System Level Design with UML: a Unified Approach

S. Rouxel, G. Gogniat, J-P. Diguet,
J-L. Philippe
LESTER. CNRS FRE 2734
University Research Laboratory
France
<rouxel, gogniat, diguet, philippe>@univ-ubs.fr

C. Moy
SCEE Group, SUPELEC
Cesson-Sévigné
France
christophe.moy@rennes.supelec.fr

Abstract

This paper describes a fast prototyping tool targeting software radio applications. It is based on the Unified Modeling Language (UML) and combines a Software Defined Radio UML profile for implementing a real MDA approach with EDA tools for multi-level verifications from the type compatibility to the scheduling and memory use analysis over an heterogeneous platform. Our approach relies on performance analysis to improve architecture and application matching thanks to non-functional criteria. The main contributions of our work are the improvement of the original meta-model of the Software Radio UML profile which is currently under standardization and its integration within a unified design framework. From a high abstraction level of a software application we perform extensive verifications and analysis to validate the designer hardware architecture choice and the corresponding implementations.

1. Introduction

Complex system-on-a-chip challenge is now achievable since both required hardware resources and integration technologies correspond to reality. The telecom domain is a great example where the SoC paradigm already enables the design of multi-standard chips (e.g. GSM, IEEE 802.11, IS-95). Such an evolution promotes the Software Radio concept for the management of multiple standards [1][2]. However, the design of such systems based on heterogeneous platforms (e.g. DSP, FPGA, GPP, memory) and intensive-computation software applications (e.g. encryption, scrambling algorithm, service management) cannot anymore be addressed with traditional CAD tools. Actually higher levels of abstraction are required to cope with the design complexity and to provide the designers with an early feedback. Such co-design tools partly exist and are based on scalable hardware and software IP reuse. Some of these can already meet the design constraints, like CoWare, that uses SystemC/C++ Hardware language specifications, or Co-fluent studio, that is based on the MCSE methodology (Co-design Methodology for Electronic Systems) [3][4]. However regarding the current initiatives our approach is original in the way that we combine an SDR UML profile for implementing a real MDA approach with EDA tools for multi-level verifications from the type compatibility to the scheduling and memory use analysis over an heterogeneous platform. Furthermore we have built very precise models through the A3S profile to perform accurate performance evaluations at the first stages of the design flow. In this paper we present our unified way to fill the gap between the specification and the prototyping phases by using UML. Our work is illustrated through an UMTS transceiver case study.

Major projects related to software radio are described in UML which enables modeling systems through a graphical approach. Furthermore UML continuously evolves to consider new specific characteristics from different activity domains thanks to the development of new profiles. A profile extends the UML language for a work context, which offers scalability. It specifies all characteristics (e.g. elements for real-time application) and relations between the UML elements. It allows model-based a priori verifications. A designer relies on the profile to analyze, generate code and specify various application and architecture constraints. Moreover, dependencies, inheritance, or groupings between profiles can be performed to promote the reuse of domain specific needs. Regarding the software radio application, three profiles are of interest: UML profile for Software Radio [5], UML profile for Schedulability Performance and Time [6] and UML profile QoS and Fault tolerance [7]. Each profile brings out some specific characteristics that are useful to perform the evaluation of the system performances. Dealing with these profiles, a system can theoretically be accurately specified by integrating various constraint types (e.g. power consumption, bounded execution time).

But the standardization of these profiles is not yet complete and existing profiles partially address the

1-4244-0777-X/06/$20.00 ©2006 IEEE

parameters required for SDR prototyping. Our work proposes to improve these different profiles through the development of a new and specific one. Its purpose is to highlight standard concepts required for prototyping and to add hardware attributes that are not currently taken into account for Software Defined Radio applications. Furthermore the goal of our A3S project (System Application Architecture Adequacy) is not limited to the definition of the A3S profile but also targets its implementation within a rapid-prototyping tool to evaluate the feasibility of complex applications over heterogeneous platforms (with DSP, FPGA components). Specification of dynamic reconfiguration is also investigated since this feature will be mandatory especially for Software Radio applications.

The remainder of this paper is the following. Section 2 presents various high level system specifications. Section 3 provides a global approach of system modeling as promoted within our project. Section 4 details the A3S profile and the UML modeling by giving the set of parameters required to compute verifications and performance evaluation. Section 5 gives an example of an UMTS application modeling. Section 6 concludes the paper and gives an overview of future work.

2. Related Work

Many tools aim at modeling systems, performing verifications, simulations, validations, and synthesis. Different modeling styles with different granularities are considered, different input specification languages as C, SystemC, VHDL, are also used to validate, verify, simulate or emulate a system [8][9]. First co-design tools, like VULCAN are using simple and limited hardware architecture models, others like COSYMA are based on dedicated hardware co-processors to speed up software execution [10][11]. COWARE and PTOLEMY consider heterogeneous specification to respectively design specific application (embedded telecommunication) and co-simulate heterogeneous HW/SW system [12]. However these approaches are limited as they require the use of different tools that must be kept updated. Actually the goal is to perform both modeling and design specification of hardware platform and software application within a single tool and through a common language to be less dependent of multiple software update [13]. The SoC Environment (SCE) developed within the University of California, Irvine provides such an approach as the design specification within each stage of the design flow is defined through a SpecC code [14]. However, the use of a generic language, common to different domains, that is enough flexible to model all co-design aspects (architectural and application specification, component properties, constrains specification) will be mandatory to accelerate the design cycle and to promote the design reuse. To target such a philosophy, the most recent rapid prototyping tools integrate methodology of hardware-software co-design into the concept of MDA (Model Driven Architecture) through UML.

