapper interfaces have to be customized as well, depending on the services the IP supports; for instance, interfaces to the IPIF/context wrapper have to be adapted, or connections for pipelining or to FPGA pins must be added.

4.1. DPR IP core modeling in MARTE.

We have chosen to concentrate our efforts in targeting EDK compatible IP cores, since we aim at generating DPR systems in a Xilinx flow. However, the approach is extensible to any kind of RTL description. The model in Figure 5 represents subset of features from a more general DPR component. The IP functionalities are chosen by the user by setting tagged values. The model presented here does not support context saving and the IP is only attached to the bus. However, a protocol adaptation component has been added between the IPIF and the DPR wrapper. For the rest of this section, we focus in describing how the components in the MARTE and IP-XACT descriptions are exploited as templates by associating them with the implementation fileSets.

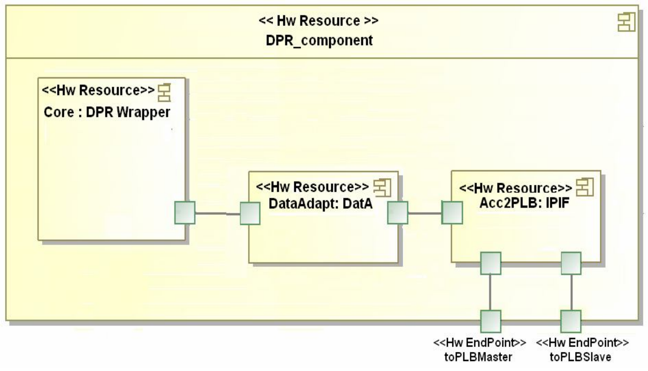


Figure 5. Modelling of a DPR IP in MARTE.

4.2. IP-XACT related concepts and modeling.

As mentioned in previous sections, the MARTE model for the platform description and the DPR cores modeling have an underlying IP-XACT representation. Each of the <<Hw Resource>> elements in the MARTE model is represented by a "component instance" in an IP-XACT design file. Similarly, component instances contain "bus interfaces" described by "abstract definitions", and "configurable elements" that are mapped to tagged values in the high-level specification. Each of the components have an associated fileSet, that references where the implementation files are located and how they to be deployed in the flow.

Figure 6 depicts an abstract representation of the IP-XACT component description for the model in Figure 5. For each customizable component in a design, IP-XACT component and hierarchical design descriptions must be defined. The component is stored in the library, which is abstracted by the corresponding MARTE models. Components in the IP-XACT design have associated implementation files; for instance, the IPIF component requires logic for implementing the read and write interfaces, each located in a VHDL file. The ports of the static wrapper also affect files such as the EDK MPD file, which contains the IP external ports description.

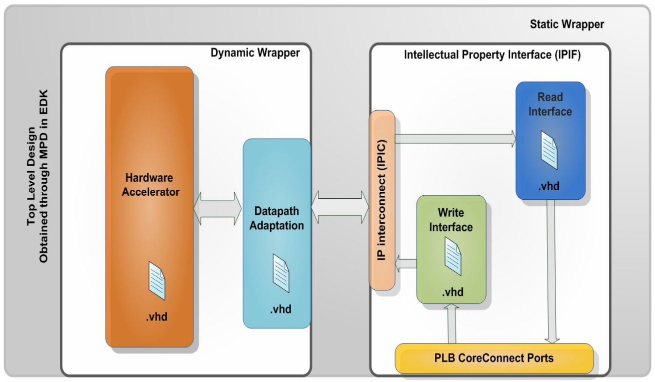


Figure 6. IP-XACT model abstract representation for the DPR IP

Depending on the choices of the designer (e.g. master/slave, burst support), the components are instantiated in the IPIF. The IP interconnect interface has to be adapted accordingly; for this, abstraction definitions for each interface have been defined, and are added to the model following parameterization options in the MARTE description.

