HAL
archives-ouvertes.fr

Managing Dynamic Partial Reconfiguration on Actual Heterogeneous Platform

Jean-Philippe Delahaye, Christophe Moy, Pierre Leray, Jacques Palicot

▶ **To cite this version:**

Jean-Philippe Delahaye, Christophe Moy, Pierre Leray, Jacques Palicot. Managing Dynamic Partial Reconfiguration on Actual Heterogeneous Platform. SDR Forum Technical Conference'05, 2005, Anaheim, CA, United States. 2005. <hal-00084143>

HAL Id: hal-00084143

https://hal.archives-ouvertes.fr/hal-00084143

Submitted on 5 Jul 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MANAGING DYNAMIC PARTIAL RECONFIGURATION ON HETEROGENEOUS SDR PLATFORMS

Jean-Philippe DELAHAYE, Christophe MOY, Pierre LERAY, Jacques PALICOT
IETR/Supelec-SCEE Laboratory, Campus of Rennes, France
{jean-philippe.delahaye, christophe.moy, pierre.leray, jacques.palicot}@supelec.fr

ABSTRACT

This paper deals with partial reconfiguration issues on heterogeneous prototyping platforms (DSP/ FPGA). An analysis of multi-standard physical layer applications in terms of reconfiguration needs permits to extract several use cases of reconfiguration in a multi-standard handset: standard switching, mode switching, bug fixing, etc. The main difference between these schemes of reconfiguration is the level of granularity of the reconfiguration. We argue that a configuration manager needs to handle this multi-granularity of reconfiguration to optimize the change of context (either reconfigurable hardware or programmable software processing components) in terms of size of code, time to reconfigure, etc. The analysis also helps to determine the accurate level of flexibility needed through the reconfigurable architecture at different scales. Within this framework partial reconfiguration is an essential feature for optimizing the reconfiguration inside FPGA. We aim indeed at providing reconfiguration adequacy between re-configurability capabilities of hardware resources and reconfiguration needs of Software Defined Radio applications. Architectural solutions are proposed to implement partial reconfiguration on existing hardware combining DSP and FPGA.

1. INTRODUCTION

SDR is expected to be the most appropriate answer to future multi-standards handsets design challenges [1]. We can predict that SDR systems will be heterogeneous in terms of computing resources, in order to deal with a wide variety of radio applications. This implies many research activities in the fields of multi-processing and heterogeneous computing. All the more so as dynamic reconfiguration is involved. But even if solutions exist for DSP [2] where it is more natural to support reconfiguration, it is less common in the field of FPGA. We propose here a combined approach, from the application side to the implementation on the platform, that particularly match to FPGA. The idea is based on the following major point: the reconfiguration request implied by SDR applications will have multiple level of granularity. Taking into account these granularity levels will be of particular importance to manage efficient and speedy reconfiguration at all levels of the system. The functional architecture proposed here takes into account application needs depending on the granularity of the reconfiguration. This functional architecture allows to extract as much as possible the flexibility offered by any reconfigurable heterogeneous platform. This is straightforward for processor-based systems and particularly suits FPGA despite FPGA's limitations in terms of dynamic reconfiguration [3]. Whereas the full flexibility of our functional architecture is not reached in an implementation using current FPGA (limitation of the column-based configuration memory structure [4] of Xilinx FPGA), it allows to obtain the best of their reconfiguration ability.
The paper is organized as follow. Next part is concerned with an algorithms analysis of a multi-standard transmitter including baseband processing functions of GSM, UMTS UTRA/TDD, and 802.11g OFDM mode. Configuration management needs are deduced. According to the fact that the flexibility is strongly related to granularity, part 3 proposes a hierarchical model of configuration management which allows to handle the multi-granularity of reconfiguration. Part 4 presents the resulting functional hierarchical architecture deduced from the previous point. A concrete implementation is detailed in part 5, insisting on the FPGA dynamic reconfiguration aspects.

2. APPLICATION ANALYSIS AND CONFIGURATION NEEDS FOR CONFIGURATION MANAGEMENT

Basically, The Software Radio aims at providing multi-standards communications accesses using a hardware platform based on shared processing resources. The Software Radio hardware requires to be reconfigurable to be able to switch from one processing mode to an other. It has also to be heterogeneous to face the wide variety of processing categories. It implies that the number of configuration contexts to manage grows as the number of standards increases multiplying the number of different processing algorithms. The number of configuration also depends on the type of reconfigurable resources of the hardware platform. First, it is necessary to do a functional baseband analysis to classify each baseband functions in order to reduce overall configuration management complexity. Having in mind the optimization of the

configuration management complexity, the classification of the baseband functions in terms of parameters, needs of hardware resources and needs of flexibility aims at reducing the number of configuration contexts. The second step of the application analysis is about the classification of the context switching cases which determines the granularity of a reconfiguration.

