HOW STANDARDS WILL ENABLE HARDWARE/SOFTWARE CO-DESIGN

Mark Genoe (Chair – Alcatel); Chris Lennard (Cadence) ; Joachim Kunkel (Synopsys) ;
Brian Bailey (Mentor Graphics); Gjalt de Jong (Alcatel) ; Grant Martin (Cadence);
Kamal Hashmi (Fujitsu – ICL), Shay Ben-Chorin (National); Anssi Haverinnen (Nokia)

Members of the SLD DWG of the VSI Alliance

Context

Reuse of Intellectual Property (IP), or Virtual Components (VCs), from different internal and external sources in Systems-on-Chip, allows companies to focus the R&D to their own core competencies, and to effectively use other companies' specialized expertise for other parts. Such a model can only work if there the microelectronics system industry worldwide can establish an unified vision with a set of open technical standards. This view is quite similar to design practices at the board level today. However, the complexities of future systems-on-chips will largely exceed the ones that we currently know at a board. Moreover, prototypes require costly silicon runs, less signals are visible for probing, less debugging facilities are available, and it will be much more difficult to analyze possible problems when combining several components. Therefore, these virtual components need specific models, to analyse, compare, debug and validate complete system chips and all their interfaces before processing the real silicon, but already starting in the early design phases. This is what is meant today with 'Virtual Prototyping'.

Virtual prototyping of complete hardware (HW) - software (SW) systems is really key, but need to be raised to much higher levels of abstraction than today's design practices, which are usually at the level of synthesizable RTL for custom hardware or Instruction Set Simulator (ISS) for programmable core processors. This shift will result in totally new system level design environments to capture requirements, to specify functionality and architectures, to explore different mappings and schedulings, to select and encapsulate reusable Virtual Components. To be used at 'system level', Virtual Components require several abstract models, expressing e.g. the performance, functionality or cycle-true behaviour.

It is exactly the goal of the System Level Design (SLD) Development Working Group (DWG) of the VSI Alliance to specify standard interfaces, standard data formats and standard methods that will help system designers to explore, debug and verify complex system architectures consisting of several Virtual Components by virtual prototyping at multiple levels of abstraction.

Discussion topics

The discussion should be organized around the main achievements of the SLD DWG of VSIA. These include following areas in the domain of HW/SW co-design and Virtual Prototyping:

Standard nomenclature and VC model taxonomy: Progress to accelerate the encapsulation of Virtual Components (VC) in co-design has hit roadblocks because of a wide diversity of model terminology in use among VC providers, VC integrators, designers, semiconductor companies, system houses and EDA companies. First experiences of integrating third-party components in HW/SW systems by several system companies have shown that different terminology has already created a lot of confusion among the participants. Some organizations use many common modeling terms with divergent meanings, while others use different words to describe the same type of models. While this confusion persists, and the electronics community lacks a common language, different teams will be unable to effectively communicate and share models. Therefore, the SLD DWG has undertaken an effort to develop a nomenclature and modeling taxonomy, which will become a common language to describe models and their attributes for the VSI membership and the electronics design community at large. It contains a classification of

system -, architectural -, hardware - and software models, and is publicly available as standard reference document. It basically modified and augmented previously defined terminology sets, broadened parochial definitions, distinguished overlapping definitions, equated close synonyms, removed inapplicable terms, added new terms, clarified poorly defined or misunderstood ones, and suggested new wording as replacements or synonyms to outdated ones. When appropriate existing definitions were lacking, the SLD DWG created them. Further evolution in design practices for VC integration into System Chips will identify the critical model types. Minimizing the number of models will reduce the effort required to produce a complete design package for a VC.

Standard Interface Behaviour Description: The design methodology promoted by the SLD DWG is based on a clear separation of the VC functionality and the VC interface. Therefore, the DWG is establishing a well-defined hierarchical and multi-level standard description of VC interfaces, covering all abstractions from high level transactions down to detailed timed component protocols, implemented in hardware and/or software. A clean separation of interface properties from VC functionality and behavior, and a clear linking of interface abstractions at the system level, is a significant step towards the achievement of such a goal. A technique for specifying such interfaces will improve VC understanding and utilization. It will reduce the time required to understand behaviors and interfaces correctly. Gaining a faster understanding of VC operational principles allows a system architect to explore many more options before committing to the design phase. This faster VC interpretation and model-integration gives more comfort in the exploration of non-legacy architectures and so will open co-designs to the third-party market. Furthermore, a complete definition of the interface abstraction hierarchy allows designers, architects and SW authors to work within their preferred area of expertise (e.g. embedded-software, RTL, etc.), but still gives them the ability to effectively communicate with the different levels (e.g. unified test-benches/test-results can be applied to any view of the design.). In addition, the standard Interface Behavior description will improve VC model supply and generation, VC integration in HW/SW co-design, and last but not least, VC protection.

Standards for Behavioural Modeling of Virtual Prototypes: For the descriptions of the VC functionality itself, the SLD DWG is shooting for a standard library of data types that are commonly used in behavioral models for virtual prototyping (e.g. in C or C++). Today, system companies are using multiple libraries, even within the same company. Third party VC vendors are offering models with specific libraries, not compliant for other environments. Different syntax and/or semantics make true exchange of models impossible. When high-level models used in virtual prototyping of HW/SW co-design systems can apply all the same data type libraries, the interoperability of these models will be largely enhanced, and co-design will become much more user-friendly.

Performance Modeling Standard for HW/SW systems: This exploration began as part of a recognition that systems containing VCs require high level modeling techniques in order to efficiently evaluate the system performance of interconnected VCs (microprocessors, DSPs, memories, caches, buses, RTOS, etc). The specific intent of this specification is to describe the basic functional and interface requirements of system-level performance models for the most common types of Virtual Components. Performance Models are often the most abstract models of Virtual Components that are used during system design. They describe the system task as well as the resources together with the abstract and physical communication channels. Each of these elements can be modeled in terms of its basic processing and communication capabilities, such as e.g. the rate of processing, the latency of operations, etc. In contrast to functional models, abstract performance models do not compute the results of the operations. System-Level Performance models are used to explore different alternative mappings of the system tasks on selected resource architectures, and to perform trade-off analysis among different hardware and software architectures. It allows the system designer to obtain a first measurement of the quality of the design, to check if the proposed architecture will satisfy the overall performance requirements and meet the constraints, and to identify possible bottlenecks or over-sized parts.