https://doi.org/10.31449/inf.v45i6.3675

A Fast Prototype for Modeling IP Cores Using in SoC with UML Marte

Benabdallah Ahcene Youcef and Boudour Rachid
Embedded Systems Laboratory, Badji Mokhtar University, Annaba, Algeria
benabdallah.ahcene@gmail.com, racboudour@yahoo.fr

Keywords: IP, SoC, MDA, UML, Marte.

Received: August 8, 2021

The gap between production systems and technological development has been growing in recent times, leading to the reuse of predesigned and pre-verified components called Intellectual Property (IP). The growth of the latter is not going to happen without encountering some difficulties; we include among some a lack of standards for the implementation of IPs, making integration difficult, only one incentive to some types of hardware or software IPs, and generally an incomplete development approach. In this paper, we present a comprehensive approach for modeling IPs starting from metamodeling. The approach is based on a Model-Driven Engineering (MDA) methodology, used standards for SoC (Systems-on-Chip) specification, MARTE, and Hardware Description Languages (HDLs). We illustrate our methodology in the case of USB 3.1, using Papyrus for modeling. Results are encouraging and show that the proposed approach allows creating cores of any size.

Povzetek: Hiter prototip za modeliranje jeder IP z uporabo v SoC z UML Marte, ponazorjen z USB3.1.

1 Introduction

The rapid evolution of society has led to the development of equipment (systems) increasingly complex, capable of processing information flows of very diverse nature and origin and whose access must remain nevertheless simple and fast.

Electronics enabled the progress of fields such as designing larger chips with more layers. Actually, it is possible to develop complex circuits for embedded systems that contain embedded processors, memory blocks, interface blocks and specific components for applications, but with difficult and time-consuming task. Currently, there are several possibilities to continue Moore's law for several decades, such as new technologies including quantum computing or 3D chips and sophisticated algorithms. [1]

Faced with these constraints, the designers are trying to promote code reuse, by moving towards assembling predesigned and pre-verified blocks referred to as virtual components. Reuse and integration of heterogeneous Intellectual Property (IP) from multiple vendors is a complex task. All thus anomalies find the problem of communication between them, which it comes to the multiple methodology of development for these components [2].

For these anomalies, VSIA (Virtual Socket Interface Alliance) [3] has recommended standardizing the design and the development of this component by defining a set of rules and using the different standards recommended in the domain of Embedded Systems (ES) and especially in System on Chips (SoCs).

In order to favor the reuse of components, architectures, and minimize the problem for developing complex embedded systems, it is required to have a tool providing flexibility, scalability and reusability capabilities, especially at the beginning of the design process [4].

For this purpose, a model described in UML Marte

MDA methodologies make the building blocks of the high-level models linked to the low implementations that embody the related behavior. This is basically an IP reuse problem, and in this way the components can be configured, and a synthesizable top-level implementation can be obtained. [5]

Absence of standard for IPs, and Lack of flexibility and interoperability gave birth to several development methods. Works related to development of IP cores using in SoC, can be categorized in several families. Therefore, we seek to make possible the design of standard virtual components in order to facilitate good operation in terms of integrating this IP into SoCs. In this work, we are interested in the design of these components in the modeling level that we try to elevate design abstraction level. Raising the level of abstraction is an optimal solution to improve design productivity.

This paper presents a fast prototype for modeling IPs cores using in SoC based on a UML MARTE, which takes into consideration both hardware and software descriptions.

To present this approach, the paper is divided into seven sections: in the sequel, we recall in section 2 the preliminary concepts about IPs, section 3 is devoted to previous works. Section 4 puts emphasis on our development approach. Section 5, describes the implementation details followed by an evaluation on a real example. Section 6 presents results and discussions; finally section 7 concludes the paper and outlines areas for future research.

2 Preliminary concepts

2.1 Intellectual property

Reuse of predesigned components is well known in the field of software design concept. This technique is applied to the design of silicon systems whose complexity and heterogeneity are growing. The complexity of hearts reused has therefore never ceased to grow, and originally began to reuse logic gates level (assembly of transistors), then evolved to use macro-functions (assemblies of logic gates) eventually leading to reuse RTL components that can go up many thousands of transistors [6].

An Intellectual Property is the specification of a component performing a clearly defined function that can be synthesized, thus reused, by a user who did not participate in the specification of this component.

VSIA defines three classes of virtual components depending on the level of abstraction of their synthesizable description: hardware, firmware and software [7]:

2.2 UML marte

UML MARTE Profile is the new standardized add semantics to UML for Real time Embedded System. It offers a number of functionalities organized as packages [8]:

(1) The Non Functional Modeling is organized around several packages: Properties (NFP) package enables to describe features that are not directly related to the business model (performance, memory usage, power consumption, etc.) and mechanisms to attach them to model elements,

(2) The Time package enables modeling of time structure and access (continuous time, clock, etc.),

(3) The Generic Resource Modeling (GRM) package enables modeling platform resources from high level perspective and mechanisms to manage access to those resources,

(4) The Allocation package enables modeling of spatial and temporal allocations,

(5) The Detailed Resource Modeling (DRM) package allows modeling of hardware and software resources at several levels of granularity; it contains two sub-packages enabling the modeling of Software Resource (SRM) and Hardware Resource (HRM),

(6) The Generic Quantitative Analysis Modeling (GQAM) package allows several types. It contains two sub packages for modeling Performance Analysis (PAM) and Schedulability Analysis (SAM).

