High Level Design of adaptive distributed controller for Partial Dynamic reconfiguration in FPGA

Sana Cherif, Chiraz Trabelsi, Samy Meftali, Jean-Luc Dekeyser

HIGH LEVEL DESIGN OF ADAPTIVE DISTRIBUTED CONTROLLER FOR PARTIAL DYNAMIC RECONFIGURATION IN FPGA

Sana Cherif, Chiraz Trabelsi, Samy Meftali, Jean-Luc Dekeyser

INRIA Lille Nord Europe - LIFL - USTL - CNRS, 40 avenue Halley, 59650 Villeneuve d'Ascq, FRANCE

ABSTRACT

Controlling dynamic and partial reconfigurations becomes one of the most important key issues in modern embedded systems design. In fact, in such systems, the reconfiguration controller can significantly affect the system performances. Indeed, the controller has to handle efficiently three major tasks during runtime: observation (monitoring), taking reconfiguration decisions and notify decisions to the rest of the system in order to realize it. We present in this paper a novel high level approach permitting to model, using MARTE UML profile, modular and flexible distributed controllers for dynamic reconfiguration management. This approach permits components/ models reuse and allows systematic code generation. It consequently makes reconfigurable systems design less tedious and reduces time to market.

Index Terms— Partial Dynamic Reconfiguration, distributed control, modularity, adaptivity, high level modeling, UML MARTE

1. INTRODUCTION

In a reconfigurable computing system, both the hardware and software parts can be reconfigured depending upon the designer requirements. Due to their several advantages, such as flexibility and performance, reconfigurable devices, mainly FPGAs, make them well adapted for diverse application domains such as satellite based systems, telecommunications, transport, etc. As the computational power increases, more functionalities are expected to be integrated into the system. Today's FPGAs are more suitable for evolving systems with Partial Dynamic Reconfiguration (PDR) feature. It enables modification of specific regions of an FPGA on the fly while others regions remain operational and functioning. Although PDR is fastly growing and gaining popularity in the domain of reconfigurable computing, it is mandatory to predict a robust control mechanism for managing reconfiguration. A good control system should have these prerequisites : speed, accuracy and robustness.

In our work, the reconfiguration decision is taken in a distributed way, which is better in terms of performance than the centralized way. In order to decrease the design complexity of our system, we will use a model-driven design methodology. UML (Unified Modeling Language) is considered as one of the main unified visual modeling languages in Model Driven Engineering (MDE). When UML is utilized in the framework of MDE, it is usually carried out through by its internal metamodeling mechanism termed as a *profile*. UML MARTE [1](Modeling and Analysis of Real-Time and Embedded Systems) is a recent industry standard UML profile of OMG, dedicated to model-driven development of embedded systems. MARTE extends UML along with added extensions (e.g. performance and scheduling analysis). The MARTE profile extends the possibilities to model the features of software and hardware parts of a real-time embedded system and their relations. MARTE is mainly organized into two packages, the MARTE *design model* and the MARTE *analysis model* which 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. The hardware concepts in MARTE are grouped in the *Hardware Resource Model* (HRM) package from MARTE design package [2]. HRM consists of a functional and a physical views. The first one (*HwLogical*) classifies hardware resources depending on their functional properties, and the second view (*HwPhysical*) concentrates on their physical nature. Both derive certain concepts from the *HwGeneral* root package in which *HwResource* is a core concept that denotes a generic hardware entity providing at least one *HwResourceService*, and may require some services from other resources. The functional view of HRM defines hardware resources as either computing, storage, communication, timing or device resources. The physical view represents physical components with details about their shape, size and power consumption among other attributes.

In this paper, we propose a high level design of an adaptive distributed controller for partial dynamic reconfiguration, using UML MARTE profile. The rest of the paper is as follows. Section 2 gives an overview of some related works while section 3 explains our motivations and our proposed approach. We validate our solution with a case study in section 4 and presents several advantages in section 5. Finally, the last section presents a conclusion and future works.

