Transaction Level Modelling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM

Grégory Gailliard, Eric Nicollet, Michel Sarlotte
Thales Communications S.A.
Colombes, France
{firstname.lastname}@fr.thalesgroup.com

François Verdier
ETIS Lab – UMR CNRS 8051
Cergy-Pontoise, France
verdier@ensea.fr

Abstract

In the scope of the US Department of Defense (DoD) Joint Tactical Radio System (JTRS) program, the portability and reconfigurability needs of Software Defined Radios (SDR) required by the Software Communications Architecture (SCA) [1] can be resolved thanks to Model Driven Architecture (MDA) and component/container paradigm to address a heterogeneous hardware and software architecture.

In this paper, we propose SystemC Transaction Level Modelling (TLM) to simulate Platform Independent Model (PIM) and Platform Specific Model (PSM) of SDRs, while keeping the component/container approach for applications portability. We show that SystemC 2.1 enables natively to simulate the waveform PIM specified in UML to obtain an executable specification, which can be reused to validate the SystemC TLM model of PSM. This latter allows radio platform virtualisation and true reuse of IPs models to validate earlier SDR waveforms and platforms.

1. Introduction to Software Defined Radio

1.1. General overview

A *Software Defined Radio* is a reconfigurable radio whose functionalities are controlled in software, which is able to run a set of waveform applications on the same radio platform, depending on the operational need. Thanks to its software definition, a SDR fosters reuse and flexibility, because evolutions can be easily performed via software updates instead of hardware replacement. The two main SDR objectives are the *portability* of waveform applications across different radio platforms and the *reconfigurability* of the hardware platform to accept several waveforms. A *waveform application* is a OSI (Open Systems Interconnection) model like layered software application implementing the waveform logical components that process the received data from the antenna to the end user and vice versa. A *radio platform* is the set of software and hardware layers, which provides the services needed by the waveform application layer through an abstraction layer API.

1.2. Military overview

The *Joint Tactical Radio System* (**JTRS**) is a US Department of Defense program aimed to create a global communication network of scalable and interoperable SDRs for US and allied terrestrial, maritime and airborne joint forces. The JTRS is built upon the *Software Communications Architecture* (**SCA**).

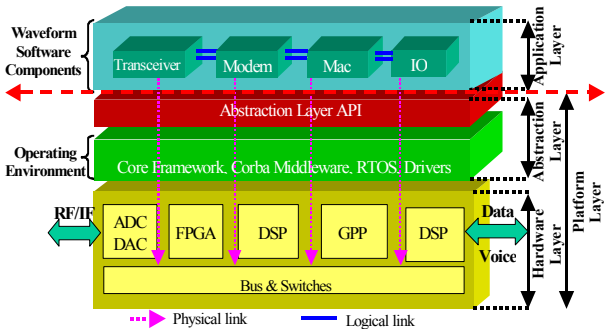

Figure 1. Heterogeneous and distributed SDR architecture.

SCA is a software architecture framework allowing some separation between waveform application and radio platform. The SCA specifies an *Operating Environment* (**OE**) in which waveform applications are executed. The OE enables to manage waveform applications and to provide radio platform services. It is composed of a *Core Framework* (**CF**), a minimum CORBA compliant middleware and a POSIX compliant *Operating System* (**OS**). The Core Framework is a set of interfaces and related behaviours used mainly to load, deploy and run waveform applications. A middleware is a software layer between the application layer and the network layer providing services and enabling transparent commu-

978-3-9810801-2-4/DATE07 © 2007 EDAA

nications between distributed applications. As depicted in the Figure 1, the SCA targets heterogeneous hardware and software architectures, which are complex to design and validate.

The official SCA 2.x versions specify a high level software architecture for GPPs (General Purpose Processor), which is not adequate for high constrained real-time resources like DSPs and FPGAs, while the unofficial SCA 3.0 tries to resolve these issues. A methodology has been proposed [2] to address the limitations of SCA 2.x and has been supported by the "Software Radio Architecture" PEA (Plan d'Etude Amont) under contract of the French DGA (Délégation Générale de l'Armement). In this project, waveforms have been mainly implemented in software and the design of radio platform consisted in assembling boards composed of GPP and DSP cores.