ZeligSoft proposes a code generator that produces SCA artifacts for Corba compliant targets. This approach is sizable regarding different aspects such as the SCA core framework [15] but no SoC meta-model is provided. In [16] they focus on the importance of the deployment design step but the analysis method is limited to IDL, type compatibility and pure software concerns. There is no analysis addressing embedded systems issues such as memory, bus, real-time, power.

The Prompt2Implementation targets an MDA for SoC design. It is based on the ISP UML profile [17] for parallelism expression at task and data-levels and on model to model engines. The main objective is to produce a simulation code (e.g. SystemC TLM) based on mapping rules. This is a very ambitious project restricted to very intensive signal processing, but the tools seem to be under development. Moreover, this approach does not address the SDR concept. The association between UML and SystemC is a promising approach, which is also explored in [18]. In this work, a UML SystemC profile is proposed and used to generate SystemC code. An object-oriented HW/SW synthesis flow based on an UML initial specification is described in [19]. The MOCCA compiler implements an MDA approach based on system, platform and deployment models. The current implementation is based on a processor/FPGA platform where SW and HW components have been implemented. This work is interesting but does not rely on SDR UML profile. In [20] the authors present a framework for software design space exploration based on performance and power estimation issued from an UML specification. The method is based on a pre-characterized platform and enables the evaluation of software implementations solutions specified by the designer. FZI is developing a framework for the communication conflict analysis in a SoC context. In this approach [21] UML and SysML are combined to specify architectures when on the other side a sequence diagram is used to specify the application. After refinement and component mappings a conflict graph is built to analyze communication scheduling. Finally, the UML2.0 profile for SoC is another initiative that reached the approval step in September 2005. No tool is currently proving the concepts that are located at a software level.

Compare to previous efforts our approach relies on our UML A3S profile that inherits from others standardized profiles and extends them. This profile improves and offers more hardware specification possibilities that are essential for software radio or other electronics systems in order to specify hardware and software architecture systems. In addition our high abstraction level specification alleviates the modeling and the validation of applications that belong to other specific application domains. Moreover, as we consider

applications as a set of IPs, components are only characterized by non-functional parameters instead of source codes (which depend on their implementation and need different tools).

3. A3S Design Approach

A3S approach proposes a UML software framework where the designer can rapidly and easily prototype his system and check if constraints are met in terms of timing, memory, area, and power consumption [22]. The main steps of our design flow for virtual prototyping are depicted in Figure 1.

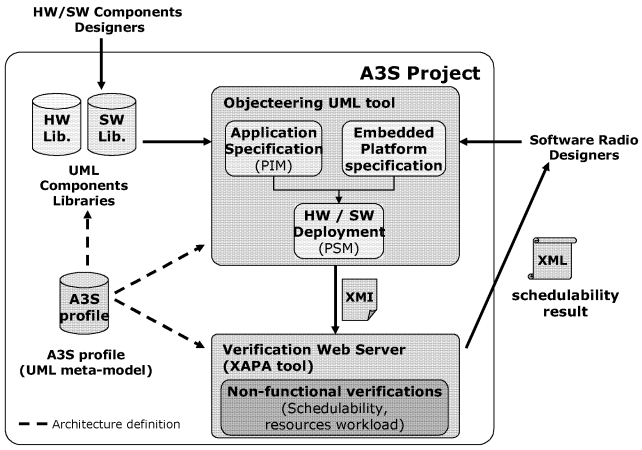

Figure 1. A3S design flow

One important question to be raised is to know who is the final user of the tool. Two kinds of actors can take benefit of the A3S framework:

- The component designers. They are concerned by the components definition (HW and SW) which takes place within the modelling and specification tool. They will create the software and hardware components libraries (IPs). For this kind of actors, some ergonomic wizards included in the design tool will help them to input the correct values when creating new hardware and software components.
- The software radio designers. Their goal is to design and tune the software radio platform and waveform. For this kind of actors, others ergonomics wizards will help them to instantiate and place the A3S software radio components with conformity to the A3S profile. They will be able to perform manually several HW/SW deployments in order to reach an optimized solution.

The verifications performed by the tools are related to the A3S profile (see section 4). They allow the designer to see in a simple glance the errors within his design during each step of the A3S design flow. It is always possible, in spite of the existence of the IHM, that the designer gives values that are not coherent. Thus, extensive verifications enable a faster and safer design flow.

Some errors can be related to the architecture of a platform, or the connection between the software application and the embedded platform. The designer can perform the verifications for the main points of a design (libraries of hardware components and software components, hardware platform and software application) or for a whole project.

Each step of the design flow is now detailed in the next sections.

