A Design Flow for Critical Embedded Systems

Vincent LEFFTZ[†], Jean BERTRAND[♯], Hugues CASSE[‡], Christophe CLIENTI[◇], Philippe COUSSY[◁],
Laurent MAILLET-CONTOZ[▷], Philippe MERCIER[⋈], Pierre MOREAU[§], Laurence PIERRE[♭], Emmanuel VAUMORIN[⋆]

[†]Astrium (Toulouse, FR), [♯]CNES (Toulouse, FR), [‡]IRIT (Toulouse, FR), [◇]Thales R&T (Palaiseau, FR),
[◁]Lab-STICC (Lorient, FR), [▷]STMicroelectronics (Grenoble, FR), [⋈]PSI (Grenoble, FR),
[§]Airbus (Toulouse, FR), [♭]TIMA (Grenoble, FR), [⋆]Magillem (Paris, FR)

***Abstract*—The SoCKET project (SoC[1] toolKit for critical Embedded sysTems)[2] gathers industrial and academic partners to address the issue of design methodologies for *critical* embedded systems. They work 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 briefly describes its application to industrial case studies.**

I. Introduction

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 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. For automotive and health applications, the ability to master the validation/certification flow of SoC's will allow to use them in these fields.

By combining the efforts of industrial and academic partners, our main goal is 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 norms (aeronautics: DO-178C, DO-254, ARP4754 - space: ECSS Q60-02, Q80, E40).

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. The proposed flow is built upon technical pillars:

- High-Level Synthesis
- Heterogeneous simulation techniques:
  - SystemC/TLM (Transaction Leval Modeling) LT (Loosely Timed), AT (Approximately Timed) and CABA (Cycle-Accurate Bit-Accurate) abstraction levels [1], [2], [3], [4], [5]
  - ISS (Instruction Set Simulator) generation and integration
- IPs encapsulation and interoperability (IP-XACT).
- Validation techniques (semi-formal methods, mutation analysis techniques, test cases automatic generation).

The main outcomes of the project should be:

- a design flow supporting critical embedded systems development,
- a draft IDE implementing this flow and tested with partners's tools (adaptable with other tools and for other applications),
- some return of experience through 4 industrial case studies,
- some Certification/Qualification kits for IPs and SoCs in each application domain, as well as some recommendations to certification and normalization bodies.

II. Related Work

This project lives into a large R&D ecosystem, exchanging information/tools and models with other projects such as SoCLib (IPs libraries) [6], OpenTLM (Modeling language and IPs libraries) [7], TWINS (HW/SW flow) [8], SPICES (tools for critical embedded systems) [9], and Topcased (tools environment) [10].

SoCLib is an open platform for *virtual prototyping of multiprocessors systems on chip*. The core of this platform is a library of SystemC simulation models for virtual components (CPUs, caches, memories, peripherals, interconnects,...) with a VCI/OCP communication interface. Each component is provided at two abstraction levels: cycle-accurate and transaction level with time (TLM 2.0 compliant). SoCLib comes with a set of associated tools: design space exploration, high level synthesis, accelerator for cycle-accurate simulation,...

The aim of OpenTLM is the development of tools dedicated to the *virtual prototyping of systems on chip at the transactional level*. Tools under consideration include solutions for simulation, test, and analysis of TLM descriptions, as well as the connection with validation environments, through the definition of appropriate SystemC semantics.

