

An MDE Approach For Implementing Partial Dynamic Reconfiguration In FPGAs

Imran-Rafiq Quadri, Samy Meftali and Jean-Luc Dekeyser
INRIA FUTURS - LIFL - University of Lille,
France
{Firstname.Lastname@lifl.fr}

Abstract

Field programmable gate arrays (FPGAs) provide an interesting solution when custom logic is needed for short time to market products. The products embedding FPGA System on chip solutions allow them to be updated once deployed. Recent FPGA architectures, such as Xilinx Virtex Series, allow for partial and dynamic run-time reconfiguration (PDR). The FPGA fabric can modify its configuration data at run-time, enabling substitution of specific portions of an implemented hardware design causing the system to be adapted to the needs of the application. Hence reliability, failure-redundancy and run-time adaptivity by usage of PDR are introduced that are critical aspects for embedded systems. In this paper, we present a work in progress to provide high level modeling of PDR and a design flow to automatically generate VHDL code from these high level models depending on QoS criteria such as reconfiguration time, performance and power consumption.

Keywords: *Reconfigurable Hardware and Systems, FPGAs, IP Reuse, Adaptivity, Partial and Dynamic Reconfiguration, Dynamic Control, Embedded Systems, MDE, UML*

1 Introduction

Today's Systems-on-Chip (SoCs) are heterogeneous and integrate computing, communication, storage and interface resources. Current SoCs regularly include reconfigurable resources, such as Field programmable gate arrays (FPGAs) to increase flexibility. An FPGA allows the realization of a computation, storage or a

communication resource. Manipulating these reconfigurable resources in a SoC design framework is essential.

Recent evolution in FPGA technology allows the designer to update/reconfigure only a specific part of the internal structure of the FPGA at run-time using a technique known as Partial Dynamic Reconfiguration (PDR). With PDR, it is possible to implement dynamic systems that adapt to run-time needs of the application allowing reuse and reconfiguration of hardware cores (IP blocks). This allows the FPGA to remain operational without comprising the integrity of the applications running on parts of the FPGA that do not undergo reconfiguration.

Model Driven Engineering (MDE) [1] at the Object Management Group (OMG) is concerned with the development of standards and technologies to support the design of model based systems. The recent UML profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) [2] is intended for SoC co-design and to represent intensive signal processing applications. In the MARTE profile, a notion of *Multidimensional Multiplicity* exists in order to define repetitive structures and to express parallelism in a factorized way. This notion is based on the proposal of Cuccuru et al. [3] and is the basis of our GASPARD Environment.

The GASPARD (Graphical Array Specification for Parallel and Distributed Computing) co-design environment [4] is more specifically oriented towards parallel hardware and software co-design. It exploits the parallelism included in repetitive constructions of hardware elements or regular constructions such as application loops.

Our contribution is to introduce the notion of PDR in the GASPARD environment, by modeling and implementing the controller responsible for managing the reconfiguration in an automated manner with integra-

tion of QoS factors using the MARTE profile in order to generate the corresponding VHDL code allowing synthesis in the targeted FPGA. PDR can be a very interesting issue in modern intensive signal processing applications such as anti-collision radar and image processing allowing necessary run-time changes when required.

The rest of this paper is organized as follows: Section 2 provides an overview of the GASPARD Environment. Some details about PDR are described in Section 3. Related works are presented in Section 4 followed by our contribution. Concluding remarks are given in section 6.

2 Gaspard Environment

GASPARD [4],[5] is a co-modeling environment for the design of massively parallel SoCs. It is based on a model driven methodology in accordance with the Y design flow and is a subset of the MARTE profile [2], [6]. Figure.1 illustrates the GASPARD Co-design Framework with our conceptual flow.

