Billion-Transistor Architectures: There and Back Again

Doug Burger
The University of Texas at Austin
James R. Goodman
University of Auckland

A look back at visionary projections made seven years ago by top researchers provides a context for the continuing debate about the future direction of computer architectures.

In September 1997, *Computer* published a special issue on billion-transistor microprocessor architectures.[1] Our goal for that issue was to frame the debate about the direction computer architectures would take as Moore's law continues its relentless drive down to atomic limits. That issue, widely cited, contained a range of visionary projections from many top researchers, covering the space of architectural possibilities at the time across a range of candidate markets and domains.

We solicited and selected two sets of papers. The first set enumerated important emerging trends that were potential drivers of architectural change in technology, applications, and interfaces. The second set described a number of visions for designs that could and would scale up to billion-transistor architectures (BTAs). What emerged from the accepted set of papers was that there was no consensus about which direction microprocessor architectures are likely to take as chip integration reaches unprecedented levels.

Seven years later, it is both interesting and instructive to look back on that debate and the projections made. What did the community get right? What did we miss? What new ideas have since emerged?

It turned out that many of the authors—a surprising number given the disparity in opinions—were exactly right about the directions that industry would take in the near term. However, none of the architectural models discussed has become dominant, and it is still unclear that any of them will be the right model for BTAs across a broad swath of future markets.

LOOKING BACKWARD AT FORWARD PROJECTIONS

Architectures are never designed in a vacuum—they are always affected by technology, cost, customers, workload, and usability constraints, as well as marketing initiatives and fads. Because of the complexity of modern systems, there is also tremendous pressure for architectures to evolve gradually; major transitions are extremely rare. Consequently, it is important to understand the specific constraints that cause evolution in architectures over time.

Architecture-affecting trends

The 1997 issue contained articles that each predicted a driving external force that would affect architectures over the next decade. These constraints fell into three categories—technology, workloads, and hardware/software interfaces.

Published by the IEEE Computer Society

Technology. Doug Matzke[2] presented a prescient study of on-chip interconnect delay, predicting that because of faster clocks and growing resistivity in shrinking wires, only small fractions of a chip would be reachable in a single cycle by 2012. Although many researchers were aware of this trend—several of the articles cited wire delays as a key driving issue—this study quantified the extent of the emerging challenge, unrecognized by many at the time. Matzke's projections still hold: The effects of slower wires continue to increase each year.

Workloads. Keith Diefendorff and Pradeep Dubey[3] made the case that multimedia workloads would be the key driver of new computer architectures. In particular, they predicted that the general-purpose processor market would subsume the high-end DSP market as BTAs inevitably incorporated support for efficient multimedia execution: real-time capabilities, loop-specific optimizations, and subword data parallelism. This unrealized convergence is still possible because graphics and signal processing systems are becoming more programmable, and future general-purpose machines are likely to exploit more fine-grained concurrency. Whether the two types of architectures will converge remains an open question.

Binary interfaces. Josh Fisher[4] made the case that fixed instruction sets would become less important due to "walk-time techniques" such as binary translation and dynamic recompilation, enabling many minor application- or market-specific variations in each family of instruction sets with full software cross-compatibility. This capability has not yet become universal, but individual companies like Transmeta—whose chips run legacy x86 code on a VLIW implementation—rely on such technology. Additionally, there is evidence that the major microprocessor vendors are moving in this direction.

Projections for future BTAs

The 1997 issue included visions of what future BTAs would be from seven top research groups, selected to cover the spectrum of leading candidate architectures. While seven years is a short time in terms of design generations—fewer than two assuming a four-year design cycle—it is simultaneously a long time in our fast-paced field.

We ordered the articles in the 1997 issue according to the granularity of parallelism exposed to software—coarsest to finest—which influences the ease of partitioning the hardware. The spectrum of granularities ranged from a single thread running on a single, enormous, wide-issue superscalar processor to a chip with numerous small, single-issue tiles in which both the computation and the interfile communication are fully exposed to software.

Software and hardware may be headed for a train wreck.