3.1. Application specification (1st step)

With the MDA approach, software application and hardware architecture can be specified independently, so 1st step and 2nd step (see Figure 1) can be exchanged. To manage complexity, an application is split into several functions that are represented by independent generic software (SW) components. It corresponds to a PIM (Platform Independent Model) since each function can be potentially mapped onto any hardware component. These SW components have specific non-functional parameters that correspond to specification constraints coming from the application or from some designer requests. An example of these parameters is the periodicity of the SW component which is independent from any implementation. More information about these parameters is detailed in section 4. At this stage of the design flow SW components can represent any function.

The application is modeled through a functional scheme based on the UML Activity Diagram which is composed of a set of action states (SW component) and transitions. Transitions correspond to dependency relations between functions and have specific parameters related to the exchanged data (e.g. number, size). For each component, the designer specifies the corresponding parameters value. An Activity Diagram has been considered since it enables the description of the dynamicity of a system. Activity Diagrams allow the modeling of the process described by activity chains with information related to transmission, connection management, and activity responsibility description.

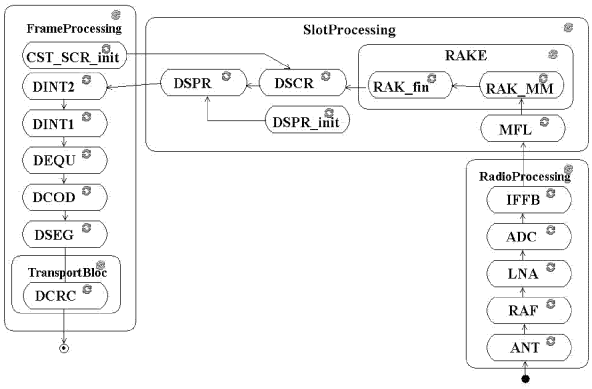

Figure 2. UMTS-FDD Receiver Activity Diagram

An activity diagram example for an UMTS-FDD receiver is given Figure 2. This diagram also addresses the links between the different SW components to specify the system radio functionality. The black dot represents the input of the application which takes place at the propagation channel side. Each arrow corresponds to an edge (transition) and represents a data-flow dependency. The UMTS-FDD receiver is mainly a data-flow application with periodic and iterative functions (FrameProcessing, SlotProcessing, RadioProcessing, TransportBloc). The black dot in the circle is the output of the application; it corresponds to the exchanged data between the physical layer and the higher layers of the OSI model.

Through this model the designer can easily replace, add, move/remove a SW component, or modify some parameters to enhance the algorithm and thus test various configurations. By this way, he can analyze the impact of different reconfigurations, which is of major importance in a software radio context. Once the application model is completed, some coherency constraints verifications are performed. Among them, the tool verifies that all connections between SW components have been correctly done, through compatible data format and that all required parameters have been settled. These verifications have been implemented within the Objecteering case tool [23].

3.2. Embedded platform specification (2nd step)

This step deals with the platform specification. Each hardware component is described in a hardware library (DSP, FPGA, GPP, memory, interconnect and ASIC) corresponding to an UML package. Each component has specific attributes defined through its stereotypes (this point is developed in section 4). The designer builds his platform by assembling hardware components instantiation (in UML sense) through a UML deployment diagram. Many hardware platforms can be realized, especially heterogeneous platforms. This kind of architecture is essential for telecommunication applications like software radio that need flexibility (offered by FPGA and DSP components for hardware and software reconfiguration) and important computation resources (multi-processor). A Deployment Diagram has been considered since it enables the description of the physical connections that exist between the hardware devices located on the platform.

3.3. Hardware/Software deployment (3rd step)

After the software application and hardware platform modeling steps completed, the designer chooses which dedicated SW component is implemented onto which hardware component. For each SW component, the designer selects the corresponding function in the software component library as a SW component corresponds to a processing element that is not dedicated to a specific target (PIM-Platform Independent Model). Thus, the function represents an implementation of the SW component on a processor (e.g. DSP, GPP, µC), a FPGA or an ASIC. The target hardware component selected to implement the SW component is obtained by defining an instance of a hardware component within the hardware platform in the UML deployment diagram.

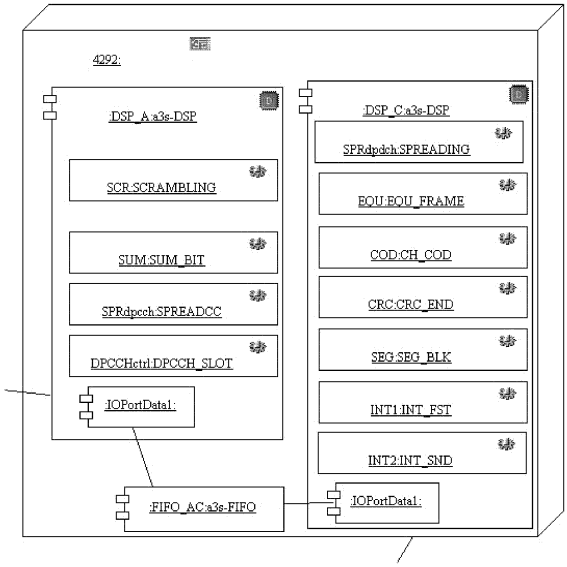

Figure 3. Deployment diagram after mapping

A broad range of implementation solutions can then be tested for a specific platform (PSM – Platform Specific Model) due to all possible combinations. The example in Figure 3 depicts an hardware platform composed of two DSPs (DSP_A, DSP_C) on which different software components are implemented (e.g. scrambling function is implemented on DSP_A). Thus the deployment diagram is refined by software component instantiation implemented into a hardware component instantiation. This partitioning is performed through links between the software components from the UML activity diagram and the hardware components from the UML deployment diagram. For example, the DSP_A that is connected to DSP_C via FIFO_AC handles four functions (SCR, SUM, SPRdpcch, DPCCHctrl).