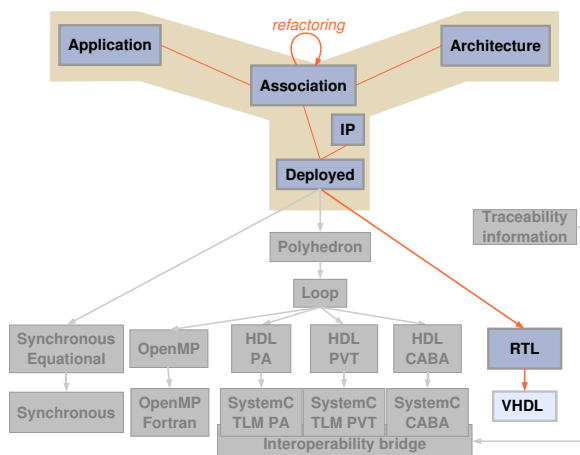


Figure 1: Our Conceptual flow in the GASPARD Environment

GASPARD uses the ARRAY-OL [7] concepts in order to express parallelism. It proposes a UML profile allowing designers to model both massively parallel applications and their architectures. An association mechanism is provided to link the two aspects together with a set of transformations for simulation and synthesis. A deployment mechanism is also used to link the elementary components (black boxes) to source code of the existing implementations [8]. This mechanism en-

riches both the application and the architecture models. Initially, application, hardware architecture and association are modeled using the GASPARD UML profile.

Using the ARRAY-OL concepts and its extensions, GASPARD then analyzes and transforms the models and allows for automatic code generation of different programming languages that allow simulation, execution or synthesis of an application mapped onto an architecture. We refer the reader to [9] for detailed presentations and demos of the GASPARD Environment. PDR is not currently possible in GASPARD and we aim to introduce this methodology in our environment.

3 Partial Dynamic Reconfiguration

Partial Dynamic Reconfiguration is the capability to reconfigure specific areas of the FPGA at run-time after its initial configuration. PDR is carried out to allow the FPGA to adapt to changing hardware algorithms, improve fault tolerance and resource utilization, to enhance performance or to reduce power consumption. Xilinx proposed the initial PDR methodology in [10] and currently only Xilinx FPGAs support Partial Dynamic Reconfiguration.

Initial Xilinx PDR design flows recommended the use of tri-state buffers for communication between dynamic and static parts of the FPGA [10]. They were not very effective leading to a new alternative in the form of predefined routed macros generally termed as *bus macros* [11].

Bus macros are placed at the edge of boundaries separating dynamic and/or static regions/modules. They allow communication between the different regions of the FPGA. The limitation to this approach was that although signals could pass through the reconfigurable area by use of bus macros, however if a module was being reconfigured, the internal signal route was also re-routed making the parts of the systems on either side of the module isolated from each other. This problem was addressed later on.

Virtex devices support *glitchless dynamic reconfiguration*: If a configuration bit holds the same value before and after reconfiguration, the resource controlled by that bit does not experience any discontinuity in operation, with the exception of LUT RAM and SRL16 primitives in Virtex-II/Pro devices [12]. This limitation has been removed in the Virtex-4 family.

The granularity of PDR is also of importance. Xilinx Virtex-II devices support frame sized (832 to 9152 bits, depending on the exact model) reconfiguration while Virtex-4 devices split up the frames in smaller regions [13].

Self reconfiguration is a special form of dynamic reconfiguration [14] and can be defined as the capability of a device to change its internal structure under the direct control of a user application. The configuration information is either generated by the device itself or by an external source.

The Virtex-II and the Virtex-II Pro are the first Xilinx architectures that support *Internal configuration access port* (ICAP) [14] which is a subset of the Xilinx SelectMAP interface having fewer signals because it only deals with partial configurations and does not have to support different configuration modes. For Virtex-II and Virtex-II Pro series, the ICAP furnishes an 8 bit Input data bus and an 8 bit Output data bus while with the Virtex-4 Series, the ICAP interface has been updated with 32 bit input and output data buses to increase its bandwidth. The ICAP allows the internal logic of the FPGA to reconfigure and to readback the configuration memory. With combination of either a hard or a soft microprocessor as a controller, Dynamic and Self-reconfiguration is carried out through the ICAP interface [14].

