SoCKET: A HW/SW Co-Design Flow: Presentation & feedbacks from aeronautic and space application domains

Vincent LEFFTZ$^{\diamond}$, Pierre MOREAU§

$^{\diamond}$Astrium Satellites S.A.S. (Toulouse, FR), §Airbus Operations S.A.S. (Toulouse, FR)

***Abstract*—The SoCKET project (SoC[1] toolKit for critical Embedded sysTems)[2] gathered industrial and academic partners to address the issue of design methodologies for *critical* embedded systems. They worked towards the definition of a "seamless" design flow which integrates qualification and certification, from the system level to integrated circuits and to software. This paper sketches such a design flow and the associated methodologies, and details the industrial experimentations on this flow and associated tools in the aeronautic and space application domains.**

I. Introduction of the SoCKET project

The evolution of technology (SoC integration) and application needs leads to design more and more complex embedded systems both at hardware and software levels. So, companies in the field of critical embedded systems (aeronautics, space, automotive, health applications) have to face some technical and industrial challenges. Mastering this complexity, in order to improve the time to market and life cycle, the costs of the design, and the validation/certification of the critical embedded systems, is a key point to ensure the success of future industrial projects. For example, in aeronautics, the deployment of distributed miniaturized computers should decrease drastically the wiring complexity and cost, and the overall mass. In space industry, the increase of embedded computing power thanks to miniaturization and high level of integration will allow new missions.

By combining the efforts of industrial and academic partners, the SoCKET project (French FUI funded) main goal was to define a *"seamless" development flow*, integrating the equipment qualification/certification, from the system level to the Integrated Circuits (ICs) and the associated embedded software, compliant with the applicable standards (such as DO/ARP for aeronautic and ECSS for space).

This "seamless" flow requires some formalisms unification (elimination of semantic holes in HW/SW interfaces), the availability of models transformation operators (skeleton generation, requirements traceability), and tools interoperability. This "seamless" design flow is built upon technical pillars:

- High Level Synthesis (under time or resources constraints).
- Heterogeneous simulation techniques (SystemC/TLM [1], [2], [3], [4], LT, AT and CABA abstraction levels, ISS generation and integration).
- IPs encapsulation and interoperability (IP-XACT).
- Validation techniques (semi-formal methods, mutation analysis techniques, test cases automatic generation).

After introducing the SoCKET HW/SW Co-Design flow and the associated methodologies, we will provide the return of experience regarding the experimentations of this flow on representative case studies of aeronautics and space application domains.

[1]SoC: System on Chip

[2]See http://socket.imag.fr/

II. HW/SW Co-Design Flow

A. Overall Description

The design flow defined in the SoCKET project targets the design of critical embedded systems. It covers important steps as system architecture exploration, and the definition of virtual prototypes at different levels of abstraction to support early embedded software development, verification of hardware blocks, and preparation of qualification/certification activities.

1) A Design Flow Relying on Standards: In the embedded system world, it is quite common to integrate IPs from various providers, and to deliver (sub)systems to third parties. As the need to exchange models is rising, it is key to guarantee model and tools interoperability, and rely on well understood and stable interfaces for the models. Industrial applications in the aerospace domains also require durability and stability of the flow among the years, and can not have strong dependency on one given tool from a specific vendor. The flow is therefore based on standards.

Several standards are defined for modeling and verifying embedded systems. The IP-XACT standard (IEEE 1685) [5] defines a XML schema to describe IP interfaces and facilitates model exchange and integration by offering a common format to be used by integration or net listing tools.

The SystemC language (more precisely a set of classes on top of C++, plus a simulation kernel) is now defined as the IEEE 1666-2005 standard [6]. It offers primitives to model hardware systems, with modules, ports, synchronization mechanisms and processes to model the behavior of the IPs. As a complement, communication APIs at the transactional level have been defined by the SystemC consortium in the TLM 1.0 and TLM 2.0 standards (Transaction Level Modeling) [7]. They will be integrated in the 2011 revision of IEEE 1666 standard.

To address verification needs, the Property Specification Language (PSL) has been standardized as IEEE 1850-2005 [8]. All these standards are used in the flow depicted below.

2) Flow Overview: The design flow represented in Fig. 1 starts from the system requirements, that can be captured either in natural languages, or through specific languages as UML/MARTE. From these requirements, the global SoC specification is produced. System properties can be defined from this step. They will be reused and refined as hardware and software properties throughout the flow to check the consistency of the models and their implementation.