During the application specification step, non-functional verifications are automatically performed thanks to the use of the A3S meta-model.

3.4. Analysis results (4th step)

Results are provided through a schematic view defined in a UML sequence diagram which is close to a Gantt diagram. The results emphasize the performances achieved by a heterogeneous platform with multiprocessor resources to perform the application. For example, execution time, resources use rate, system evolution (scheduling), allocated memory resources are exhibited. Scheduling information is very important as if

the system cannot be scheduled or if it does not reach the required timing constraints, the solution is not relevant.

If the solution built does not satisfy the constraints, it is easy to modify the implementation choices just by modifying the links between software and hardware components in the UML activity diagram without modifying the diagram. As several applications and platforms can be specified it enables testing an application on different platforms and with different implementations for a same platform. It also promotes testing different configurations and re-configurations of the system. The design space exploration is performed manually and iteratively in the current methodology. It is also possible to modify some hardware characteristics by changing hardware component parameters values. Moreover, this CAD tool returns results that help designer to make modifications according to identified critical functions.

As the previous steps, coherency verifications are performed to check the solution. After this step, the system is completely specified and a functional analysis can be launched.

3.5. Ergonomics – wizard – GUI

To provide an intuitive verification tool, the checking preserves the hierarchy of the elements within a project (components, application/platform, deployment) and indicates through a message the possible errors or warnings. Thus, it is easy for the designer to analyze where the problem comes from and to further help him the tool points out the element affected by the error when clicking on an error message.

Figure 4 shows the consistency report as it is provided to the designer. As we can see an error is highlighted which enable the designer to correct his specification before going through the non-functional verification tool that analyzes the schedulability of the system.

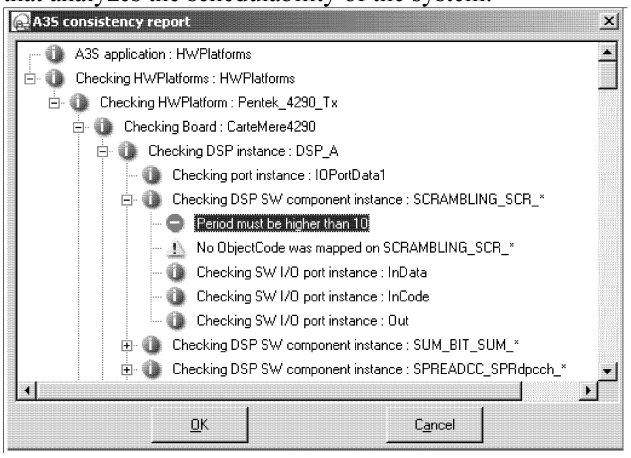

Figure 4. Checking procedure of a SW application after deployment – first part (PIM) – second part (PSM) for each HW platform

4. A3S profile and UML Modeling

4.1. A3S Profile

One of the goals of A3S is to apply non-functional description elements on PIM and on PSM and to confront them during the verification phase.

Currently, major software radio projects are described by UML class diagrams for architecture and sequence diagrams for chaining behavior. The UML profile for software radio proposes a set of PIM and PSM stereotypes for allowing the description of platform independent or dependent architectures of radio systems on a functional view side which is relatively close to the SCA specification. To allow the fine definition of signal processing behaviors, it can be extended by the additions of stereotypes coming from QoS profile and Real Time Scheduling and Performances profile on each of the components addressed by the Software Radio profile, describing their quality of service behavior offered or desired. In order to address the DSP/FPGA specific domain of study, it can be possible to extend the software radio profile by introducing specific DSP and FPGA stereotypes representing DSP and FPGA components derived from the processor stereotype of the software radio profile. These new stereotypes will be then tagged with stereotype extracted from the QoS profile to describe the quality of service behavior of the DSP and the FPGA.

Using standardized profiles and the components they introduce, we will allow the designer to reuse some legacy components by wrapping them into a standard component exhibiting the compliant interfaces. It will make the designer possible to focus on the architecture or system composition, instead of being compelled to discover and/or create new components from scratch. This method is already used for a long time by software developers to de-couple from third-party provided components. Such an approach is the only way to enable a smooth transition from existing methods to new one. It also makes possible the integration of non-compliant external provided component.

The A3S profile formalizes through a rigorous semantic the elements that will be used to build the software radio architecture models that are verified by the A3S tool. Our formalization enables the definition of the verification rules. These elements extend or use some elements extracted from the previously explained OMG standard profiles, as illustrated in Figure 5. This warranties the timelessness, the interchange and the reusability of the A3S models. Since the interfaces can be standardized by this way, it is then possible to work and verify any A3S model assuming that the tools have the A3S profile.

