If You Build It, Will They Come?

Srilatha Manne
Cavium

Bryan Chin
University of California, San Diego

Steven K. Reinhardt
Microsoft

All hardware companies face a conundrum. Should they continue the evolutionary trend of their current products, or build riskier products that have the potential for greater reward but carry a higher probability of failure? The safe course, and one that many customers ask for, is the former. However, as Clayton Christensen points out in *The Innovator's Dilemma*, "most companies with a practiced discipline of listening to their best customers and identifying new products that promise greater profitability and growth are rarely able to build a case for investing in disruptive technologies until it is too late."[1]

Computer hardware companies expend enormous resources to successfully improve their products in an evolutionary fashion. Single-threaded processor performance has been improving at a rate of 15 to 20 percent per year by utilizing both process technology and architectural improvements.[2] These improvements, however, are increasingly difficult to achieve. Using data from Moein Khazraee and colleagues,[3] Figure 1 shows that a processor's cost per operation, as defined by a combination of fabrication, nonrecurring engineering (NRE), and packaging costs, has not significantly improved in the past decade. However, performance improvements are flattening out due to power restrictions and the breakdown of Dennard scaling. For instance, Intel is no longer relying on the tick-tock model, which it rode to market dominance for the past decade, due to the declining benefits of process technology scaling.[4]

Sustaining versus Disruptive Technology

Christensen describes the evolutionary process of improvements using the *sustaining technology S-curve* (see Figure 2). For every successful technology, the performance metric is initially flat during development, rapidly improves for a period of time, and flattens out again when the product and/or technology reaches maturity. Sustaining technologies are dominating the processor industry, and these technologies are reaching a plateau.

Sometimes a disruptive technology with a new S-curve will enter the landscape, as shown in Figure 2. Disruptive technologies do not go head to head with mainstream technologies, but they do have features that a few fringe markets value. Typically, disruptive technologies initially underperform, but then rapidly match and exceed the previous technology. Successful companies not only ride their sustaining S-curves but generate new, disruptive curves to improve performance as the current technology curve flattens out. Microprocessors were once a disruptive technology,[1] and the computing landscape over the past few decades is littered with disruptive technologies, from minicomputers to PCs to smartphones to cloud computing. In all these cases, the disruptive technology yielded worse performance in the near-term when using the same cost function as mainstream technology. However, as Christensen maintains, disruptive technologies eventually redefine how performance is measured.

Recent examples of disruptive technologies in processor architecture include GPUs and Arm servers. GPUs were originally designed for 3D graphics processing, but have made significant inroads first in high-performance computing (HPC) and more recently in machine learning. For applications that are similar to those found in SPECint, GPUs underperform general-purpose processors. However, for targeted HPC applications and machine learning, GPUs are overwhelmingly superior.

Arm processors originally targeted power-constrained embedded domains, but have more recently entered the server market with product offerings from companies such as

Published by the IEEE Computer Society

Cavium and Qualcomm that address multicore throughput computing.[5,6] A new S-curve could develop for these specialized throughput-based server products—enabled by highly parallelizable shared-memory applications—just as it did with HPC and machine learning in the GPU market.

It took the GPU market nearly two decades to make headway outside of graphics applications, and the Arm server market has resulted in several failures. Christensen notes that this commonplace in disruptive markets is where "[it] is simply impossible to predict with any useful degree of precision how disruptive products will be used or how large their markets will be." So, how does one innovate in a rapidly changing technology landscape where the underlying cost function is in flux? How does a company keep up with the necessary and expensive evolutionary changes, yet also prepare for and justify expending valuable resources investigating disruptive technologies that are inevitable?

The Case for Agility

Companies and their mainstream customers alike are notoriously bad at predicting what disruptive products will take root in the marketplace. There are many instances of high-profile developments that flopped. For example, it is unlikely you are reading this article on your Apple Newton while listening to music on your Microsoft Zune. Conversely, some disruptive technologies have found success in surprising places such as GPUs. Innovation in a rapidly changing landscape is difficult and prone to failure. Therefore, we posit that architects, rather than trying to predict the future, should pursue agility in order to accelerate innovation while minimizing costs. Hardware companies, architects, and the underlying design methodologies and infrastructure must be nimble enough to deal with disruptive technologies that come from within and outside the current technology landscape. The rest of the article presents some ideas on how this may be accomplished.