Similarly, the datapath adaptation logic is added if the data width of the hardware accelerator is different from the bus width. Currently, this logic is not generated automatically; the designer of the IP has to create it accordingly to the needs of the application, but we envision exploring automatic protocol synthesis techniques to create the wrapper. This wrapper will be discussed later on the context of the intended application. Together with the HW accelerator, the protocol adaptation logic comprises the dynamic wrapper, or the static interface in the static design.

4.3. IP customization conception flow

The methodology for IP customization is explained in more detail in this section. This work is an extension of the approach presented in [16], which was used for system generation of the top level description of the DPR design. As mentioned before, the DPR design flow requires, as inputs, the netlists for the top level design and for the reconfigurable IPs. In that previous work, we discussed how to generate the SoC platform; here the discussion revolves around the DPR IPs as hierarchical components.

The framework used in our methodology is depicted in Figure 7. The designer of a DPR application starts by composing a system using a MARTE description in a tool such as Papyrus. An extension to the MARTE Profile for DPR has been created, details can be found in [17]. The static part of the system is created by integrating non DPR components such as the processor, memory controller, and communication IPs. The DPR IPs are also defined, with special tag values, as described in previous sections.

After the system has been composed and the DPR IPs defined, the MARTE model is converted into an IP-XACT design, retrieving the IP information from an abstract model library. The obtained design description is fed to a tool, Sodious MDWorkbench [19], a MDE platform which enables to define meta-models and to perform model transformations. We have created meta-models for the different Xilinx files used in the EDK environment, such as XBD, MHS, MPD, UCF, PAO, among others. As described in our previous work, we use these meta-models to perform model to model transformation between the IP-XACT description and the aforementioned files. These files are then fed to the Xilinx Xflow, that generates the top level and IP netlists that are similarly used as inputs to PlanAhead for implementing the DPR system. An in-depth discussion of the role of the different files is out of the scope of this paper; please refer to the Xilinx PSF Guide [18].

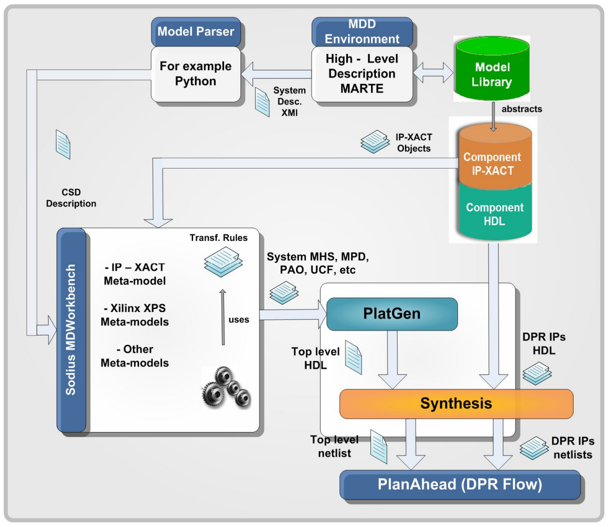


Figure 7. MDE framework for DPR system and IP composition

5. CASE STUDY: DPR IP INTEGRATION.

We have made use of the methodology previously described to implement a simple DPR architecture. The system contains two partial reconfigurable regions (PRR) in which we have mapped two different pixel based tasks. The architecture integrates a few simple components used to test the two Partial Reconfigurable Modules (PRMs, image inversion and binarization). The partial bitstreams are stored in the compact flash, and retrieved by the MicroBlaze, that writes the configuration information to the ICAP. A program in the Microblaze controls the configuration under request by the UART.

Figure 8 shows the implementation for the PRMs. The program running in the processor reads an image from memory and sends it line by line to the IP, where is stored in internal FIFOs. The datapath adaptation consists then in the module that handles this functionality for the input and output pixel information. Together with the hardware IP, it comprises the dynamic wrapper, that is at first completely implemented for testing, but which is implemented as a black box for DPR implementation.

