Guest Editors' Introduction:

CHALLENGES IN PROCESSOR MODELING AND VALIDATION

Pradip Bose
IBM T.J. Watson Research Center

Thomas M. Conte
North Carolina State University

Todd M. Austin
Intel Corporation

• • • Today's methodology for designing state-of-the-art microprocessors involves modeling at various levels of abstraction. In the presynthesis phase, this can range from early-stage, performance-only models to final-stage, detailed register-transfer-level (RTL) models. Hierarchical modeling requires the use of an elaborate validation methodology to ensure inter- and intralevel model integrity. The RTL model, often coded in a hardware description language (for example, Verilog or VHDL), captures the logical behavior of the entire chip, both in terms of function and cycle-by-cycle pipeline flow timing. It is this model that is subjected to simulation-based architectural validation prior to actual tape-out of the processor. The validated RTL specification serves as the source reference model for synthesizing the gate- and circuit-level processor descriptions.

So what ends up as the final chip (hardware) is really a systematic refinement game played in the domain of software. How do we make all of these presilicon models and abstractions work together? How do we ensure that critical information relevant to a lower level is not lost during the refinement process? How do we ensure consistency across levels of abstraction? How do we manage the escalating cost of simulation and verification at various levels? Are future designs forever destined to be "verification-gated"? (Al-Ashari provides recent verification cost data.[1]) What are the future trends in microarchitecture design that may further complicate (or ease) the modeling and validation bottleneck? These were some of the questions that inspired the three of us to put together this special issue.

Current trends

In introducing the theme articles in *Computer*'s May 1998 performance analysis issue, two of us had focused only on performance modeling.[2] At this high level of abstraction, lead performance and design microarchitects want to define the best microarchitecture implementing the given instruction-set architecture, or ISA. "Best" often implies the highest architectural performance measured in terms of instructions per cycle or its inverse metric, CPI (cycles per instruction). An example of a simulation toolkit that is widely used in academic research today is the SimpleScalar model developed by Burger and Austin.[3]

Increasingly, however, we are witnessing the need to factor in more and more lower-level design constraints into early-stage, high-level modeling and analysis. This trend is ultimately due to the increasing levels of integration afforded by the relentless progress of the underlying semiconductor technology.[4] This progress has resulted in current chip designs that use many millions of transistors and operate at near-gigahertz clock frequencies. At these clock speeds, wiring and interconnect delays become a significant determinant of cycle time. As such, developers must treat careful partitioning and layout of logical blocks as an issue at the highest level of design to avoid later-stage surprises. An example of such impact on high-level design is the emergence of clustered register files in high-frequency microprocessors such as the Compaq Alpha 21264.[5] Due to the relatively large access times of centralized register files, smaller (partitioned) register sets are cross-coupled to feed independent clusters of functional