2.3 USB 3.1

The original Universal Serial Bus (USB) was driven by the need to provide a user-friendly plugand- play way to attach external peripherals to a Personal Computer (PC). USB 3.0 was a revolutionary step for USB.

USB 3.1 is the next evolutionary step to increase the bandwidth. The goal remains the same; end users view it as the same as they viewed USB 2.0 and USB 3.0, just faster, and to enable devices from different vendors to interoperate in an open architecture [9].

USB 3.1 Gen 2 is a new high-speed IO interface running at 10 Gbps. It is expected to be widely used for connection of external hard drives and flash drives. The previous generation USB 3.1 Gen 1 design is used as a starting point and a divide and conquer approach is used to analyze the link. [10]

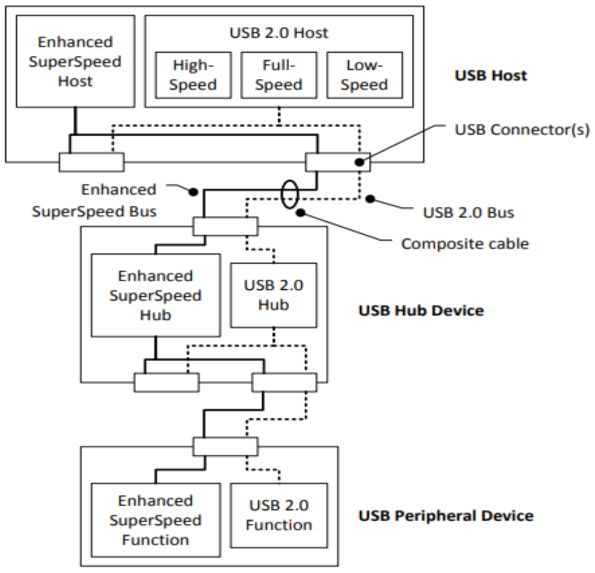


Figure 1: USB3.1 Dual Bus System Architecture [9].

The USB 3.1 system architecture in Figure 1 is comprised of two simultaneously active buses: a USB 2.0 bus and an Enhanced SuperSpeed bus.

Version	Described as	Data Rate
USB1.1	Low Speed (LS) Full Speed (FS)	1.5 Mbit/s
USB2.0	High Speed (HS)	480Mbit/s
USB3.0	*Super Speed (SS)*	*5 Gbit/s*
USB3.1	*Super Speed Plus (SSP)*	*10 Gbit/s*

Table 1: The evolution of USB terminology.

3 Related works

We can mention several initiatives that preceded Works related to modeling an IPs and trying to elevate design abstraction levels, such as providing specification in system level language like UML and profile MARTE to accelerate the production and decreasing the complexity related to the creation of virtual component can be categorized in several families. In this section, we mention the following works:

In [11] authors have proposed a soft IP customization framework based on the concept of design process and metaprogramming. The framework is independent of specific HW synthesis tools, proprietary technological libraries and HW description languages, and allows implementing customization of soft IP components taken from different providers and sources. HW and SW design can be unified at a high level of abstraction using UML. Different high-level abstractions can be used in the same system design flow, and in [2] they have presented a Design Process Model for adopting object- oriented design concepts in HW design domain and implementing HW design processes. The model combines UML class diagrams, metaprogramming and SW generators. The generation of HW models from UML class diagrams combined with the metaprogramming-based generation allowed us to raise the level of abstraction above the gate-level HDL specifications usually generated from UML behavioral models to system level soft IP-based design, and to separate the compositional aspects of design from the behavioral ones.

In [12], a method for synthesizing interfaces from UML models for heterogeneous IP (Intellectual property) integration is proposed. In order to maximize IP integration, the framework offers both interface protocol customisation and glue logic generation. The framework also allows for the creation of communication links between system blocks using UML profiles that are used to define system-level communication interfaces.

The Gaspard Methodology [13] is intended to provide a framework for developing parallel and distributed applications implemented on SoC. This methodology is an implementation of the MDA approach in the eclipse framework and provides a set of transformation rules allowing generation of optimized SystemC Code for repetitive structure architecture.

This alternative approach was used in an extended version of the MARTE profile and using model transformations, able to integrate configurations at SoC application and deployment design levels for different purposes and execution platforms, such as implementing run-time execution in state of the art FPGA based reconfigurable SoCs[14].

