Brief Announcement: Extracting Models from Design Documents with Mapster

David James
Palo Alto

Tim Leonard
Intel
tim.leonard@charter.net

John O'Leary
Intel
john.w.oleary@intel.com

Murali Talupur
Intel
murali.talupur@intel.com

Mark R. Tuttle
Intel
tuttle@acm.org

Abstract: We cannot apply PODC methodologies to industrial designs without formal models of the designs. Formal models are usually hard to find. We have built a tool that extracts formal models directly from design documents.

Categories and Subject Descriptors: B.6.3 [**Logic Design**]: Design Aids—*Verification*
General Terms: Algorithms, Design, Verification
Keywords: Industrial Design, Models, Verification

Protocol verification consists of three easy steps: find a protocol, model the protocol, and verify the protocol. Go ahead, laugh at the joke, but it is not much of an exaggeration to say that getting the protocol model has traditionally been the hardest of the three steps at companies like Digital, Compaq, HP, and Intel. We could talk for an hour on the reasons, but the bottom line is that the tools we build assume the existence of a model, and we think very little about where these models come from, yet our tools will never gain widespread use if we don't address the problem of building the model of an industrial design.

Industry often uses the "Superman approach" to model building. (Superman is a fictional comic book superhero once popular in America.) In this approach, a single verification expert joins a group (usually late), spends months learning the protocol, building a model, changing the model as the design changes, and only then gets to do any verification. This is not the right approach. For one thing, it doesn't scale well (there aren't many supermen), it burns out the supermen we have (leaving even fewer supermen), and it never gets past the "technology demonstration" phase ("That was great, Superman, you wanna do it for me again?").

While not building models, designers are producing a number of design artifacts with significant technical content: microarchitecture specification documents containing state transition tables, block diagrams, pipeline diagrams, timing diagrams, message flows, etc. We should be able to use this information to build the formal model we need from what the designers are already willing to produce. And once the model is built, it could be highly attractive to the designers to include it in their documents, leading to more precise, less ambiguous design documents.

Copyright is held by the author/owner(s).
PODC'08, August 18–21, 2008, Toronto, Ontario, Canada.
ACM 978-1-59593-989-0/08/08.

This is our dream:

- A front-end that can extract from a specification document the transition tables, block diagrams, pipeline diagrams, etc, and build a formal protocol model.
- A back-end that can take this protocol model and produce input to formal verification tools, and produce reference models for traditional simulation.

What we have done is build a tool that mechanically extracts tabular information from a design document — state transition tables, state definitions, type definitions, etc — and builds a mathematical protocol model, and from that model generates a Murphi model [3] for model checking. We have applied this approach to a cache coherence protocol that is unusual (caches are maintained on doubly-linked lists) and complex (37 cache states) called SIMPL.

We are not the first to do table-based design [1, 2, 4], but it is one thing to ask designers to write tables in a style intended for formal analysis, and it is another to take tables as they are written by the designers and make formal sense of them. We require no language or GUI for building the model, and impose no rigid philosophy on how protocols should be described. Our aim is to take the protocol description in a form that makes most sense to the designer, and to make formal sense of that. Our approach involves two key ideas: a way to assign semantics to table columns, and a way to rewrite the resulting model when our method of assigning semantics to columns fails to express the intended semantics exactly.

[1] M. Azimi, C.-T. Chou, A. Kumar, V. W. Lee, P. K. Mannava, and S. Park. Experience with applying formal methods to protocol specification and system architecture. *Formal Methods in System Design*, 22(2):109–116, Mar. 2003.

[2] C. Heitmeyer, M. Archer, R. Bharadwaj, and R. Jeffords. Tools for constructing requirements specifications: The SCR toolset at the age of ten. *International Journal of Computer Systems Science and Engineering*, 20(1):19–35, Jan. 2005.

[3] C. N. Ip and D. L. Dill. Better verification through symmetry. In *Computer Hardware Description Languages and their Applications*, pages 97–111, Apr. 1993.

[4] D. Parnas. Inspection of safety-critical software using program function tables. In *Proceedings of the IFIP 13th World Computer Congress*, pages 270–277, Aug. 1994.