However, waveforms and platforms design becomes more and more complex (high data rate antenna processing, programmable RF…) and involves new methodological needs, that we try to address now. Indeed, executable specifications are needed to validate waveform consistency and compliance against specifications. Hardware/software partitioning and architecture exploration have to be studied to achieve the required performances, which have to be estimated and compared to Quality of Service (QoS) and real-time constraints. Another need is to foster reuse of waveform components to reduce time-to-market and risks.

1.3. Proposition

This paper presents an improvement of the existing SCA compliant SDR design flow and methodology, which is based on Model Driven Architecture and component/container paradigm, thanks to SystemC TLM. SystemC TLM allows a tradeoff between modelling accuracy and simulation speed in keeping only the relevant details at a given abstraction level. It allows the design and validation of waveform applications and radio platforms, earlier in the development cycle, thanks to hardware virtualisation and IP models reuse. The proposed design flow consists in simulating waveform applications using the abstraction layer API on a quick and sufficiently accurate SystemC TLM model of the hardware platform, whose IP models can be easily reused. After early waveform and platform validation, hardware and software developments can thus begin in parallel.

This paper is organised as follows: in Section 2 we present the existing design flow and methodology for SCA compliant SDR. Then we show in Section 3 the interest of SystemC TLM in this methodology. Section 4 presents how SystemC TLM models of PIM and PSM can be performed, while keeping the component/ container paradigm. Finally, our future work is presented as a conclusion to this paper.

2. Existing design flow and methodology for SDR

The existing methodology is based on two concepts: *Model Driven Architecture* (**MDA**) and component/ container paradigm.

MDA is a methodology standardised by the OMG which advocates the separation between system functionalities specification defined in a *Platform Independent Model* (**PIM**) from their implementation specification on a target platform captured by a *Platform Specific Model* (**PSM**). The PIM can be expressed in UML with domain specific profiles, while the associated PSMs are described in a target programming language (C/C++/Java/VHDL…). MDA enables to raise the level of abstraction, to increase reusability and foster portability. The PIM/PSM approach is useful in a SCA context because of the amalgam between logical (interfaces and behaviours) and implementation choices (type of middleware, POSIX profile…).

The component/container paradigm enables a clear separation of concerns between behavioural and technical properties. The component is a reusable and composable entity implementing the business logic, which provides and requires services through defined interfaces. The container makes the adaptation between component and execution infrastructure interfaces. Whatever waveform or platform changes, container guaranties the portability and reconfigurability of SDR.

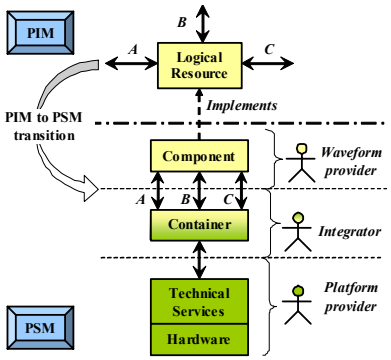

Figure 2. Methodology inspired from MDA and component/container paradigm.

The methodology illustrated in the Figure 2 consists in defining in the PIM the logical interfaces A, B and C between abstract processing resources (waveform management components, waveform resources, radio services). Then the PIM-to-PSM transition is performed. A component implements the useful treatments specified at logical level and communicate with its environment thanks to a container through the interfaces A, B and C. The container implements the transition between logical interfaces semantics and available technical services. These latter provide the software and hardware services

used by the implementation of the logical semantics.

This approach is compliant with SCA for GPPs and has been successfully applied on DSPs. It seems to be promising for FPGAs.

The design flow based on this methodology starts at PIM level with waveform specifications. The logical interfaces and real-time constraints between the logical waveform components themselves and with the radio sub-system are modelled in UML thanks to use case, class, statechart and scenario graphical diagrams. The breakdown into waveform logical components and its granularity are performed according to system requirements (characteristics of usable GPPs, DSPs, FPGAs) and business experience. Then subsequent hardware and software design flow can begin. The waveform application can be lately validated on an SCA compliant workstation. The final validation can only be done once the hardware platform is available.