This debate about the correct degree of partitioning is timely because software and hardware may be headed for a train wreck. The increasing wire delays that Matzke described are forcing greater partitioning of hardware, which could in turn force more partitioning of software. Because many applications are still monumentally difficult to parallelize, hardware designers may provide more processing units but pass the buck to either compilers or programmers to figure out how to use them. The right point in this space (for each application class) must carefully balance this tension between hardware and software partitioning.

Wide-issue superscalar processors. Yale Patt and his group[5] advocated ultrawide-issue, out-of-order superscalar processors as the best alternative for BTAs. They predicted that the first BTAs will contain a single 16- or 32-wide-issue processing core using out-of-order fetch, large trace caches, and huge branch predictors to sustain good instruction-level parallelism (ILP).

At present, industry is not moving toward the wide-issue superscalar model; the termination of the Alpha 21464 design—an 8-wide-issue, multithreaded out-of-order core—was a significant setback. This model suffers from high design complexity and low power efficiency, which are both currently of enormous concern to product groups. Since these issues have not been mitigated, industry is moving in other directions: The desktop market has continued with narrow-issue, ultra-high-frequency cores; the server market has begun using multithreaded chip multiprocessors; and the graphics market is starting to use CMPs that are more fine-grained than server processors. New types of instruction-set architectures may move wide-issue superscalar processors back into favor.

Superspeculative superscalar processors. Mikko Lipasti and John Shen[6] proposed Superflow, a wide-issue superscalar architecture that relied on heavy data speculation to achieve high performance. Like Patt's group, they assumed an aggressive front end that used a trace, but differed by proposing a data speculation engine that used value prediction for loads, load addresses, and arithmetic instructions, along with load/store dependence prediction for memory ordering.

Aggressive speculation has become commonplace throughout microprocessor pipelines, but it

Designers need to find the sweet spot between single-thread execution semantics and a distributed architecture.

has not yet broadly incorporated value speculation. Most modern predictors mitigate performance losses due to deeper pipelines; as industry has progressively shortened the clock period, state previously reachable from a given point becomes unreachable in a single cycle, forcing the microarchitecture either to wait or to guess. Thus, of the speculative techniques that Lipasti and Shen advocated, those that facilitated deeper pipelines have generally been implemented, but most of the techniques intended to support high ILP in a wide-issue machine have not.

Simultaneous multithreaded processors. SMT processors share a superscalar core dynamically and concurrently, increasing its utilization. Susan Eggers and coauthors[7] accurately predicted that SMT processors would appear in the near future—both the Intel Pentium 4 and IBM's Power5 processor use SMT technology. However, the number of threads per individual core is unlikely to increase much beyond the small number currently appearing, making SMT an unlikely first-order paradigm for BTAs. All superscalar-style cores likely will have some form of SMT capability, but SMT is not a model that will provide long-term scalability for future implementations.

Distributed processors. James E. Smith and Sriram Vajapeyam[8] advocated trace processors as a viable candidate for BTAs. They argued that logical uniprocessors—running a single thread—are desirable, but because hardware trends will increase the necessary partitioning, microarchitectures will inevitably start to resemble parallel processors. They described trace processors as an example of a "fourth-generation architecture" in which a single logical thread feeds multiple discrete processing engines, one trace at a time. Trace processors are one approach to finding the sweet spot between single-thread execution semantics and a necessarily distributed microarchitecture.

Aside from limited clustering in the Alpha 21264, designers have not yet adopted aggressive microarchitectural partitioning, although recent academic literature frequently describes clustered microarchitectures. To tolerate wire delays, high-frequency processor designers have instead added pipeline stages for communication—for example, the Pentium 4—rather than clustering the execution core. Adding pipeline stages is a short-term solution for wire delays, so clustering is inevitable for large processors that support single threads.

Vector IRAM processors. Christoforos Kozyrakis and colleagues[9] advocated placing enormous, high-bandwidth memories on the processor die—built using dynamic RAM (DRAM) technology—integrating physical memory with the processor and thus increasing main memory bandwidth appreciably. They proposed using vector processors to exploit this additional bandwidth and developing new compiler techniques to vectorize many applications previously deemed unvectorizable.