The *global SoC architecture* is defined by system architects from their expertise. Several complementary tools assist them in this crucial task. To dimension correctly the interconnect and the memory subsystem, bandwidth and latency are usually key figures. Cycle accurate models are required to produce these figures with the required level of accuracy. It is not affordable for complex systems to model all the IPs at this level of abstraction. So the strategy is to have a cycle accurate

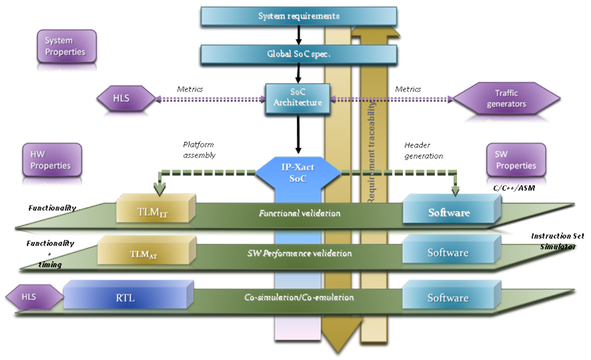


Fig. 1: SoCKET design flow

model of the interconnect and the memory subsystem, and generate traffic into the system by using traffic generators to mimic the profiles of the actual IPs. High Level Synthesis tools can also be adopted during this phase to get early figures of the HW resources consumed by the implementation of the various IPs. System architects loop on this iterative work until they reach a situation where system performances constraints are met.

A *Virtual Prototype* is developed as soon as the functional specification of the IPs is available. IP models are implemented first as Loosely Timed models, to enable early functional software development and development of the verification test suite for the hardware IPs. IP-XACT descriptions of the various IPs are used as a pivot format to keep consistency between the specification, the TLM models, and the software. It is indeed possible to generate from them parts of the TLM model, Interface Control documentation, header files for the software, and sanity checks to be run both on the RTL (Register Transfer Level) or the Silicon. IP-XACT descriptions are also used to automate the platform integration process and generate the top net list. In a second step, Approximately Timed models are implemented to validate performance figures of the hardware + software system, and tune the real-time aspects of the software implementation. This step can be complemented further with TLM/RTL co-simulation or co-emulation platforms, depending on the progressive availability of RTL models.

Validation and faithfulness of the virtual prototype is obviously a key concern. Validation should give good level of confidence that the model is functionally correct, and will enable pre-silicon software development. This software should execute in the same conditions on the virtual prototype or the actual device, with no modification. Moreover, the virtual prototype should not be more permissive than the target, to avoid any delay in the system bring-up. Assertions can be used to validate the behavior of the virtual prototype, and can be also reused to monitor the system in operation. Fault injection into the Virtual Prototype is also be considered to assess the robustness of the embedded software.

B. Methods and Tools

1) Modeling and Heterogeneous Simulation: Transaction-level modeling (TLM) is a novel technique motivated by the practical need of providing an early virtual prototype. These models are written in SystemC [6], a class library on top of C++, using the TLM standard communication APIs [7], with the following features:

- High simulation speed: 100x to 1 000x improvement compared to RTL as a matter of thumb are observed, enabling pre-silicon software development and interactive activities (e.g. software debug)
- Model accuracy: bit accuracy and register accuracy are required, as well as the representation of the system synchronization events [9]. Timing accuracy is not always a strong requirement: functional software development or functional validation may be operated with Loosely Timed models, whereas time accurate models may be required to optimize software implementation or deal with some real-time aspects of the system.
- Early availability: to enable pre-silicon activities, high level models must be available early.

Early development of SoC embedded software requires processor models to be integrated in the virtual prototype. Different solutions have been explored:
(1) Integration of an external ISS through a dedicated SystemC wrapper,
(2) Use of the ISS provided by the CAD vendor tool-suite,
(3) Generate ad-hoc ISS from its Instruction Set Architecture (ISA) description including some micro-architecture details (see [10]).

2) IPs Encapsulation and Interoperability: The methodology aspects (modeling, assembly, safety, validation, certification, SoC and embedded SW) which are targeted in SoCKET drive several needs, concerning IP encapsulation and interoperability.
These needs have been split in four categories:
(1) Interfaces standardization for each IP used in a company, to ease IP assembly and reuse and to enable models interoperability,
(2) IP library management to provide the users with a set of coherent IPs facilitating the assembly,
(3) IP configuration management is also required, with the description of the configuration levels for the components of the library,
(4) Automation and verification of the assembly
are simplified thanks to the IP-XACT (IEEE 1685) description of the platform and its components. Thus the process can be fully or partially automated and qualified.

3) High-Level Synthesis: HLS raises the abstraction level by taking abstract specifications that focus on functionality rather than cycle accuracy specified at the Register Transfer Level (RTL). HLS tools allow designers to rapidly generate complex RTL hardware architectures that are optimized to various performance, area and power requirements.

HLS tools transform an untimed (or partially timed) high-level specification into a fully timed implementation [11], [12].
They automatically or semi-automatically generate a potentially pipelined architecture.
In addition to memory banks, and communication interfaces, the generated architecture is described at the RT Level and contains a data-path (registers, multiplexers, functional units, buses) and a controller as required by the given specification and the design constraints.
SystemC simulation models are also generated at both Cycle accurate and Transactional (TLM) level of abstraction for fast virtual prototyping.