4 Related Works

A large amount of research has been carried out with regards to PDR in FPGAs. Xilinx initially proposed two types of dynamic reconfiguration [10]. The modular method divides the FPGA in modular regions (static and dynamic) for required functions with the limitation that the modules must occupy the full height of the device and thus the connectivity and topology were limited to 1D. Resources placed inside the modules could not be shared by other modules nor any routing was permitted through the module. Communication between modules was carried out by bus macros placed at the edge of the modules. The second approach, the difference based method is used for minute manual changes to only the design and is not suitable for large applications.

Bieser *et al* [15] used a shared bus to connect to modules which extended to the height of the device. The bus utilized the tristate drivers in the Virtex-II.

Bobda [16] *et al* have proposed two alternatives. In the first approach, the modules (based on vertical slots)

are connected to a reconfigurable multiple bus (RMB). In the second approach, they have carried out a network on chip (NoC) architecture in an FPGA that is capable of 2D placement. The NoC which is of an irregular structure allows to handle the routing aspects.

Sedcole *et al* [17] presented a modular approach which is more flexible than described by Xilinx [10] and were able to carry out 2D reconfiguration by placing hardware cores above each other. The layout (size and placement) of these cores is predetermined. They made use of reserved static routing to which the signals are connected by means of bus macros, allowing communication between modules by using the principle of glitchless dynamic reconfiguration.

Huebner *et al* [18] implemented 1D modular reconfiguration using a horizontal slice based Bus Macro. All the reconfigurable modules that stretched vertically to the height of the device were connected with the bus macro for communication. They followed by providing 2D placement of modules of any rectangular size by using routing primitives that stretch vertically through the device [19]. A module can be attached to the primitive at any location, thus providing arbitrary placement of modules. The routing primitives are LUT based and need to be reconfigured at the region where they connect to the modules. A drawback of this approach is that the number of signals passing through the primitives are limited due to the use of LUTs.

In March of 2006, Xilinx then introduced the *early access partial reconfiguration flow* (EAPR flow) [20] along with the introduction of slice based bus macros which are pre-routed IP cores. The concepts introduced in [17] were integrated in this flow. The restriction of full column modular PDR was removed allowing reconfigurable modules of any arbitrary rectangular size to be created. The EAPR flow also allows signals from the static regions to cross through the partially reconfigurable regions via the bus macros. Using the principle of glitchless reconfiguration, no glitches will occur in signal routes as long as they are implemented identically in every reconfigurable module for a region. The only limitation of this approach is that all the partial bitstreams for a module to be executed on a reconfigurable region must be predetermined.

Raaijmakers *et al* [21] provided a methodology to carry out general routing and placement on the Virtex-II Pro series. They carry out the reconfiguration aspect at bitstream level only without the use of Xilinx tools. This is done to avoid lengthy synthesis cycles. Arbitrary module placement is carried out with no restriction for

shape and size of the modules. The limitation is that only CLB resources are targeted and manipulation of the bitstream is carried out manually.

5 Implementing PDR in GASPARD Environment

Current methodologies for PDR entail extracting the difference between the old and new configurations through a method termed as *Read-Modify-Writeback* (RMW) method [17],[19]. We plan to make use of this method and our work is based on the methodologies introduced in [17],[18] and [20].

Because reconfiguration factors are specific for a device family, we have chosen an easily accessible FPGA (the Xilinx Virtex-II Pro family) to work on. It is commonly used, has an economical price tag, can carry out PDR and has an XC2VP30 as its main device with 2 PowerPC Cores, 30K+ Logic cells, 136 Multipliers and 428kBit of RAM. Further information can be found at [22].

As can be seen in the figure.2, the system is connected to an external Flash memory (which stores the partial bitstreams to carry out the reconfiguration) via a Flash controller which provides read/write access to the external memory. The ICAP module is present to read/write configuration data to/from the devices' configuration memory. An internal on-chip Block RAM can be utilized to store the recently read configuration data (not shown in the figure). The modules can be allocated any rectangular shape and non-adjacent modules are connected with each other and to the system using reserved static routing preventing isolation and allowing communication between the static and dynamic regions.