The importance of vector-like media processing has clearly increased, and vector processors have remained important at the ultrahigh end of the computing spectrum—for example, the Japanese Earth simulator. However, the continued divergence of DRAM and logic processes makes vector intelligent RAM (VIRAM)-like parts unlikely to subsume general-purpose processors anytime soon. Vector-like processors with dedicated and integrated memories are good candidates for data-parallel workloads in the embedded space.

Chip multiprocessors. Like many of the other authors, Lance Hammond and coauthors[10] argued that wire delays and changing workloads will force a shift to distributed hardware, which in their model consists of a large number of simple processors on each chip. Unlike other authors, they extended that argument to software, claiming that the programming model is likely to change to exploit explicit parallelism because a CMP uses transistors more efficiently than a superscalar processor only when parallel tasks are available.

In the high-performance commercial sphere, CMPs are becoming ubiquitous. IBM's Power4 has two processors, Compaq WRL's proposed Piranha processor had eight, and Intel has announced plans to build CMP-based IA-64 processors. In the desktop space, however, single-chip uniprocessors are currently still dominant. A key question is whether CMPs—made up of simple processors—can scale effectively to large numbers of processors for nonserver workloads. Computer architecture historians may be interested to know that the 1997 *Computer* issue was where the now widely used CMP acronym was popularized, although we had first used the term a few months before in a paper presented at ISCA.

Raw microprocessors. Finally, Elliot Waingold and coauthors[11] proposed Raw microprocessors as the right model for BTAs. These processors have the flavor of a highly clustered, two-dimensional VLIW processor in which all of the clusters have independent sequencers. Raw processors push partitioning to an extreme, with numerous extremely simple and highly distributed processing tiles managed wholly by software. Statically scheduled instruction streams at each intertile router manage interprocessor communication.

These systems achieve terrific scalability and efficiency for codes exhibiting statically discoverable concurrency, such as regular signal processing applications. However, they still cannot deal effectively with runtime ambiguity, such as statically unpredictable cache misses or dynamically determined control, making them unlikely candidates for BTAs except in specialized domains.

EMERGING TRENDS

A number of constraints and trends, the significance of which many researchers (including us) did not foresee, have emerged since 1997. Some of these new directions are affecting the march toward balanced and scalable BTAs.

Superclocked processors

The extent to which faster clocks were driving designs was known but underappreciated seven years ago. Since then, industry has continued along the high-frequency path, emphasizing faster clock rates over most other factors. This emphasis is most clearly evident in the Intel x86 family of processors.

In 1989, Intel released the 80386, implemented in approximately 1-µm technology, with a 33-MHz clock rate. That frequency corresponded roughly to 80 fan-out-of-four (FO4) inverters' worth of logic per clock cycle, with each inverter driving a load four times that of its own. By 2003, Intel was selling 3.2-GHz Pentium 4 chips, implemented in roughly 90-nm (or .09-µm) technology—a 100-fold increase in frequency. This speed increase came from two sources: smaller, faster transistors and deeper pipelines that chopped the logic up into smaller pieces. The Pentium 4 has between 12 and 16 FO4 inverters per clock cycle, a decrease of 80 to 85 percent compared to the 80386.

This rapid clock speed increase—40 percent per year over the past 15 years—has provided most of the performance gains as well as being the primary driver of microarchitectural changes, a result that few researchers predicted. Most of the new structures and predictors appearing in complex microarchitectures, such as load latency predictors in the Alpha 21264 and the Pentium 4, are there solely to support high frequencies, mitigating the ILP losses resulting from deeper pipelines.

The emphasis on frequency increases has had three major implications. First, it has hastened the emergence of power bottlenecks. Second, it has deferred the need to change instruction sets; since RISC instruction sets, and the x86 µop equivalents, were intended to support pipelining effectively, industry was able to focus on clock scaling without incurring the pain of changing industry-standard architectures. The dearth of new ISAs in the past 15 years is more attributable to the explosion of clock frequency than to a fundamental end of ISA innovations. Once design-enabled frequency improvements are no longer viable, we are likely to see a resurgence of ISA changes, although they will likely be hidden behind a virtual machine with an x86 interface.