2. RELATED WORKS

There are a few works that have carried out research using the MARTE profile for specification of reconfigurable systems. The MoPCoM project [3] aims to target modeling and code generation of dynamically reconfigurable embedded systems using the MARTE UML profile for SoC Co-Design [4]. However, the targeted applications are extremely simplistic in nature, and do not represent complex application domains normally targeted in the SoC industry. The authors in the OverSoC project [5] also provide a high level modeling methodology for implementing dynamic reconfigurable architectures. They integrate an operating system for providing and handling the reconfiguration mechanism.

Robust control mechanisms for managing reconfiguration must be developed in order to take into account SoC designer needs as well as QoS choices. In [6], authors reuse the design methodology based on Automation Objects presented in [7] to address formal approaches to verify the control architecture in a hierarchical way. Therefore, they propose an hierarchical control approach by means of a multi-layered architecture. Authors in [8] present control semantics at the deployment level where each configuration can be viewed as an implementation of the modeled application: comprised of unique collection of Intellectual Properties (IPs). Each IP is related to an application elementary component. The control at deployment level can be extended for complex configurations. A comparison of several design approaches of distributed controllers in industrial automation systems is presented in [9] . Authors discussed essentially two different approaches: decomposition of a centralized controller onto several communicating distributed controllers and integration of predefined controllers of components to the control of a system. In [10], authors present a Multiple Models Switching and Tuning approach(MMST) for adaptive control to provide greater flexibility in design. They also consider deterministic and stochastic systems and discuss some problems of convergence and stability.

3. PROPOSED APPROACH

3.1. Motivations

Partial Dynamic Reconfiguration (PDR) gives the designer the ability to reconfigure a certain portion of the FPGA during run-time without influencing the other parts. This special feature allows the hardware to be adaptable to any potential situation especially when a high degree of flexibility and parallelism are required to achieve real-time constraints. Normally, the reconfiguration is carried out in complex systems by means of a control mechanism (for example a RTOS or middleware), which can depend on QoS choices: such as changes in executing functionalities due to designer requirements, or changes due to resource constraints of targeted hardware/platforms. The controller can be implemented in different manners. It can be external or internal; and can be an hardware module written in an HDL acting as a controller. It also can be integrated in different levels in a SoC Co-Design framework [8]. Control integration at the application level has a local impact and is independent of the architecture or the allocation. The reconfiguration at this level may arise due to QoS criteria as in video processing application where switching from a high resolution mode to a lower one is required. Control integration in an architecture can be mainly local and used for QoS such as modification of the hardware parameters (voltage or frequency) for manipulating power consumption levels. The second type of control can be used to modify the system structure either globally or partially. If either the application or the architecture is changed, the allocation must be adapted accordingly. Nevertheless, dynamic reconfiguration is more architecture driven as compared to being application oriented, and is influenced by target platform low-level details. Hence, we are interested in control integration at the architecture level. Dynamic adaptation depends on the context required by designer and can be determined by different QoS criteria [11]. Self adaptation is also present in some component models. Usually this type of adaptation is executed by an adaptation manager or controller, typically a separate process or component that is responsible for managing the adaptations. The aim of our proposed approach is to model at a high abstraction level, a distributed modular controller which will carry out some of QoS factors such as changes in executing functionalities due to designer requirements, or changes due to resource constraints of targeted hardware/platforms. The changes can also take place due to other environmental criteria such as communication quality, time and area consumed for reconfiguration and energy consumption.

3.2. Controller modeling

Our proposed controller consists of a distributed and local device which manages each partial reconfigurable region (PRR) in an FPGA. While as the use of a master controller is unavoidable, a master controller (processor) is mandatory for overall system level control. The detailed block diagram of controller architecture is shown in Figure 1.