4) Verification – Monitoring Temporal Properties: *Assertion-Based Verification* (ABV) aims at guaranteeing that designs obey properties, usually expressed as logico-temporal assertions that capture the design intent [13].
These assertions can be checked using static (model-checking) or dynamic (simulation-based) techniques.
The shortcoming of runtime methods is that they do not enable an exhaustive check, but the advantage is that they are not limited by the design complexity.

The SoCKET design flow makes use of dynamic ABV solutions, based on the construction of *observation monitors* from PSL [8] assertions.
At the transactional (TLM) level, we thus check assertions that express properties regarding transactions in any kind of communication channels (for instance, *a given transfer eventually completes*, *data are transferred at the right place*, etc.) [14].
At the RTL level, we monitor more accurate properties on the design signals [15].

III. Astrium's Experimentations

A. Overview

Astrium's use case was in the Guidance/Navigation/Control domain, specified as an image processing algorithm supporting mobile object extraction/tracking. The main goal was to define the optimal data processing architectures for different sets of parameters of this algorithm.

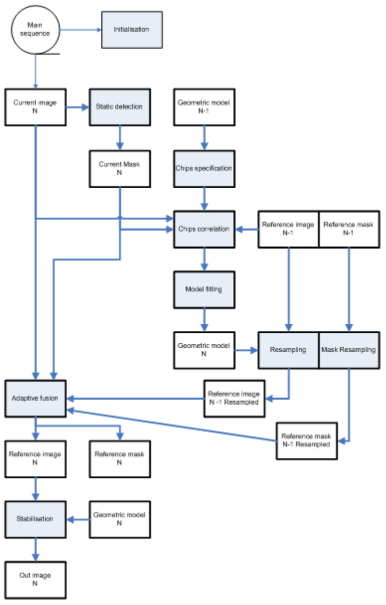


Fig. 2: Astrium Use Case Algorithm

In a first phase, we focused on the Design space Exploration (using SystemC/TLM modeling), with fast prototyping of HW functions (using HLS), and then on the verification of major HW IPs at TLM level (using Monitoring Temporal Properties techniques). The capture of all HW/SW interfaces are performed into IP-XACT formalism, and used to auto-code skeletons for SystemC/TLM, OBSW and HDL models.

We focused on the two following architectures:

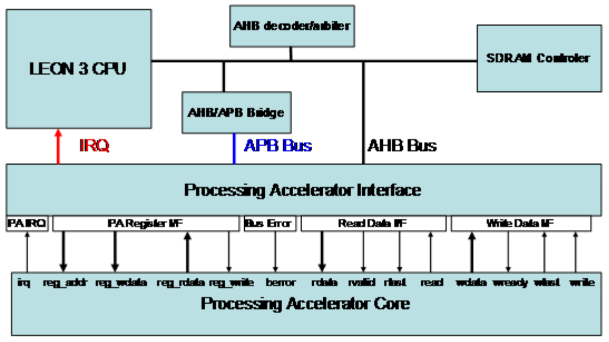


Fig. 3: Accelerator Architecture

In this first architecture (Fig. 3), we mapped a complete algorithm step into a HW accelerator. The full algorithm dataflow is a sequence of SW steps interlacing with HW steps.

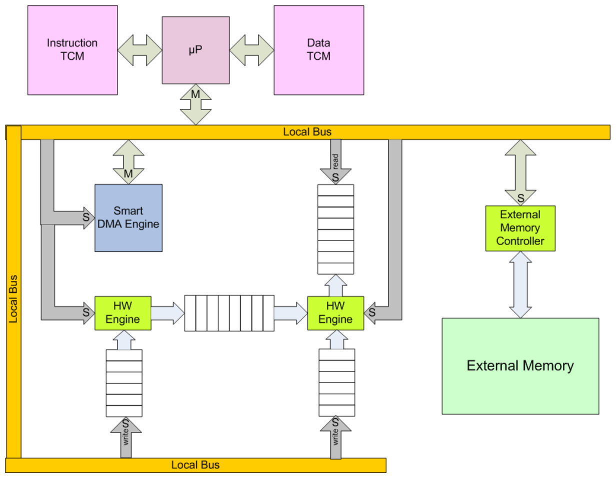


Fig. 4: Streamed Operators Architecture

In this second targeted architecture (Fig. 4), the SW is in charge of the full algorithm pipeline by using shared streamed operators to accelerate the processing.

The main evaluation criteria (improvements) are the overall performance and power consumption, but also the equipment testability, and mainly the smoothing of the collaboration between the various involved development teams.

To this day, dedicated numerical simulators for on-board software and avionics validation were used. Another major evaluation criteria was to build them from the SystemC models, instead of using paper HW specification documents.

B. Golden Model creation

The SoCKET flow is built upon the principle that each refinement step is validated from the result/testbench of the previous one (incremental flow).
The entry point of Astrium's use case is a functional model built in Mathworks Matlab^{TM}(and a paper specification). To check easily the behavior of a given step with the test-bench of the previous one, we need a bit accurate check-ing. So, a C functional model has been written from the Matlab^{TM}implementation in order to use this C model as a reference (Golden Model) for the functional and bit accuracy behavior for the full incremental flow. This model has been validated against the Matlab^{TM}one thanks to test scenarii with quality criteria and intermediate (inter algorithm stages) results (e.g: floating to fixed point migration).