Thanks to SystemC TLM, we propose now a PIM/ PSM design flow to design and validate earlier waveform applications on virtual SDR platforms.

3. Proposed simulation methodology and design flow for SDR

MDA and component/container paradigm approach can be simulated thanks to SystemC TLM as illustrated in the Figure 3. The logical resource business services are implemented in a C/C++ model. This independent and reusable model is encapsulated in a SystemC TLM container, which abstracts it from its virtual execution environment. For a software component, the container could use an API, while for a hardware component, it could contain a register bank (see Section 4). Developers can validate that the SystemC TLM virtual module provides the required behaviour and perform a partitioning based on performances estimation. The model can be either refined until RTL for a hardware component or extracted to be optimized and cross-compiled for a software component. If the partitioning was already performed, the C/C++ model can be used without a SystemC TLM encapsulation directly on an Instruction Set Simulator (**ISS**) if available, otherwise the SystemC kernel could be used. This SystemC TLM based methodology allows to simulate the heterogeneous architecture of SDR platforms.

Thus, we introduce SystemC TLM for the simulation and validation of PIM and PSM. In the SDR context, an important need is to be able to simulate the real-time constraints of a radio system, from its specifications to its final validation. SystemC appears, as far we know, as the only language addressing this kind of needs.

3.1 Proposed simulation methodology

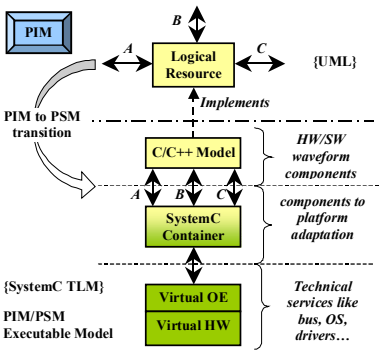

Figure 3. SystemC TLM model of PIMs/PSMs.

At PIM side, SystemC at functional level can be used to model the logical interfaces and simulate the real-time constraints between the waveform logical components themselves and with the radio sub-system. The resulting SystemC TLM modules could include a behavioural model or acts as dummy black box. This model can be used as a useful and undeniedable executable specification between hardware and software teams that anticipates the PSM and accelerates the development cycle.

The concepts of required and provided interfaces specified in the PIM can be natively found in SystemC 2.1 and used to simulate a PIM. Indeed, a *sc_port* specifies that an *sc_interface* is **required** by a *sc_module*. A *sc_export* indicates that an *sc_interface* is **provided** by another one, which implements it. The *sc_port* can be bound directly to a *sc_export*. The first *sc_module* calls the second one *sc_interface* method through the *sc_port*. Event and time driven systems can be modelled with *sc_event* and *wait* statements, while UML synchronous and asynchronous calls can be respectively modelled with SystemC TLM blocking and non-blocking calls [3].

At PSM side, SystemC at cycle level enables performance estimation (profiling) of waveform software components executed on a virtual radio platform composed of ISS for GPP and DSP and Bus Functional Models (**BFM**) if available. Thus, real-time constraints simulated in the PIM model can be reused to validate the PSM model. SystemC TLM allows to simulate and to validate only critical parts instead of the entire radio system model. A concrete example will be seen in the last section to illustrate the modelling of PIM and PSM in SystemC TLM.

3.2. Proposed design flow

Based on these considerations, we propose a SystemC TLM based design flow, whose approach is depicted in the Figure 4, in order to address the SDR needs. It starts with the capture of waveform and platform requirements

in UML PIMs. Both PIMs are associated thanks to an abstract interface (see Section 4.1), which defines the Abstraction Layer (AL) API. The PIMs are then translated in SystemC TLM containers, where a functional model is encapsulated to validate algorithms choice, by simulating the behaviour of the system and measuring its performances (e.g. in terms of Bit Error Rate for a modem application). This executable specification can be reused for example to validate waveform compliance to specifications, particularly real-time constraints.

