AN EMBEDDED CONTROLLED FLEXIBLE BASEBAND PROCESSING APPROACH

Thomas Loewel (Bell Labs, Alcatel-Lucent Deutschland AG, Berlin, Germany; thomas.loewel@alcatel-lucent.de); Andreas Wich (Bell Labs, Alcatel-Lucent Deutschland AG, Stuttgart, Germany; andreas.wich@alcatel-lucent.de); Wolfgang Koenig (Bell Labs, Alcatel-Lucent Deutschland AG, Stuttgart, Germany; wolfgang.koenig@alcatel-lucent.de); Ferenc Noack (Bell Labs, Alcatel-Lucent Deutschland AG, Berlin, Germany; ferenc.noack@alcatel-lucent.de); Christian Lange (Bell Labs, Alcatel-Lucent Deutschland AG, Berlin, Germany; christian.lange@alcatel-lucent.de)

ABSTRACT

A flexible base station (FBS) is an enabling element for the realization of cognitive radio networks. It consists of a multi standard transceiver and a flexible baseband processing (FBP) unit. The key features of such a FBP unit are the support of multiple radio standard processing chains (e. g. Universal Mobile Telecommunications System (UMTS), Long Term Evolution (LTE)), the dynamic load balancing between these standards, the flexible assembly of radio functional modules and the simple extension of radio functional modules and thus the radio standards.

The paper describes an experimental FBP design and the necessary applications for the control of this design. The described design will be mainly realized with field programmable gate arrays (FPGA) and the partial reconfiguration (PR) technology. For that, the FPGA area will be divided into smaller variable reconfigurable areas and an interconnection system between these variable areas will be presented. The flexibility of the baseband processing can be further increased by introducing parameterization of the functional modules where appropriate.

A framework is required to control the FBP. The corresponding approach presented in this paper is the introduction of an FPGA embedded framework. This framework operates on an embedded processor and will be controlled by higher layer applications. The corresponding interface based on the Object Management Group (OMG) software radio specification will be also briefly described in this paper. Furthermore an application for the direct control of the framework will be introduced. This application offers a graphical user interface (GUI) and uses the OMG-based interface mentioned above. The last topic inside this paper is an approach on modeling and performance analysis for verification of flexible baseband processing systems. This topic discusses the exploration of the design space for flexible reconfiguration architectures.

1. INTRODUCTION

A flexible base station supports more than one radio technology, possibly future radio technologies and should be reconfigurable according the requirements of management modules interacting with the FBS [1]. The FBS itself consists of two major elements, the radio frontend and the baseband processing. There exist a lot of flexible solutions (e. g. regarding frequencies, radio standards, bandwidths etc.) for both elements. For the baseband processing the flexibility can be reached by usage of FPGAs. Assembly and reconfiguration of the FBP chains using of FPGA is a complex and difficult procedure. Thus, a special architecture is needed. A possible approach is described in this paper. The architecture for the approach consists of three major elements [1] (see also Figure 1):

- A control application, which establishes the communication with management modules,
- An embedded control system on the FPGA, so-called System on a Programmable Chip (SoPC), which controls the reconfiguration,
- An element located near by the SoPC - the FBP.