Reconfiguration of the dynamic region is managed by a Reconfiguration Controller which can be either external or internal (self configuration) to the FPGA. It is responsible for the partial reconfiguration bitstream that must be loaded onto the reconfigurable modular region. However, self configuration can be carried out through the internal ICAP parallel interface and reduces reconfiguration time as compared to external reconfiguration. For this reason, our reconfiguration controller is present inside the FPGA and communicates to the rest of the system via the Processor Local Bus (PLB).

Other peripherals are attached to a slower On-Chip Peripheral Bus (OPB) via the PLB-OPB Bridge. A

UART controller is also present to read the initial configuration from an external PC. The reconfigurable modules (attached to the bus macros) communicate with the OPB by means of a simplified Intellectual Property Interface (IPIF).

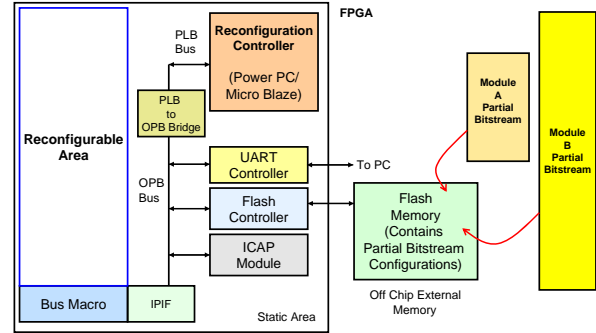


Figure 2: *Conceptual model of complete system to carry out PDR*

Our Controller will be based on the following notions. *Observation of the behavior of the system:* which depends strongly on the type of application and its adapting needs and modes (Performance requirements, power levels, time and area required for reconfiguration, etc.). *Evaluation:* after analysis of the above mentioned QoS factors, the controller should evaluate the cost of possible reconfigurations with respect to the QoS factors and finally *Implementation:* the controller should carry out the best possible reconfigurations. Although work has been carried out in PDR domain related to optimization of energy consumption of IP blocks [23], they used an external microcontroller to carry out the reconfiguration leading to increased reconfiguration times.

Currently, GASPARD supports the notion of static control as defined in [24]. We plan to introduce the notion of *Dynamic Control* in order to carry out PDR. We aim to initially carry out the reconfiguration manually at the RTL level with an application of our choice. The application will have several different implementations with each one corresponding to a specific QoS factor. We thus regard each implementation as a different reconfiguration and will carry out each reconfiguration manually to determine the result of QoS criteria for each implementation with regards to the application. Once we have carried out the reconfigurations manually to get the QoS results, we would be in a position to determine the choice of the implementation on the ba-

sis of the QoS criteria. As GASPARD deals primarily with intensive signal processing applications, applications such as anti-collision radar system or image processing are ideal to test the partial dynamic flow.

Afterwards, we need to link the QoS criteria determined in the first phase with the control aspect in order to create a controller that carries out the specific reconfiguration (implementation) based on these criteria. An interesting question is when to integrate the notion of dynamic control in the GASPARD framework. As in the GASPARD environment, the deployment level deals with the implementation details, we have decided to integrate the notion of a dynamic control at this level. This control(ler) will be linked to pre-determined QoS factors and their respective reconfigurations (implementations).

The final phase is to carry out the modeling aspect of the controller at the highest abstraction layer using the MARTE UML profile. This basically includes the modeling of the application, the architecture and the control aspect. This will be followed by a set of transformations in order to automatically generate the corresponding VHDL code for the reconfiguration controller, the partial reconfiguration bitstreams and the rest of the system. Depending upon the choice made by the designer, the controller will select a QoS and thus the reconfiguration associated with it. The PDR flow will aid to introduce PDR in MARTE using an automated design flow, making it possible to evolve the PDR approach for future versions of the profile.