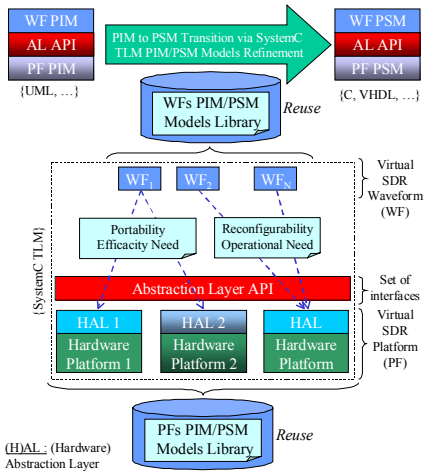

Figure 4. Proposed approach.

During hardware/software partitioning, several architectures and partitioning choices can be evaluated for the system regarding for example the interconnect strategy and the memory dimensioning. Associated tools perform for instance profiling or bottlenecks check. This partitioning enables to ease the definition of waveform breakdown/granularity, and thus reusable/reconfigurable components.

In architecture exploration, a key point is the feasibility study. If the requirements cannot be met, a feedback is given to the system engineers in order to review their partitioning or eventually their algorithms. After several iterations, this step results in a system architecture definition, which provides a breakdown of the application with associated hardware targets and the interfaces of each component.

Then hardware modelling and software developments can start in parallel. The system can be refined progressively to platform specific models in SystemC TLM, for both hardware and software modules, keeping the same testbench to functionally verify the successive translations. This refinement can be realized through timing annotations or use of another bus/topology and results in a better design. Moreover, these models can be reused earlier in steps of future developments. In software, application code is developed, eventually for managing the partial reconfigurability (bitstream download, application deployment, configuration...). However, in the scope of this paper, we only focus on static reconfiguration, as required by SCA.

Co-simulations can be performed to verify the behaviour of the application in a specific mode. Then the hardware workflow can start (synthesis, place and route) and finally the platform integration on the real target is performed. A validation shall then be led to check that all system requirements are met.

4. Experimental results

To validate this new system engineering flow, we present two test cases corresponding to the simulation of a PIM and a PSM in SystemC TLM. The following section shows how SystemC TLM can be easily used to model a UML PIM in SystemC TLM.

4.1. PIM model in SystemC TLM

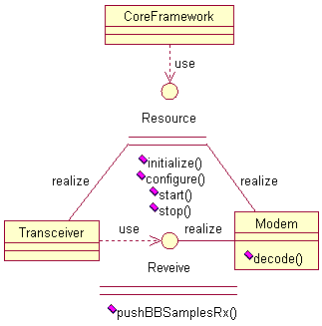

Figure 5. Waveform PIM class diagram.

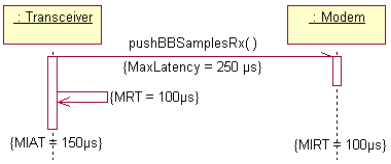

Figure 6. Waveform PIM scenario diagram.

An example of a partial waveform PIM in UML is depicted in Figure 5 and Figure 6. In the PIM class diagram given by the Figure 5, a circle represents an abstract interface, which is **required** (*use* dashed arrow) by a component and **provided** (*realize* association) by another one. The real-time constraints are captured in a UML sequence diagram given by the Figure 6. According to this PIM, the Transceiver Resource shall asynchronously call (truncated arrow) the Modem Resource method with a maximum latency of 250µs in order to push the received baseband samples. This call shall return with a Maximum Return Time (MRT) of 100µs. Moreover, the Minimum Inter-Arrival Time (MIAT) is 150µs between two Transceiver calls, and the Minimum

Inter-Return Time (MIRT) is 100µs between two Modem returns.

In the PIM, the Transceiver component requires the Receive interface implemented by the Modem component. In the simplified SystemC TLM model of the PIM depicted in the Figure 7 this interface (`line 0`) is required by the Transceiver module port (`1`) and is implemented inside the Modem module (`5`), which is called by the Transceiver port through a channel (`4`). This channel models the media real-time constraints (throughput, latency (`4`)). Modules real-time constraints MIAT and MIRT can be contained in the SystemC TLM container itself (*Explicit Timing Annotation*) or in a transactor between the module and the channel, which can be better to separate timing from behaviour (*Implicit Timing Annotation*) [4]. A transactor is a hierarchical channel, which acts as a TLM protocol and interface converter (see Section 4.2). Traffic generators can be used to transfer samples according to various probability laws (uniform, Poisson…) to observe the behaviour of the communication chain following operational conditions (link overload, Signal-to-Noise Ratio decrease…). The PIM to SystemC TLM transition can be automated with tools like *modtransf* [5]. After validation of real-time constraints, the SystemC TLM models of container and component PIM can be refined to their PSM models.