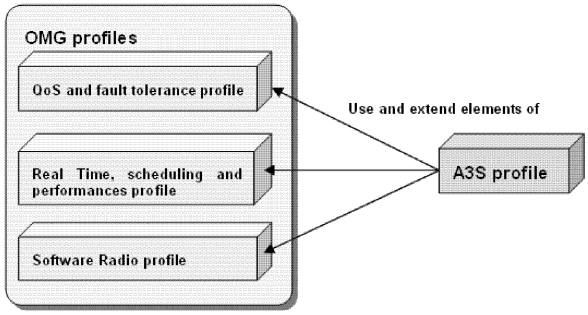

Figure 5. Relation between A3S profile and the OMG standard profiles

This A3S profile's main interest resides in the fact that all the interfaces may be standardized, and that all the elements are redefined from the basic types, warranting an automatically generation of interface specification through the IDL syntax language.

Figure 6 illustrates the hardware meta-model of the profile that defines all the stereotypes that will be used to design Software Radio platforms. It extends the OMG software radio model (on the right part of the figure), by defining new stereotypes prefixed by the "a3s-" keyword inheriting from each of the main components of the OMG software radio profile and providing some non-functional information (on the left part of the figure).

For instance, according to the OMG software radio profile, each hardware component of a software radio can be stereotyped by the CommEquipment element and that the CommEquipment are connected to each other through some CommEquipmentConnectors linked to their DigitalPort, A3S provides the same elements extended with QoS characteristics, that may range from data size and processing frequency to power consumption. Such an inheritance is generic enough to envision the future addition of new QoS characteristics to an element of the a3s profile, without disturbing all the models of the software radio platform.

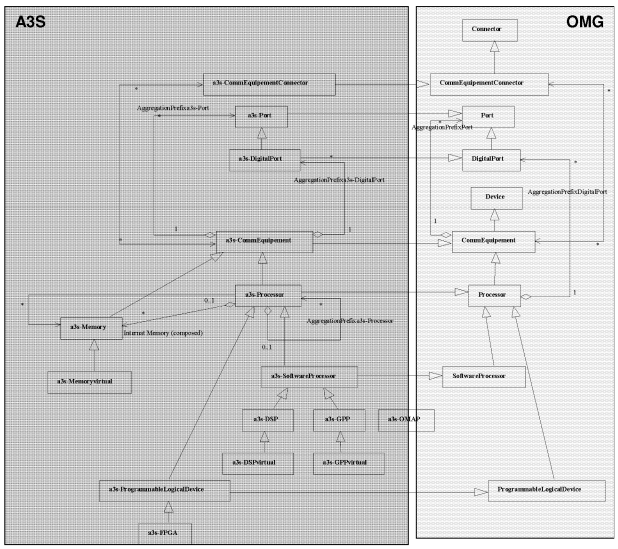

Figure 6. A3S profile's hardware meta-model

Figure 7 describes the software meta-model of the profile that defines all the stereotypes that will be used to design Software Radio waveforms.

The first step for QoS definition of software radio elements is to specify the QoS language that will be used during the modeling phase. For our purpose, it will allow the specification of a particular kind of software radio component, the fields that are relevant to quality of service and that must be filled with measured values in the PSM model. The definition of such a QoS language specific to the A3S issues is performed using the QoSCharacteristic elements of the QoS and Fault Tolerance profile. QoSCharacteritics can be extracted directly from the catalog of well known QoSCharacteristics of the QoSProfile, but can also be defined from scratch, inherited from other QoSCharacteristics or aggregating by others. The set of QoSCharacteristics obtained by this way, is then stored in a QoSCatalog dedicated to the A3S needs. At the design time, these QoSCharacteristics will be implemented into QoSValues which will be applied to PSM software radio components. Figure 7 illustrates the definition of the QoS characteristic of an FPGA, and Figure 8 illustrates how it is possible to describe the QoS offered by the platform specific FPGA-pentek-3292.

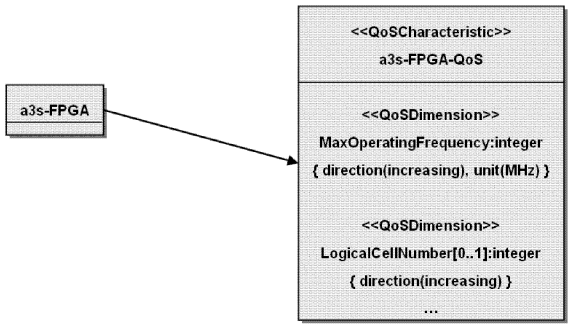

Figure 7. Definition of the QoS characteristic of a FPGA

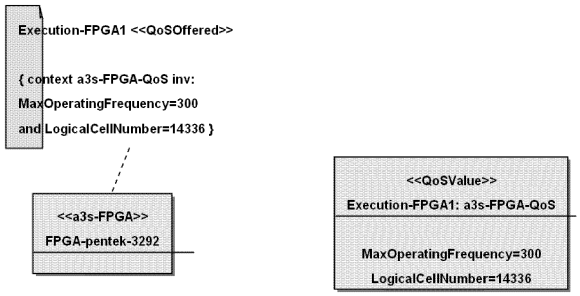

Figure 8. Definition of the QoS value that a specific FPGA may have

4.2. UML Modeling

During the application and platform specification steps, the designer provides the values of the software and the hardware component attributes to perform the

coherency verification and analysis of the system. Each component (software and hardware) can be characterized in three parts as described in Figure 9. First part concerns the non-functional characteristics (attributes) of the component.