MoPCoM [15,16] is a design environnement that presented an automatic generation scheme from an unique UML/MARTE model that targets MPSoPC. This approach defines three models: functional, platform and allocation. In the functional model the designer specifies the behavior of the systems by means of an object oriented model. The platform is a set of hardware components where behavior will resides. The allocation maps the behavior onto the platform components, where the HW/SW partition is done. The code generation tool’s extract the new hardware components to be generated and writes VHDL code for each one. In order to generate code they define three different parts to be generated: structure, behavior and communication.

Three design levels are used in this approaches: AML Abstract Modeling Level, EML Execution Modeling Level and DML Detailed Modeling, which allows code generation to be done[17].

In [18], authors introduced a new logic circuit design methodology witch applied UML class diagrams on hardware design and proposed an automatic skeleton generator for hardware description language from UML.. They use UML tools to model hierarchical hardware modules, by defining some attributes and visibilities for this purpose. First they defined simple UML notation for logic LSI designers. It defines correspondence of UML to digital LSI modules. After they capture the requirements of the target design with UML diagram such as use case diagrams as software people. In the architectural design process, we will produce architectural design diagram such as class diagrams, state charts or activity diagrams.

In [19] The suggested method proposes a method for automatically generating SW code for complex embedded systems from a UML/MARTE model. The automatic synthesis process makes it simple to experiment with different SW component allocations in real physical platforms. Initially, the system is described using the UML/MARTE standard. The generated model has all of the information needed to automated synthesis, including functionality, HW platform, and allocation. A generator uses this concept to create communication wrappers that are totally ad-hoc and have a lower overhead than more general alternatives. The application can be split into clearly separable and reusable components, enhancing the product's organization as well as its reusability and modularity. In addition, each component's internal behavior should be considered during the specification process.

The work reported in [20] have detailed the approach for enabling IP reuse and modeling of DRSoC capabilities to the FAMOUS framework, in which a UML MARTE modeling frontend allows the design of reconfigurable platforms by importing component templates, which are then used to specify the DRSoC structure and behavior. The goal is to make it easy to reuse IP metadata between vendors while also allowing the tool to integrate with other back-end implementation tools. A MARTE « Platform model », which may be developed by importing IP blocks from the high-level library, connecting the bus interfaces of the various IP blocks to a bus component, and configuring the variable parts as needed, provides as the entry point. Reco- MARTE, which facilitates the deployment of IP-XACT components, including wrappers with persistence management services, is used in the application presented in this work.

Nevertheless, all the previous solutions are oriented toa fixed models, that depend on designer experience. These works are related to some types of software IPs whose UML models proposed cannot be generalized to all IPs.

The contributions of this paper are the introduction of an MDA approach that uses the UML MARTE profile for modeling IP reuse in SoC, and that enables moving from high level models to HDL code generation The novelty of our approach is that take into account aspects of the both software and hardware description with the third types of IPs.

4 Prototype description

The methodology of our approach for mofeling IP combines two aspects. The first one is applying Model-Driven Architecture (MDA) principales in the development of this component and getting the three models of this , and the second one is present a metamodel UML/Marte inspired of the architecture and functionality of a reusable IP, defined as a common starting point between all developers of this type of component. This metamodel mainly contains the majority the composition and information found in these components.

4.1 Graphical representation of an IP

We choose this block like representation of the IP-XACT standard [21]to facilitate their comprehension, and communication in the phase of integration in a platform to create a System on Chips, and to increase the automation of IP development and getting the automatic code generation to implement Intellectual Property (IP) blocks for System on Chip (SoC). The elements contained in the component schema are intended for describing as many different kinds of IP cores as possible, but it is obvious that not all of them will be required in all instances [22].

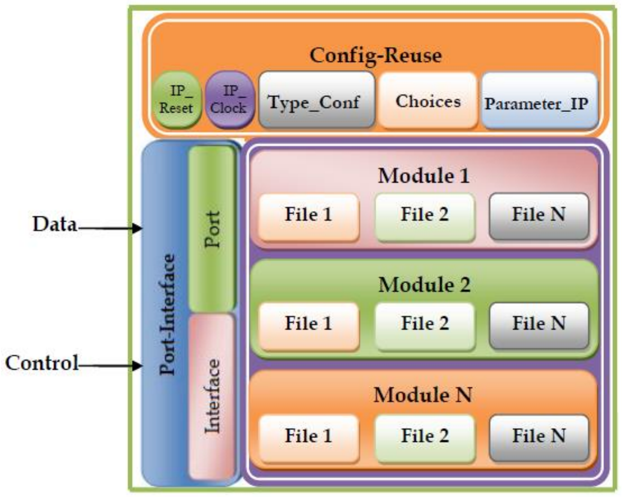


Figure 2: Graphical representation of an IP.