Third, reductions in the logic-per-clock period are nearing a hard limit; prior work has shown that reducing the clock period much below 10 FO4 inverters per cycle is undesirable.[12,13] We are thus quite close to a microarchitectural bound on frequency improvement. Further, leakage power is likely to bound the rate of device-driven frequency improvement. These two factors suggest that the rate of frequency increases is about to slow dramatically, forcing a shift to other strategies for achieving performance.

The rate of frequency increases is about to slow dramatically, forcing a shift to other strategies for achieving performance.

Power

One factor that has become drastically more important than any of the 1997 authors predicted is power consumption, both dynamic and static. Power issues have moved from being a factor that designers must simply consider to become a first-order design constraint in future processors.

The primary cause of the sudden emergence of dynamic power as a constraint is the extraordinarily rapid and continued growth in clock speeds. Future BTA designs must consider power efficiency as a factor in determining the right way to extract performance from a given software workload—a necessity that penalizes the conventional wide-issue superscalar approach.

Static power is just beginning to emerge as a serious design constraint, but it could be more fundamental by limiting the number of devices available for use at any given time.

Intel's recent announcement of new materials—presumably improved dielectrics—offers some hope that leakage will not limit available devices as soon as some thought. However, we could still eventually find ourselves in a domain in which transistors continue to shrink but do not get faster, putting more pressure on extraction of concurrency for performance rather than raw clock speed.

These potential new power constraints imply that designers must balance high performance with efficient use of transistors, adding another new constraint—wire delays being the other—to options for BTAs.

Researchers are actively exploring two directions: making processors *faster* and making them *better*.

LOOKING FORWARD AGAIN: SOME NEW DIRECTIONS

Semiconductor process experts predict a continued increase in transistor counts for at least another decade. These increases will enable an enormous degree of integration, but the pressing question is, what should we do with all of this hardware? To answer this question, researchers are actively exploring two directions: making processors *faster*, which was the focus of the 1997 articles, and making them *better*.

Making systems better, not faster

As on-chip devices become extraordinarily small and more numerous, using them intrinsically becomes more difficult. They are less reliable, fail more often, and can consume too much power. Furthermore, programmers, languages, and compilers may not be able to use them all effectively. Numerous ongoing research efforts are addressing these challenges by allocating a fraction of future hardware budgets to mitigate the downsides of such enormous device counts.

Assist threads. Since enough explicit parallel threads often are not available, researchers have begun using the parallel thread slots available in SMT processors for "helper" threads. These helper threads are designed to improve performance and have been called variously subordinate threads, slipstreaming, speculative data-driven threads, or master-slave threads.[14-17] Like SMT, this approach could benefit a few generations of designs, but it is not a substitute for scalable hardware or more effective parallel programming.

Reliability, debugging, and security. David Patterson has recently been making the case that reliability in future systems will be paramount and should be more of a focus for researchers than improved performance. Certainly, many recent reports in the literature have focused on providing reliable execution, whether with a result checker,[18] reliability-enhancing redundant threads,[19,20] or a system that supports execution near the edge of tolerable voltage limits.[21]

Researchers have also begun using threads to support software debugging. In related efforts, they have proposed using hardware support to enhance security, for example, detecting and preventing buffer overflows and stack smashing, or providing fine-grained memory protection.[22] Detecting bugs, recovering from faults, and foiling intruders (malevolent and otherwise) are all likely to be important uses for future hardware resources.

Parallel programming productivity. A major underlying theme that emerged from the articles in the 1997 issue was the tension between the difficulty of explicitly partitioning software and the need to partition future hardware. It is clear that the ability of software—either compilers or programmers—to discover concurrency will have a first-order effect on the direction of BTAs in each market. If parallel programming remains intractably difficult for many applications, chips with small numbers of wide-issue processors will dominate, bounded only by complexity and efficiency limits.