Our Controller characterized by its modularity and adaptability, will be based on the following notions:

- Ctrl2Proc_Adapter: adapts signals transmitted from the controller to the processor via bus macro.
- Event_Observer: has as input the processor's requests, e.g., read, write, etc. or user inputs sent from the hyperTerminal. It treats these data and transforms them into a list of events.
- Decision module: analyzes received events coming from Observer module and decides which configuration can be run. This choice depends on defined

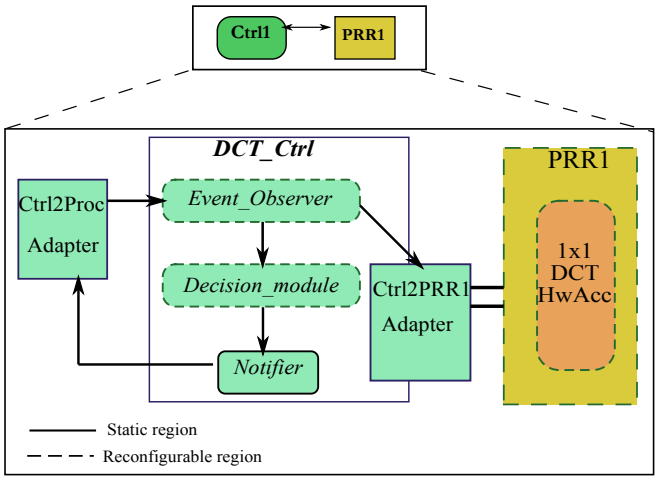


Fig. 1. Detailed block diagram of our proposed modular controller

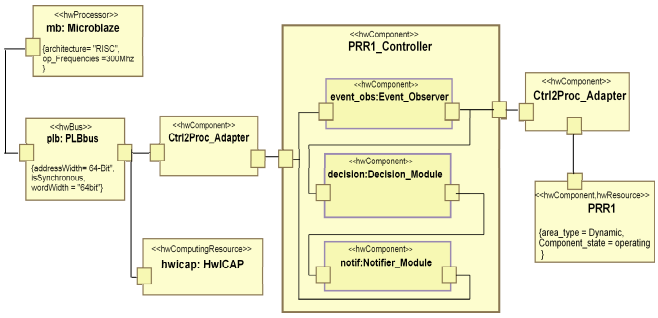


Fig. 2. Modeling of our proposed controller

criteria such as power consumption, performance, etc. This module can be adaptive according to constraints on which it's based.

- Notifier : this module forwards the configuration decision to the ICAP (Internal Configuration Port Access) which will load the chosen configuration.

- Ctrl2PRR_Adapter :adapts signals between our modular controller and its corresponding reconfigurable region.

Our proposed controller is highly flexible thanks to its adaptivity and modularity. In fact, the Event_Observer and the Decision module are adaptable and each one can contain different functionalities according to observed data and events,respectively. Figure 2 shows a high level model of our proposed modular controller using MARTE profile concepts. Each component contains the MARTE HRM *HwComponent* stereotype to indicate that these modules are physical in nature.

As shown in Figure 2, we use a softcore Microblaze processor as example of a master controller which will monitor all the distributed controllers. It has MARTE HRM *HwProcessor* stereotype and is connected to the *controllerToProcessor_Adapter* via the Processor Local Bus (PLBbus). The Partial reconfigurable region (PRR) typed as the generic *HwResource* type in order to illustrate that the partially reconfigurable region can be either generic or have a specific functionality. This PRR consists of a reconfigurable Hardware Accelerator (HwAcc) defined as *HwPLD*, as it is reconfigurable compared to a typical hardware accelerator in a large-scale SoC design which can be seen as a HwASIC (after fabrication).

4. CASE STUDY

We present here a case study of a complete FPGA model to illustrate our modeling methodology.

4.1. Application level

The H.264 is currently one of the most commonly used formats for the recording, compression, and distribution of high definition video [12]. This standard has already been adopted for a wide range of applications, from low bit-rate mobile video to high-definition TV. Like other previous video coding standards, the H.264 utilizes video coding algorithms based on block-based motion compensation and transform based spatial coding framework. The H.264 video encoder treats the frames as I (Integral frame) or as P (Partial frame)[13]. The I frames are encoded without taking into consideration the previous frame, while the P frames are treated taking into account the information existing in the previous frame. Figure 3 describes the encoding principle for a PFrame. Just like encoding an I frame, the algorithm partitions the frame into 16x16 pixels MacroBlocks (MB).