Here, we have included the most widely used concepts for structural and logical implementation and parameterization. As it is illustrated in Figure 2, the virtual component consists essentially of:

Config-Reuse: This element is responsible for setting the IP, we find several fields:

Parameter_IP: Parameterization of the IP in relationship to the system with values,

Choices: Parameterization of the IP relative to the system with functions ,

Type_Conf: It configures IP as master, slave or system ,

IP_Clock: The clock of the component in the hardware description ,

IP_Reset: Resetting IP during the operation.

Port-Interface: This item is designed to exchange data between the IP and the system, it is composed of:

Port: Area of exchanging data and messages in the hardware description ,

Interface: Area of exchanging data and messages in the software description.

Module: It represents the internal components of the IP. Each module consisting of several codes Files providing functionality in the IP.

Data: It is a set of data exchanged between the component and the system.

Control: It is a set of features to control and adapt the IP to the integration environment.

4.2 Methodology overview

According to the graphical representation of an IP presented in Figure 2, our proposed model covers the virtual components of different nature hardware and software descriptions.

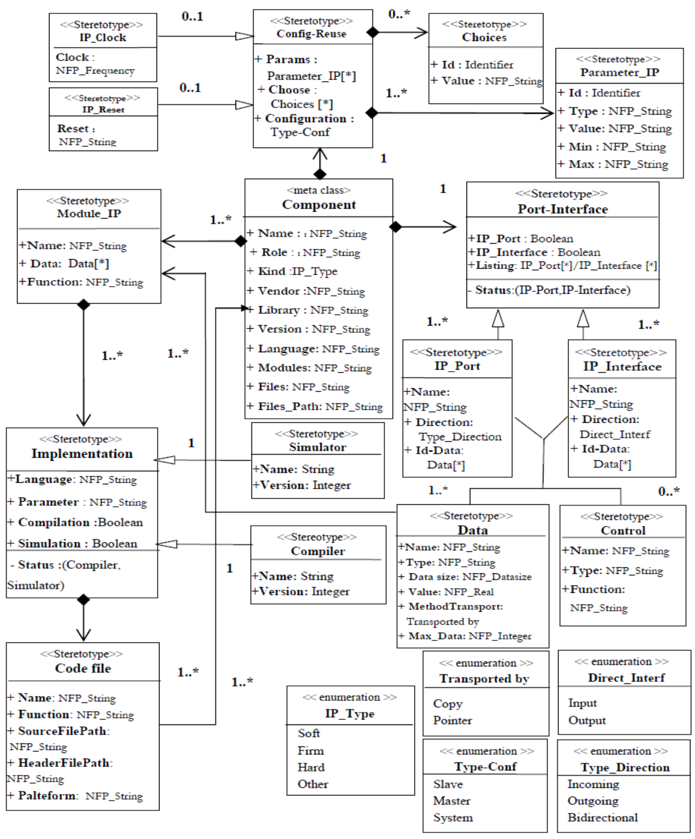


Figure 3: Our model with UML marte.

In a first step we model the virtual component from his specification using the model of UML MARTE proposed in Figure 3, to obtain an instantiation of the desired component using one of the tools supporting this profile.

To do so, the modeling methodology must fulfill three requirements. First, it has to minimize the modeling effort and to reduce the number of mistakes; the modeling methodology should be simple [4]

However, at the same time, the model must include all the information required to automatically perform the evaluation of each design alternative the designer selects, as described in next section. Additionally, the capability to reuse components or to integrate legacy or third-party components requires the use of standards. Thus, in this paper, the OMG standards UML and MARTE, are used to specify the system.

The system, which is intended to be developed, is represented by the metaclass component, which includes the following stereotypes:

Component**:** Application components are modular parts of the system that indentify pieces of functionality that represent a certain behavior. The functionality of the system is separated into a series of interconnected application components in this methodology. As a result, application components are the center of the platform-independent model

Port-Interface: This element captures the characteristics of the services provided by an application component in order to establish the communication. These interfaces are modeled by means of UML interfaces specified by the MARTE <<Port-Interface>> stereotype presented in Figure 4.

The ports are specified by the MARTE stereotype <<IP-Port>> in order to define the required/provided interface which is used for communicating with other application components, and managing the control between the IP and the system. Port Interface is composed of:

IP-port: iis reserved for hardware description in HDL.

IP-Interface is reserved for the description of the software in C ++. In both of these components we find stereotype and Control Data.

Confi-Reuse Its role is the configuring and the reuse of the component on the system that we want to integrate the IP on it. We have IP-Parameter, Choices, IP-IP-Clock and Reset.

Module-IP: A set of modules constituting our IP, each module has an implementation, which is made up of several files that are described in software or hardware programming language.