2.1. Multi-standards baseband functions analysis

The multi-standard functional analysis starts with a first study of the baseband functions of the uplink transmitter of three standards (GSM 900, UMTS UTRA/FDD, WLAN 802.11g mode). These standards have been chosen for their wide variety of baseband signal processing. GSM is a classical mono-carrier TDMA system, UMTS uses CDMA techniques, and finally 802.11g offers a multi-carrier mode (OFDM). Such a multi-standards transmitter involves all of the conceptual challenges, to define a configuration management architecture. We group the multi-standards baseband functions into three functional classes presented below. Gathering functions into classes highly favorises the opportunity to use common operators [5] that can address several processing roles by a simple change of parameters.

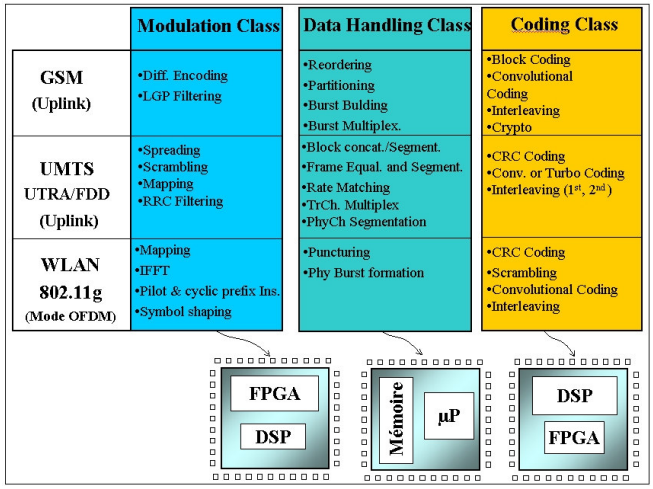

Fig 1: 3 Standard Baseband Functions Classification and the computing demand resources

The **Coding Class** includes functions like cyclic coding, convolutional and turbo coding. Standards use a wide range of coding schemata. Despite the extensive set of parameters to handle the coding functions, these are generally based on few operators as 1-bit linear shift register and modulo-2 adders. A high flexibility of the processing resources is mandatory to reuse efficiently the common operators.

The **Data Handling Class** puts together the functions which manipulate data packets. Due to the nature of functions (concatenation, segmentation, multiplexing, etc.), the data packet lengths are very different. These functions are exclusively dedicated to data transfers. This class of algorithms is largely control-oriented. It mainly requires a lot of memory resources and flexibility to process different kinds of data.

The **Modulation Class** corresponds to the baseband functions which take place before transposition on frequency carrier. In this class, the functions are computation-oriented and data are often oversampled. High throughput is needed by the filtering functions.

The general goal in each class of processing is to group similar processing to maximize the reuse of hardware. Grouping functions into classes helps when dealing with configuration management. Despite grouping the functions in three classes, parameters from a function to an other are quite different. This is the reason why we associated to each function a specific Configuration Management Unit (CMU). Explanations about our configuration management architecture is detailed more in depth in the next section.

2.2. The reconfiguration needs of multi-standards applications.

The general goal of our studies is to reduce de configuration management complexity as the configuration overload by minimizing the resources to reconfigure during any context switching. Many design considerations are involved to optimize the reconfiguration such as the architecture, the design methodologies or the parameterization studies, with a common goal: to enhance the reuse of the processing blocks [6]. In this part, we discuss about the multi-standards application switching needs and afterwards we lean on this analysis to propose our configuration management architecture. The different types of application switches that we define are the following:

Standard Switching: The standard switch is the most demanding one. The transmitting chain between 2 standards are hugely different. Almost, all the baseband processing have to be reconfigured. Moreover, a standard switch not only implies the physical layer but also the other higher layer of the protocol stack.

Mode Switching: A mode switch is considered to be an intra-standard context switching. Some standards define several functional modes. In most of the cases, the mode switch does not include changes of the higher level layer and the mode parameters are often defined by the MAC Layer. For instance in 802.11g is defined with several type of mode DSSS, FHSS, OFDM. So in the case of mode switch, reconfiguration are lighter than in standard switch and it is not worth performing the same huge reconfiguration.