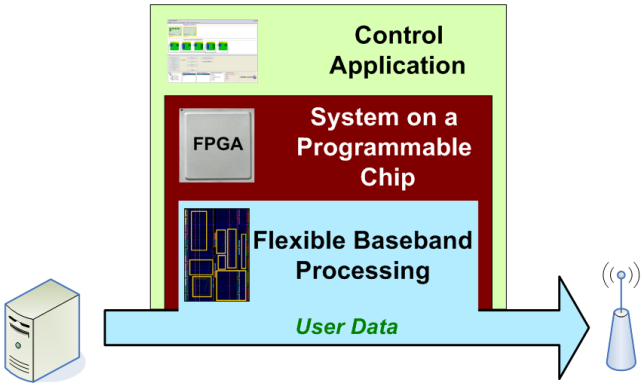

Figure 1: Architecture of the Flexible Baseband Processing

2. THE CONTROL APPLICATION

The first element of the envisaged architecture is the control application [1], [2]. This application represents an interface between different network management applications and the FBS. One major task of this application is the translation of reconfiguration commands from the different management modules into well-known FBS statements. For that, a lot of configuration commands between the application and the management modules are specified. Every command will be analyzed and a set of OMG-based commands [3], [4] will be composed. These commands (Allocate Capacity, Initiate Load, SendSegment, Execute, Initialize, GetProvidedPorts, ConnectPort, ConnectExternalPort, Configure, Start, Stop, Disconnect, Release, Terminate, Unload, DeAllocate) [1], [2] are OMG compliant with some modification regarding the applicability with FPGAs. The control application itself offers a GUI for the direct control and monitoring of the FBS, see Figure 2.

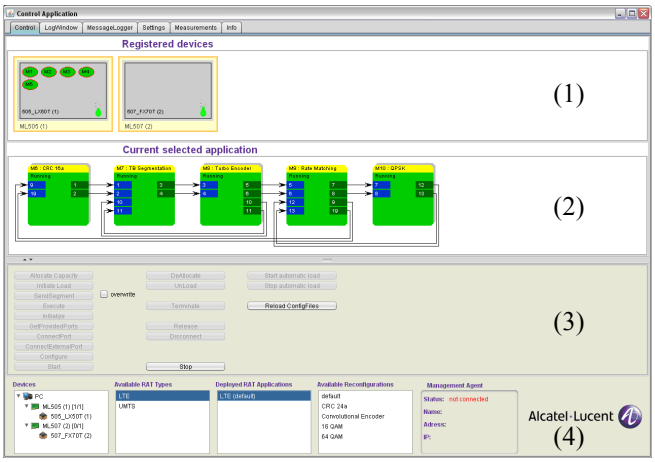

Figure 2: Control Application Overview

Window (1) gives an overview about all registered and working devices. These devices can be for example baseband processing or transmitter boards. Furthermore all running radio modules on each baseband device are shown. Window (2) gives a detailed overview about the current shown radio technology (selected in the window (4)). This overview includes the status of each module and the interconnections between the modules. Window (4) gives an overview over all available devices, the available radio access technologies (RATs) and the activated RATs. Furthermore detailed hardware information about all available devices are shown and a dedicated RAT for the detailed view in window (2) can be selected. Windows (1), (2) and (4) are of status and an informational character only. Window (3) provides direct control functionality for the FBS. Within this window all suitable OMG-based commands offer an easy and detailed setup of the baseband processing chain.

3. THE SYSTEM ON A PROGRAMMABLE CHIP

The System on a Programmable Chip (Figure 3) is one major element for the flexibility and serviceability of the flexible baseband processing [1] and has physical interfaces to the outside, e. g. to the control application. Currently, especially the PCI Express and the Gigabit Ethernet interfaces are supported by the SoPC. The master device of the SoPC is a microprocessor (e. g. MicroBlaze or PowerPC (PPC)). This microprocessor receives the control and reconfiguration commands and performs measurements. Furthermore, the microprocessor executes control and reconfiguration commands and thus the setup and management of the baseband processing. A special POSIX-based Kernel, the Xilkernel [5], is running on this system. Primarily a lot of threads are executed on this Kernel. There are threads to communicate with higher layer control modules, to perform load measurements, to execute the partial reconfiguration [6], [7], to provide parameters etc. All these threads enable the features of the whole system and form the FPGA embedded framework.