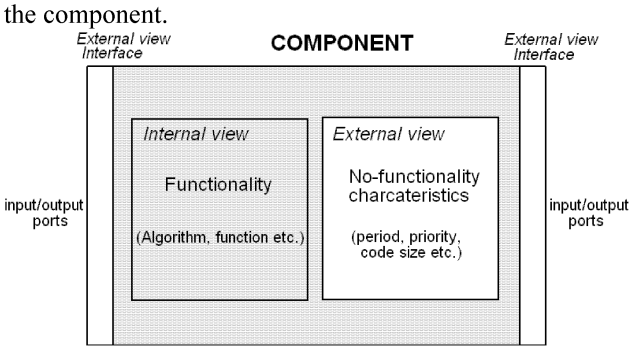

Figure 9. Component Views

For software components it represents the temporal aspects of the function (e.g. period) and the data characteristics. The second part describes its interface (its I/O port) with significant attributes relative to exchanged signal. The third part is relative to the functionality of the component. For hardware components it corresponds to the clock frequency, the type and quantity of internal/external memory. This view mainly corresponds to specification constraints. As our approach relies on IP cores use, the internal view of the component is not explicitly represented since we assume that IP cores functional behavior (C, C++, SystemC, VHDL) is validated through other means that are not in the scope of this paper. In our case attributes can be provided using the IP characteristics.

UML stereotypes permit to identify and characterize any element by assigning different parameters called "attribute". So each element of UML can be specialized by using different stereotypes that are used to define component parameters. Generic SW components which composed the UML activity diagram, HW components, ports of HW components, dedicated SW component, ports of dedicated SW components have different stereotypes, which give them specific attributes.

Basically, generic SW components which are not yet implemented have different attribute (e.g. a function is periodic or not, it has an initialization part or not) to a dedicated SW component which represents one implemented choice of one generic SW component. Each implementation choice brings some specifics constraints that are highlighted through the non-functional attributes. They deal with function periodicity, execution time, size code of IP core, priority level if a RTOS is used, and other attributes like data and code localization, and access memory type. HW components have different stereotypes, which make the difference between HW processing components (DSP, ASIC, processor), memory components (FIFO, RAM, ROM), reconfigurable components (FPGA) with their associated ports, and communication components (Bus, wire). Specific performance parameters are considered according to the hardware component (frequency, data/program memory size, port type, data width, throughput)

So all identified parameters are required to perform analysis. They are used during the scheduling analysis step (see figure 1), to compute resources use rates, to perform constraints verification and to check the coherency of the system.

4.3. SDR-related non-functional verification

Once specification and mapping have been completed and coherency verifications have been performed (i.e. no error about HW/SW connection, all attribute settled), the A3S tool generates a XML file gathering the information about the system. The file contains the diagrams (activity, deployment), the hardware/software component allocated, and the attribute values. More precisely the UML activity diagram that represents the functional application scheme of the system is encompassed in the XML file. Thanks to an XML parser this diagram is converted into a task graph. Thus, the parsing of this file enables building a General Task Graph (GTG) based on the Radha Ratan model since we consider the corresponding method to perform period derivation [24]. This method computes the period of each task within the GTG even if some are previously unknown. The GTG nodes represent tasks (functions), and the GTG oriented edges are channels from producers (tasks) to consumers (tasks). Each task can be triggered by a data or control. Each edge contains producer and consumer information corresponding to data/control to exchange between functions. For applications that use multiprocessor, the choice of function implementation can lead to additional communication tasks (in case of two tasks connected each other and implemented on different hardware devices). The period derivation step is performed to compute the timing constraints (periods) that have not been settled by the designer during the specification steps. This point is important, since this kind of computation is very error prone and can be efficiently done with our CAD tool.

The GTG obtained from the XML processing is then used with the HW architecture characteristics within our real time analysis tool RTDT [25]. This tool performs automatically complex scheduling verification and provides performance analysis results which help designer to drive his choices. Such automatic bridges and tools are essential to improve time to market and quality designs.

5. UMTS FDD Case Study

The A3S profile has been created to specify the software defined radio physical layer in the UML 2.0 meta-model. An UMTS FDD channel in uplink mode has been chosen as a first reference to determine which

software and hardware components could be included in the software and hardware components library from the A3S UML profile. This application has also been tested to validate the A3S tool. The UMTS transmitter and receiver applications have been modeled through two different activity diagrams (UMTS receiver modeled in Figure 2). The chosen hardware platform corresponds to a standard board composed of multiple DSPs connected to FPGAs via FIFO and SDRAM memories. The deployment diagram represents the Pentek board (4292) described thanks to the hardware components from the A3S hardware components library (an overview is given on Figure 3). For this case study the partitioning has been determined manually by the designer. After specifying the hardware components attributes using the Pentek board components characteristics, and the software components attributes using software IP core characteristics, the validation step and the schedulability performance analysis are performed.