Service Switching: A service switch on the physical layer remains a intra-standard switch rather than a service switch to the application level that could implies a standard switch. A standard switch could be a request to increase the data

rate. For instance in 802.11g mode OFDM, a request to change the data rate from 6Mbit/s to 54Mbit/s implies to modify at least the mapping function from BPSK to 64-QAM. So the service switch could be perform with parameterization and reconfiguration of some of the baseband functions. Then, a partial reconfiguration of the transmitting chain is generally sufficient.

Performance enhancement, bug fixing:

The ability to reconfigure small parts of a processing chain is necessary to allow some performances enhancement or bug fixing when the system is already in use . The possibility to change any processing patterns in chain is closely related to design methodologies which have been used during the definition of the processing chain architecture and the possibilities of the hardware resources to change such small parts of a given function. This type of reconfiguration has to be performed without any discontinuity of service for the user.

Actually, the different kinds of context switching, detailed above should be supported in a SDR system. The given configuration management architecture has to be able to manage the different granularity levels of configuration to optimize the reconfiguration. We propose, in the next section, a hierarchical configuration management architecture that will efficiently enable the handling of the multi-granularity of configuration.

3. HIERARCHICAL CONFIGURATION MANAGEMENT MODELLING

3.1. Config-Data Path Model

The communication applications are dataflow oriented, then our approach, detailed in [7], is based on a data path model. The functions of the SDR transmitting chain are mapped into several Processing Block Units (PBU). Following this approach, each PBU is optimized using specific reconfigurable hardware resources.

In addition, a configuration path, also split into several Configuration Manager Units (CMU), controls the reconfigurable processing path. Each CMU, dedicated to a type of PBUs, manages the configuration of a type of baseband function in the chain. The split configuration path offers the possibility to partially reconfigure the transmitting chain by an independent reconfiguration of each PBU. The distributed configuration management approach also decreases its design complexity.

3.2. Hierarchical Configuration Management Architecture

The section above presented the benefits of the configuration datapath model. It is a base to manage the heterogeneity of baseband processing functions and enable the partial reconfiguration of the SDR transmitting chain. The hierarchical configuration management model illustrated in the Figure 2 is based on the config-data path approach. This model is necessary to manage the multi-granularity of configuration required by the different context switching detailed in the section 2.2. It is composed of three levels of hierarchy that are detailed below.

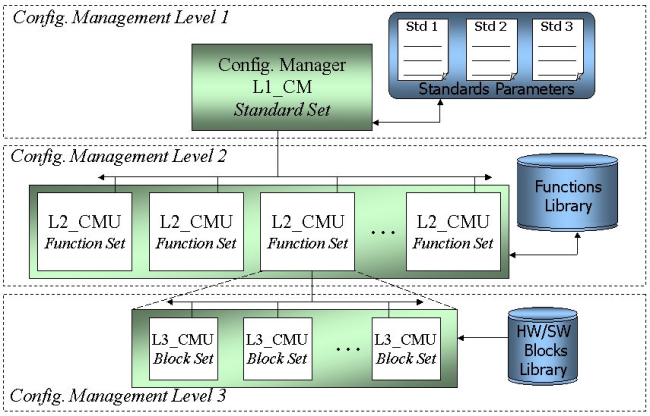

Figure 2: Hierarchical Configuration Management Model

Hierarchical level 1

The first level is split into 3 entities following the functional classification described in Fig 1. This first high level classification allows a control of category-specific functions to manage parameters at the standard level. The *Configuration Manager L1* (*L1_CM*) works at the standard level as host toward the underlying levels of management. This entity is in charge of choosing the functional units which will constitute the entire configuration of the baseband processing chain. At this level, generic functions are handled as generic components. Any hardware implementation is not yet considered.

Hierarchical level 2

The generic functions selected at level 1 are parameterized at level 2 in accordance with standard specifications. The set of attributes of each function is handled by the *Configuration Manager Unit L2* (L2_CMU) to create each functional context of the entire processing chain. For example, the generic function "bit to symbol mapping" of the modulation class is set at level 2 to fit the selected standard. In the case of 802.11g standard this setting could be BPSK, QPSK, 16-QAM, 64-QAM constellation. In the case of a mode switching the needed reconfiguration directly involved the concerning L2_CMUs.