TWINS (OpTimizing HW-SW co-desIgn flow for software iNtensive System development) is a project of the ITEA2

[1]SoC: System on Chip
[2]See http://socket.imag.fr/

978-1-4244-5841-7/10/$26.00 ©2010 IEEE

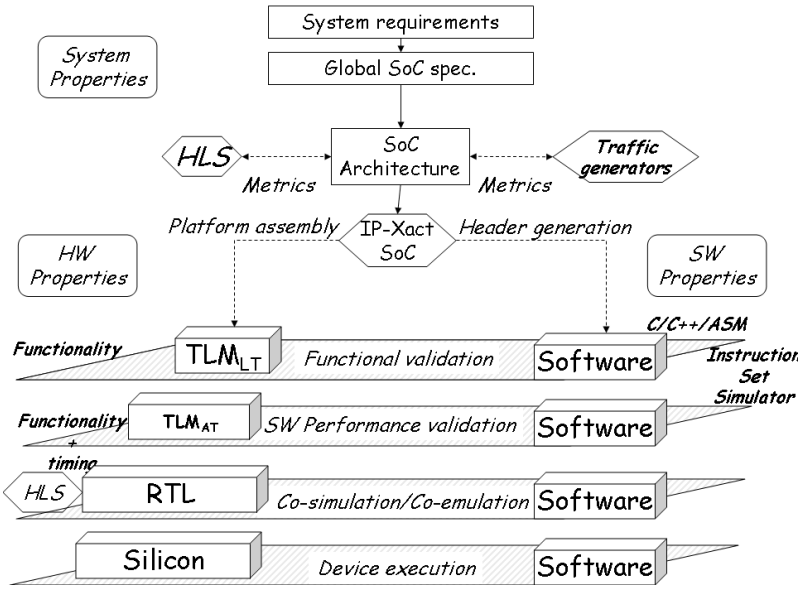

Fig. 1. SoCKET design flow

European program (devoted to embedded software). It focuses on *co-design in the context of integrated hardware/software development*. Methods of interest are not specific to critical systems.

The ITEA SPICES project (Support for Predictable Integration of mission Critical Embedded Systems) ended in 2009. It resulted in the development of various tools that may be of interest for the SoCKET project, such as an AADL to SystemC generator [11].

Topcased (The Open-Source Toolkit for Critical Systems) is another French project, devoted to the production of *an open-source environment for the development of critical hardware/software embedded systems*. It mainly follows a model-based approach. Its outcomes may be beneficial to the SoCKET project e.g., the tools for requirements traceability.

Safety-critical applications become also key in the automotive area. The objective of the SASHA project [12] is the development of tools that help complying accurately with both the present and future safety standards. The most important one is the ISO26262 that shall help to harmonize the approaches in the automotive industry. The main objective of the project is to apply the life cycle described within the standard ISO 26262 onto a pratical safety function: monitoring of the consistency of the value of the target engine torque.

III. 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 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) [13] 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 [14]. 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 TLM2.0 standards (Transaction Level Modeling) [5]. They should be integrated in the 2010 revision of IEEE 1666 standard. To address verification needs, the Property Specification Language (PSL) has been standardized as IEEE 1850-2005 [15]. All these standards are used in the flow depicted below.

2) Flow Overview: The design flow represented in Figure 1 starts from the system requirements, that can be captured either in natural languages, or through specific languages as UML or 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 the 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 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 resources consumed by the implementation of the various IPs. System architects loop on this 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, 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.

Automatic refinement of models is the Holy Grail, but is in practice very difficult to obtain. As a consequence, appropriate validation means are integrated in the flow to ensure functional equivalence of LT and AT views. In particular, the reuse of the tests suite at the various levels of abstraction is a good way to validate that the functionality remains unchanged after each refinement step. Advanced weaving techiques can also be applied to generate a AT model from a LT model and micro-architecture information.

From system properties, it is also possible to express assertions that are used to validate the behavior of the virtual prototype during the simulation. They can be reused to monitor the system in operation. Fault injection can 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 technique motivated by the practical need of providing an early virtual prototype. These models are written in SystemC [14], using the TLM standard communication APIs [5], 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 [16]. 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. The usual target is 6 months before the availability of RTL models

From the early 2000's, STMicroelectronics has setup internally a full TLM methodology to develop virtual prototypes. It is based on a TLM modeling API, called TAC [4]. Initially fully proprietary, this API has been donated to OSCI as a contribution to the standardization efforts, and is now aligned to the TLM standards. It offers support for Loosely Timed modeling style, targets early embedded software development and functional verification activities. It is used for production in the various application areas of the company to model complex Systems On Chips.