C. High Level Synthesis techniques

Astrium used HLS technology (GAUT) in order to fast prototype the cost (achieved performance, silicon surface,..) of dedicated parts of the overall algorithm candidate to an HW implementation. This step allows also to optimize the HW/SW interfaces by prototyping earlier in the design flow, the inefficient round-trips between the HW and SW part.

We use also HLS tool to identify common factorisable oper-ators between the algorithm stages by using Data Flow Graph and internal representation Abstract Syntax Tree produced by GAUT tool.

Using HLS helps to get an early confidence in the design, and to delay the design freeze thanks to the ease of iteration process. Another interest is to ease the IP maintenance/evolution.

Using HLS requires both hardware know-how and SW skills: Handling IO exchanges (FIFO, Cache, prefetch, ...) and data organization in memory remains one of the main challenge.

D. IP-XACT and skeletons/documentation generation

When designing an embedded system, one of the major source of issues is the misunderstanding at interfaces (HW/HW and HW/SW).
The use of IP-XACT, as proposed by SoCKET flow, allows to capture, in a well defined and standardized formalism, all the semantics of these interfaces (with vendor extensions if needed). These descriptions can be shared between HW and SW teams.

The first usage of IP-XACT is for platform assembly (virtual (SystemC/TLM) or physical (HDL) one) by generating top file, checking for any interfaces incoherences, and instantiating required interface adapters.

At Astrium, we focused on skeletons and documentation generation. Indeed, thanks to M2T (Model to Text) transformations (the Magillem ToolSuite evaluated in the frame of SoCKET project is built upon Eclipse and Template Jet mechanisms), we are able to automatically generate skeletons for SystemC/TLM components, HDL ones, and also any header files for low level on-board Software HW abstraction layer. By automating this transformation, we insure the full coherency between the HW and SW understanding (address mapping/decoding, bit-fields meaning, ...).
The need of requirements traceability, common to all critical embedded systems domains, can be supported by generating the correct tags into the methods headers (for parsing by traceability tools like Reqtify).
In the same way, most part of the datasheet, and user's manual documentation can be generated from this IP-XACT description.

Due to its history, IP-XACT standard needs some extensions to be able to capture all HW/SW interface semantics (e.g: Reset on Read characterisation for bit field, or the ability to describe (keep the link to) the description of the different modes of the IPs and how to configure them).
In an other hand, in order to ease the deployment of an IP-XACT based flow, it is mandatory that all IP providers distribute also their IP-XACT descriptions.

E. Modeling techniques

Regarding modeling techniques, Astrium's activities focused on 2 subjects:

- define a modeling infrastructure (coding rules on models, building environment, existing IPs library with a focus on bus model, convenience functions hiding the complexity of generic TLM interfaces, accelerating the simulation,...)
- define the proper way to add some timing annotation on pure functional models.

As described into the SoCKET flow (see Fig. 1), modeling techniques are used at different steps:

- architecture exploration: In this case, we need mainly to have a Cycle Accurate Bit Accurate interconnect model with some traffic generator miming the external (bus exchange) behavior of the given IPs, in order to identify any bottlenecks in the architecture.
- HW/SW co simulation (functional Validation): In this case, the models of the platform shall be fully bit accurate (address map and registers mapping fully representative). The time modeling is minimal in order to run an Operating System.
- Performance Validation: In this case, the models shall mimic the temporal behavior to be able to extract some performance trends.

At Astrium, regarding time modeling, we developed a solution based on the separation of functional and non-functional aspect in transactional level models [16]. With this solution, name TTP (Timed TLM Protocol), for each IP, we have a pure functional model coupled with a timed one. The switching between the phase PV (Programmer View) and T (Timed) are made when all functional models reached a SSP (system Synchronization Point: given register access, interrupt and all state alternating accesses). During the functional phase, the Timed models record any transactions with their related meaningful information (address, exchange size, ...). During Timed phase, the T models replay the last execution PV phase by simulating the time following the records.

We validated this concept on multi-port memory controller (with timing information extracted from well defined test cases running on the real HDL model). The TTP solution has the big advantage to allow the independent development and debug of PV and T models. In an other hand, the modeling is more abstracted, so more complex , and sporadic events (IRQs) are harder to model. In its current form, TTP cannot be used on an industrial scale. After this experiment, we conclude that heterogeneous simulations (mixing SystemC/TLM model with HDL ones) are more fitted to fulfill the need of performance validation.

We experimented also the HW-In-The-Loop co-emulation solution with two objectives:
(1) Deploying an incremental validation flow (SC/TLM testbench, some sub-blocks running on FPGA hardware, the others ones still running as SC functional models).
(2) Improving the simulation speed.
The main concern for this co-emulation activity is how to

manage the clock. The easiest implementation, uncontrolled mode, requires to decouple HW blocks (FIFO interconnections), but blocks with strong timing requirements on IO are hardly compatible with this principle. Another feedback is to use a standard API like SceMi for co-emulation in order to capitalize on transactors reuse.