Figure 4 presents all these Stereotypes used in this metamodel

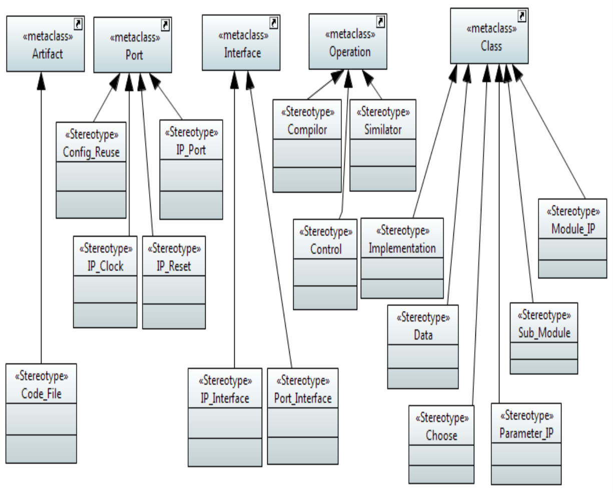


Figure 4: Different stereotypes used.

The second step is applying Model-Driven Architecture (MDA) principales in the development of HW/SW Component presented in C/C++ or VHDL or Verlog System.

System models are divided into three sub-models, following of Y chart structure [23]. Designs start by defining the two main starting points. The platform-independent model is formed by the expected system functionality, which contains all platform-independent details (PIM). A platform description model (PDM) is used to describe the HW/SW platform, which includes all of the details needed to implement communication and deployment codes. Finally, the platform-specific model is created when these two models converge to define how to enable that feature in the HW platform (PSM) [24].

5 Case study

In this section, we present a case study in which we show how the methodology is used. The application test consists of a USB 3.1 described in section 2.3. The global processing of this application is divided into three main phases following in Figure 5.

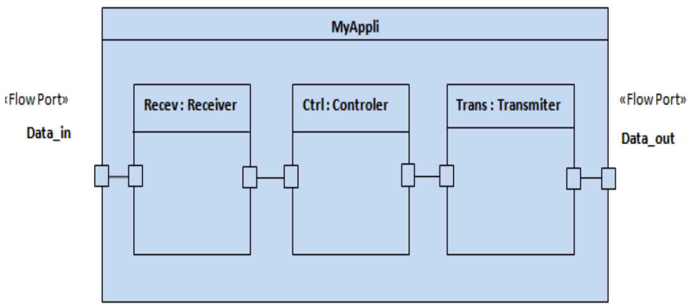


Figure 5: Global processing of the application.

The detail of this application is presented in figure 6.

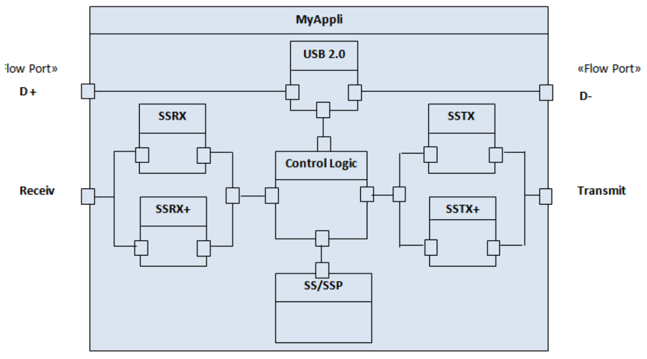


Figure 6: Application modeling in detail.

The major blocks and functions of our case are [25] :

Super Speed: In addition to the three existing transfer modes, the SuperSpeed bus includes a transfer mode with a nominal rate of 5.0 Gbit/s. Physical symbol encoding and connection level overhead are two aspects that influence its performance. Each byte requires 10 bits to transmit at a 5 Gbit/s signaling rate with 8b/10b encoding, thus the raw throughput is 500 MB/s.

Super Speed Plus: USB 3.1 has two variants. The first one preserves USB 3.0's SuperSpeed transfer mode and is labeled USB 3.1 Gen 1, and the second version introduces a new SuperSpeed+ transfer mode under the label of USB 3.1 Gen 2. SuperSpeed+ doubles the maximum data signaling rate to 10 Gbit/s, while reducing line encoding overhead to just 3% by changing the encoding scheme to 128b/132b.

USB 2.0 (High Speed and Full Speed) : The high-speed and full-speed controller implements the USB 2.0 protocol for peripheral devices. Data transactions, suspend and resume behavior, and interrupt generation are all handled by the high-speed and full-speed controller.

End Point Logic : The Endpoint Logic generates control signals for one synchronous, single-port RAM component, which is used for both IN and OUT endpoints. RAM size is fully configurable for the number, size, and buffering requirements of endpoints. Control transfers are handled through a dedicated IN/OUT endpoint pair with a 512 byte (64 byte for Full-Speed and High-Speed) buffer.

Physic Interface : The Device Controller IP connects to the IP for USB 3.1 PHY (Only Super Speed or Super Speed Plus) through a configurable 8-, 16-, or 32-bit PIPE 3.1 interface, and the IP for USB 2.0 PHY (High-Speed and Full-Speed) through an 8- or 16-bit UTMI or 8-bit ULPI interface.

Application Interface :The Device Controller IP supports an 8-, 16- or 32-bit AHB (Advanced High-performance Bus) or APB (Advanced Peripheral Bus) slave interface for configuration and access to the DMA Engine is through a 32 or 64-bit AXI (Advanced Extensible Interface) or AHB master interface.The DMA engine supports scatter-gather data transfers between endpoint buffers and external memory.