Hierarchical level 3

The processing data path architecture of this third level will depend on the reconfigurable computing resources of the hardware architecture. The complete processing path could be formed by different types of reconfigurable resources in accordance to the studies detailed in the first section. It could correspond to configurable accelerators, array of DSP

blocks or a fine grain reconfigurable data path. The main task of the (*L3_CMUs*) in the configuration path is to find the available processing resources and configure them to enable the execution of the functional context created at the above level.

4. HIERACHICAL FONCTIONAL ARCHITECTURE

This section presents a functional hardware architecture made up of three main reconfigurable clusters. After an overview of this heterogeneous architecture, each cluster is described in further details.

This architecture fits the hierarchical configuration management considerations, proposed in the previous section. It allows first to answer the needs of handling the multi-granularity of the context switching. Second, the hardware architecture is separated into three main hardware clusters. The cluster resources are specialized according to the classification of the baseband functions into three classes presented in section 2.1 and required performances.

These three clusters have the same general architecture: a cluster is designed around a dedicated central processor with separated data bus and program bus *(as a typical Harvard architecture)*. This allows to distinguish the configuration data path from the processing data path. The processor unit of the CM_L1 controls the three clusters to work together. The CM_L1 also manages the whole application parameters.

The configurations of each cluster are controlled by the L2_CM processor units. Each L2_CM is interfaced to a processor designated as master. In addition to this master processing unit, some other reconfigurable units are used to speed up the data processing of the cluster. These reconfigurable accelerators are various as the baseband functions have different needs. So a L3_CM is associated with each different reconfigurable accelerators.

Coding Cluster: Software implementations, for the functions of the coding class, are possible. But hardware will be more efficient and will consume fewer resources. Since data width is only one bit for this class of functions, software implementations are generally under optimized because of using 16/32 bits processors. Furthermore hardware implementation is necessary to reach the expected high throughput performances of 54Mbps in 802.11g. On the other hand, facing the wide variety of coding schemes, software flexibility (DSP) is mandatory. Consequently, the coding cluster is composed of DSP aided by reconfigurable co-accelerators accelerated through efficient dedicated coprocessors corresponds to a suitable architecture for this class of processing functions.

Data Handling Cluster: The high flexibility offered by the GPP processors is interesting to run the wide variety of handling functions of every standards. The data handling functions almost perform only memory transfers, so a DMA interface is useful between the processor and the memory. The memory is a sizable arrays of SRAM blocks, it allows to resize the array by switching-off the unused memory blocks, depending on the handling function. Power saving architecture design, for this class of functions, is a key factor, as memory consumes a lot of power.

Modulation Cluster: Most functions handle data, up to 32 bits, which takes a lot of resources. Consequently, some of these functions, like pulse shaping filtering will take advantage of an implementation in dedicated configurable hardware accelerators. One other point is the intensive processing requirements of the Modulation class functions which often necessitates a HW implementation.

The main goal of this model is to help defining features needed by the hardware platform for SDR applications. The Fig 3 depicts the model as an architecture with lots of duplicated resources (multiple microprocessors and buses). This just illustrates, for a good understanding, our hierarchical view of the hardware platform.

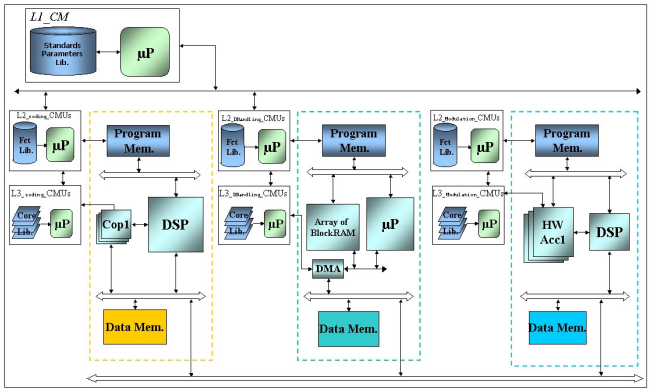

Fig 3: Hierarchical functional architecture

For instance, in the CM part of the model, the multiple microprocessors (called µP) are instantiated as multiple tasks running on a processor of the hardware platform.

The next section illustrates in a more detailed manner the effective implementation of this functional architecture in a real platform.