Regarding modeling by itself, our experimentation highlight the fact that an upper abstraction layer (read/write like) upon TLM 2 one is mandatory for architect use, in order to mask all low level transaction mechanisms provided by TLM library. TLM 2 is currently a memory mapped platform oriented protocol. We need to have some abstraction to model easily interrupts and streamed data exchange.
Having a building environment supporting fast iteration during architecture exploration phase is also a must have (pure IP-XACT based solution requires IP-XACT description that are not available during architecture exploration phase for new IPs).
Moreover, some well defined modeling guidelines have been proposed in order to insure model exchange and maintenance.

Astrium's On-Board SW validation process is based since several years on Virtual Platform built on C models coded from HW paper specification.
Another return of experience of the SoCKET project is to use of SC models (representative by construction of the final HW) as golden model for these C models validation.

SystemC/TLM modeling is today deployed in Astrium for feasibility study and architecture exploration activities. The produced Virtual Platform is provided as annex of the paper specification for disambiguate/clarify it. The use of this platform into the HW IPs verification flow is under consolidation.

F. Assertion Based Verification techniques

At different phases of the SoCKET design flow (see Fig. 1), the satisfaction of dedicated properties has to be guaranteed (namely at the System level, and after HW/SW partitioning). Some of these requirements, originally provided as textual descriptions, can be fully disambiguated when translated into PSL assertions.

The ultimate goal of Astrium is to verify these properties at the System Level using TLM-based Virtual Platforms, but also at the implementation level using RTL designs. The ISIS tool fits the Functional Validation and SW Performance Validation steps.

Following the standard validation (test) process, we usually run one or more test scenarii (related to a given requirement) and check the results. The ABV approach gives the possibility to complement this process by automatically instrumenting the virtual platform code with functional properties in order to check them while executing the full non-regression test suite (e.g., to verify that a client does not read data in a DMA destination area when a transfer is on-going).

After these experiments, Astrium issued a first noticeable return of experience, summarized as follows:

- the expressivity of PSL is fine, and the benefits provided by the Modeling layer are undoubtable,
- the moderate overhead induced by the ISIS monitors is attractive (5-8%),
- there is a need to define strict rules for the textual (natural language) description of the properties (observation/connection points identification, disambiguation of transaction meaning in terms of function calls and parameters involved, ...)

Regarding ABV technologies, our future works include the refinement of these property definition rules. We will also focus on the relation between such PSL properties at the TLM level and their counterparts at the Register Transfer level, as well as on prototyping the embedding of the automatically generated monitors into the final design in order to mitigate the space radiation environment effects at the architecture level (today, most of the mitigation is performed at the silicon technology level).

G. Conclusion

For Astrium, thanks to the SoCKET project, we matured our design flow based upon models, and how to use these models in order to improve and smooth the communication between involved teams (functional algorithm definition, system, SW and HW development teams) for a new payload processing mission.

IV. Airbus' Experimentations

A. *Overview*

AIRBUS Avionics and Simulation Products Department aims to design a flight control remote module.

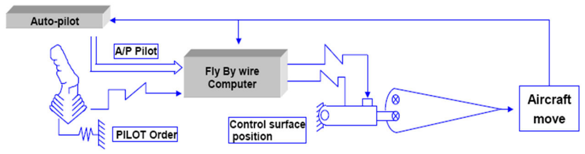


Fig. 5: Case Study Environment

This Avionics Computer is dedicated to process data from sensors and to control actuators according to predefined flight laws. The goal was to implement the architecture in a SoC technologies in a Hardware & Software critical avionics context (DO-254 and DO-178 : DAL A).

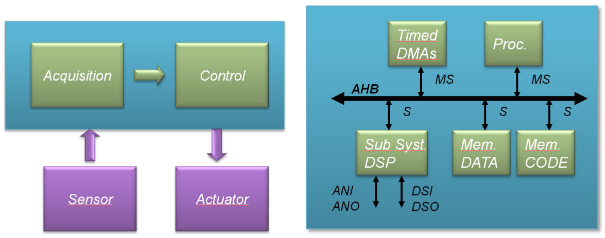


Fig. 6: Functional description & Targeted

Due to SoC targeted solution, new certification issues and specific safety requirements must be taken into account.

The focus of the first phase is on requirements validation at the platform level, in order to mature IP specifications (golden model concept), and to allow earlier software development by providing virtual platform (SystemC/TLM) to SW designers. Second, we reuse these golden models in order to verify design results using co-simulation. One of the objectives was also to integrate and validate safety requirements by using Assertion Based Verification.

Throughout this design process, DO-254 and DO-178 certification compliance must be achieved (e.g., requirements traceability). This will contribute to highlight relevant evidences for future SoC certification review.