Early development of SoC embedded software requires processor models to be integrated in the virtual prototype. GLISS [17] is a generator of Instruction Set Simulators (ISS). The Instruction Set Architecture (ISA) is expressed in SimNML [18], a specific language designed to lighten this hard and error-prone work. The code, the disassembly syntax and the semantics of each instruction is given in a form close to microprocessor ISA handbooks but SimNML provides also a way to factorise this description as much as possible: common behaviour (like memory access modes) may be grouped and re-used. These features allow to speed up and to make safer the ISA description work that is, in a second step, processed to generate automatically different tools like a functional simulator, a disassembler, a debugger, etc. GLISS has been successfully used to support instruction sets (PowerPC, TriCore, Sparc, x86, Sharc, ARM, ...).

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 assembly are simplified thanks to the IP-XACT description of the platform and its components. Thus the process can be fully or partially automated and qualified.

IP-XACT (IEEE 1685) is the right standard for answering lot of these needs: documentation of interfaces, description of file sets and management of views, parameters and configuration at platform or IP levels, meta modeling and scripting of the flow.

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 [19], [20], [21]. 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 level (TLM) 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 [22]. These assertions can be checked using static methods (typically, 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 design flow makes use of dynamic ABV solutions, based on the automatic construction of *observation monitors* from PSL [15] assertions. At the transactional (TLM) level, we thus check assertions that express properties regarding transactions in any kind of communication channels [23], [24]. Here are two simple examples: (1) Regarding a DMA component: *any time the DMA control register is programmed (transfer starts), an end-of-transfer notification occurs before the next writing into the control register (next transfer)*, (2) Regarding a multicast packet switch: *every packet sent by any sender X to any receiver Y will be delivered to Y, within a given time limit*. At the RT level, we monitor more accurate properties on the design signals [25]. Any assertion of the simple subset of PSL (which conforms to the notion of monotonic advancement of time) is practicable.

5) Observability and Debug: Debugging an application running on a system on chip can quickly become a nightmare if the SoC does not provide enough debug capabilities. The definition of the debug resources, that will be further used by Software and System engineers to tune their application, shall be addressed very early in the design phase during the architecture definition. Debug shall be addressed at different levels:

- Hardware level : this includes access to processor debug support units, on-chip bus access and monitoring, ...
- Software level : For instance communication with the GNU debugger GDB [26] server.

Generally, debug at hardware and software levels are completely separated. The SoCKET project investigates the possibility to use the various debug resources to address the debug at system level and by using a common interface to minimize the pin count. The possibility to reuse debug resources in operation to monitor the application is also addressed within the project.

IV. Industrial Case Studies and the Design Flow

We give here a brief overview of the case studies that will be used by the industrial partners to evaluate the proposed design flow and its technological solutions.

A. Space High Resolution Image Processing

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

In a first phase, we focus on the Design space Exploration (using SystemC/TLM modeling), the fast prototyping of HW functions (using HLS), the verification of major HW IPs at TLM level (using Monitoring Temporal Properties techniques). 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 will be to build them from the SystemC models, instead of using paper HW specification documents.

B. Avionics Flight Control Remote Module

AIRBUS aims to design a flight control remote module. This Avionics Computer is dedicated to process data from sensors and to control actuators according to predefined flight laws. Due to SoC targeted solution, new certification issues and specific safety requirements must be taken into account. We first

plan to focus 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 plan to reuse these golden models in order to verify design results using co-simulation. 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 is expected and improved exchanges between designer teams are foreseen by using shared formalisms and standards.

