EDA tools and methods required for formal system level representation

By Mario Diaz Nava, Laurent Maillet-Contoz STMicroelectronics and Adam Morawiec, ECSICorporation

Nowadays the electronic systems industry observes an ever increasing customer demand for innovative products integrating multiple heterogeneous functions. Such systems become more and more complex but can nevertheless be implemented due to the advances in nanometer technology enabling the integration of hundreds of millions of transistors in a single chip. However, the system and microelectronic companies, in order to remain competitive or in critical cases to survive, need to acquire and continuously expand their design capabilities and dispose of manifold expertise to design and integrate these complex systems – and this at the pressure of lowest possible cost in an extremely limited time frame.

In this context, companies mainly differentiate in their design capabilities based on designs tools, methods, standards and flows to provide unique efficient design services and competitive products.

Systems-on-Chip (SoCs) are contributing to the implementation of systems with ever increasing complexity. It is now commonly understood that traditional design techniques are no longer suited, and there is an obvious need for new design flows starting at system level.

System Level Design Challenges

As this is well known now, systems become more and more complex, due to several factors:

» Increase of the number of transistors integrated on one chip (Moore's Law).
» Increase of the heterogeneity of components, by the integration of RF, analog, digital blocks, and involving multi-processor systems in charge to run huge software stacks, co-operating to control hardware blocks, but also offering significant parts of the algorithmic part of the system.
» Difficulties for the integration of subsystems coming from different providers (internal first, but also third parties).
» Reuse of hardware and software components in different contexts.
» Testing the system implies (re)testing of all subsystems involved.

This creates new challenges in term of design flows, to master the complexity, increase the reliability of the system, while addressing time to market constraints as well. Whereas the design at RTL is now well understood and stable in terms of design processes, System Level Design (SDL) is still in its infancy. The techniques associated to this level provide lots of advantages, but are at the moment limited to experts, who have developed their own, and usually in-house expertise, methods and tools. SDL techniques still have to mature to be deployed widely in the industry by targeting several objectives:

» Moving one step forward in the definition of methods and tools to capture the overall system functionality, and provide efficient and proven means to refine the system representation.
» Offering formal methods to assess the validity of system properties at different levels of abstraction, and to provide means to check that refinement of the system has not introduced any issue, if the refinement itself can not be proven as correct-by-construction.
» Improving the high level synthesis methods to provide effective and general purpose tools, able to deal with regular C/C++ code without limiting themselves to a very constrained subset of these languages.
» Completing the standardization effort, in some areas standardization process has been already carried out. For example, the SPIRIT consortium defines the design and IP exchange formats, or the OSCI consortium defines standard memory-mapped bus APIs. Other areas need to be standardized addressing higher levels of abstraction to capture the overall functionality of the entire system, before hardware/software partitioning.
» Creating, as an ultimate goal, a truly interoperable ecosystem of tools, models and related standards to support a component-of-the-shelf approach, valid both for software and hardware components (RF, analogue and digital) in a unified way.

System Level Design methods will only be adopted with major changes in the culture and organization of semiconductor companies. Core competencies will move from purely hardware experts, toward software skilled engineers, and people able to capture the overall complexity of the mixed hardware/software systems. Other competencies to define, design and optimize system architectures or to analyze overall system performances will gain more and more importance in the entire design process.

One important consequence will also be the dramatic change in term of project management practices, to shift from allocating software resources late in the project to massively investing in system and software resources in parallel from the very beginning of the project.

Abstraction levels for system modeling

People have worked for years to define appropriate levels of abstractions, modeling languages and tools to shift from one level to another. It is now widely accepted that one can define the following levels (see figure 1.07).

System functionality

Capture of the overall system functionality. There is so far no notion of hardware or software architecture, only functional blocks are connected together to provide the functionality. This level is usually described as a document, written in natural language, sometimes also accompanied with some C or Matlab code.

Subsystem level

The system is decomposed into several subsystems. First system architecture is defined. The functionality is modeled at the subsystem level. Still some open options remain between hardware or software. From this level, Transaction Level Models (TLM) might be implemented. They might be black boxes representing the overall functionality of the block (regardless of the hardware or software final implementation), or white boxes that effectively execute the real software on top of a processor model, in conjunction with models of the hardware components of the subsystem.

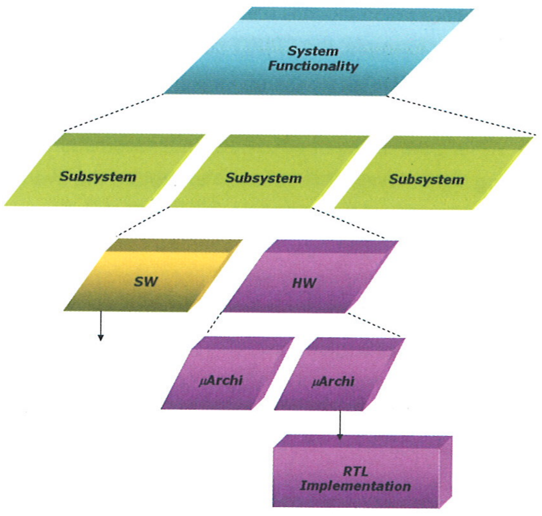

Fig. 1.07: System components and its higher levels of representation.

Component level

Hardware blocks are refined towards Register Transfer Level (RTL), where the traditional design methods are applied.

A continuous path from very early system specification down to implementation is today a dream. Several flows are therefore needed to bridge the gap between these levels. This causes significant risks of discrepancies between the levels, leading to broken silicon. As of today, there is a certain gap between tools and methods used before hardware/software partitioning, dealing with system functionality, and those used after.