Faster specification maturity achievement was expected and improved exchanges between designer teams are foreseen by using shared formalisms and standards.

B. *Modeling techniques*

The following figure gives an overview of the Socket Flow application perimeter for Airbus case study:

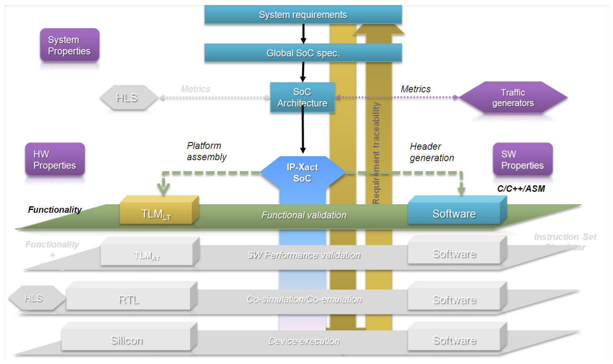


Fig. 7: SoCKET Flow application perimeter for Airbus case study

The main Airbus axes of the SocKET flow application were:

- HW/SW architecture Requirements capture
- Functional requirement modeling in a virtual executable specification (code generation, makefile, netlist,etc,...)
- HW/SW Functional Validation based on the virtual platform
- Safety requirements capture in assertion
- Requirements Traceability

C. *SystemC/TLM Virtual Platform Development*

The first phase of the experiment began with the development of different modules that constitute the platform of the case study. To ensure interoperability between these modules and other modules developed by third parties, the SystemC/TLM transactional interface library was used to develop the models of the platform. The modeling approach chosen is functional LT (Loosely Timed).

During the project, several possibilities were investigated in order to find a CPU model to execute the platform of the case study. For example, the trap-gen tool allows the generation of processor models (ISS) from the description of the instruction set and architecture files (Python).

With TLM2, SystemC language presents a new and important step, and more and more models are available with this interface. In our case study, the model of the bus and the CPU, from third-party development, could be connected to those developed in-house without difficulty. However, to allow a better and easy use of SystemC, the library should be completed to facilitate the construction of transactional model, like the concepts in the library SCML2.

The second phase of the experimentation was used to assess the ability of IP-XACT formalism to describe the components and assembly of the platform. The possibilities for code generation, supplied with the tool Magillem, were also explored.

The modeling step of the platform has removed some ambiguities (clarity, completeness, consistency, etc.) against the requirements specification of the case study. With the virtual platform, the different stakeholders HW / SW has an executable specification that allows a first validation of the structure and dynamics of the system.

D. Requirement Traceability with IP-XACT format

One of Avionics item is to prove that no requirement is forgotten through the design process and that there is a perfect match between customer specification requirements and the design implementation. Therefore Requirements Traceability must be fulfilled in order to achieve DO-254 and DO-178 certification compliance. Each requirement is allocated, refined and verified throughout the design process (Figure 1). In addition to requirements coverage, traceability can also allow to perform impact analysis upon requirement modifications.

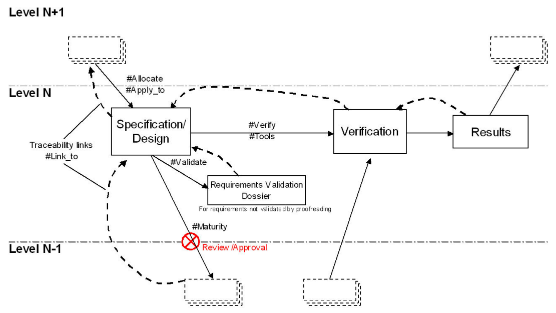


Fig. 8: General Traceability flow

More than ever in the System On Chip design context, involved requirements are heterogeneous (hardware, software,...). They include also requirements for Safety which must be taken into account in the development. It is necessary to achieve traceability between all kinds of requirements. Tools involved in the SoCKET project take traceability into account.

Magillem provides an Eclipse-based framework which mainly relies on the IP-XACT formalism (IEEE 1685). In order to cope with traceability issues, a specific flow has been defined. This framework allows to integrate requirements inside the global design flow defined by the SoCKET project.

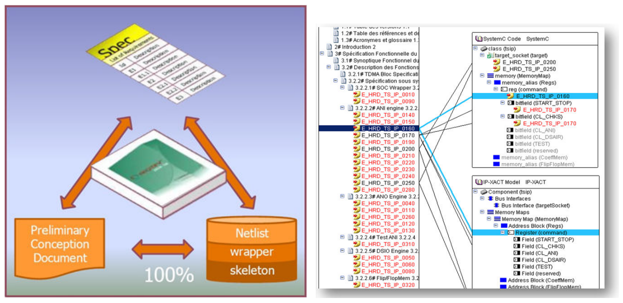


Fig. 9: SoCKET Traceability flow

The first step is to be able to import specification requirements into the Magillem framework, and then to build the solution architecture based on IP-XACT complying with these requirements. Magillem framework must provide ability: (1) to define design requirements, (2) to map them on the IP blocks defined in the architecture, and (3) to link them with the appropriate specification requirements.