Non-functional attributes are first checked to verify the system coherency. Then the A3S tool generates the GTG file (.gtg) and provides the HW architecture characteristics to perform the schedulability and the power consumption analysis. Our real time analysis tool has been first designed for mono-processor architectures with hardware accelerators; it has been modified to support schedulability analysis in the context of multi-DSP/Processor architectures.

To prototype the UMTS FDD transmitter and receiver we have considered a hardware platform composed of four TMS320 C6203 DSP running at 300Mhz. Each DSP is connected to a XILINX Virtex XC2V3000 FPGA running at 100Mhz. Each DSP is also connected to an external shared SDRAM memory. The two UMTS FDD software applications, transmitter (SW 1) and receiver (SW 2) are implemented into the hardware platform described above. SW 1 is composed of 11 functions (pulse shaping, scrambling, coding, spreading, integrating …) and SW 2 is composed of 14 functions (matched_filter, rake, descrambling, despreading, decoding …).

Different implementations are considered to verify and validate the efficiency of the A3S tool. These experiments have been iteratively performed in order to meet both architectural and application constraints. The first experience consists in implementing all the functions for SW 1 and SW 2 into the DSPs (software solution), and then in modifying the data rate frequency to see the limits of such a solution. The second experience consists in partitioning the functions implementation between DSPs (DSP_A, DSP_C) and their respective associated FPGAs (FPGA_A, FPGA_C). The critical functions within each application are implemented into the FPGAs and the remainder into the DSPs. For both experiences, the two different data rates (117 kbits/s and 950 kbits/s) are tested. The UMTS design under consideration is not a complete fully realistic UMTS system but comprise enough processing element to evaluate the tool.

For each experience, a possible scheduling was determined and the corresponding hardware components utilization rates were computed. The results for one radio frame are given in Table 1. An overall 100% means that all the processing power of all HW devices is necessary to run the application in real time. Less than 100% means that real-time is also reached. When the workload is more than 100% it means that a single HW device (DSP or FPGA) is not enough to run the application. In that case it is necessary to provide a new partitioning to be able to reach the constraints.

In the UMTS standard, a radio frame must be computed every 10 ms. In this first experience, we only consider the execution time issue. It shows that software-only solution is adequate for SW 1 as the rate is correct (<100%) for the two configurations (117kbits/950kbits). Actually this solution is not correct for the 950kbits configuration since the timing constraint is not respected as the execution time exceeds 10 ms (10.33>10). In the case of SW 2, the software-only solution cannot be realized because of the DSP overload (185%). Thus to respect both the DSP load and the timing constraint we have to define a new implementation.

The result analysis helps to identify function and/or data exchanges that affect the global system performance. Changing the implementation is straightforward with the A3S tool since it just requires to modify some links (corresponding to the critical functions) and not to rebuild the whole system. Thus only two critical functions (PSH for SW1, MFL for SW2) that were previously implemented onto the DSPs are implemented onto the FPGAs (hardware solution corresponding to the 2nd experience). The remainder functions are still implemented onto the same DSPs. The new results show that for each case (SW 1, SW 2), the DSP load was reduced (e.g. from 96% to 11% for SW 1 and from 185% to 17% for SW 2). This implementation also reduces the execution time, and the timing constraint (<10ms) is respected in each case. Thanks to the tool, the designer performs a fast analysis and is able to compare the most appropriate implementations satisfying the application and architecture constraints.

	data rate 117 kbits			data rate 950 kbits		
	DSP_A	DSP_C	time (ms)	DSP_A	DSP_C	time (ms)
tansmitter (SW 1)						
1st experience (all DSP)	96,6%	3,4%	9,99	96,6%	5,1%	10,33
2nd experience (DSP + FPGA)	11,4%	3,4%	7,96	11,4%	5,1%	8,29
receiver (SW 2)						
1st experience (all DSP)	185,5%	4,6%	19,27	185,5%	5,0%	19,33
2nd experience (DSP + FPGA)	17,1%	4,6%	9,44	17,2%	5,0%	9,49

TABLE I. Hardware component utilization rate

Figure 10 shows the Gantt diagram provided to the designer to analyze the resources workloads.

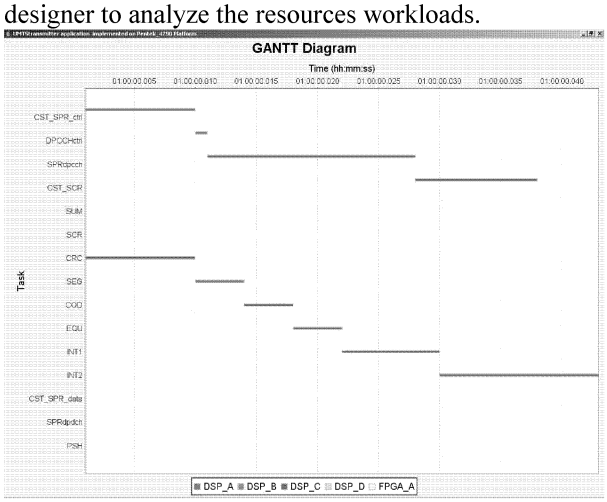

Figure 10. Gantt diagram provided to enable the analysis of the resources workloads