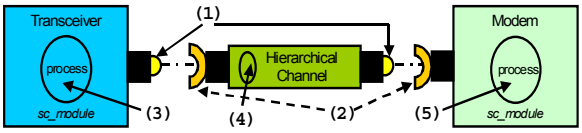

Figure 7. Simplified SystemC TLM model of PIM.

```
(0) struct Receive : virtual sc_interface{
      virtual void pushBBSamplesRx() = 0 ;
    };
(1) sc_port<Receive> pM;
(2) sc_export<Receive> pT;
(3) pM->pushBBSamplesRx();
(4) struct Channel : Receive, sc_module {
      void transfer() {
        wait(latency);
        pM->pushBBSamplesRx();
      }
    };
(5) struct Modem : Receive, sc_module {
      void pushBBSamplesRx() {
        // C/C++ model functions calls
      }
    };
```

4.2. PSM model in SystemC TLM

This section presents the IP and interconnect refinement of the SystemC TLM model of a PSM. This PSM model is inspired from a digital AM receiver MPSoC called DiMITRI [6].

We present the modelling of a previously validated RTL IP and its simulation on a virtual platform. The business code in C of a Viterbi decoder has been encapsulated in a SystemC TLM container with a register/bit accurate hardware interface. The resulting SystemC TLM IP has been reused without any modification during the refinement and exploration of the virtual platform depicted in the Figure 8. This first virtual platform is composed of an ARM968-ES, an OCP TL2 [7] bus, two OCP memories (ROM and RAM) and an OCP TL2-to-PV transactor connected to the Viterbi model.

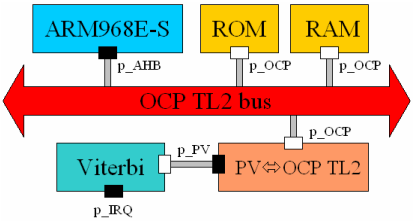

Figure 8. First platform modelling.

4.2.1. IP refinement. Starting from a functional model in C of the Viterbi decoder IP, we developed a SystemC TLM IP at *Programmer View* (**PV**) level. At this abstraction level, the model is untimed, memory map / register accurate and can support interrupt handling.

We model the Viterbi registers with the same bit accuracy as in the VHDL code and Viterbi specifications. A register bank and its registers are declared and bound in the Viterbi module constructor. Bitfields allow to select a precise set of bits inside a register. At each register or bitfield, a read or write callback function is associated, which is triggered when a read or write access is performed. The register bank is bound to a PV target port, which uses the OSCI (Open SystemC Initiative) TLM API [3].

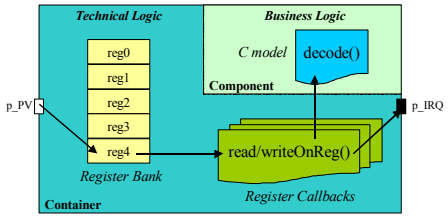

Figure 9. Modelling methodology.

As shown in the Figure 9, the C model implements the business logic of the Viterbi decoder and its services are called by the container through register and bitfield callback functions. These callbacks functions implement the technical logic (storage and synchronization). The C code file is compiled with SystemC files keeping the model intact that constitutes a reusable golden source. Thus there is a clean separation of concerns between business and technical logic. The resulting SystemC TLM

model is untimed and event-driven.

The Viterbi model was validated with the testbench already used to verify the hardware IP. The ISS executes the testbench code and accesses to the Viterbi registers thanks to the original driver in C.

The modelling effort to follow the component/ container paradigm from a C model is not really important, but it enables portability and reuse of business logic.