0272-1732/99/$10.00 © 1999 IEEE

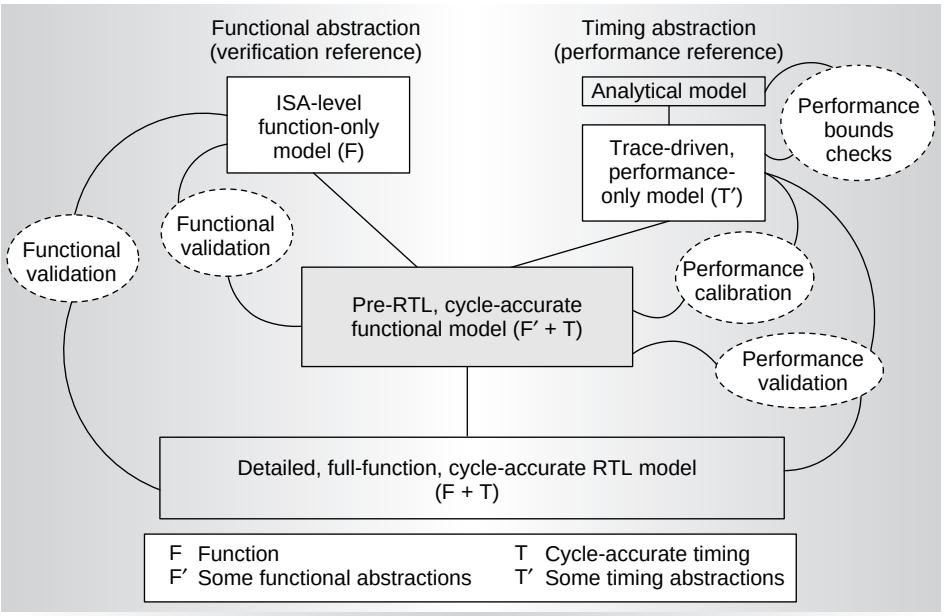

Figure 1. Levels of abstraction in pre-RTL processor modeling: an example.

units. Such decisions may result in a CPI-level performance hit, while protecting the clock frequency target. Architecture-level power estimation and power-driven design methodologies reflect another trend in which performance may need to be curbed (adaptively?) to meet power budgets.

In view of these trends, we need better integration between the modeling and validation methodologies at various levels. Figure 1 shows a hierarchy of levels in the range of interest to us in this theme issue. The highest—the function-only level of abstraction—defines an ISA-level functional simulation model. The performance-only simulation model abstraction is the one available in a trace-driven, cycle-by-cycle simulator of the microarchitecture. This models the microarchitectural implementation of the ISA, but only at the level of pipeline instruction flow timing; functional semantics and data state information are not maintained. Such a simulation model is sometimes referred to as a timer.

At the lowest (and most detailed) level of the hierarchy is the full-blown RTL model. This incorporates full function as well as latch-accurate pipeline flow timing. Depending on the particular project, we may envisage an intermediate, pre-RTL level in which function and performance are combined, but certain refinements of the inter-latch gate-level logic specifications may be abstracted out. Typically, such an intermediate (full-function, behavioral) model would be written in a language like C or C++ (as in the higher-level models). During early estimation, architects may typically use analytical models for setting achievable performance targets or bounds;[2] this may serve as a reference for validating the simulation model.

On the issue of presilicon validation, we limit discussion to only two of the many design dimensions alluded to earlier:

- functional integrity at the architectural level—verifying that the microarchitectural implementation is faithful to the functional semantics of the source ISA; and
- performance integrity in a CPI-centric view—making sure that the initial CPI projections and trade-off decisions are accurate in light of the postsilicon measurements.

Figure 1 indicates the interlayer validation requirements in a typical setting.

In this issue

To address the problems we've mentioned, we provide articles that describe some of the innovative leading-edge technologies in academia and industry. Moudgill et al. describes Turandot, a novel, high-speed, execution-driven (performance-only) simulator that achieves an order of magnitude speedup over prior PowerPC processor timers within IBM. The model is amenable to validation against a pre-RTL reference model by using systematically generated performance test cases. The authors show that their validation methodology allows users to calibrate a model quickly, without losing its innate speed efficiency.

Bechem and colleagues describe an integrated modeling approach that combines per-

formance and function. Here, modeling occurs essentially at the pre-RTL behavioral level (see Figure 1 again). Again, the authors treat the problem of validation in conjunction with developing the modeling strategy. They describe a finite-state machine (FSM)-based method of modeling the control logic; this is used as the reference for generating directed test cases to cover all possible state transitions. The authors demonstrate the power of this approach in terms of generating a small test suite, which covers the FSM transitions fully. Benchmarks and other synthetic test case suites used in standard methodology are much larger but exhibit much poorer coverage.

Arvind and Shen address a key problem in microarchitectural modeling: formal specification. They describe a specification language based on term-rewriting systems. They show the use of such systems to specify the behavior of complex functional execution semantics of modern processors. This formalism is shown to point the way toward the automated synthesis and verification of complex microarchitectures. This example of leading-edge academic research is still too new to be part of an industrial processor development methodology. However, it reflects an increasing trend and need for higher-level formalisms to weed out basic design bugs early.