6. Conclusion

In this paper we have presented the UML compliant rapid prototyping A3S tool based on the UML A3S profile. The UML A3S framework provides designers with a unified, fast and easy method to specify software applications and hardware architectures. Such an approach significantly decreases the prototyping time and enhances system reuse, which is a major metric for software radio applications. It also provides the designer with an exploration tool for rapidly testing various HW/SW mappings. Furthermore, the A3S project proposes more than a design framework. It also provides a design methodology to validate complex systems with a step by step design approach from PIM to PSM. This CAD tool simplifies designer job by making automatic usually heavy tasks, like coherency verifications, period task and timing computations as well as scheduling verifications. The A3S tool is under development and some improvements are still to be done. The consumption parameters for power analysis are required to target mobile devices instead of base stations; they are already present in our analysis tool. They have been recently added to the A3S profile and some experiments are performs to compare estimations and measures. Another improvement will be the automatic generation of appropriate compiling scripts and binding edition for logical synthesis.

References

[1] J. Mitola, "The Software Radio Architecture," IEEE Communications Magazine, vol.. 33, no. 5, pp. 26-38, 1995.

[2] [online] Software Defined Radio: http://www.sdrforum.org

[3] I. Bolsen et al," Hardware/software co-design of digital telecommunication systems," Proceedings of IEEE, vol. 85, no. 3, pp. 391-418, 1997.

[4] J.P Calvez, MCSE : Spécification et conception des systèmes : une méthodologie, Masson, 1990.

[5] UML™ profile for Software Radio - OMG draft.

[6] UML™ profile for Schedulability, Performance and Time Specification – ptc/02-03-02 OMG draft.

[7] UML™ profile for QoS and Fault Tolerance Characteristics and Mechanisms- OMG revision submission.

[8] S. K. Shula et al., "High Level Modeling and Validation Methodologies for Embedded Systems : Bridging the Productivity Gap," 16th International Conference on VLSI design, pp. 9-14, 2003.

[9] S. Edwards et al., "Design of Embedded Systems : Formal Models, Validation and Synthesis," Proceedings of IEEE, Vol. 85,N°3, pp. 366-390, 1997.

[10] R. Gupta, G. De Micheli, "Hardware-Software Cosynthesis for Digital Systems," IEEE Design and Test of Computers, pp. 29-41, 1993.

[11] R. Ernst et al., "Hardware-Software Cosynthesis for Microcontrollers," IEEE Journal Design and Test of Computers, pp. 64-75, 1993.

[12] J. Davis et al., "Ptolemy II - Heterogeneous Concurrent Modeling and Design in JAVA," University of California at Berkeley, September 2000.

[13] D. Araki et al., "Rapid prototyping with HW/SW codesign tool," Proceedings. Engineering of Computer-Based Systems (ECBS), pp. 114-121, 1999.

[14] D. Gajski, "System-Level Design Methodology," ASP-DAC 2004 Pacifico Yokohama, Yokohama, Japan, January 27, 2004.

[15] [online] "Code Generation for SCA Components", white paper, http://www.zeligsoft.com/, 2005.

[16] [online] "SCA Deployment Management", White paper, http://www.zeligsoft.com/, 2005.

[17] P.Boulet, J-L.Dekeyser, C.Dumoulin and P.Marquet. "MDA for soc design, intensive signal processing experiment". FDL'03, Frankfurt, September 2003. ECSI.

[18] E.Riccobene, P. Scandurra, A. Rosti, S. Bocchio, "A SoC Design Methodology Involving a UML 2.0 Profile for SystemC", Design, Automation & Test in Eur. Conf. (DATE), 2005.

[19] B. Steinbach, Ch. Dorotska, D. Fröhlich "Hardware Synthesis of UML-Models". Workshop on UML for System-on-Chip Design (UML-SOC'05) at Design Automation Conference (DAC'05), Anaheim, USA

[20] M.Oliveira, L.Brisolara, L.Carro, F.Wagner, "Embedded SW Design Exploration Using UML-based Estimation Tools", Workshop on UML for System-on-Chip Design (UML-SOC'05) at Design Automation Conference (DAC'05), Anaheim, USA

[21] A.Viehl, O. Bringmann, W.Rosenstiel, "Performance Analysis of Sequence Diagrams for SoC Design", Workshop on UML for System-on-Chip Design (UML-

SOC'05) at Design Automation Conference (DAC'05), Anaheim, USA

[22] [online] A3S project home page: http://web.univ-ubs.fr/lester/www-lester/Projets/Codesign/A3S/English/A3Shome.htm

[23] [online] Objecteering software: http://www.objecteering.com/

[24] A. Dasdan, "Timing Analysis of Embedded Real-Time Systems," Ph.D. dissertation, University of Illinois, 1999.

[25] Tmar H., Diguet J-P., Azzedine A., Abid M., Philippe J-L., RTDT : a Static QoS Manager, RT Scheduling, HW/SW Partitioning CAD Tool, ICM, 2004