Evolution and Impact of a Large Industrial Proof

Robert B. Jones
Strategic CAD Labs
Intel Corporation
Hillsboro, OR, USA
robert.b.jones@intel.com

Noppanunt Utamaphethai
Low Power Technologies Group
Intel Corporation
Austin, Texas, USA
noppanunt.utamaphethai@intel.com

The Intel® IA-32 instruction-set architecture includes several hundred opcodes of varying length [3]. Certain instructions have optional bytes that specify register modes, memory modes, and address offsets. Instructions vary in length from one to twelve bytes. An additional complication arises from *prefix bytes* that can change the semantics and even length of the subsequent instruction.

Decoding the IA-32 instruction-set in a high-frequency pipeline is challenging. Recent processor implementations divide the decoding process into separate activities; the first is an *instruction-length decoder* (ILD) that marks instruction boundaries.

This talk at DCC 2006 will overview the evolution and impact of a formal proof about the ILD. The proof has evolved as it has been applied to multiple microprocessor designs over almost a decade. The proof has detected bugs in almost every design it has been applied to, and has largely replaced simulation-based validation of ILD functionality on some projects. This talk will consider the evolution of the proof in the face of hardware changes and additions to the instruction set. Technical details of an early version of the proof have been published previously [1, 2, 4].

We have learned several important lessons about specification and verification during the evolution of the ILD proof.

- Formal specifications should avoid implementation details when possible. Current ILD implementations are significantly different from the first ILD pipeline that was verified. Writing the formal specification to reason about the ILD as a "black box" has been an important aspect of applying it on multiple hardware designs.

- Proofs need to be amenable to variations in proof complexity induced by design changes. We found that certain hardware changes made the BDDs underlying the proof simpler. On the other hand, changes usually made the proof BDDs more complex. Managing this complexity was one of the main challenges as the proof evolved. With the benefit of hindsight, we can see techniques that would have made complexity management easier.

- Certain classes of specifications must be written in an extensible way. It was fortunate that the original ILD specification was extensible. In the years since the original specification was created, multiple features have been added to Intel microprocessors that required new instructions. More recently, the introduction of a 64-bit mode to the Intel architecture resulted in extensive additions to the instruction set–and to its formal specification.

The ILD proof has been very successful and a wide range of bugs have been found over the proof's lifetime. Some might have escaped simulation-based validation, and, as far as we know, the formal proof has not missed any bugs in its targeted area. We will highlight examples of bugs and their underlying causes. As the proof has matured, the ability to apply it early in the design process has proven particularly useful.

References

[1] M. D. Aagaard, R. B. Jones, and C.-J. H. Seger. Combining theorem proving and trajectory evaluation in an industrial environment. In *Design Automation Conference (DAC)*, pages 538–541. ACM Press, June 1998.

[2] M. D. Aagaard, R. B. Jones, and C.-J. H. Seger. Formal verification using parametric representations of Boolean constraints. In *Design Automation Conference (DAC)*, pages 402–407. ACM Press, June 1999.

[3] *IA-32 Intel® Architecture Software Developer's Manual, Volumes 2A and 2B: Instruction Set Reference*. Intel Corporation, September 2005. Document numbers 253666 and 254667. Available at `http://www.intel.com`.

[4] R. B. Jones. *Symbolic Simulation Methods for Industrial Formal Verification*. Kluwer Academic Publishers, 2002.