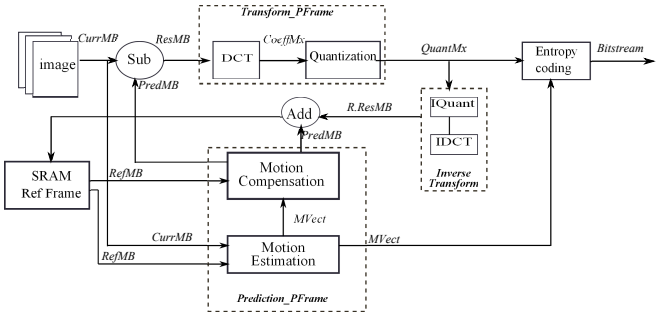
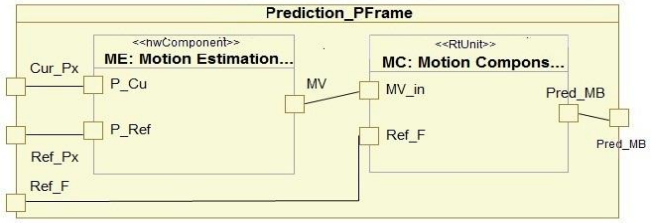


Fig. 3. Simplified block diagram of a P frame H.264 video coding

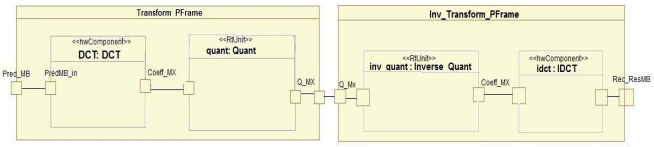
Motion Estimation (ME) compresses the temporal redundancies between consecutive frames [14]. The algorithm compares a *Current MB* (CurrMB)(16x16 pixels) from the current frame to all the candidate blocks in a search window in the previous frame and the best match is used to estimate the motion. For each CurrMB, the ME module can obtain

two possible results. The first possibility is that this MB is similar to the previous one called *referential MB* (RefMB) (with possible presence of small differences). The second possibility is that they are completely different, i.e., the current MB contains 16x16 new pixels. In the case of similarity, the ME will provide a *Motion Vector* (MVect) for the CurrMB to indicate its origin from the previous decoded frame. Then, the *Motion Compensation* module (MC) takes as inputs this MVect and the RefMB. Then, using temporal prediction, it provides a *Predicted MB* (PredMB), which will store the full information needed to transform the previous frame into the next frame. Nonetheless, the similarity between CurrMB and PredMB is not absolute. This leads to use *Subtractor module* (Subtract) to compute differences between them in order to obtain a MB that contains only residues (extra data) called *Residual MB* (ResMB). Therefore, after the ME, MC and difference phases, the current MB which initially contains a 16x16 original pixels, will contain only some few data. The rest can already be found somewhere in the previous decoded frame. By using the Discrete Cosine Transform (DCT), the frame values stored into a MacroBlock will be transformed into frequency values. The DCT video coding, particularly the 2D DCT, can be decomposed as two 1D DCT computations together with a transpose memory in order to reduce the computational complexity. The DCT block implements a row-column Distributed Arithmetic (DA) [12] version of the Chen fast [15] DCT algorithm. Furthermore, 1-D DCT is first performed row by row on the input data and the results are saved in a transpose memory. Then, 1-D DCT is performed column by column on the results stored in the transpose memory. The outputs obtained from the second 1-D DCT are the coefficients of the 2-D DCT, i.e., a Coefficient Matrix (CoeffMX). Using the transformed MB after the DCT function, the next step is to quantize (*Quant*) these values by removing a specific number of bits from all the 8x8 DCT values. This number can vary from frame to frame but is unique during the quantization of the MBs from the same frame. As a result, we obtain a quantization matrix (QuantMx). After that, an inverse transform (inverse quantization (IQuant) and inverse DCT (IDCT)) is applied to this matrix to generate a reconstructed residual MacroBlock (R.ResMB). This one in sent to an *adder module* (Add) and finally stored in the SRAM memory. The last step is to code quantized values using *Entropy Coding* (EC) techniques. Entropy coding is a term referring to a lossless data compression scheme that replace data elements with coded representations which, in combination with the previously-described predictions, transformations, and quantization, can result in significantly reduced data size [16]. In H.264, two modes of EC designs are used in a *context adaptive* (CA) way: CA *Variable Length Coding CAVLC* and CA *Binary Arithmetic Coding CABAC*. After processing all the MBs of the frame using the P frame coding, the final result that we obtain is a bitstream.