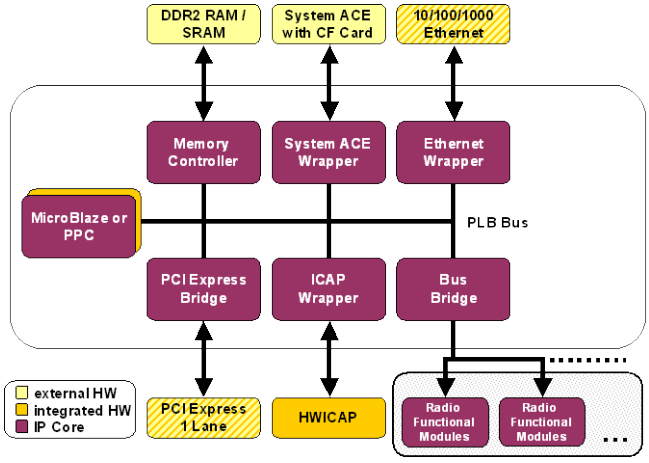

Figure 3: Architecture of the SoPC

The Processor Local Bus (PLB) of the IBM CoreConnect architecture [8] has been chosen as bus system for the SoPC. Its performance is sufficient for the data exchange between all SoPC components. This data includes all kinds of control and measurements, but not the user data. The user data will be processed within the baseband processing chains and not in the SoPC. Nevertheless the SoPC controls the baseband processing and thus indirectly the processing of the user data.

Besides the microprocessor a set of standard components complete the SoPC [1]. The memory controller for instance is needed to provide additional memory to the system. Especially the microprocessor requires a lot of additional memory and the SoPC internal memory is sometimes insufficient. An important component is the Internal Configuration Access Port (ICAP).

This component performs the partial reconfiguration [6], [7] and thus the setup of the baseband processing chains. A special wrapper in the SoPC enables the access to this hardware module. The System Advanced Configuration Environment (System ACE) wrapper enables the access to Compact Flash cards. These cards relate to local databases and store all needed files for the normal and partial reconfiguration. The Bus Bridge is a special intellectual property (IP) Core for the communication between the SoPC and the FBP chains. This communication includes, for instance, commands for the controlled setup of radio standard processing chains, for the reconfiguration of these chains, for the reconfiguration of one or several radio functional modules or for the provision of parameters.

4. THE FLEXIBLE BASEBAND PROCESSING

The baseband processing is one component for the realization of the radio access technology. This processing should be as flexible as possible for a FBS. A couple of well-know architectures for the realization of the baseband processing (see Figure 4 and Figure 5) can be considered.

One architecture is a classical bus system, which is good upgradeable and serviceable, but the throughput is limited due to common bus for all radio functional modules.

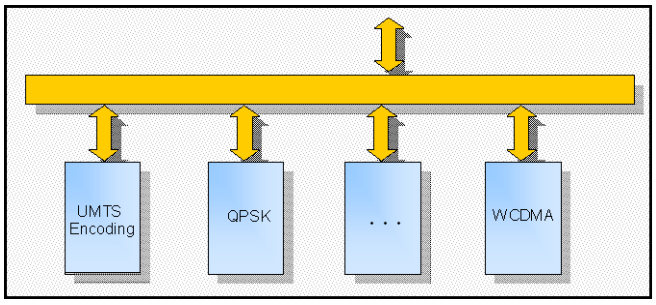

Figure 4: Bus System

The second architecture is the classical signal processing chain. This chain is less flexible, but the throughput can be much higher compared to the bus system.

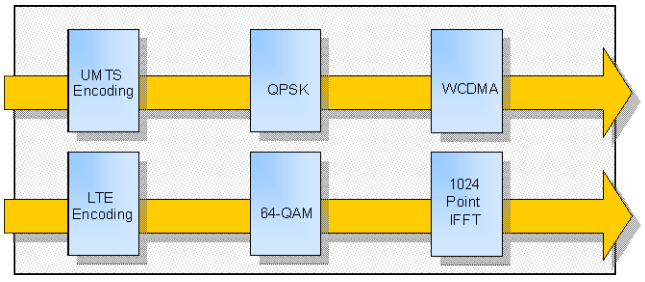

Figure 5: Signal Processing Chains

One limitation of the signal processing chains is the static assembly of the radio functional modules and thus the fixing of a radio technology.

One approach to bring more flexibility to this architecture is introduction of parameterizeable radio functional modules. Consequently some possible variations of a radio access technology can be supported with only one baseband processing chain.