This enables a dedicated tool to cross check the requirements between the specifications, the preliminary design document, the source codes, and the verification manuals.

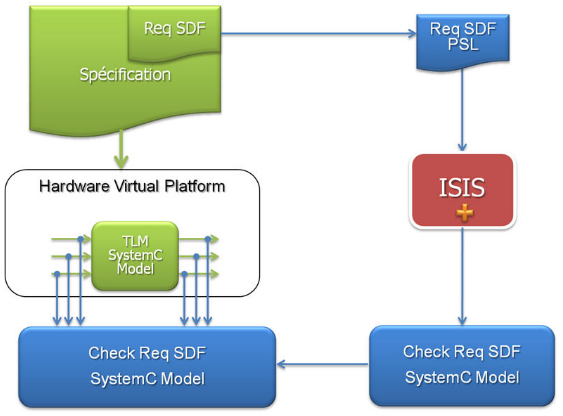
IP-XACT requirements description allows only one single source along the flow, during conception, implementation and verification steps. It also makes possible to automate document generation (especially the generation of preliminary design document) and the generation of code (Netlist, Wrapper, Skeleton) with requirement tags included.

In order to implement these features, the IP-XACT standard has to be extended with the concept of requirement. In addition the Magillem framework has to be upgraded in order to manage this concept and to provide interface with requirement management tools like Reqtify (GeenSoft).

E. Assertion Based Verification techniques for Safety Requirements

One target of Socket project was to define methods and tools to verify that the HW/SW requirements developed in compliance with Safety Requirements.

The methodology of SoCKET Project is to translate the Safety requirements in functional properties, based on the PLS standard language. With the tool ISIS of TIMA, we can then generate SystemC monitors from these properties, and integrate them into the virtual platform. During the execution of the platform and instrumented, the traces produced by the monitors, inform us of any violations of the properties.



Req SDF
Spécification
Req SDF PSL
Hardware Virtual Platform
TLM SystemC Model
ISIS
Check Req SDF SystemC Model
Check Req SDF SystemC Model

Fig. 10: SystemC Monitors Generation

After a selection of safety requirements, we had to refine these requirements so that they are easily translated into PSL properties. For example, the functional behaviors requirements, such as monitoring requirements are easy to translate into language PSL. For experimentation with the tool ISIS from TIMA, Safety requirements for the integrity of the memory of the DSP CoeffMem were selected. All of the Safety requirements can not be expressed directly in PSL properties. They must first be translated according to the

chosen implementation in the platform. Modelling with the layer of PSL, it is possible to use C++ expressions to facilitate the writing of the property.
We then generate the monitors using the ISIS tool TIMA and integrate these models into the platform. We were then able to simulate various erroneous functions in the hardware monitoring mechanism in order to check the reactivity of monitors generated with ISIS and to validate the robustness of safety requirements. (ex : HW & SW fault injection).

F. Conclusion

The experiment conducted in the Airbus case study allowed to highlight some advantages to using the SoCKET Flow and the virtual prototyping in the development phase.

1) Data obsolescence management: The use of standard (SystemC / TLM, IP-XACT, PSL) guarantees a certain continuity of data required in the aviation context. The standards also limit the dependence on the suppliers of tool.

2) Complexity management: The validation of the architecture and functional specifications of the system is more efficient because the virtual platform is a shared reference between the HW and SW teams. This executable specification can remove some ambiguities, unclear, and other inconsistencies that can be encountered in a specification document text. The observability offered by the virtual platform is enhanced and offers new features such as the ability to freeze the implementation of the platform, or change the condition of equipment at any time. In addition, the monitors generated by assertions can be used to verify properties at the system level, and ensuring the right functional behaviour of hard and soft. The use of the platform makes it possible to provide an additional improvement in terms of safety and certification context.

3) Productivity gain: In the validation phase of HW/SW requirements, the use of the platform can reduce the iterations, because the development of a TLM platform is shorter than at RTL. By having the virtual platform in phase advance of the equipment, software development environment (compiler, debugger) can be prepared. Also in parallel with hardware design, software layers depend on the material can be developed on the virtual platform. The maturity of the software is improved when the physical platform is available.

V. Global Conclusion

The SoCKET flow provides a real improvement regarding HW/SW co-design methods. Moreover, the usage of Open Source standards, such as SystemC/TLM, IP-XACT and PSL, insures data continuity. these standards limits also the dependence with tools vendors.
The main objective of SoCKET flow was to transmit to critical embedded systems companies the know-how of semiconductor ones in order to define an automated flow with adequate tools.

An efficient deployment of such flow requires the building of a substantial toolbox, including standard models library (processor core, local bus, DMA, peripherals,...), a mature development/assembling/debugging environment, and also all coding rules and other guidelines for using HLS, SystemC/TLM modeling and ABV techniques.

