Design Approach for Cognitive Radio on Heterogeneous Platform
Oussama LAZRAK, Xiguang WU, Christophe MOY
SUPELEC/IETR
Cesson-Sévigné, France
Email: {firstname.name}@supelec.fr

Abstract-- We propose a MDA design approach for Cognitive Radio, that enables to specify and design the management architecture necessary to support the real-time adaptation and reconfiguration of the radio processing, whatever the implementation. In particular, we proposed HDCRAM to specify how to include sensing and decision making (including learning) features into CR equipments.

Index terms—Model Driven Architecture, Cognitive Radio, HDCRAM, Meta-Model

I. INTRODUCTION

CR equipments are typical examples of devices requiring an Hw/Sw co-design approach in an embedded real-time context [1]. CR implementation indeed advantageously benefits from SDR flexibility which relies on the mix of flexible Hw and Sw technologies [2]. The design, modeling and simulation of such complex dynamic embedded systems, regardless of the method used needs to go through a number of phases. The first phase of Hw/Sw co-design focuses on the definition and identification of the system. This step is most often done by designers, defining the specifications of the system and present the tender specifications to the developers. The next step is the system implementation. When a traditional development approach is used, developers implement the different functionalities, for example by coding software for an embedded system in C language and the hardware part in VHDL. This phase is the most risky because it is the largest source of error. In this paper we propose a design methodology that reduces time and errors in coding through automatic code generation.

II. HIGH LEVEL APPROACH FOR CR DESIGN

The design of cognitive radio equipment requires the involvement of multiple actors and multiple domains, electronics, software, embedded system, etc. A design approach understandable by all these actors is necessary. For this reason we adopt a high level modeling methodology that allows to reduce the complexity of the system and offers several levels of representation of the system. Abstractions of the equipment should rely on established modeling means. MDA (Model Driven Architecture) [3][4] is a standardized approach, based on UML (Unified Modeling Language), to describe a system composed of software and hardware pieces as described in Fig. 1. MDA first step consists in modeling the application regardless of the platform of execution (PIM: Platform Independent Model) [4][5]. This model may be specific to a domain or more generalist. In a MDA approach, PIM is followed by a description of the platform (PDM: Platform Description Model). Platform can be a software platform such as .NET, JAVA, or hardware like DSP, FPGA, GPP, etc. Then in order to go to the implementation step, PIM is translated to one or more Platform Specific Models (PSM).

Fig. 1. MDA Approach for cognitive radio design

III. PLATFORM INDEPENDENT MODEL

The role of PIM is to provide a structural and dynamic view of the application, always regardless of the technical design of the application [4][5].
PIM can be described with two different modeling approaches: either through a standard model or a specific domain model.

A. STANDARD MODELS

UML is a language commonly used for software engineering, this language offers thirteen different charts that could be necessary during the design cycle. Each of these diagrams targets a different aspect of the system. UML is the result of the work of Grady Booch, James Rumbaugh, and Ivar Jacobson. UML is applicable for different fields including cognitive radio [6][7].

Software Communications Architecture (SCA) [8] is proposed by the JPEO (Joint Program Executive Office) as an alternative for future military radio equipment. SCA is an open architecture for SDR design. SCA architecture does not support self-adaptive features of radio equipment as the major SCA challenge is to ensure portability and compatibility between the components of software radio equipment[9] in order to create a universal radio equipment. SCA uses UML in order to specify interfaces, components, operational scenarios, use cases and the behavior of system.

B. CR DOMAIN MODELS

Cognitive radio, as it was defined by J. Mitola [10], asserts that future radio equipments and systems should be more autonomous. This is possible through flexible technologies. While acquiring a certain degree of self-autonomy that allows dynamically modifying their functionality. Radio equipments should also be able to analyze their environment, detect certain variations, and make the right decision of adaptation. Therefore we conclude that CR equipment must be composed of [11]:

- a flexible execution platform, executing flexible signal processing operations,
- with in addition, sensors,
- decision making means (including learning).

But we claim that all these obvious features needed for self-adaptation, cannot be efficiently deployed inside equipments without a management architecture, as a director to coordinate all previous four points. In our previous work, we proposed HDCRAM as a Hierarchical and Distributed Cognitive Radio Architecture Management[12]. It is the necessary infrastructure to be added to the signal processing (for radio, decision or sensing processing) inside CR equipments in order to make them support self-adaptation capabilities. This management infrastructure, presented in Fig. 2 is mostly made of software, but could also be made of hardware pieces in a heterogeneous implementation context.

