A Role for Theorem Proving in Multi-Processor Design

Albert J. Camilleri

Hewlett-Packard Company, 11000 Wolfe Rd, Cupertino, CA 94179
ac@cup.hp.com

1 Overview

The role of theorem proving in industry is far from obvious. Especially in the absence of critical requirements such as safety or security, many of the potential advantages it offers, such as the power of expressibility and thorough analysis, can be quickly outweighed by the disadvantages of long interaction times and demands for high levels of expertise.

At Hewlett-Packard we have been using theorem proving as part of our verification arsenal for several years [1], constantly searching for new ways of fitting it into our verification methodology with the goal of improving design quality without adversely affecting time to market. In [1] we describe a *lightweight* style of specification to facilitate quick analysis and proof. We advocate using theorem proving not as an end in itself, but as a means to an end, focussing on analysis, and feedback to designers, rather than on delivering proof as an ironclad guarantee. We still favour this approach, and we have continued to build on it and the specification style described.

One way of addressing some of the shortcomings of theorem proving is to use it alongside other verification techniques, invoking the true power of the theorem prover only where necessary and using alternative techniques where they are more efficient. In [3], a hybrid approach of theorem proving and model checking is used to verify *liveness* for a commercial, symmetric multi-processor. Liveness properties of Runway based multiprocessors [2] are analysed by means of theorem proving because of the large number of queues and dependencies involved in the model, while fairness, inherently a property concerning finite state machine descriptions, is efficiently addressed using model checking.

2 Fitting Theorem Proving into the Design Cycle

Currently, at the Cupertino Systems Laboratory of Hewlett-Packard, theorem proving is being used selectively to verify various aspects of multi processor design. The commercial schedules and pressures do not permit a full scale, hierarchical verification effort using theorem proving. Indeed, simulation technology is still considered the most efficient verification technique for most parts of the designs, so

careful selection of problems is done to decide which aspects need to be formally checked, and of those, which require theorem proving.

There are currently no sweeping generalizations that can be made to define a general purpose methodology advocating where best to apply theorem proving. Some rules of thumb can be provided, however, regarding criteria to look for when considering theorem proving. These are best presented by means of a simplified view of a typical VLSI design cycle, as illustrated in Figure 1.

Figure 1. A Simplified View of the VLSI, Pre-silicon Design Cycle.

The diagram depicts the definition of computer *architecture* as the first stage in the design cycle. Architectures are intended to last a relatively long time (over 10 years, say) so they are often intentionally loosely defined and partially specified. They are aimed at providing a framework for the definition of several products as defined by their *micro-architectures*; the next phase in the design cycle. Micro-architectures define the chip sets, topology, protocols and requirements for a set of products to ensure the correct functionality of the system with respect to the architecture. For example, protocols for coherency, flow-control and ordering would be included in a micro-architecture.

Theorem proving plays a natural role during these early stages of the design cycle because of its ability to deal with partial specifications, its expressiveness at very high levels of abstraction and its ability to handle system-level (multiple-chip) complexity

which often causes state explosion for model checking or infeasibly long simulation runs.

The third phase in the design cycle shown above involves the definition of the individual chips (such as microprocessors, I/O adapters, memory interface chips, etc.). They are defined and implemented once the micro-architecture is reasonably stable, and are typically done by means of synthesizable hardware description languages (often referred to as the RTL, or register-transfer level).

In an industrial environment, quick and timely feedback to designers is of the essence. The use of theorem provers at the chip implementation level can currently be very time consuming because of the amount of low level detail and lack of abstraction. Therefore, serious consideration should be given here to other technologies such as model checkers, which we have found to work very well at this level, especially when finite state machine representations can be extracted directly from the HDL, and checked automatically.

At the higher levels of abstraction of the architecture and micro-architecture, even though theorem proving can play a natural role, interaction time with theorem provers can still remain a problem. To overcome this problem we have tended to use at least three techniques:

1. Use of a lightweight specification style as advocated in [1].
2. When appropriate, doing pen and paper proof for quick analysis to avoid the overhead of a theorem prover [4].
3. Use of theorem proving as a 'glue' for other methods, e.g. using the theorem prover to verify the high-level interaction of multiple components and to generate invariants (or implementation requirements) for verification of the separate components by model checking or simulation.

Of course, with more automation and efficient decision procedures (tailored for VLSI verification), theorem provers will become more straightforward to use and will require less methodological care.

3 Conclusions

Overall, at this stage, theorem provers are still a long way from becoming a standard design tool. They require expertise and good judgment both on how to use them as well as where. They have, however, unique abilities for verification not present in other verification techniques. For now, they remain a tool for experts, but if carefully deployed, they can be used to increase confidence in today's designs.

References

1. Bainbridge S., Camilleri A., Fleming R., 'Theorem Proving as an Industrial Tool for System Level Design', in *Theorem Provers in Circuit Design*, Stavridou V., Melham T., Boute R. (eds.), North-Holland, 1992, pp. 253-274, Proceedings of the IFIP TC10/WG 10.2 International Conference, Nijmegen, 22-24 June 1992.
2. Bryg W., Chan K., Fiduccia N., 'A High-Performance, Low-Cost Multiprocessor Bus for Workstations and Midrange Servers', *Hewlett-Packard Journal*, 47 (1), February 1996.
3. Camilleri, A., 'A Hybrid Approach to Verifying Liveness in a Symmetric Multi-Processor', in *Theorem Proving in Higher Order Logics*, Gunter E., Felty A. (eds.), Springer-Verlag, 1997, pp. 49-69, Proceedings of the 10th International Conference, New Jersey, 19-22 August 1997.
4. Corella F., Shaw R., Zhang C., 'A formal proof of absence of deadlock for any acyclic network of PCI buses', in *Computer Hardware Description Languages and their Applications*, Chapman & Hall, 1997, Proceedings of the 13th International Conference.