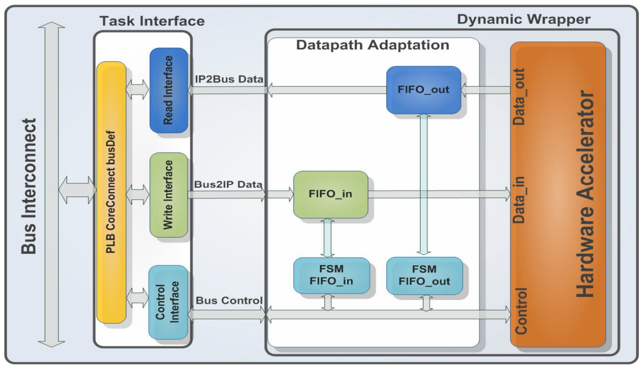
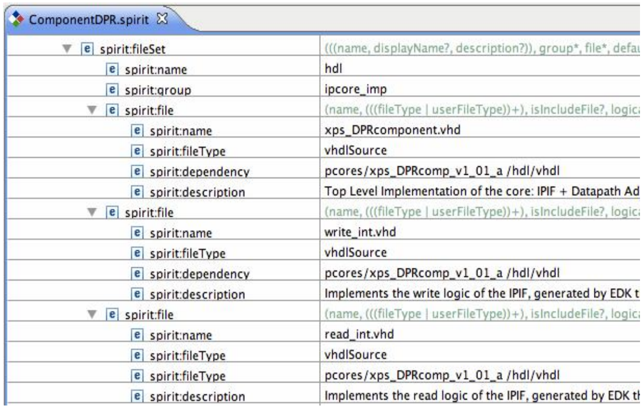


Figure 8. Implemented image processing IP with dynamic wrapper.

As described in Section 4.2, the basic architecture consists of several templates, each associated with one or several VHDL files. Figure 9 shows a snapshot of the fileSets in the IP-XACT component description.


ComponentDPR.spirit

spirit:fileSet	(((name, displayName?, description?)), group*, file*, defau
spirit:name	hdl
spirit:group	ipcore_imp
spirit:file	(name, (((fileType | userFileType))+), isIncludeFile?, logic
spirit:name	xps_DPRcomponent.vhd
spirit:fileType	vhdlSource
spirit:dependency	pcores/xps_DPRcomp_v1_01_a /hdl/vhdl
spirit:description	Top Level Implementation of the core: IPIF + Datapath Ad
spirit:file	(name, (((fileType | userFileType))+), isIncludeFile?, logic
spirit:name	write_int.vhd
spirit:fileType	vhdlSource
spirit:dependency	pcores/xps_DPRcomp_v1_01_a /hdl/vhdl
spirit:description	Implements the write logic of the IPIF, generated by EDK t
spirit:file	(name, (((fileType | userFileType))+), isIncludeFile?, logic
spirit:name	read_int.vhd
spirit:fileType	vhdlSource
spirit:fileType	pcores/xps_DPRcomp_v1_01_a /hdl/vhdl
spirit:description	Implements the read logic of the IPIF, generated by EDK t

Figure 9. Snapshot of the fileSets IP-XACT component description.

Table 1 summarizes the synthesis results for each partial reconfigurable module, the wrapper and the incurred overhead by using the DPR wrapper. The functionality of the system has been tested in an FPGA card incorporating a Virtex FX50.

Resources	I.I	I.B	Wrapper	Overhead	
LUT	1190	2297	90	8.7%	4%
Slice L	206	422	11	6%	2.8%
Slice M	94	167	12	16%	7.6%
Bram36	0	0	2	0%	0%

Table 1: Synthesis results for the two implementation of the DPR IP.

6. CONCLUSIONS

In this paper we have presented a design methodology that enables the parameterization and customization of DPR IP cores from a high-level specification. The presented approach is based in two widely used standards, UML MARTE and IP-XACT that until recent years had been developed in parallel; a great deal of research have been carried out to unify both standards, given the opportunities offered by the IP-XACT standard for interchanging IP descriptions among EDA tools.