Another approach to bring more flexibility to the baseband processing is the use of the partial reconfiguration technology from Xilinx [6], [7]. For that, a static signal processing chain will be designed. This chain has one or more placeholders without any functionality, so-called partial reconfiguration regions (PRRs). The embedded microprocessor sets up a baseband processing via the ICAP and initializes them via the Bus Bridge. All needed radio functional modules will be placed in the PRRs. A good design of the static chain is important to reach a maximum of flexibility. Figure 6 shows a possible segmentation of an FPGA into smaller PRRs.

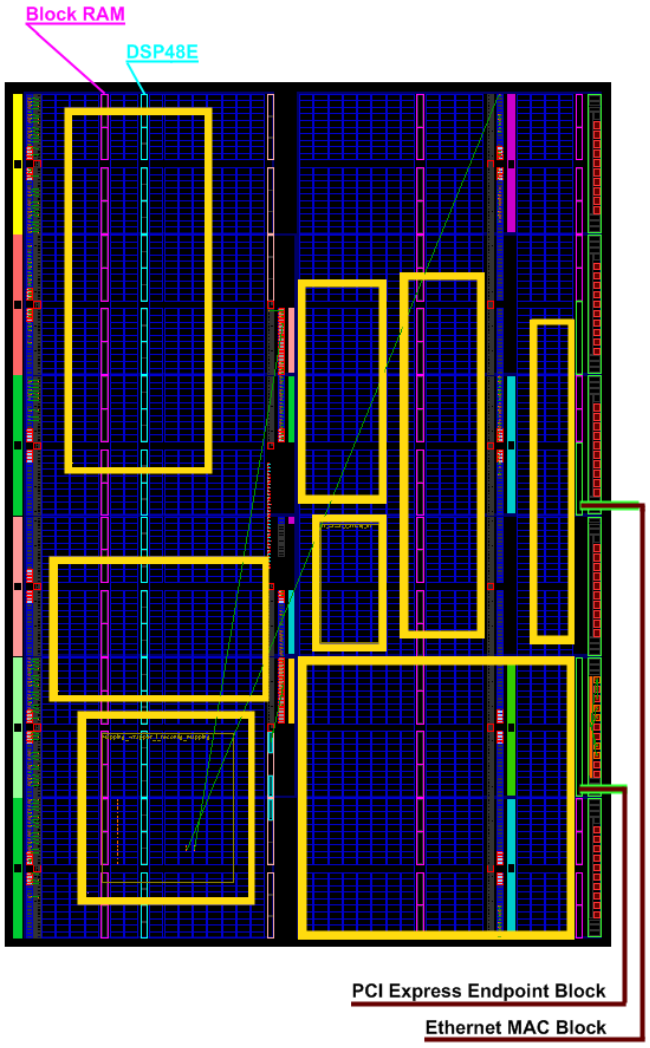

Figure 6: FPGA with PRR

The Segmentation considers the resources of an FPGA. Thus there exist regions with Block RAMs (BRAMs) and DSP48E elements, regions without DSP48E elements, regions with needed resources regarding the PCI Express and the Gigabit Ethernet interfaces, regions with no special resources (logic only) etc.

The combination of the parameterizability and the partial reconfiguration is one key enabler of the flexible baseband processing. These two approaches offer the following options to setup or reconfigure a baseband processing chain of a FBS [1], [2]:

- Setup a radio standard processing chain,
- Replace a radio standard processing chain by another,
- Change functionalities of a processing chain by replacing several modules,
- Change parameters of a radio technology module.

For replacement, setup or change of functionalities of a radio standard processing chain a lot of radio modules must be arranged and configured. Three major independent cases can be separated [1], [2].

The first case is related to completely different radio modules when switching to a different radio technology. In this case all radio modules of a chain must be changed. This process can be done via partial reconfiguration. Thus all other processing chains keep alive and only the addressed processing chain has an outage period. All needed radio modules are stored in the local module database or can be downloaded via the control application.

The second case is related to very common and good customizable radio modules among different technologies. In this case the reconfiguration can be done by changing parameters only. The parameters will be provided by the control application and the SoPC. The outage time in this case is very small compared to first case. In analogy to the first case all other processing chains will be not affected.