C. Scientific Payload On-board Processing

CNES has decided to apply the SoCKET design flow to challenge the design of an on-board computer dedicated to scientific payload. The use case is the Digital Processing Unit of the SWARM satellite. Work will focus on HW/SW partitioning considering both performance and safety issues but will also address other topics such as co-simulation or generation of Hardware Abstraction Layer. Another objective of this study is to evaluate the new design flow from an inexperienced point of view such as that of entities having little knowledge in the field of critical embedded systems.

Tasks to be performed during the use case implementation and evaluation criteria will be mostly similar to those identified by Astrium. However as the algorithms to be implemented and the expertise and experience brought by the design teams will be different, the results are expected to complement each other.

D. Pedestrian Tracking with Smart Cameras

Thales R&T uses a smart camera application to demonstrate the SoCKET flow. This application aims to monitor and track pedestrians in public areas by using well known AdaBoost algorithms from machine learning domain[27].

The SoCKET flow is applied in two main steps : the first one aims to explore software and hardware partitioning at TLM level by using tools to describe both SoC components and application building blocks. The second step is in charge of platform assembly at RTL level by mixing standard vendor IP and dedicated IP generated with HLS tools. To generate these IPs, HLS tools use application building blocks selected during HW/SW mapping.

The flow raises the developer team productivity thanks to rapid solution exploration by giving metrics to evaluate power consumption and number of gates but also helps to maintain requirements traceability.

V. Conclusion

Motivated by the "seamless" design flow imperative, and by the variety of industrial requirements (Space Qualification and Avionics Certification), the standards-based methods and tools involved in the flow of section III are being enhanced in various directions e.g.,

- to better characterize the major features to be modeled at the highest abstraction levels to ease qualification and certification,
- to alleviate the main restriction of High Level Synthesis i.e., the inefficiency to deal with control and data flow oriented applications.

Assertion-Based Verification also has to evolve towards giving the possibility to establish the relation and traceability between TLM and RTL properties.

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] "TLM-2.0 Language Reference Manual [online]," http://www.systemc.org/downloads/standards.

[6] http://www.soclib.fr.

[7] http://www.minalogic.com/en/posters/OpenTLM_eng-web.pdf.

[8] http://www.twins-itea.org.

[9] http://www.spices-itea.org.

[10] http://www.topcased.org.

[11] http://www.teisa.unican.es/gim/en/AADS2.php.

[12] J. Langheim, B. Guegan, L. Maillet-Contoz, K. Maaziz, G. Zeppa, S. Boutin, H. Aboutaleb, F. Philippot, and P. David, "System architecture, tools and modeling for Safety critical automotive applications the R&D Project SASHA," in *ERTS 2010*, May 2010.

[13] "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.

[14] "IEEE Standard SystemC Language Reference Manual, [Online]," http://standards.ieee.org/getieee/1666/download/1666-2005.pdf, December 2005.

[15] *IEEE Std 1850-2005, IEEE Standard for Property Specification Language (PSL)*. IEEE, 2005.

[16] 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.

[17] R. Ratsiambahotra, H. Cassé, and P. Sainrat, "A Versatile Generator of Instruction Set Simulators and Disassemblers," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2009)*. IEEE, Jul. 2009, pp. 65–72.

[18] S. Basu and R. Moona, "High Level Synthesis from Sim-nML Processor Models," in *VLSI 2003*, Jan. 2003, pp. 255–260.

[19] P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuits*. Springer, 2008.

[20] P. Coussy and A. Takach, *Special Issue on High-Level Synthesis, IEEE Design and Test of Computers*. IEEE Computer Society, 2009, vol. 25.

[21] http://lab-sticc.fr/www-gaut.

[22] H. Foster, "Applied Assertion-Based Verification: An Industry Perspective," *Foundations and Trends in Electronic Design Automation*, vol. 3, no. 1, January 2009.

[23] 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.

[24] L. Pierre and L. Ferro, "Enhancing the Assertion-Based Verification of TLM Designs with Reentrancy," in *Proc. MEMOCODE'10*, July 2010.

[25] 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.

[26] http://www.gnu.org/software/gdb/.

[27] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.