4.2.2. Interconnect refinement. The refinement and exploration of the first virtual platform have been performed, while keeping the same SystemC TLM IP as can be seen in the Figure 10. The refinement consisted in replacing the OCP TL2 bus model by an AHB bus model and the exploration in replacing the ARM968-ES core by an ARM926-EJS core. Both steps enable us to be more specific to the existing platform. Thanks to the use of an AHB-to-PV transactor and cross-compilation, a true reuse of the Viterbi SystemC TLM IP has been performed because no modification has been necessary. Refinement from one platform to another is thus highly simplified. The same SystemC TLM IP has been used on another virtual platform at a different level of abstraction.

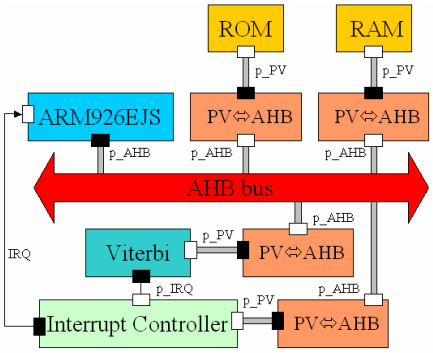

Figure 10. Second platform modelling.

Moreover, a real software application has been executed on this second virtual platform. We validate successfully an application of iterative decoding executed on the real SoC, which uses the Viterbi decoder many consecutive times.

Unlike the OCP-based platform, which used polling, the second platform contains also an interrupt controller. The Viterbi decoder uses it to inform the ARM processor core that decoded samples are available and can be read from registers. It is another step in the accuracy of the virtual platform.

Thus the contract at PV level - into which waveform software components developers could enter - has been fulfilled: software validation, interrupt handling and modelling accuracy have been achieved. Now this SystemC TLM IP/component can be reused to model more quickly and easily other Viterbi-based virtual SDR platforms.

At the moment, this test case is not sufficiently complex to achieve significant profiling results, but this issue will be addressed soon with an sizeable extended platform.

5. Conclusion & future work

In this paper, we have proposed the use of SystemC TLM to simulate PIM and PSM for design and validation of both SCA compliant SDR applications and platforms. The PIM in UML can be translated into an executable specification enabling the validation of waveform real-time constraints, while the SystemC TLM PSM model enables the exploration and refinement of virtual SDR platform, whose the IPs models can be easily reused.

Although this approach targets military SDR, it can be apply on other domains of applications. Indeed SDR is representative of modern applications needs : reuse, real-time, heterogeneousness…

For the next steps, we will model an SCA compliant radio platform to target a multiprocessing architecture on FPGA. The OE would be executed on a virtual hardware platform composed of a PowerPC ISS, a CoreConnect BFM and waveform IP models. The Core Framework could simulate the deployment, installation and running of waveform applications using the virtual hardware through the OE. Switching to another waveform application could lead to the exploration of FPGA partial reconfiguration. After validation of waveform application on the virtual platform, we could evaluate if the software and hardware integration on the target has been seamless and complies with portability and reconfiguration requirements of SDR.

6. References

[1] Joint Program Executive Office (JPEO) JTRS, *Software Communications Architecture Specification v2.2.2*, http://jtrs.spawar.navy.mil/sca/.

[2] C. Serra, B. Sourdillat, E. Nicollet, "Waveform Portability - Return of Experience on Implementing the SCA", *SDR'05 Technical Conference*, 2005.

[3] A. Rose, S. Swan, J. Pierce, J.-M. Fernandez, *Transaction Level Modelling in SystemC*, OSCI white-paper, 2004.

[4] T. Kogel, A. Haverinen, J. Altis, *OCP TLM for Architectural Modelling*, OCP-IP white-paper, 2005.

[5] J.-L. Dekeyser, P. Marquet, S. Meftali, C. Dumoulin, P. Boulet, et S. Niar, "Why to do without Model Driven Architecture in embedded system codesign?", 1st *IEEE BENELUX/DSP Valley Signal Processing Symposium*, 2005.

[6] J. Quévremont, M. Sarlotte, B. Candaele, "Development process of a DRM digital broadcast SoC receiver platform", *Annals of Telecommunications*, 2004.

[7] Open Core Protocol (OCP)-International Partnership (IP), *A SystemC™ OCP Transaction Level Communication Channel*, v2.1.2, OCP-IP white-paper, 2006.