The model of our application is creating by Papyrus of Eclipse that supports the UML/MARTE model in [26]. First we instantiate the proposed model described in MARTE which is presented in Figure 3. The Figure7 presents the modelisation of USB 3.1

The functional blocks that constitute the USB 3.1's capabilities have been modeled as UML components using the MARTE stereotype <<RtUnit>> as illustrated in Figure 8.

The component starting triggers the application's execution, after which the files that implement each component's functionality are modeled and associated with the appropriate components. as shown in Figure 9.

The plateform independant model (PIM) of our application is illustrated in Figure 10, describes the functionnal components interne (their interface, fonctionnal code) and the interconnections among them.

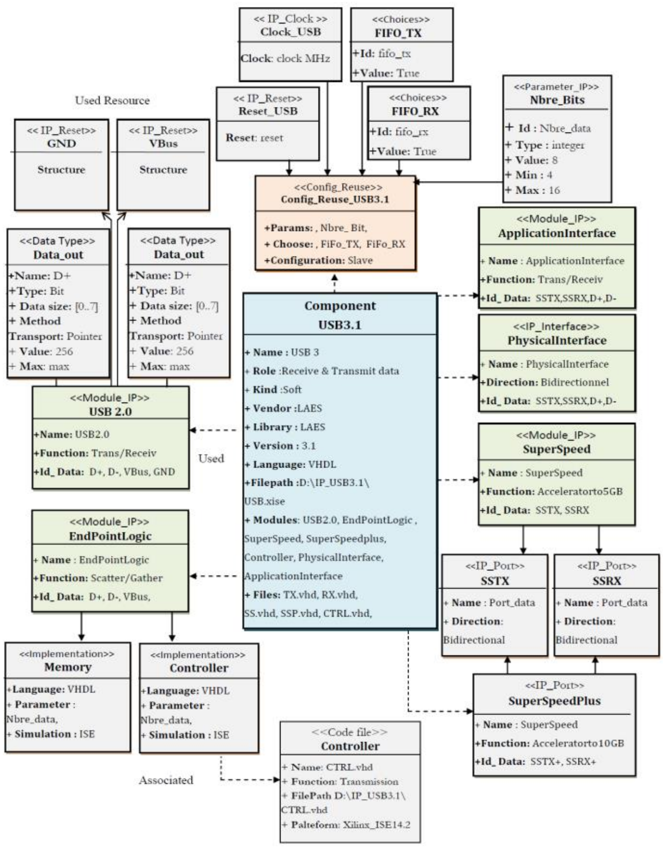
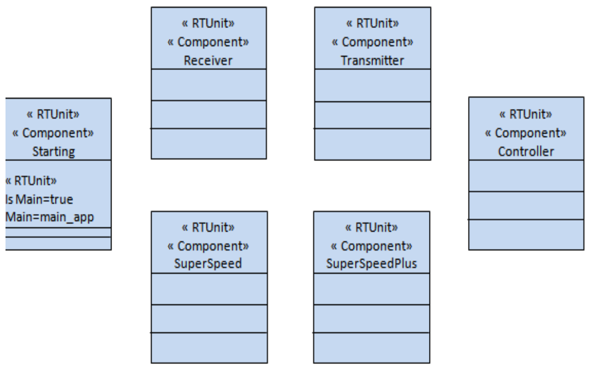


Figure 7: The model of USB 3.1.



« RTUnit» « Component» Receiver
« RTUnit» « Component» Transmitter
« RTUnit» « Component» Starting
« RTUnit»
Is Main=true
Main=main_app
« RTUnit» « Component» Controller
« RTUnit» « Component» SuperSpeed
« RTUnit» « Component» SuperSpeedPlus

Figure 8: Application Component.

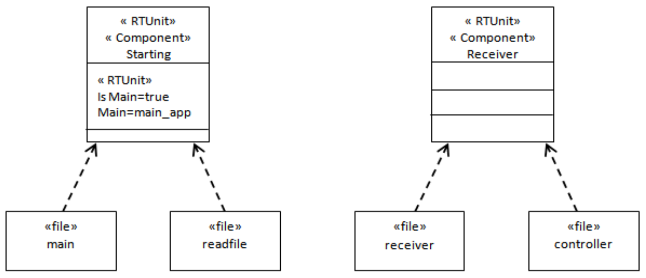


Figure 9: Application components and the associated files.