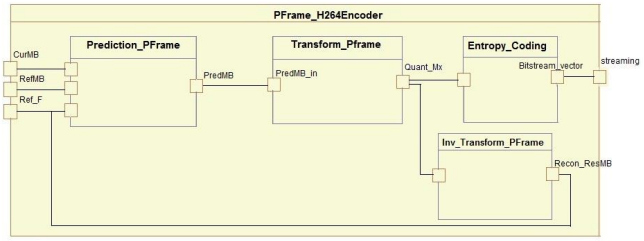
This application is modeled with UML MARTE concepts using HLAM (*High-Level Application Modeling*) and HRM (*Hardware Resource Modeling*) packages. As shown in Figure 4, we apply MARTE HLAM *RtUnit* stereotype to Motion Compensation, Quantization and Coding tasks, to indicate that the component is a realtime unit which means that they are allocated to a platform computing component (in our case : a processor). While we apply MARTE HRM *HwComponent* stereotype to DCT and ME tasks in order to specify that these tasks are performed on hardware accelerators.



(a) Prediction PFrame modeling



(b)Modeling of Transform and Inverse Transform PFrame



(c) PFrame modeling

Fig. 4. Modeling of the H.264 video coding

4.2. System architecture

In Figure 5, we present the global structure of our reconfigurable architecture, implemented on the Xilinx Virtex 4 XC4VFX20. Our system can be mainly divided into two regions. The static region and the dynamically reconfigurable one. The static region mainly consists of a softcore Microblaze processor which is selected as the global reconfiguration controller and other necessary peripherals for dynamic reconfiguration. The Microblaze processor connects directly to the high speed 64-bit Processor Local Bus (PLB) and links with On-Chip Peripheral Bus (OPB) via a PLB to OPB Bridge (plb2opbBridge). Several slave peripherals are connected to the OPB bus, such as SystemACE controller for accessing the partial bitstreams placed in an external on board Com-

pact Flash (CF) card, an ICAP core (Xilinx OPB HwIcap module) which carries out partial reconfiguration using the read-modify-write mechanism. An UART controller module is connected to a hyperTerminal from which the user inputs/events are sent to the processor. Finally, four processors generated from the modeled application are connected to PLB bus, i.e.,*quant_pr* (quantization), *cod_pr* (coding), *sub_pr* (subtractor) and *MC_pr* (Motion Compensation).

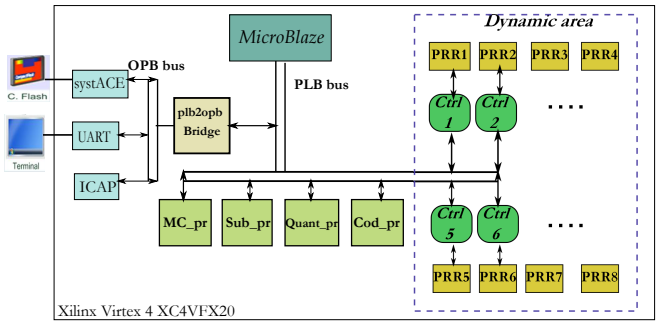


Fig. 5. Block diagram of the architecture of our reconfigurable system

In video coding application, ME and DCT are the most compute intensive processes. Inspired by proposed architecture in [17], dynamically reconfigurable region contains eight hardware accelerators (HwAcc) connected to PLB bus via bus macros. Each one serves as a partial reconfigurable region (PRR) which can be dynamically reconfigured using partial reconfiguration. To achieve quality scalability of DCT computations, hardware Accelerators can be reconfigured in two ways. They can be removed or added to achieve zonal coding for the DCT coefficients (from 1x1 to 8x8), or we can simply change the internal logic of the HwAcc through dynamic partial reconfiguration to reduce the precision of resultant DCT coefficients. Since, ME is a highly parallel application, it should be implemented on a Hardware accelerator (HwAcc). Therefore, we can use ME computation as an example to demonstrate how to take advantage of the unused HwAccs.