The third case is a mix between the first and the second case. One example for this case is the multiplexing in current and future radio technologies. In UMTS the multiplexing will be realized via Code Division Multiple Access (CDMA). The multiplexing in Worldwide Interoperability for Microwave Access (WiMAX) is completely different and will be realized via the Inverse Fast Fourier Transform (IFFT). Thus, the complete multiplexing module must be exchanged (via partial reconfiguration, first case) during the reconfiguration from UMTS to WiMAX. In the same context (regarding the multiplexing module) only the IFFT size must be configured during the reconfiguration from WiMAX to LTE. This can be done via parameters (second case). Figure 7 summarizes all cases and approaches.

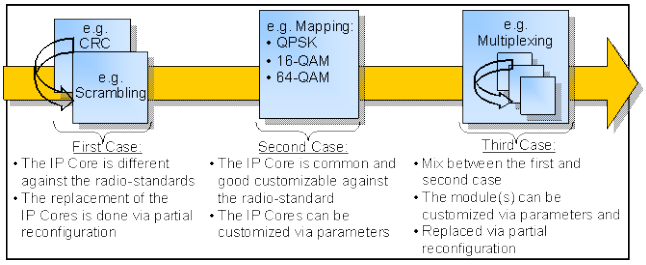

Figure 7: Flexible Baseband Processing Chain

5. SYSTEM MODELING AND PERFORMANCE ANALYSIS FOR VERIFICATION

5.1. System Modeling

A crucial aspect in a flexible multiple radio standard approach is the flawless functional performance of the radio standards. This includes the proper functionality of single modules and the overall processing flow as well as the meeting of real time constraints inherent to the standards. Together with dynamic reconfiguration and a considered hardware-based load management, this poses new challenges to the design, verification, simulation, test and debugging of such flexible, heterogeneous hardware systems. The question arises how to introduce these aspects early into the system design process and how to keep they streamlined throughout the process. Several methodologies and approaches exist which could be utilized to achieve this. Most of them are utilized in the upcoming system level design (SLD). An overview of SLD methodologies and tools is given in [9] and a thorough consideration of the approach is presented in [10]. In short, a good modeling approach must guide us through most of the design process, must be able to support different abstraction levels within one model and allows for successive model refinement, must include functional, functional timing and if necessary hardware timing aspects, includes the separation of functionality/computation and communication and should allow us to automatically verify the design against specified properties and constraints. Finally, generation of code for a given hardware platform, with the possibility of exploiting different architectural options ("design space exploration" (DSE)) would complete the "perfect" modeling approach.

As for today, no single modeling methodology or tool is known that supports this. Still many commercial and academic tools and languages exist that cover this or parts of this approach, e. g. Berkeley's Ptolemy [11], which includes modeling and model execution of concurrent real-time embedded systems.

In a pragmatic approach, existing languages like SystemC, VHDL or SystemVerilog [12] could be used for this purpose, since they allow for abstracted modeling even of analog and mixed signal components (VHDL-AMS).

Still there is a strong focus on (hardware-)implementation of those languages which makes it difficult and time-consuming to use them on system level and for abstracted multi-level designs.

So in general, working with a system level model leaves us with nothing executable but has the "little-change-big-impact" potential. Also, good performance estimation is difficult to achieve since most of the system's performance is determined by the translation to architecture, both software and hardware, with potentially many possible implementation tradeoffs options like complexity vs. throughput. Once we have an implementation on a platform, changes are tedious, time consuming and error prone. Our requirement on system modeling and verification, especially in terms of performance analysis, thus is to include architectural aspects in a functional model.

A language which is capable of modeling structure and behavior, is extensible (profiles), well supported and easy enough to use is Unified Modeling Language (UML) 2.0 [13]. Various profiles for UML exist (e. g. for Scheduling, Time, and Synchronization) and even a profile for platform independent executable UML models has been defined. Since many UML tools support the XML Metadata Interchange (XMI) exchange standard (which is Extensible Markup Language (XML)), processing of models generated from different tools is possible. It has been proposed [14] to use UML and executable UML for System on Chip hardware-software-co-design and early behavioral modeling.