Two categories of complementary approaches are used for different activities, during different project phases:

Modeling of non functional attributes of the system:

- Usually developed very early in the design cycle, before or just after hardware/software partitioning.
- Define profiles/scenarios/use cases to be investigated (best, worst, average, peak, etc.).
- Suited for architecture analysis (power, latency, bandwidth analysis, etc.).

Architectural model of the system:

- Developed after hardware/software partitioning.
- Transaction Level models, suited for functional verification and firmware/software development.

Architectural models may themselves be refined from purely untimed, to approximately timed, through loosely timed models. As a complement, high level synthesis techniques offer support to move from a C based model to an RTL model. At the moment, these techniques still impose significant constraints on the form of the input code.

The approaches above are sometimes coupled to create hybrid platforms, where the functional model drives the non-functional part of the simulation.

Different approaches for "formal"system representation

Virtual prototypes are now in deployment to address needs for pre-silicon software development, and early functional verification. This raises new challenges to ensure consistency between the evolutions of the specification, evolutions of the virtual platform integrating various IP models, and evolutions of the embedded software.

This calls for getting new tools and methods for efficient Hardware-Software co-design. In particular, the synchronization of the various views of the model (functional, architectural, implementation) and the reusability of them in different contexts is a key challenge to be shortly addressed. Characterization of the views of a model is now required, to get a systematic understanding of: the programming interfaces of the model, the list of supported features (and the associated level of completeness and test) and the known limitations and issues.

Another challenge appears to be able to globally capture and maintain the representation of the system functionality. Traditional techniques, usually using specifications written in natural languages are no more suited. The ever growing complexity of the systems makes the document difficult to keep up-to-date, and is subject to different interpretation by the various stakeholders. One step forward has been the availability of TLM models that provide a de facto golden model shared by different activities, and IP-XACT representations of the interfaces that can be extracted from a textual specification. Still, they can not be easily coupled with system level use cases that are captured far before in the design cycle.

Several approaches to capture system functionality and requirements are currently investigated. Among them, we can list:

» Top/down approaches, aiming at breaking the complexity of the system under development into subsystems.
» Bottom/up approach, by integrating components to create complex system functionality. This raises new challenges, by considering SoCs as Components off the Shelf (COTS), which must adhere to well identified interfaces, and offer proper functional contracts. This requires expressing properties at the component level, and to provide mechanisms to guarantee that they are still valid once the components are integrated in a complex system.
» Techniques coming from the software engineering community are also considered. In particular, one observes these days some convergence between Model Driven Engineering (MDE) and design practices from hardware community. MDE is seen as a way to capture system functionality and requirements, and provide support to derive software and hardware parts of the system. From a project perspective, it helps in keeping consistency between capture of high level, early requirements and implementation, tests and validation of the system. System houses as Airbus, ASTRIUM, or Thales are very involved in this area. On academic side, labs as KU Leuven, Eindhoven University of Technology, University of Milan, INRIA, or CEA-LIST have key competencies to address these topics, whereas several CAD vendors like CoFluent Design are also in the picture.
» Graphic-based system requirements and system function representation enabling to raise abstraction level on which the design process is initiated and making possible to express system functionality in the context of its environment (e.g. use cases).
» Formal system specification and refinement methods enabling to define the system starting from the set of requirements and refining it to representations from which software and hardware components can be derived with state-of-the art methods (RTL implementation and SW compilation)

Identification of key system properties and verification that they are still valid at each level of abstraction are today a promising research topic. Formal methods though are not mature enough to be applied at the system level, for several fundamental reasons: input languages are general purpose, no assertions or property support, and system-wide approaches create state explosion, calling for more modular concepts.

Conclusions

European competence lies in system design, a comprehensive approach how to specify, model and implement complex systems. This competence should be supported by advanced design methods, appropriate specification formalisms, design data representations and efficient tools. Moreover, these methods and representations should be supported wherever appropriate by standards to enable to leverage from the commonly accepted rather than specific and proprietary solutions. All these elements brought together will form an ecosystem in which the systems can be designed and validated in the most effective way and all involved parties in the product development value chain will find their clear market positioning.

Author & Cont@ct:
Mario Diaz Nava is a STMicroelectronics-Grenoble R&D Cooperative Programs Manager. He has worked at STMicroelectronics for 18 years and has more than 25 years of experience in the system architecture and design of communication circuits and design methodologies. He has a PhD in Computer Science from National Institute Polytechnic of Grenoble and can be reached at mario.diaznava@st.com.

Laurent Maillet-Contoz is a CAD Manager at STMicroelectronics, responsible for the transaction-level modeling (TLM) activity in the System Platforms Group. After a PhD thesis on Hardware/Software CoDesign at the University of Montpellier, France, he has been involved in the definition of the TLM methodology within ST since 2000, and has contributed to the definition of the Open SystemC Initiative's (OSCI) TLM standard. His main interests are system modeling and simulation, and formal methods for SoC verification. He can be reached at laurent.maillet-contoz@st.com.

Adam Morawiec received his MSc degree in electronic system design in 1993 from the Silesian Technical University in Gliwice, Poland and his DEA (Diplome d'Etudes Approfondies) in 1996 and PhD in 2000 in Microelectronics at the TIMA Laboratory / Université Joseph Fourier, Grenoble, France in the domain of verification and simulation performance methods. He works for ECSI (initially part time since 1994 and then full time since 2000) in the R&D project development and project management in the domain of system design methods and standards, in setting up industry and research consortia, in organization of advanced training and workshop in system design area. Since September 2005 he is the director of ECSI. He can be reached at Adam.Morawiec@ecsi.org.