A simplified view of our flow can be seen in figure.3. As described before, an application can have multiple implementations (different modes) associated with QoS factors. An example corresponding to the figure can be of a radar application, operating at either mode A (prolonged range with increased performance and power consumption) or mode B (decreased range with decreased performance and power). Mode C can be a mode with average power and performance levels as shown in the figure.3. The left hand side of the figure shows the work already implemented before. The static controller is able to choose between the different implementations (IP blocks) A, B and C by means of a switch. The right hand side of the figure shows the work to be implemented. The controller is dynamic in nature and can choose between the IP blocks A, B and C during run-time according to the choice of the designer.

Future works will include the introduction of *Physical* and *Logical* properties in the GASPARD framework as defined in the MARTE profile. This is to in-

troduce physical and logical aspects such as power consumption, performance and other aspects currently not present in the GASPARD framework which are vital for our work.

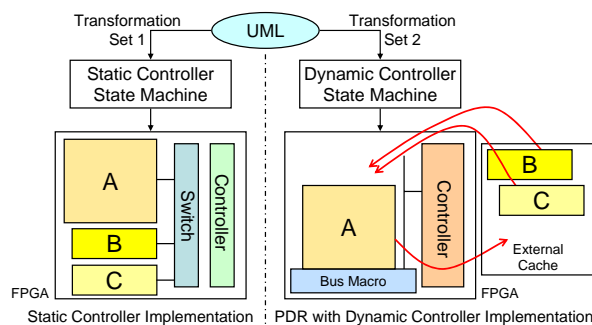


Figure 3: A simplified overview of the Dynamic and Static design flow

6 Conclusion

In this paper, we have described the work in process to implement the PDR in Xilinx Virtex-II Pro FPGAs. The originality of this approach is that as compared to the existing partial reconfigurable methodologies, the reconfiguration partial reconfiguration controller is to be initially modeled at a high abstraction level with the introduction of QoS factors, and then is to be implemented automatically, using the model to model transformations present in the GASPARD design flow to carry out PDR.

References

- [1] Planet MDE, *Model Driven Engineering*. <http://planetmde.org/>, 2007.
- [2] L. Rioux, T. Saunier, S. Gerard, A. Radermacher, R. de Simone, T. Gautier, Y. Sorel, J. Forget, J.-L. Dekeyser, A. Cuccuru, C. Dumoulin, and C. Andre, "MARTE: A new profile RFP for the modeling and analysis of real-time embedded systems," in *UML-SoC'05, DAC 2005 Workshop UML for SoC Design*, Anaheim, CA, June 2005.
- [3] A. Cuccuru, "Modélisation unifiée des aspects répétitifs dans la conception conjointe logicielle/matérielle des systèmes sur puce à hautes performances," Ph.D. dissertation, Université des sciences et technologies de Lille, Laboratoire d'informatique fondamentale de Lille, Nov. 2005, (In French). [Online]. Available: <http://www.lifl.fr/west/publi/Cucc05phd.pdf>
- [4] WEST Team LIFL, Lille, France, "Graphical Array Specification for Parallel and Distributed Computing (GASPARD-2)," <http://www.lifl.fr/west/gaspard/>, 2005.