However, as it is demonstrated in this paper, IP-XACT can also be exploited as a means for providing an intermediate system description that can be used to pass from UML MARTE models to HDL code generation. We have showed how the IEEE 1685 standard can be used to facilitate the customization of IP core by using sub-component templates that are linked to MARTE models. This approach also promotes IP reuse by automatically integrating IP cores into DPR designs. The generated IP-XACT design description for the customized IPs is then used, in an MDE platform environment, to generate several files. Among them are the files used by the Xilinx EDK Xflow to produce the system and IP cores implementations. This step produces the netlists that are used as inputs for the Xilinx DPR design flow, effectively leveraging the design entry phase of the DPR design flow. However, it must be noted that our approach is nor limited to EDK: different tools and flows can be addressed by modifying the transformation rules.

7. ACKNOWLEDGMENTS

This work has been supported by the ANR FAMOUS Project (ANR-09-SEGI-003)

8. REFERENCES

[1] P. Manet, “An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications”, EUSASIP Journal of Embedded Systems, 2008.

[2] A. Donlin, “Applications, Design Tools and Low Power Issues in FPGA Reconfiguration”, Chapter 22 in Designing Embedded Processors A Low Power Perspective, Springer, 513 -541, 2007.

[3]. Xilinx Corporation, Partial Reconfiguration User Guide, Xilinx UG208, 2011.

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

[5] "IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tools Flows," IEEE Std 1685-2009, Feb. 18 2010.

[6] W. Kruijtzer et al., “Industrial IP Integration Flows based on IP-XACT Standards,” in DATE’08, March 2008, pp. 32–37.

[7] C. Lennard, “Industrially Proving the SPIRIT Consortium Specifications for Design Chain Integration,” in DATE’06, March 2006, pp. 1–6.

[8] J-L. Dekeyser, P. Boulet, P. Marquet, and S. Meftali, "Model driven engineering for SoC co-design," IEEE-NEWCAS Conference, 2005. The 3rd International, vol., no., pp. 21- 25, 19-22 June 2005 doi: 10.1109/NEWCAS.2005.1496724.

[9] 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.

[10] J. Vidal, F. de Lamotte, G. Gogniat, J.- P. Diguet, and P. Soulard, "IP reuse in an MDA MPSoPC co-design approach," Microelectronics (ICM), 2009 International Conference on , vol., no., pp.256-259, 19-22 Dec. 2009.

[11] West team of LIFL laboratory. Graphical array specification for parallel and distributed computing (Gaspard). [Online]. http://www2.lifl.fr/west/gaspard/.

[12] I. R. Quadri, A. Muller, S. Meftali, and J.-L. Dekeyser, “MARTE based design flow for Partially Reconfigurable Systems-on-Chips,” in 17th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 09), 2009.

[13] I. R. Quadri, S. Meftali, and J.-L. Dekeyser, "Designing dynamically reconfigurable SoCs: From UML MARTE models to automatic code generation," Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on, vol., no., pp.68-75, 26-28 Oct. 2010.

[14] C. André, F. Mallet, A. M. Khan, and R. de Simone, "Modeling SPIRIT IP-XACT with UML MARTE", In: Proc. DATE Workshop on Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile, 2008.

[15].F. Herrera, E. Villar. “A Framework for the Generation from UML/MARTE Models of IP-XACT HW Platform Descriptions for Multi-Level Performance”. In the *Proceedings of the Forum on Specification & Design Languages (FDL'2011)*. Oldenburg, DE, September 2011.

[16]. G. Ochoa, E.B. Bourennane, H. Rabah, O. Labbani, “ High-Level Modeling and Automatic Generation of Dinamically Reconfigurable Systems,” in Proceedings of the DASIP Conference, Tampere Finland. November 2011

[17] S. Cherif, I. R Quadri, S. Meftali, J-L Dekeyser: Modeling Reconfigurable Systems-on-Chips with UML MARTE Profile: An Exploratory Analysis. DSD 2010.

[18]. Xilinx Corporation, Embedded System Tools Reference Guide, Xilinx UG111,September 2009

[19]. Sodius Corporation, MDWorkbench, http://www.mdworkbench.com/, 2011.