5.2. Performance Analysis

Our reconfiguration methodology is based on the unification of the replacement of a processing functionality within a processing flow (e. g. a chain with streaming processing). The two considered main aspects in our work were the parameterization of radio modules and the partial reconfiguration of an FPGA area with a partial bit stream. Analog to the well known task manager/scheduler in operating systems, we define states and a flow in such a system reconfiguration process, see Figure 8.

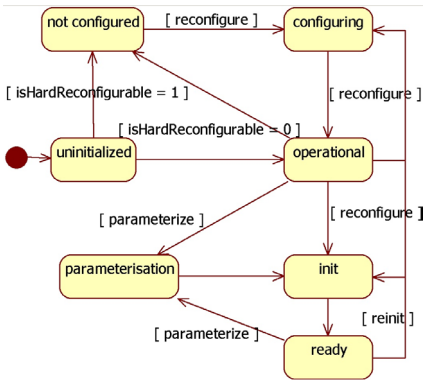

Figure 8: Configuration States of a Reconfigurable Module

Also, for modeling the interface of the processing and eventually the processing itself activity diagrams can be used. Figure 9 shows the control flow model of an input interface which is able to exhibit the behavior of several interface varieties, e. g. single token input, streaming input and even double buffered/multiple data input.

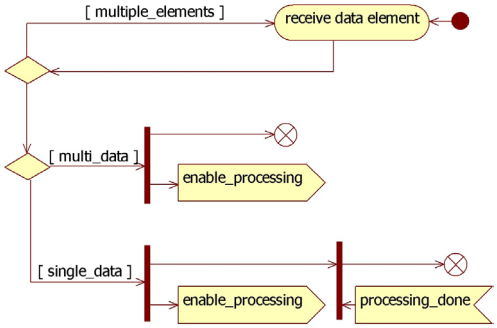

Figure 9: Activity Model of a Flexible Input Interface

A similar model can be used for the output of the function. Processing is triggered as soon as an appropriate data element was received. Note that buffers have to be modeled separately as processing elements. Annotating these models with time aspects and constraints, e. g. processing time, interface data rate, reconfiguration, and parameterization and initialization time allows for static and dynamic performance analysis, either by using tools for converting UML models to executable models or by transforming the models (semi-)manually to e. g. SystemC.

We have to explore and analyze the processing flow including dynamic reconfiguration by partial reconfiguration or parameterization, and evaluate different architectural realizations of modules. The processing time of a module can very often be estimated quite well, even for different module architectures, or the designs already exist (in different variations e. g. IP Cores) and their performance numbers are known. That leaves us with the reconfiguration or parameterization times and the respective initialization times as unknown factors. Initialization in the case of partial reconfiguration is an important factor for a safety-aware design. Ensuring that the module has been deployed correctly and is ready for processing is a vital part of the PR-concept. Many different approaches exist to this, from the processing of test data an its comparison to the known correct result to simply employing a small shift register with a known sequence that has to be sampled correctly after module loading. When setting parameters of a parameterizeable module, it's possible that the module has to do some calculations, e. g. of matrices, which may take several cycles. Note that it's possible that a PR module also has to be parameterized before it becomes functional.

For analysis we assume the following parameters: the FPGAs partial reconfiguration bus is 32 bit wide and runs at 100 MHz, giving 3.2 Gbps reconfiguration rate.

In order to estimate the partial bit stream size for an FPGA we first have to estimate the resources needed for the intended functionality. A major role plays the required number of lookup tables, flip-flops but most of all DSP48E elements and BRAMs. Usually, these numbers can be estimated quite well even for different implementation solutions. Once those estimates exist, the actual partial bit stream length for the estimation can be obtained by using the PR-toolset, which allows for defining an area containing the estimated resources and returns the matching partial bit stream length. A rough estimate is also possible by taking into account the FPGAs configuration architecture, an approach which we will omit here.