Figure 6 finally shows our reconfigurable architecture (Xilinx Virtex 4 XC4VFX20 chip) using MARTE concepts in a merged functional/physical view to express all the necessary attributes related to the corresponding physical/logical stereotypes. We use 4x4 DCT with 4 HwAccs for ME as an example for modeling with MARTE profile.

5. ADVANTAGES OF OUR SOLUTION

Our proposed solution combines mainly three aspects : *distribution*, *adaptability* and *modularity*. In this section, we illustrate the advantages of our approach with some examples.

5.1. Spatial distribution:

- *Performance:* In modern embedded systems, the reconfiguration controller can significantly affect the system performance. When the performance of a centralized controller is not sufficient, often it is a good idea to divide the control program to several pieces and run them in different devices. Indeed, using only one global controller to manage several partial reconfigurable regions can lead to deadlocks and consequently performance degradation.
- *Complexity:* Partial dynamic reconfiguration becomes one of the most key issues in today's FPGAs. These systems are becoming more agile to meet the requirements of ever changing markets. Due to the presence of several partial dynamic regions (PRR) in an FPGA, it is mandatory to efficiently manage them. Therefore, implementing a distributed controller to manage each PRR enhances flexible reconfiguration and offers reusability of the controllers. Such a splitting may bring an immediate gain in the productivity and the complexity of a global reconfiguration management is reduced significantly.

This solution makes the control design less complex and increases the performance, compared to a centralized solution.

5.2. Modularity and adaptivity:

- *Reconfiguration Time:* Three main functionalities are handled by our modular controller: observation, taking reconfiguration decision and notification. The main objective for proposing a modular controller is to ease the reconfiguration of some sub-modules of our distributed controller without needing to change the others. Since an unused HwAcc can be utilized by dynamic partial reconfiguration for other functions, Motion Estimation and DCT can be performed concurrently. In this case, this accelerator controller will not totally change. Furthermore, the kind of data observed by Event_Observer module is a pixel, i.e., MacroBlock. So that only the decision functionality changes and adapts to the ME's constraints. Thus, in this case, we will gain in terms of time reconfiguration.
- *Energy consumption gain:* The Event_Observer module can detect the current activity of the processing element (hardware accelerator) and the related power consumption. Thereafter, it sends this information to the decision module. If the total consumption exceed a certain threshold, the monitor can decide that a reconfiguration is needed. So, it sends the decision to the processor responsible for the reconfiguration launching. A reconfiguration can, for example, load another version of the same algorithm but with a lower performance. For an image processing application this would be done by loading, for example, a filter that is less precise than the previous one. Moreover, in order to process a High Definition frame (Event_Observer module's input is a HD frame), we are dealing then with intensive computing thus increasing parallelism degree. Therefore, we need to