On the other hand, Hunt and Sawada highlight the promise of formal specification and verification by describing their work in verifying an actual pipelined, superscalar microprocessor. The authors use a different formalism than Arvind and Shen to specify the processor execution semantics. Their work is distinguished by the fact that they were successful in applying the verification methods to a nontrivial, modern processor design. Also, the authors point to the temporal correctness checks obtainable (at the CPI level) when using their method as well as the functional correctness.

In Kunkel et al. we see a case study of a real-life processor/system performance-tuning project portrayed across several generations of IBM's AS/400 product line. This article takes the reader through a post-hardware tuning process in which the layered software architecture supporting the AS/400 hardware had to be tuned and validated to attain target performance.

To cope with complexity, design teams may increasingly be forced to rely on higher-level modeling to weed out early bugs and to avoid late-stage physical design surprises.

Hangal and O'Connor provide a tutorial perspective. They detail a typical industrial modeling and validation methodology related to the picoJava processor designed at Sun Microsystems. The authors describe some newer trends in cosimulation and parallel simulation of the RTL model to tackle the simulation speed and efficiency bottlenecks. They address both functional and performance verification issues.

Modeling issues

To cope with complexity, design teams may increasingly be forced to rely on higher-level modeling to weed out early bugs and to avoid late-stage physical design surprises. Another way of looking at this is to infer that there may be a need to introduce additional layers of abstraction. As shown in Figure 1, the pre-RTL, cycle-accurate functional model abstraction may become a necessary step in most design methodologies. The question, then, is what's the right level of abstraction of this high-level model? If it is too close in net semantics to the final RTL, it may be hard to justify the economics. This is because the net development and validation expense will (likely) not be reduced. Rather, it may increase due to the duplication of the modeling and validation tasks at that low level. This would especially be true if the pre-RTL and RTL models are written in different languages without automated means for intermodel consistency checks. Also, the simulation speed will be too slow to allow viable performance studies. If the pre-RTL abstraction is too high, however, it may not reduce the burden of the RTL-level validation team very much. The detected

functional and performance bugs may be valuable, but too few to avoid the "billions of simulation cycles" syndrome at the RTL level.

Design teams must toil through this methodology transition phase, possibly at the expense of a delayed project or two, before working out the right high-level modeling abstraction. Hangal and O'Connor, for example, talk about an ISA-level functional reference model, which also incorporates a detailed (logical) cache hierarchy model to factor in key performance issues. For that particular project, this intermediate level seems to have proved to be a reasonable choice to support their cosimulation and hierarchical model validation.

The modeling challenges posed by emerging microarchitectures are not fully covered in this theme issue. The Moudgill and Bechem articles touch on the performance model accuracy challenges posed by today's dynamic superscalar processors. However, future trends and their impacts are not covered.

The philosophy behind EPIC (explicitly parallel instruction computing) and VLIW (very long instruction word) processors, for example, is that complexity can be moved out of the microarchitecture and into the compiler wherever possible. Thus, the modeling of EPIC/VLIW processors must put the compiler in the loop although traditional (superscalar) modeling does not.

Speeding up the modeling of such processors poses new challenges. To see the challenges, we need to look inside the compiler. An EPIC compiler necessarily has greater complexity in its code-generation phase. This phase must assemble instructions into parallel groups of independent operations and also manage the hardware resources of the machine. Its complexity is often $O(N^2)$, $O(N^3)$, or in some cases nonpolynomial (that is, worse). Here, N is the number of instructions or operations in the scheduling window. Thus, putting the compiler into the modeling loop can slow down the simulation significantly, despite simpler hardware. A major challenge in such modeling is to determine the equivalent of trace sampling (used in the superscalar world) in the EPIC domain. The use of profile-directed "hot regions" to accelerate EPIC simulation holds some promise. It will require some effort to tie its use back to statistical sampling theory. This remains an open research topic.