- [5] P. Boulet, C. Dumoulin, and A. Honoré, *From MDD concepts to experiments and illustrations*. ISTE, International scientific and technical encyclopedia, Hermes science and Lavoisier, Sept. 2006, ch. Model Driven Engineering for System-on-Chip Design. [Online]. Available: <http://iste.co.uk/index.php?f=a&ACTION=View&id=147>
- [6] P. Boulet, P. Marquet, E. Piel, and J. Taillard, "Repetitive Allocation Modeling with MARTE," in *Forum on specification and design languages (FDL'07)*, Barcelona, Spain, Sept. 2007, invited paper.
- [7] P. Boulet, "Array-OL revisited, multidimensional intensive signal processing specification," INRIA, Research Report RR-6113, Feb. 2007. [Online]. Available: <http://hal.inria.fr/inria-00128840/en/>
- [8] R. B. Atallah, E. Piel, S. Niar, P. Marquet, and J.-L. Dekeyser, "Multilevel MPSoC simulation using an MDE approach," in *IEEE International SoC Conference (SoCC 2007)*, Hsinchu, Taiwan, Sept. 2007. [Online]. Available: <http://www2.lifl.fr/west/publi/BPT+07.pdf>
- [9] INRIA, "DaRT short presentations and demos," <http://www.lifl.fr/west/DaRTShortPresentations/>, 2007.
- [10] "Two Flows for Partial Reconfiguration: Module Based or Difference Based," in *Xilinx Application Note XAPP290, Version 1.1*, 2003.
- [11] "Two Flows for Partial Reconfiguration: Module Based or Difference Based," in *Xilinx Application Note XAPP290, Version 1.2*, Sept. 2004. [Online]. Available: <http://www.xilinx.com/bvdocs/appnotes/xapp290.pdf>
- [12] B. Blodget and C. Bobda and M. Huebner and A. Niyonkuru, "Partial and dynamically reconfiguration of Xilinx Virtex-II FPGAs," in *FPL'04*, 2004, pp. 801–810.
- [13] P. Lysaght and B. Blodget and J. Mason and J. Young and B. Bridgford, "Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs," in *FPL'06*, Madrid, Spain, Aug. 2006.
- [14] B. Blodget and S. McMillan and P. Lysaght, "A lightweight approach for embedded reconfiguration of FPGAs," in *Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 399–400.
- [15] C. Bieser and M. Bahlinger and M. Heinz and C. Stops and K.D. Mueller-Glaser, "A Novel Partial Bitstream Merging Methodology Accelerating Xilinx Virtex-II FGPA Based RP System Setup," in *FPL*. IEEE Circuits and Systems Society, 2004, pp. 701–704.
- [16] C. Bobda and A. Ahmadinia, "Dynamic interconnection of reconfigurable modules on reconfigurable devices," in *IEEE Design and Testing*, vol. 22, no. 5, 2005, pp. 443–451.
- [17] P. Sedcole and B. Blodget and T. Becker and J. Anderson and P. Lysaght, "Modular Partial Reconfiguration in Virtex FPGAs," in *FPL'05*, 2005, pp. 211–216.
- [18] J. Becker and M. Huebner and M. Ullmann, "Real-Time Dynamically Run-Time Reconfigurations for Power/Cost-optimized Virtex FPGA Realizations," in *VLSI'03*, Darmstadt, Sept. 2003.
- [19] M. Huebner and C. Schuck and M. Kuhnle and J. Becker, "New 2-Dimensional Partial Dynamic Reconfiguration Techniques for Real-Time Adaptive Microelectronic Circuits," in *ISVLSI'06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*. IEEE Computer Society, 2006, p. 97.
- [20] Xilinx, "Early Access Partial Reconfiguration Flow," <http://www.xilinx.com/support/prealounge/protected/index.htm>, 2006.
- [21] S. Raaijmakers and S. Wong, "Run-Time Partial Reconfiguration for Removal, Placement and Routing on the Virtex-II Pro," in *FPL'07*, Amsterdam, Netherlands, Aug. 2007.
- [22] The Xilinx XUP-V2Pro Board, <http://www.xilinx.com/univ/xupv2p.html>.
- [23] D. Elléouet, "Méthodes de modélisation, d'estimation et d'optimisation de la consommation d'applications du TDSI pour la conception de systèmes reconfigurables de type FPGA," Ph.D. dissertation, LESTER / UBS Lorient / IETR Rennes, Laboratoire d'informatique fondamentale de Lille, Dec. 2006. (In French).
- [24] S. Le Beux, "Un flot de conception pour applications de traitement du signal systématique implémentées sur FPGA à base d'Ingénierie Dirigée par les Modèles," Ph.D. dissertation, Laboratoire d'informatique fondamentale de Lille, Université des sciences et technologies de Lille, France, Dec. 2007. (To Appear).