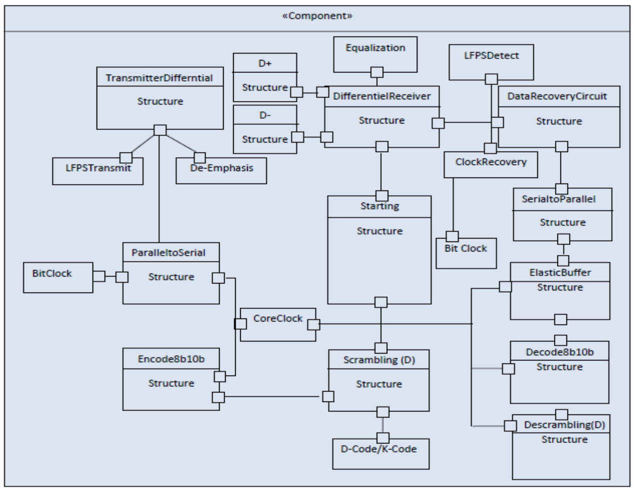


Figure 10: Application PIM.

6 Result and discussion

The method that we have proposed has been successfully applied to the USB 3.1 Core (Family of Serial Interface Communication), which contains five modules: USB2.0(TX, RX), EndPoint, SS, SSP,CTRL, and one interface with six principal files will be described in VHDL tx.vhd, rx.vhd, ss.vhd, ssp.vhd, ctrl.vhd and the main file will USB3.1.vhd assembled all these files.

Compared to the related works of this domain, we have defined our proper MARTE model according to the standard of IP-XACT, which contains interface, component, design description, parameterization and set of files. This model can be explored to instance and getting the specification of the IP desired to development.

Our approach is clear and comprehensive that allows the development of IPs understandable in their schema and easy to verify and integrate them thanks to their modular architecture as shown in Figure 3 and also thanks to the use of UML MARTE as a modeling language starting from our model which includes both descriptions, software in C ++ and HDL hardware.

This approach allows generating both hardware and software descriptions with the three types of IP (Soft, Firm, Hard), which is absent in other approaches presented in related works. A set of transformation rules accomplishes the transition from one level to another to generate target code or model. The use of high-level unified modeling notation accelerates up design time and system integration, and we can create all UML MARTE models for our virtual component whether of time or architecture, and creating cores of any size and shape, and it is easy to migrate to new technology..

Works exist are related to some types of software IPs whose UML models proposed cannot be generalized to all IPs. All the previous solutions are oriented to the generation of previously fixed models, leaving architectural decisions that depend on designer experience.

7 Conclusion

In this paper, a draft is proposed as a guide for the modelisation of IPs, recommended by the VSIA to reduce the gap between production of systems and technological change that has steadily increased in recent decades. This draft, in addition to its clarity and interoperability, offers the possibility to create good quality components, reliable and easy to be implemented in a system.

Our goal was to create IP cores that can be developed using the standard modeling tools to elevate the interoperability of these components that in the future we will reuse them easily and quickly in several systems and platforms without posing the problem of communication between other cores. This approach allows backtracking when errors appeared. The experiments were carried out successfully on a set of USB3.1 and we have provided a correct component thanks to the modular architecture of our model, the clarity of our approach, and the separating tasks of modeling and behavioral description to achieve the standardization goal recommended by VSIA.

This model, inspired by the architecture and functionality of a reusable Virtual Component, defined as a common starting point between all developers of this type of component. It makes the component understandable, clear and flexible, which facilitates the detection of errors and anomalies in the validation and verification phase. The model allows the scalability of the IP in the event of re-integration into a SoC.

Our future works will focus firstly on enriching the our model, secondly, developping the Physical communication links described in the PDM model, architectural mappings among PIM components and PDM resources lead to PSM models. Secondly proposed a set of rules for generating automatically the code (C++, VHDL), and finally get a series of syntheses of models produced by ISE of Xilinx thanks to its very rich component library.

References

[1] M. Gams, T. Kolenik. (2021) Relations between Electronics, Artificial Intelligence and Information Society through Information Society Rules. Electronics 2021, 10, 514. https://doi.org/10.3390/electronics 10040514

[2] R. Damaševičius, and V. Štuikys (2005). Soft IP Customization Models Based on High-Level Abstractions. *Information Technology and Control*, 34(2), 125-134. https://doi.org/10.15388/informatica.2004.049

[3] http://www.vsi.org

[4] H. Posadas, J. Merino and E. Villar (2020). Data flow analysis from UML/MARTE models based on binary traces. *Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 1-6, https://doi.org/10.1109/dcis51330.2020.9268671

[5] G. Ochoa-Ruiz, O. Labbani, E. Bourennane, Cherif, S. Meftali, and J. Dekeyser (2012). Enabling Partially Reconfiguration IP Cores Parametrisation and integration Marte and IP-XACT *.23rd IEEE International Symposium on Rapid System*

Prototyping (RSP), (pp.107-113),Finland. PA :IEEE Publishing.
https://doi.org/10.1109/rsp.2012.6380698

[6] M. Keating and P. Bricaud (2002). *Reuse Methodology manual for System-on-chip designs*, Kluwer Academic Publishers. 3th editions, (pp. 312).
http://www.ime.cas.cn/icac/learning/learning_3/201 9 07/P020190716574161163126.pdf