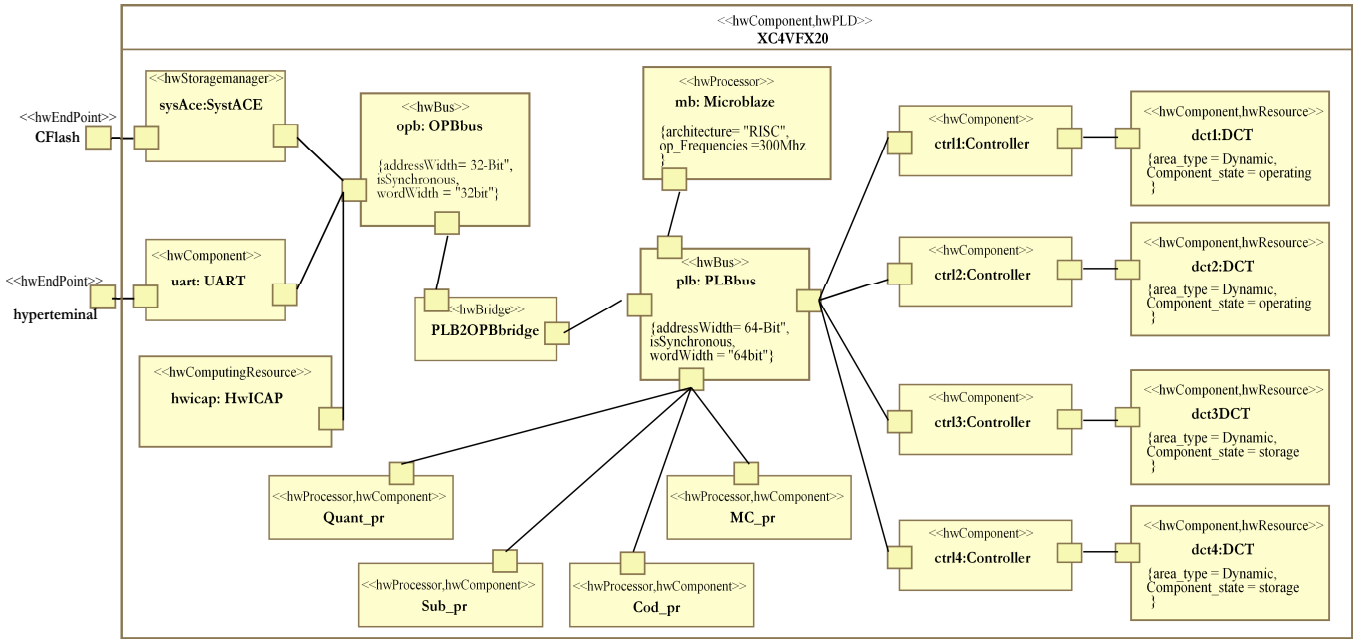


Fig. 6. Modeling of our reconfigurable architecture

increase the number of processing elements. This leads to a higher performance. However, when there is not enough energy in the battery to carry on with the same parallelism degree, decreasing the number of the operational processing elements is required. This reconfiguration consists of switching from a high resolution mode to a lower one (Standard Definition frame). In this case, such information (battery level) is transmitted from the Event_Observer module to the decision module which decides to turn off its corresponding accelerator in order to not consume clock cycles. Obviously a controller of a partial reconfigurable region (PRR) has to collaborate and communicate with other PRR controllers (as any local controller). As in many real situations, the use of a master controller is unavoidable. In our architecture, the Microblaze is considered as the master controller. While the control mechanism depends on QoS choices, the Microblaze can provide changes due to resource constraints of targeted hardware or configuration constraints to local controllers in order to turn on/off processing elements for each reconfigurable region. In the design model, these changes are modeled using *Component_state* attribute from MARTE HRM HwComponent stereotype as shown in Figure 6. A *Component_state* of a partial dynamic region (PRR) set to *Storage* instead of *Operating* means that this PRR is disabled or inactive and does not consume clock cycles. The changes can also depend on other environmental criteria such as communication quality, time and area consumed for reconfiguration; and energy consumption levels. In our proposed solution, reduction of accelerators number (e.g.,DCT dimension from 8x8 to 1x1) can reduce overall power consumption, total number of clocks cycles required for the running module. Thus, our proposed solution can be beneficial in terms of hardware resources and power consumption. The FPGA does not need to be stopped while changing the configuration which is important for many image/video applications. Finally, an efficient reconfiguration would find a trade-off between performance and power.