5. HETEROGENEOUS HARDWARE SDR PLATFORM

The use of reprogrammable (DSP), reconfigurable (FPGA) devices [8] and more generally reconfigurable computing architectures is commonly admitted for Software Radio. We present the heterogeneous (GPP/DSP/FPGA) platform and especially FPGA reconfiguration requirements to enable the implementation of our model. We focus on the FPGA, because it is foreseen as a good intermediate architectural possibility between DSP and ASIC. FPGA combines the flexibility of DSP and the throughput efficiency [9] of

ASIC. But this really makes sense only if we can efficiently manage reconfiguration of FPGA. Mandatory requirements are partial and dynamic capabilities of reconfiguration for FPGA. Currently, a few FPGA devices enable dynamic and partial reconfiguration [10]. Moreover, the system architecture [11], [12] is a key factor to make embedded these kind of reconfigurations.

5.1 From the Architecture Model to the platform.

The implementations on the platform are currently under development. We present in this part the platform and the drawn up implementations.
Our prototyping platform is composed of a GPP, a DSP, a FPGA and different types of memories. The functional architecture of the Fig 3 is mapped into this platform as it is illustrated in the Fig. 4. It consists in gathering on each category of HW processing device (GPP, DSP, FPGA) on the one hand the PBU (in white on Fig. 4) and on the other hand the CMU depending on their hierarchical level.

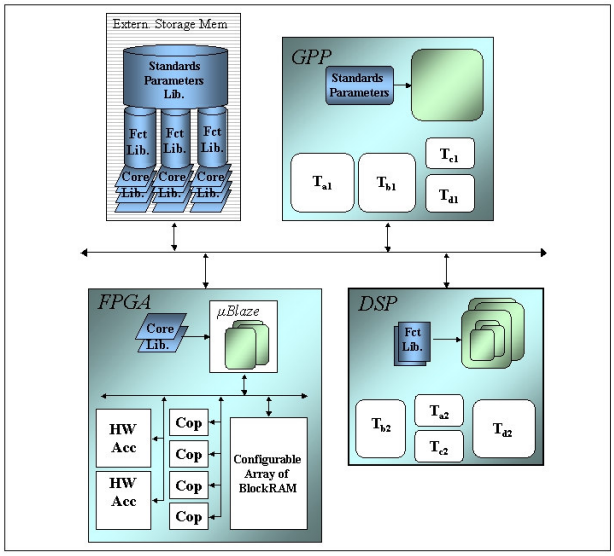

Fig. 4: Platform architecture

The **GPP** and the external storage memories are resources used from a standard PC station. The GPP is the host of the rest of the platform. The L1_CM is a task running on the GPP which controls the configuration the other platform resources (DSP, PFGA). The control-oriented functions as some Data Handling functions take advantages of the flexibility of GPP.
The **DSP** is a C64 from TI. The DSP parts from the both clusters Modulation and Coding Clusters are tasks running of the C64. The DSP works as the master of the (DSP/FPGA) subpart of the platform, Then the L2_CMUs of the three cluster are also mapped on the DSP. The position of master allows the DSP to manage the overall configuration of the functions that run on the cluster resources.
The **FPGA** of our platform is a Virtex -II device from Xilinx. The following components of the functional model: the hardware accelerators, the small co-accelerators and the sizable array of blockRAMs are mapped into the FPGAs. The partial reconfigurability is of course a mandatory features to allow reconfiguration of a single component. The L3_CMUs responsible for the configuration of the co-accelerators are implemented as task into the µBlaze soft processor. It allows to perform fine grain reconfiguration of the FPGA without involving any external resources. Next section details furthermore the internal architecture of the FPGA to enable the partial dynamic reconfiguration.
Components Library: The Hardware and software designs of the processing functions are stored in the external storage memory of the platform where the configuration management takes them.

5.2 FPGA architecture on the embedded platform.

At the initial stage the CM of the Host (GPP) downloads the DSP boot program that includes in its data memory the initial full configuration of the FPGA. It consist in the FPGA design architecture. It includes an internal configuration controller (µBlaze soft processor), the internal reconfiguration interface (ICAP), the initial instantiations of PBUs and the communication interfaces with the DSP.