The trend toward data-dependent microarchitecture optimizations (for example, value prediction[6]) creates further challenges in the construction of accurate models. Traditional high-level models (such as those based on instruction traces) often lack the fidelity necessary to accurately reproduce the complex nature of these optimizations. For example, in a model with value speculation and partial reexecution recovery, the model must correctly reproduce function on both the correct and incorrect paths of execution. Modeling the correct path gauges the benefits of correct value predictions, while incorrect path modeling measures resource contention due to misspeculation recovery.

Validation issues

Current microprocessor design teams use a combination of simulation-based and formal verification techniques to validate (functionally) the RTL and pre-RTL models. They apply formal verification methods to well-defined combinational parts of the gate- or circuit-level implementation. Also, higher-level formal analysis (for example, Clarke and Kurshan,[7] Arvind and Shen, and Hunt and Sawada in this issue) is of increasing use to reduce the burden of late-stage simulation-based validation. We expect this trend to continue. Nonetheless, pseudorandom test case generation to cover the architectural space is still relied upon as the principal means to identify the design bugs at the RTL or pre-RTL levels. See examples in Kantrowitz and Noack,[8] Aharon et al.,[9] and Hangal and O'Connor in this issue. We also expect the typical verification team to maintain this status quo, at the very least in the coming couple of processor design cycles. To alleviate the simulation speed bottleneck created by increased workload and microarchitecture complexity, more and more workstations will be thrown into the typical simulation farm.

Other innovations in parallel simulation at all levels (Hangal and O'Connor, this issue) will be required. It is apparent, however, (Bechem et al., this issue) that relying on pseudorandom generation techniques alone is going to be increasingly inadequate. Also, relying on manually generated, specialized test cases to cover hard-to-detect corner cases is going to be risky for tomorrow's complex

hardware. We foresee the use of more sophisticated, formal, model-driven test case generation and coverage techniques[10] to replace or significantly augment current methods. This blend of formal, and simulation-based methods is a clear future trend.

If checking for functional correctness is hard, ensuring the lack of performance bugs is seemingly even more difficult to automate. These bugs are of two types:

1. a design or model defect that causes observable inaccuracies in specific latency or bandwidth parameters or other timing characteristics; and
2. a design deficiency, as reflected in the presilicon model, that causes a significant gap between actual and expected performance of a key benchmark or kernel.

An example of type 1 is a model defect that may cause the back-to-back, dependent floating-point operation issue (pipeline bubble) latency to be greater or less than the design specification. As a class 2 bug example, the model may correctly implement an initial design specification; however, this may result in a large, inadvertent negative impact on a key benchmark, such as Linpack, or a dominant loop kernel, such as Daxpy. In later-stage design, type 2 bugs that are correctable via minor redesigns are of significance; major redesign recommendations are taken seriously only in early-stage analysis. Type 2 bugs may also expose tuning opportunities for the compiler. As we discussed earlier,[2] we foresee a trend of automation in which the presilicon validation exercise will use automatic test-case generation, which combines "golden" signatures to qualify error-free function and performance. Researchers are experimenting with such integrated testing methodologies in some development projects. Again, formal models to aid in correctness checks may be of use in early (high-level) performance models. Timing properties or FSM-level formalisms may be used to weed out fundamental problems such as deadlock scenarios and race conditions.

Hardware-specific tuning of software (at the application and system support level) continues to offer a major opportunity for increasing delivered system performance. Tools for application tuning and heuristics generation for compiler optimization promise to offer significant advantages in the future. Kunkel et al. (this issue) depicts the magnitude of software-tuning opportunities in a commercial system built using PowerPC superscalar RISC nodes.

> We foresee the use of more sophisticated, formal model-driven test case generation and coverage techniques to replace or significantly augment current methods.

We hope this theme issue will help you appreciate the significance of each topic we've presented. Our introduction may also help you understand some of the future trends and requirements that are not addressed in the articles in this issue. MICRO

Acknowledgments

We thank all the anonymous referees for their help in reviewing the articles submitted for this special issue and Ken Sakamura, *IEEE Micro* Editor-in-Chief, for his help and support.