6. CONCLUSION AND FUTURE WORKS

Dynamic and partial reconfiguration is supported by more and more recent FPGA platforms. Controlling reconfigurations in such systems becomes one of the most difficult and important issues, because of its impact on systems performances. We presented in this paper a new high level approach for modeling and designing dynamic reconfiguration controllers, starting from UML MARTE profile. We validated our design approach with a case study which consisting of a P Frame (Partial Frame) H.264 video encoding. Finally, we illustrated the advantages of our solution with some examples. Our proposed controller is implemented as a reconfiguration distributed hardware component. Its modular and flexible structure make it reusable and permits implementation code generation using MDE approach. Thus, these properties help to decrease design effort and time to market of complex partial and dynamic reconfigurable systems.

As future works, we plan to carry out MDE model transformations to enable automatic code generation of our distributed and adaptive controller in a targeted FPGA. The code can then be used as input for commercial tools for final FPGA

synthesis.

7. REFERENCES

[1] OMG. Modeling and analysis of real-time and embedded systems (MARTE). http://www.omgmarte.org/.

[2] S. Taha, A. Radermacher, S. Gerard, and J.-L.Dekeyser. An open framework for detailed hardware modeling. In *IEEE proceedings SIES'2007*, pages 118 – 125. IEEE, 2007.

[3] A. Koudri et al. Using marte in the mopcom soc/sopc co-methodology. In *MARTE Workshop at DATE'08*, 2008.

[4] J. Vidal, 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)*, pages 226–231, 2009.

[5] S. Pillement and D. Chillet. High-level model of dynamically reconfigurable architectures. In *Conference on Design and Architectures for Signal and Image Processing, (DASIP'09)*, pages 1–7, 2009.

[6] D.Missal, M.Hirsch, and H .-M. Hanisch. Hierarchical distributed controllers design and verification. *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, page 657, September 2007.

[7] S. Karras T. Pfeiffer V. Vyatkin, H.-M. Hanisch and V. Dubinin. Rapid engineering and reconfiguration of automation objects aided by formal modelling and verification. *International Journal of Manufacturing Research*, 1:382 – 404, 2006.

[8] J.-L. Dekeyser, A. Gamatié, S. Meftali, and I. R. Quadri. *Heterogeneous Embedded Systems - Design Theory and Practice*, chapter High-level modeling of dynamically reconfigurable heterogeneous systems. Springer, 2010.

[9] V.Vyatkin and M.Hirsch and H .-M. Hanisch. Systematic Design and Implementation of Distributed Controllers in Industrial Automation. *Emerging Technologies and Factory Automation, 2006. IEEE Conference on*, page 633, September 2006.

[10] Kumpati S. Narendra and Osvaldo A. Driollet. Adaptive control using multiple models, switching, and tuning. *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, page 159 – 164, October 2000.

[11] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J. Stefani. The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. *Software: Practice and Experience*, 36(11-12):1257–1284, 2006.

[12] V. Bhaskaran and K. Konstantinides. Image and Video Compression Standards - Algorithms and Architectures. *Kluwer Academic Publishers*,1997.

[13] Yuri V. Ivanov and C. J. Bleakley. Real-time h.264 video encoding in software with fast mode decision and dynamic complexity control. *ACM Transactions on Multimedia Computing, Communications and Applications*, 6:Article 5, February 2010.

[14] W.Burleson, P.Jain, and S.Venkatraman. Dynamically parameterized architectures for power-aware video coding: Motion estimation and dct. *Digital and Computational Video, 2001. Proceedings. Second International Workshop on*, page 4, Feb 2001.

[15] C.H. Smith W.H. Chen and S. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Trans. Commun.*, COM-25:1004–1009, September 1977.

[16] G. J. Sullivan, P. Topiwala, and A. Luthra. The h.264/avc advanced video coding standard:overview and introduction to the fidelity range extensions. *SPIE Conference on Applications of Digital Image Processing XXVII, Special Session on Advances in the New Emerging Standard: H.264/AVC*, 5558:454–474, August 2004.

[17] J. Huang and M.Parris and J. Lee and R. F. DeMara. Scalable FPGA-based Architecture for DCT Computation Using Dynamic Partial Reconfiguration. *ACM Transactions on Embedded Computing Systems*, pages 1–8, December 2008.