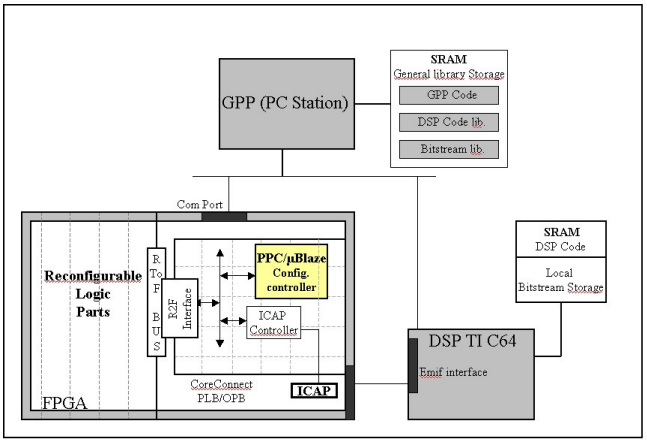

Fig. 5: Details of the FPGA architecture

The Fig. 5 illustrates this platform architecture with details of the internal FPGA design that enable two types of dynamic partial reconfiguration depending on their granularity level. One stays internal in case of limited-scale reconfiguration (for co-accelerators configuration) or design parameterization (auto-reconfiguration). This implies to interconnect the µBlaze to the ICAP internal configuration interface. This kind of reconfiguration of the FPGA by an processor (µBlaze) embedded in the FPGA is called self-

reconfiguration or auto-reconfiguration [13]. In this case, small partial bitstreams are stored inside the FPGA, and the use of auto-configuration let free the other HW resources of the platform. At a larger scale reconfiguration for the HW Accelerator is external. This implies to interface to interconnect the DSP to the external SelectMap or internal ICAP reconfiguration interfaces. The Bitstream corresponding to the design of HW Acc. are stored in an external SRAM memory.

6. CONCLUSION

Through the proposed models, this paper shows the trade-offs to provide a full reconfiguration flexibility with a reasonable hardware complexity from configuration management to the hardware architecture. Actually, the hierarchical functional architecture offers the maximum of the reconfiguration flexibility. It allows to take as much flexibility as possible from any kind of heterogeneous platforms. As this functional architecture is not specific to any platform or device, it will improve itself and will easily take advantages of the future evolutions of reconfigurable computing technologies in terms of flexibility. Following this idea we already achieve to take into account existing limitations concerning the partial reconfiguration of FPGA.

7. REFERENCES

[1] J. Mitola, "The software Radio architecture," *IEEE Comms Mag*, vol. 33, no. 5, pp. 26--38, May 1995.

[2] C. Moy, A. Kountouris, A. Bisiaux, "HW and SW Architectures for Over-The-Air Dynamic Reconfiguration by Software Download," SDR Workshop of the IEEE Radio and Wireless Conference, Boston, USA, Aug. 2003

[3] J.P. Delahaye, G. Gogniat, C. Roland, P. Bomel, "Software Radio and Dynamic Reconfiguration on a DSP/FPGA Platform," 3rd Karlsruhe Workshop on Software Radios, proc. pp 143-151, Karlsruhe Germany, March 17-18 2004.

[4] "Virtex Series Configuration Architecture User Guide," Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124,XAPP151 (v1.6) March 24, 2003

http://www.xilinx.com/bvdocs/appnotes/xapp151.pdf.

[5] J. Palicot, C. Roland, "FFT a basic Function for a Reconfigurable receiver," ICT Conf., Tahiti, 2003.

[6] DeHon, J. Adams, M. DeLorimier, N. Kapre, Y. Matsuda, H. Naeimi, M. Vanier, and M. Wrighton., "Design Patterns for Reconfigurable Computing," fccm, pp. 13-23, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04), 2004.

[7] J.-P. Delahaye, J. Palicot, P. Leray, "A Hierarchical Modeling Approach in Software Defined Radio System Design," SIPS 2005, Athens-Greece, Nov. 2005.

[8] M. Cummings, S. Haruyama, "FPGA in the Software Radio," *IEEE Comms. Mag.*, vol. 37, no. 2, pp. 108-112, Feb. 1999.

[9] T. Claasen, "High Speed: Not the Only Way to Exploit the Intrinsic Computational Power of Silicon," Digest of Tech. Papers ISSCC 99, IEEE Press, pp. 22-25, 1999.

[10] S. Donthi and R.L. Haggard, "A survey of dynamically reconfigurable FPGA devices," Proc. IEEE Symposium on System Theory, pp. 422-426, 2003.

[11] M. Ullmann, M. Huebner, B. Grimm, J. Becker; "An FPGA run-time system for dynamical on-demand reconfiguration," 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.pp.135, 26-30 April 2004.

[12] J.C. Ferreira, M. M. Silva, "Run-time reconfiguration support for FPGAs with embedded CPUs: The hardware layer," Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05).

[13] B. Blodget, P. James-Roxby, E. Keller, S. McMillan and P. Sundararajan. A Self-reconfiguration Platform. In proceeding of 13th International Conference on Field- Programmable Logic and Applications, FPL'2003, pp. 565-574. Sept. 2003, Lisbon, Portugal.