An Introduction to SLDL and Rosetta

(ASP−DAC 2000 extended abstract)

Steven E. Schulz, P.E.
Advanced EDA Technologist / Senior Member Technical Staff
CAD Flow, Methodology, and Architecture
WW ASIC Division / Texas Instruments USA
email: ses@ti.com

The Rosetta Stone (permanently displayed at the British Museum of Art) has played a pivotal role in enabling scientists to translate across a number of widely varying languages used around the world throughout our history. Through this mapping process, linguists have been able to more fully understand the semantic meaning embodied in the written word of our culture. Similarly, the Rosetta language (as we will refer to it in this text) represents a modern−day effort to map across semantic domains within the electronics−centric systems engineering world, motivated by the trend towards System−Level Integration, or System−on−a−Chip (SoC). Rosetta plays a key role in the System Level Design Language (SLDL) industry standards initiative.

Rosetta has borrowed many concepts from numerous existing system−level research efforts across Europe and the U.S., including declarative semantics, domain theories, partial specifications, and parametric abstraction. However, what is new in Rosetta is the integration of multiple domain theories into a common semantic framework, rather than attempting to force the system description into a singular semantic. Why is this useful? Because using a "least common denominator" (LCD) diluted approach to describing aspects of a system is necessarily lossy −− with loss of conciseness, expressive power, and analytical power. A simple example would be to mix analog and digital "views" of a complex IC into a single LCD semantic, say SPICE. Not only would the merged SPICE syntax be difficult to express and difficult to understand for digital functionality, analysis through SPICE simulation would have prohibitively long runtimes. Yet even worse, much of the semantic meaning of the digital abstraction would be lost, so that a mux, adder, or ALU would just be treated as more transistor voltage waveforms.

A system is composed of many views −− digital function, analog behavior, timing (e.g. delay, latency, queuing), bandwidth or throughput, power consumption, cost, etc. Engineers usually prefer to manage this complexity by focusing on specific modeling of one view at a time, then studying this view in relationship to other views. In Rosetta, a model of a specific view into a system is called a "facet". Modeling facets become more important when multiple facets interact in some (possibly complex) way. Rosetta permits multiple facet models to be integrated together −− both in terms of mixing portions of the system together, as well as mixing multiple perspectives of the system together. This enables the engineer to consider more of the tradeoffs involved when making a design decision that will directly impact the final product.

Of course, robust and concise modeling does little good if the models cannot be parsed and analyzed by EDA tools. Rosetta, like the rest of SLDL, is based on a semantic foundation intended to enable the use of formal methods to help ensure thorough coverage, and compositional reasoning techniques for scalability to very large systems. Some of these methods are well−proven, while others will require much more learning to exploit in commercial applications. As part of the DARPA SLDL contract, a Java−based parser for Rosetta is now in the final stages of development. This parser will enable EDA tool suppliers to actually prototype with Rosetta / SLDL as a path towards commercially robust solutions over the next 1−2 years.

Some highlights of the Rosetta syntax:

- Pre−defined facet domains supporting semantic definitions for frequently−used views
- Facets may be user−defined, for highly flexible modeling of concerns
- Physical variables define externally visible component state, while logical variables define local values (visibility may be exported to other facets)

- Terms in a facet define behavior using parameters and variables; the semantics of a term are defined by its domain
- Facets are typed; pre−defined types support most common uses (numeric, records, sets, sequences, arrays, functions, vectors, etc.)
- Assertion of a defined facet makes the claim that the facet is "True" in the system
- Complex behavior is modeled by combining facets (AND, OR, NOT, and IMPLIES)
- Facets may be combined across domains in the exact same way
- Structure is created by including and re−labeling facets within components
- Requirements are defined by using one or more requirements domains (logic, state−based, discrete time, continuous time)
- Constraints are defined by using a constraints domain (many are possible: heat, power, clockspeed, etc.); these will be defined through the System−level DCDL effort
- The complete system thus consists of components and facets, where the systrem model describes:
  - Assumptions on the external environment
  - Definitions for the system
  - Verifications describing and justifying system behavior
- Logic for a complete system model:
  (Assumptions and Definitions) implies (Verifications)

While the syntax of Rosetta does borrow on several proven constructs found through experience to be useful in VHDL, it is entirely separate from any HDL. Rosetta / SLDL is a systems language, driven to be complementary with both Verilog and VHDL, as well as familiar software programming languages (C, C++, Java). Rosetta does not compete with any of these, since no other language describes declarative design constraints at the system level.

Rosetta supports the Phase I SLDL goals for a high−level, multi−domain constraints language capability. Rosetta is complemented by the Design Constraints Description Language (DCDL) effort, supported through OVI, VSIA, and VI. DCDL will provide the constraints content semantics, while Rosetta enables an integrated semantic framework for applying these constraints to different views of a system and analyzing the results.

So, what can industry expect for Phase II of SLDL? Much of the work is already beginning, as the semantic foundation is being laid for describing diverse views of behavior and architecture in a constraint−driven environment. The formally−based semantic model will serve as a "backplane" for allowing all compliant system−level languages to communicate with each other, whether they already exist today or have yet to be defined to meet future needs. It is imperative that the electronics industry worldwide avoid a virtual "Tower of Babel" in the system design arena. Much as the PC industry has benefited tremendously from alignment on the PCI bus standard, enabling new "plug−in" cards that can communicate and coordinate activities, we too should agree to leverage a common semantic backplane enabling various modeling domains to work in harmony with each other. This level of interoperability is not easy to achieve, but is critically important for the electronics industry to reap the full rewards of silicon advancement through greatly enhanced design capability and productivity.

SLDL has been greatly supported by numerous industry consortia, and is now an integral part of VHDL International and it’s vision to extend the capabilties and value of HDL−based design. Critical support for SLDL is also provided courtesy of ECSI, VSIA, DARPA, MARCO, GSRC, and OVI.

More information on SLDL, including requirements documents, workshop minutes and presentations, industrial driving design examples, Rosetta white papers, summary slides, and organizational structure can be found at <http://www.inmet.com/SLDL>.