We (Jim Goodman, along with his colleague, Ravi Rajwar) have been developing hardware support that improves the ease of productive parallel programming by enabling concurrent execution of transactions.[23] Speculative Lock Elision allows programmers to include locks that suffer no performance penalty if no lock contention occurs, and the more aggressive Transactional Lock Removal[24] provides lock-free execution of critical sections. Programmers can thus concentrate on getting the synchronization code right, with a generous use of locks less likely to kill a program's performance.

Continuing the quest for performance

As frequency improvements diminish, increased concurrency must become the primary source of improved performance. The key concern that architects must address is the number and size of processors on future CMP chips. Scaling the number of simple processors in a CMP beyond a few tens simply doesn't make sense given the state of software parallelization, and it will result in asymptotically diminishing returns. Similarly, scaling a single core to billions of transistors will also be highly inefficient, given the ILP limits in single threads. In our view, future BTAs should have small numbers of cores that are each as large as efficiently possible.

The sizes and capabilities of these large future processors are an open question. The Imagine processor[25] and the follow-on streaming supercomputer effort[26] both use large numbers of arithmetic logic units to exploit the data-level parallelism prevalent in steaming and vector codes, with high power efficiency per operation. We (Doug Burger, along with his colleague, Steve Keckler) have proposed an alternative approach that exploits concurrency from irregular codes and from individual threads using large, coarse-grained processing cores. These large cores rely on a new class of dataflow-like instructions sets called EDGE architectures (for explicit data graph execution—a term that Chuck Moore coined

while he was at the University of Texas at Austin), of which the TRIPS architecture will be the first instance.[27] By enabling much larger cores to exploit concurrency both within and across threads (and vectors), the hope is that this class of architectures will permit future BTAs to continue effective performance scaling while avoiding the need to build "CMPPs" (chip massively parallel processors).

Future BTAs will be judged by how efficiently they support distributed hardware without placing intractable demands on programmers. This balance must also factor in efficiency; hardware that matches the available concurrency's granularity provides the best power and performance efficiency. Researchers will doubtless continue to propose new models as they seek to find the right balance among partitioning, complexity, and efficiency. Whether the right model for general-purpose BTAs ends up being one of those advocated in 1997, a more recent one such as some of those described in this article, or one that has not yet been discovered, the future for interesting architectures has never been more open.

What is even more exciting—or scary, depending on the reader's perspective—is that the solution to these problems could have fundamental implications for both the software stack and software developers. When efficient, transparent solutions to hardware partitioning reach their scalability limit, hardware designers must pass the buck to software, placing the onus for more performance on the programming model.

The next decade in both architecture and software systems research promises to be even more interesting than the last. ■