The evaluated academics tools are high quality ones, answering the expressed needs. Some improvement linked with SoCKET flow have been identified, and their deeper and smother integration in the flow will be studied.

In the frame of critical embedded systems, the experimentation highlighted the interest of Virtual Platforms in the development phase. The architecture solution validation and the functional specification of the system are more efficient, since the virtual platform is the shared reference between HW and SW teams. This executable specification allows to remove any remaining ambiguities, incoherences and clarify fuzzy points often met with textual paper specifications.

The SoCKET project has also investigated the propagation of traceability tags between the paper specification and the virtual platform thanks to the definition of some ”vendor extensions” for the IP-XACT standard. It would be interesting to have these extension, related to traceability support, included into next release of IP-XACT standard. The participation of System Companies to normalisation bodies/consortium such as Accellera/OSCI for IP-XACT and SystemC/TLM could be a way to insure the taking into account of these needs.

The observability provided by the virtual platform is better than on silicon one, and offers some added-value features, such as platform execution freezing, or HW state modification at any time.
Moreover, the usage of assertions, implemented by monitors, allows to perform some properties checking at system level, insuring the common behavioral correctness of SW and HW. Then, the use of a Virtual Platform allows to add some certification credentials against safety requirements. During HW/SW requirements validation phase, a Virtual Platform allows shortening the iterations cycle. Indeed, modeling at transactional level is faster than at RTL one.

With a Virtual Platform available sooner than Silicon one, the SW development environment (debugger, compiler, OS, ...) can be set up earlier.
In parallel with HW design phase, the HW dependent SW layer can be developed and first debugged on the Virtual Platform, improving the overall maturity of the SW, and also the HW, by discovering any discrepancies earlier.

The performed experiments highlight that the SoCKET flow (SystemC/TLM, ABV, HLS, ..) can be directly applied also to a complete critical embedded equipment or at board level. Indeed, this flow is not tightly coupled with to System On Chip (SoC) technology.
A complementary analysis shall be made to challenge the representativity of SystemC/TLM models to replace partially or fully the physical verification against the requirements for HW and SW certification(DO178 and DO254).
The space applicable standards (ECSS) will not be impacted by the deployment of a such flow (SoCKET). Indeed, ECSS documents describe mainly the design/development/verification flow with a fully defined documentation structure, and take already into account the development/verification process based upon virtual platform. The deployment of the SoCKET flow to other critical embedded systems domains such as IEC 62566 (on-going) for nuclear power industry, ISO TS16949, ISO26262 for automotive domain, or CENELEC EN50126/8/9 for rail transport, should be straightforward.

In an other hand, as future work, we can identify the running P project studying how to connect the SoCKET flow with a more large System to SW/HW one.

References

[1] *IEEE Std 1666-2005, IEEE Standard System C Language Reference Manual*. IEEE, 2005.
[2] T. Grötker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Kluwer Academic Pub., 2002.
[3] D. Black and J. Donovan, *SystemC: From the Ground Up*. Springer, 2004.
[4] F. Ghenassia, Ed., *Transaction-Level Modeling with SystemC*. Springer, 2005.
[5] "Standard for IP-XACT, Standard Structure for Packaging, Integrating and Re-Using IP Within Tool-Flows [online]," http://standards.ieee.org/announcements/2010/pr_pr_1685.html.
[6] "IEEE Standard SystemC Language Reference Manual, [Online]," http://standards.ieee.org/getieee/1666/download/1666-2005.pdf, December 2005.
[7] "TLM-2.0 Language Reference Manual [online]," http://www.systemc.org/downloads/standards.
[8] *IEEE Std 1850-2005, IEEE Standard for Property Specification Language (PSL)*. IEEE, 2005.
[9] M. Moy, F. Maraninchi, and L. Maillet-Contoz, "LusSy: an open tool for the analysis of systems-on-a-chip at the transaction level," *Design Automation for Embedded Systems, special issue on SystemC-based systems*, 2006.
[10] "TRAP TRansaction level Automatic Processor generator, [online]," http://code.google.com/p/trap-gen.
[11] P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuits*. Springer, 2008.
[12] P. Coussy and A. Takach, *Special Issue on High-Level Synthesis, IEEE Design and Test of Computers*. IEEE Computer Society, 2008, vol. 25.
[13] H. Foster, "Applied Assertion-Based Verification: An Industry Perspective," *Foundations and Trends in Electronic Design Automation*, vol. 3, no. 1, January 2009.
[14] L. Ferro and L. Pierre, *Advances in Design Methods from Modeling Languages for Embedded Systems and SoC's (Selected Contributions from FDL'09)*. Springer, 2010, ch. ISIS: Runtime Verification of TLM Platforms.
[15] K. Morin-Allory, Y. Oddos, and D. Borrione, "Horus: A tool for Assertion-Based Verification and on-line testing," in *Proc. MEMOCODE'08*, June 2008.
[16] J. Cornet, "Separation of functional and non-functional aspects in transactional level models of systems-on-chip," Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2008.