[7] VSI Alliance. (1997). Architecture Document – Version 1.0. Technical report.
http://xilinx.pe.kr/_hdl/1/VSI/vsi-or.pdf

[8] OMG. (2011). UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. Version 1.1, 11-06-02, June 2011 http://www.omg.org/spec/MARTE/1.1

[9] Universal Serial Bus 3.1 Specification July 26, 2013.https://manuais.iessanclemente.net/images/b/bc/USB_3_1_r1.0.pdf

[10] N. Srivastava, A. C. Scogna and H. Shim (2018). Enabling the next generation USB 3.1 (Gen 2) in mobile devices. *IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, pp. 18-23,
https://doi.org/10.1109/isemc.2018.8393730

[11] R. Damasevicius and V. Stuikys (2004). Application of UML for hardware design based on design process model. *ASP-DAC 2004 : Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753)*, pp. 244-249.
https://doi.org/10.1109/aspdac.2004.1337574

[12] S. Zhenxin and W. Weng-Fai (2009).A UML-based approach for heterogeneous IP integration. *ASP-DAC09.*
https://doi.org/10.1109/aspdac.2009.4796473

[13] E. Piel , R B. Atitallah, P. Marquet , S. Meftali, S.Niar, A. Etien, J. Dekeyser and P. Boulet (2008). Gaspard2: from MARTE to SystemC Simulation, *Proceedings of the DATE'08 workshop on Modeling and Analyzis of RealTime and Embedded Systems with the MARTE UML profile DATE'08, USA, 26-31.*

[14] A. Koudri, D. Vojtsiek, P. Soulard, C. Moy, J.Champeau, J. Vidal and J C. Le lann (2008). Using MARTE in the mopcom soc/sopc methodology. *In workshop MARTE, Germany*.

[15] I.R. Quadri, A. Gamatié, P. Boulet, S. Meftali, J.L. Dekeyser (2012). Expressing embedded systems configurations at high abstraction levels with UML MARTE profile: advantages, limitations and alternatives. *Journal of System and Architecture*. 58.
https://doi.org/10.1016/j.sysarc.2012.01.001

[16] J. Vidal, F. De Lamotte, G. Gogniat, J P. Diguet and P. Soulard, P. (2009). IP reuse in an MDA MPSoPC co-design approach. *International Conference on Microelectronics (ICM). (pp.256-259). PA: IEEE Publishing*.
https://doi.org/10.1109/icm.2009.5418638

[17] J. Vidal, F. De Lamotte, G. Gogniat, P. Soulard, J.P. Diguet. (2009). A Co-design approach for Embedded System Modeling and code generation with UML and MARTE. In *Proceedings of Design Automation and Test in Europe, (DATE'09), (pp. 226– 231). Dresden, PA: IEEE Publishing.*
https://doi.org/10.1109/date.2009.5090662

[18] N. Shimizu, M. Ikura, W. Wiriya, and S. Chivapreecha (2009). A new logic circuit design methodology with UML. *In Proceedings of ITC-CSCC'09. (pp.62-65).*
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.467.333&rep=rep1&type=pdf

[19] H. Posadas, P. Peñil, A. Nicolás, and E. Villar (2013).Automatic synthesis of embedded SW for evaluating physical implementation alternatives from UML/MARTE models supporting memory space separation. *Microelectronics Journal*. ISSN 0026-2692.
https://doi.org/10.1016/j.mejo.2013.11.003

[20] G. Ochoa-Ruiz, O. Labbani, E. Bourennane, P. Soulard, and S. Cherif (2012). A high-level methodology for automatically generating dynamic partially reconfigurable systems using IP-XACT and the UML MARTE profile. *Journal of Design Automation for Embedded Systems*, 16(3), 93-128, PA: Springer Publishing.
https://doi.org/10.1007/s10617-012-9098-6

[21] IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tools Flows," *IEEE Std 1685-2014*, vol., no., pp.C1-510, June. 12 2014.
https://doi.org/10.1109/ieeestd.2014.6898803

[22] A. Rautakoura, M. Käyrä, T. D. Hämäläinen, W. Ecker, E. Pekkarinen and M. Teuho, "Kamel: IP-XACT compatible intermediate meta-model for IP generation," *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 325-331.
https://doi.org/10.1109/dsd51259.2020.00060

[23] D D. Gajski and R. Kuhn (1983). *Introduction: New VLSI Tools*. Guest Editors', *IEEE Computer, 16*(12), 11-14.
https://doi.org/10.1109/mc.1983.1654264

[24] H. Posadas, P. Peñil, A. Nicolás and E. Villar (2015). Automatic synthesis of communication and concurrency for exploring component-based system implementations considering UML channel semantics. *Journal of Systems Architecture*. Volume 61, Issue 8, 2015, Pages 341-360, ISSN 1383-7621.
https://doi.org/10.1016/j.sysarc.2015.07.002

[25] http://www.cadence.com

[26] http://www.eclipse.org/papyrus