CR equipment is made of a set of software and hardware components interconnected in order to perform the expected functions, e.g. radio processing. These are called operators at the bottom of Fig. 2.

Fig. 2. HDCRAM Architecture deployment example [12]
Fig. 2. Cognitive managers analyze data and give adequate orders of reconfiguration to the (flexible) processing operators, through the reconfiguration managers. This architecture is described with a meta modeling language. Fig. 3 represents the HDCRAM meta-model described with MOF (Meta-Object Facility).

Fig. 3. HDCRAM Meta-Model

We intend to create a modeling environment specifically for cognitive radio equipments design. This environment would allow designers creating diagrams following the rules and formalism imposed by HDCRAM meta-model [13] shown in Fig. 3. To create this environment, we need tools dedicated to the implementation of a Domain Specific Language (DSL) compatible with MOF [14][15]. Several commercial tools that help making DSL are available such as MetaEdit+, Microsoft DSL, Obeo Designer or Poseidon for DSLs. Our choice is an open source tool that is available as a plugin for eclipse: GMF (Graphical Modeling Framework), which provides a framework for developing a graphical editor. The implementation of a graphical editor with GMF is explained in [16].

IV. PLATFORM DESCRIPTION MODEL

Rarely used without code generation process, a PDM describes the structure and technical functions related to an execution platform (file systems, memory, dataBase, etc.) and explains how to use them. The PDM is associated with the PIM in order to generate a PSM. Concerning CR systems, it is important to define the execution platform that runs the application itself. For the definition of platforms, we propose a diagram based on the UML deployment diagram as shown in Fig. 4.

Fig. 4. Platform Meta-model

The PDM is mainly composed of nodes and connections. Nodes correspond to a device or an execution environment. Devices are a physical systems or hardware components. Regarding the execution environment, it can be a software platform or an OS. We also include the notion of artifact existing in the UML deployment diagram to present the elements outside the platform, which are essential to its execution, such as program configuration file or FPGA bitstream. This model facilitates the deployment of the application on the target platform but unfortunately it does not allow the automatic code simply because it does not contain information concerning the programing language used (C, C++, Java, VHDL, etc.). We have therefore chosen to use other tools that permit generate code.

One of the advantages for proposing a modeling environment shown in Fig. 5 in the Eclipse plug-in form is the ability to use other existing plug-ins. Eclipse has several plug-ins, and their uses are diverse, as languages compilers, linkers or text generator as OxygenXML which is a tool for documentations generating. We have opted for a tool generating text from a model. This tool, called Acceleo is an open source tool [17], compatible with our modeling environment, which is able to integrate any XMI written Model. Acceleo already provides several templates to support languages such as Java, C or C#.
It also offers the ability to integrate other platforms by creating new templates in order to generate C++, VHDL or SystemC code. Fig. 6 represents code generation project based on C++ templates. We use these templates to define the PDM, which allows to generate code from Acceleo [17]. In order to ensure the code generation, Acceleo needs a .ecore file, representing HDCRAM metamodel, as well as a HDCRAM-compilant model as a file with .hdcram extension. The .hdcram model represents the scenario that the equipment should realize through its operations.

V. MODEL TRANSFORMATION

The PSM is a formal description of the system. It is the result of integration between a PIM and a PDM. This obviously means that we can have as many PSM and PDM as desired, and that is the major advantage of the MDA approach. Indeed we are able to change the platform without changing anything in the application and even reuse an existing platform for new applications, in particular for maintenance. It is for sure easier to maintain models than maintaining lines of code.

Up to now, it remains to transform each component of the PSM to source code. For this purpose Acceleo proposes model to text (M2T)
transformation. This M2T transformation is based on templates, which enables to simplify the PDM models. Indeed, the development complexity is included in the templates, not in the models. It is imperative that the template may be reusable by the implementation of generic components. It is important to indicate that this approach could be coupled to others such as the model to model (M2M) transformation approach, which is more focused on models and based on modeling on any target platform (by including symmetric target language in the model).