Our proposed parameterization bus is an 8 bit wide bus running at 100 MHz for minimum resource usage. This results in a data rate of 800 Mbps. It is sensible to have one bus for each functional complex which has to be parameterized, but of course it could also be used for all chains and modules commonly. Overhead is introduced on the bus for addressing and verification flows. For now we assume that we need 2 bytes per transfer for addressing, one byte length field and a "turnaround" byte that flows from the parameterized module to the control and marks the end of the transmission. In total that gives 4 bytes + data length per transfer. The equivalent net message rate, assuming four 32 bit registers will be configured per message, then is 5 million messages per second or a message every 200 ns. In a timeframe of 0.1 ms this translates to 500 configuration messages. With these numbers introduced into our models we achieve an early system verification and performance analysis.

6. CONCLUSION

In this paper an approach for a flexible base station, especially for the flexible baseband processing was presented. The introduced architecture consists mainly of three elements. The combination of these elements paired with additional approaches inside these elements offers the possibility for a dynamic load balancing between radio standards, for the flexible assembly of radio functional modules and the simple extension of radio functional modules and thus the radio standards.

Furthermore a systematic approach to modeling and performance analysis of a flexibly reconfigurable baseband processing was derived. Possible modeling solutions have been evaluated and an approach to modeling of partial reconfiguration, parameterization and processing flow, including timing aspects and using UML, was proposed. Timing aspects for reconfiguration were discussed and realistic parameters have been given.

7. ACKNOWLEDGMENT

This work was partly performed in the MxMobile project, which has received research funding from the Federal Ministry of Education and Research, respectively. This paper reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein. The contributions of colleagues from the project consortia are hereby acknowledged.

8. REFERENCES

[1] T. Loewel, F. Noack, C. Lange, V. Mérat, A. Sanchez, P. Magdalinos, D. Makris, C. Tsilopoulos, N. Koutsouris, J. Pérez-Romero, "E3 White Paper: FBS and SDR platforms for integration in the E3 prototyping environment," May 2009.

[2] P. Magdalinos, S. Panagiotis, N. Koutsouris, P. Demestichas, A. Saatsakis, D. Petromanolakis, J. Pérez-Romero, O. Sallent, V. Mérat, T. Loewel, F. Noack, C. Lange, J. Gebert, A. Sanchez, Z. Feng, Q. Zhang, B. Mouhouche, "E3 Deliverable D6.2: First version of E3 Prototyping environment; Scenarios, interfaces for integration of cognitive systems," December 2008.

[3] "Component Document Type Definitions Specification, v1.0, OMG Available Specification," March 2007; http://www.omg.org/docs/formal/07-03-03.pdf

[4] "Component Framework Specification, OMG Available Specification," March 2007; http://www.omg.org/docs/formal/07-03-04.pdf

[5] XILINX, "OS and Libraries Document Collection," June 2009, http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/oslib_rm.pdf

[6] M. Goosman, N. Doraiaj, E. Shiflet, "How to take advantage of partial reconfiguration in FPGA designs," http://www.pldesignline.com/showArticle.jhtml;jsessionid=Y2RD52UELAFRWQSNDLPCKH0CJUNN2JVN?articleID=179100555&queryText=partial+reconfiguration

[7] E. Shiflet, L. Hansen, "Using FPGA partial reconfiguration capability to mitigate design variability," http://www.pldesignline.com/showArticle.jhtml;jsessionid=Y2RD52UELAFRWQSNDLPCKH0CJUNN2JVN?articleID=184428547&queryText=partial+reconfiguration

[8] Rudolf Usselmann, "OpenCores SoC Bus Review," Rev. 1.0, January 2001.

[9] D. Densmore, A. Sangiovanni-Vincentelli, R. Passerone, "A Platform-Based Taxonomy for ESL Design," *IEEE Design & Test of Computers*, pp. 359-374, Oct. 2006.

[10] A. Sangiovanni-Vincentelli, "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design," *Proceedings of the IEEE*, Vol. 95, No. 3, pp. 467-506, March 2007.

[11] The Ptolemy Project, http://ptolemy.eecs.berkeley.edu/

[12] http://www.systemverilog.org/

[13] http://www.uml.org/

[14] S.J. Mellor, J.R. Wolfe, C. McCausland, "Why systems-on-chip needs more UML like a hole in the head," *Proceedings of the Design, Automation and Test in Europe*, Vol. 2, pp. 834 - 835, 2005.