References

1. S. Al-Ashari, "System Verification from the Ground Up," in the online magazine, *Integrated System Design Magazine,* Jan. 1999; http://www.isdmag.com/Editorial/1999/coverstory9901.html.
2. P. Bose and T.M. Conte, "Performance Analysis and Its Impact on Design," *Computer,* Vol. 31, No. 5, May 1998, pp. 41-49.
3. D.C. Burger and T.M. Austin, "The SimpleScalar Tool Set, Version 2.0," *Computer Architecture News,* Vol. 25, No. 3, June 1997, pp. 13-25; also extended version Univ. of Wisconsin-Madison, Computer Sciences Tech. Report 1342, June 1997.
4. K. Diefendorff, "The Race to Point One Eight (0.18 Micron)," *Microprocessor Report,* Vol. 12, No. 12, Sept. 1998, pp. 1-8.
5. D. Leibholz and R. Razdan, "The Alpha

21264: A 500 MHz Out-of-Order Execution Microprocessor," *Proc. IEEE Compcon,* IEEE Computer Society Press, Los Alamitos, Calif., 1997, pp. 28-36.

6. M.H. Lipasti and J.P. Shen, "Exceeding the Dataflow Limit via Value Prediction," *Proc. 29th Ann. ACM/IEEE Int'l. Symp. Microarchitecture,* IEEE CS Press, Dec. 1996, pp. 226-237.
7. E. Clarke and R. Kurshan, "Computer-Aided Verification," *IEEE Spectrum,* Vol. 33, No. 6, 1996, pp. 61-67.
8. M. Kantrowitz and L.M. Noack, "Functional Verification of a Multiple-Issue, Pipelined Superscalar Alpha Processor—the Alpha 21164 CPU Chip," *Digital Tech. J.,* Vol. 7, No. 1, 1995, pp. 136-144.
9. A. Aharon et al., "Test Program Generation for Functional Verification of PowerPC Processors in IBM," *Proc. 32nd ACM/IEEE Design Automation Conf.,* ACM, New York, 1995, pp. 279-285.
10. R. Grinwald et al., "User Defined Coverage—A Tool Supported Methodology for Design Verification," *Proc. 35th IEEE/ACM Design Automation Conf.,* ACM, June 1998, pp. 1-6.

Pradip Bose is a research staff member at the IBM T.J. Watson Research Center. During 1992-94, he was assigned to IBM Austin as the lead performance engineer for the definition and evaluation of a processor core that evolved into the Power3 microprocessor. His research interests include high-performance computer architectures and performance modeling, verification, and application tuning. Bose received a BTech degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, and an MS and a PhD in electrical and computer engineering from the University of Illinois, Urbana-Champaign. He is a senior member of the IEEE and the Computer Society.

Thomas M. Conte is currently an associate professor of North Carolina State University. He has also consulted on microarchitecture-related issues for several companies, including AT&T, IBM, and S3. He currently directs 11 PhD students on the TINKER project in topics spanning advanced microarchitecture, compiler scheduling, and performance analysis. Conte received his BEE degree from the University of Delaware, and his MS and PhD in electrical engineering from the University of Illinois at Urbana-Champaign. He is widely published in the areas of microarchitecture and performance evaluation, has served on several program committees, and chairs the IEEE CS Technical Committee on Microprogramming and Microarchitecture (TC-MARCH).

Todd M. Austin is a computer architect with Intel's Microcomputer Research Labs in Hillsboro, Oregon. His work includes product-oriented research for future-generation microprocessors, development of university relations, and design and implementation of performance analysis tools. He also teaches systems courses at the Oregon Graduate Institute, where he is an adjunct assistant professor of computer science and engineering. His research interests include microarchitecture design, memory system optimization, compiler design, computer system validation, and performance analysis of instruction-level parallel processors. Austin received his PhD from the University of Wisconsin-Madison.

Direct comments about this theme issue to Pradip Bose, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598; pbose@us.ibm.com; Tom Conte at conte@eos.ncsu.edu, or Todd Austin at tma@acm.org.