A. HDCRAM DEPLOYMENT ON X86 PLATFORM

We have seen earlier that HDCRAM architecture has been modeled with a high-level language called MOF that allows to define a framework by defining entities and the relationships between them. This framework is essentially composed of classes and interfaces (abstract class in C++ language). C++ classes are directly available by the user of the framework through new object creation. The framework offers several mechanisms for communication between different units, such as message queues between cognitive managers and associated reconfiguration managers. This link is represented by the link manage in the meta-model. The transition metric between cognitive units is send_metric whereas it is send_order between reconfiguration units among different reconfiguration levels.

The developer on HDCRAM framework does not care about the messages interpretation in the framework. The developer just creates the message in the correct format and the framework manages routing and execution by the target. HDCRAM Framework provides several processing elements ready for the realization of some signal processing chain. Processing elements supporting the USRP platform and UDP interface are also available for a fast implementation on software radio USRP platforms from Ettus Research[18]. Links between blocks are made thanks to FIFO buffers. We should mention that it is still possible for the developer to add another mechanism of data exchange by adding new interconnection type.

B. HDCRAM DEPLOYMENT ON FPGA PLATFORM

In order to implement a heterogeneous Hw implementation scenarios, the reconfiguration management platform is implemented on Xilinx ML506 board, which is connected to PC by Ethernet protocol. A level 1 HDCRAM management is implemented on PC, therefore, the highest HDCRAM level on FPGA platform is level 2. The block diagram of the reconfiguration management is shown in Fig. 7. In order to make the communication among different platforms of HDCRAM easier and flexible, we choose the UDP protocol. By this approach, different platforms could be connected with each other through Ethernet only requiring their IP address. It is scalable so that we can add new devices easily.

Fig. 7. An example of multi-platform reconfiguration management functionality.

There are some important components that enable the implementation of HDCRAM on FPGA platform. The hardware UDP core works at 1 Gbits/s and supports UDP protocol and ARP protocol. Depending on the destination port of the incoming UDP packet, the data is sent to different components. But there is only one receiver in the UDP core; therefore a demultiplexer is developed to solve this limitation. Similarly, when sending data, an arbiter is used to decide what kind of data should be sent to the transmitter. A level 2 ReMU is implemented in Microblaze. The goal is to have the hardware processing elements as flexible as software and at the same time keep their performance. In other words, we need hardware processing elements that are easy to configure, e.g. hardware processing elements supporting Dynamic Partial Reconfiguration (DPR) of FPGA [19]. Therefore, we have developed a hardware operator controller, to control the reconfiguration of hardware processing elements. When it only needs
to change the general parameters, hardware operator controller sends new values of theses parameters to hardware processing elements. When it needs to change the overall functionality of the hardware processing elements, it is better to choose Dynamic Partial Reconfiguration approach. Besides, when it needs to delete the hardware processing elements, it downloads its corresponding blank partial bitstream. Internal Configuration Access Port (ICAP) and ICAP Controller are responsible for partially reconfiguring the specific portion of the FPGA. A bitstream Controller in charge of downloading and storing Partial bitstream into the SRAM. Partial reconfiguration of FPGA is out of the scope of this paper and more can be found in [19].

C. COMPLETION OF THE DESIGN FLOW

The capability to implement radio equipments on heterogeneous HW/SW platforms with intrinsic flexibility management features from a high level view is a key point for cognitive radio design. We have shown here how we go through the flow of Fig. 1, from a meta-model perpective, down to execution code for both processors and FPGAs supporting DPR allowing the maximum flexibility.

VI. CONCLUSIONS AND FUTURE WORK

Economic benefit of using MDA is driving more and more companies to adopt it. In fact, it reduces the risk of project failure, and allows the product to reach the market faster and with a less cost compared to products made with traditional approach. The high price of the tools can be justified by the enormous contribution for designers and developers. That is the reason why we opted for an open source tools solution. This environment is still in beta version and has not yet reached maturity. However, it is one of the few that allows heterogeneous design, combining both software and hardware components, which is of major importance in CR perspective. This process needs a lot of work to reach industrial tooling level. However we assert in this paper that the future of cognitive radio design will be high level-oriented in the long term.

VII. ACKNOWLEDGMENT

Authors thank French Brittany Council (Région Bretagne) for their support.

BIBLIOGRAPHY