References

1. D. Burger and J.R. Goodman, "Billion-Transistor Architectures," *Computer*, Sept. 1997, pp. 46-48.
2. D. Matzke, "Will Physical Scalability Sabotage Performance Gains?" *Computer*, Sept. 1997, pp. 37-39.
3. K. Diefendorff and P.K. Dubey, "How Multimedia Workloads Will Change Processor Design," *Computer*, Sept. 1997, pp. 43-45.
4. J.A. Fisher, "Walk-Time Techniques: Catalyst for Architectural Change," *Computer*, Sept. 1997, pp. 40-42.
5. Y.N. Patt et al., "One Billion Transistors, One Uniprocessor, One Chip," *Computer*, Sept. 1997, pp. 51-58.
6. M.H. Lipasti and J.P. Shen, "Superspeculative Microarchitecture for Beyond AD 2000," *Computer*, Sept. 1997, pp. 59-66.
7. S.J. Eggers et al., "Simultaneous Multithreading: A Platform for Next-Generation Processors," *Computer*, Sept. 1997, p. 49.
8. J.E. Smith and S. Vajapeyam, "Trace Processors: Moving to Fourth-Generation Microarchitectures," *Computer*, Sept. 1997, pp. 68-74.
9. C. Kozyrakis et al., "Scalable Processors in the Billion-Transistor Era: IRAM," *Computer*, Sept. 1997, pp. 75-78.
10. L. Hammond, B.A. Nayfeh, and K. Olukotun, "A Single-Chip Multiprocessor," *Computer*, Sept. 1997, pp. 79-85.
11. E. Waingold et al., "Baring It All to Software: Raw Machines," *Computer*, Sept. 1997, pp. 86-93.
12. A. Hartstein and T.R. Puzak, "The Optimum Pipeline Depth for a Microprocessor," *Proc. 29th Int'l Symp. Computer Architecture*, IEEE CS Press, 2002, pp. 7-13.
13. M.S. Hrishikesh et al., "The Optimal Logic Depth Per Pipeline Stage Is 6 to 8 FO4 Inverter Delays," *Proc. 29th Int'l Symp. Computer Architecture*, IEEE CS Press, 2002, pp. 14-24.
14. R.S. Chappell et al., "Simultaneous Subordinate Microthreading (SSMT)," *Proc. 26th Int'l Symp. Computer Architecture*, IEEE CS Press, 1999, pp. 186-195.
15. Z. Purser, K. Sundaramoorthy, and E. Rotenberg, "A Study of Slipsteam Processors," *Proc. 33rd Int'l Symp. Microarchitecture*, IEEE CS Press, 2000, pp. 269-280.
16. A. Roth and G.S. Sohi, "Speculative Data-Driven Multithreading," *Proc. 7th Int'l Symp. High-Performance Computer Architecture*, IEEE CS Press, 2001, pp. 37-48.
17. C. Zilles and G.S. Sohi, "Master/Slave Speculative Parallelization," *Proc. 35th Int'l Symp. Microarchitecture*, IEEE CS Press, 2002, pp. 85-96.
18. T.M. Austin, "Diva: A Reliable Substrate for Deep Submicron Microarchitecture Design," *Proc. 32nd Int'l Symp. Microarchitecture*, IEEE CS Press, 1999, pp. 196-207.
19. S.K. Reinhardt and S.S. Mukherjee, "Transient Fault Detection via Simultaneous Multithreading," *Proc. 27th Int'l Symp. Computer Architecture*, IEEE CS Press, 2000, pp. 25-36.
20. E. Rotenberg, "AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors," *Proc. 29th Int'l Symp. Fault-Tolerant Computing*, IEEE CS Press, 1999, pp. 84-91.
21. D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *Proc. 36th Int'l Symp. Microarchitecture*, IEEE CS Press, 2003, pp. 7-18.

22. E. Witchel, J. Cates, and K. Asanovic, "Mondrian Memory Protection," *Proc. 10th Int'l Symp. Architectural Support for Programming Languages and Operating Systems*, ACM Press, 2002, pp. 304-315.
23. R. Rajwar and J.R. Goodman, "Speculative Lock Elision: Enabling Highly Concurrent Multithreaded Execution," *Proc. 34th Int'l Symp. Microarchitecture*, IEEE CS Press, 2001, pp. 294-305.
24. R. Rajwar and J.R. Goodman, "Transactional Lock-Free Execution of Lock-Based Programs," *Proc. 10th Int'l Symp. Architectural Support for Programming Languages and Operating Systems*, ACM Press, 2002, pp. 5-16.
25. S. Rixner et al., "A Bandwidth-Efficient Architecture for Media Processing," *Proc. 31st Int'l Symp. Microarchitecture*, IEEE CS Press, 1998, pp. 3-13.
26. W.J. Dally, P. Hanrahan, and R. Fedkiw, *A Streaming Supercomputer*, white paper, Computer Systems Laboratory, Stanford Univ., 2001.
27. K. Sankaralingam et al., "Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture," *Proc. 30th Int'l Symp. Computer Architecture*, IEEE CS Press, 2003, pp. 422-433.

Doug Burger *is an assistant professor in the Department of Computer Sciences at the University of Texas at Austin. His research interests are computer architecture, advanced compilation, and high-performance computing. Burger received a PhD in computer sciences from the University of Wisconsin-Madison. He is a member of the ACM, a senior member of the IEEE, and a Sloan Foundation Research Fellow. Contact him at dburger@cs.utexas.edu.*

James R. Goodman *is a professor in the Department of Computer Science at the University of Auckland. His research interests focus on computer architecture. Goodman received a PhD in electrical engineering and computer science from the University of California, Berkeley. Contact him at goodman@cs.auckland.ac.nz.*