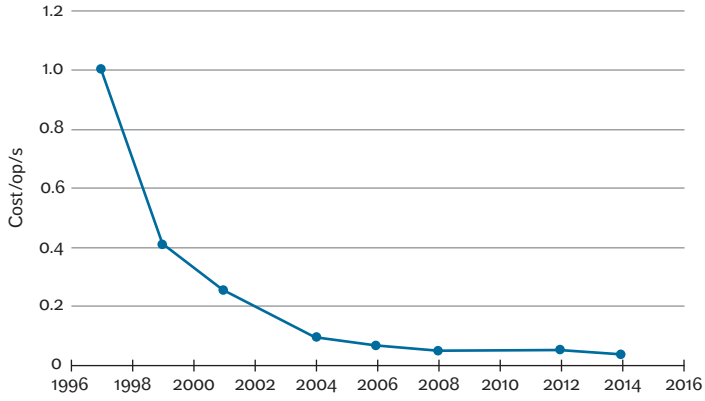

Figure 1. Processor computation cost as a function of time. Cost is defined as a combination of fabrication, nonrecurring engineering (NRE), and packaging costs.[3]

Agile Architecture

In his book *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Business*,[7] Eric Ries writes about software companies that use agile software development strategies. The premise is to deliver prototypes as quickly as possible, even if haphazardly put together, to get early customer feedback. The goal is to use customer feedback to drive product features and direction through a process of continuous development. If you consider how frequently the apps on your phone are updated, or the look and feel of social networking sites evolve, you have seen agile software practices in action.

Facebook, for example, uses agile coding practices. As Kent Beck explains,[8] one of the basic practices at Facebook is reversibility. If a decision is reversible, it does not require the rigorous testing that irreversible decisions require. Code is also released incrementally to a small subset of users, which enables changes to be rolled back with minimal disruption if a problem is found.[9] The challenge for the hardware industry is how to adapt a similar agile methodology without incurring large overheads. We address this challenge in both traditional processor hardware methodologies and innovative methodologies utilized by large computing companies.

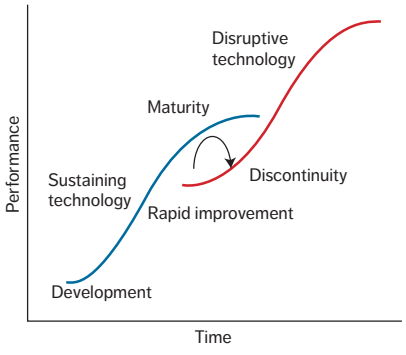

Figure 2. The sustaining technology and disruptive technology S-curves.

Processor Agility

Prior to the ASIC revolution of the past few decades, hardware prototypes were a common means of achieving the rapid development and early feedback cycle. Old technologies such as wire-wrap, breadboards, programmable logic devices (PLDs), and low-cost printed circuit boards (PCBs) enabled hardware companies to quickly build and iterate on products. This methodology is no longer feasible given the complexity and cost of processor development both in terms of engineering time and fabrication costs.[3]

Automated design methodology and reuse. Companies today rely on improved design methodologies and reusability to reduce design time and cost. Design methodologies have made great strides in the past two decades, resulting in shorter design cycle times and an expanded product portfolio using the same fundamental components. Most processors, even those designed for high performance, are mostly or completely synthesized. The Arm roadmap has synthesized cores operating at 3 GHz, and AMD, Intel, and IBM extensively use automated tools throughout their design.[10–12] In addition, companies utilize a modular design methodology such that multiple products can be developed using the same basic components.

Both Intel and AMD use their respective base core designs and innovative packaging technologies to build products ranging from low-power mobile parts to multicore server products.[13] Similarly, silicon companies such as Cavium and Nvidia have been able to create a family of devices with varying price/performance points from the same basic design by utilizing flexible chip layouts that let designers vary the number of computational units and/or the amount of on-die memory. Intel has taken this one step further by collaborating with Facebook to develop a specialized version of Broadwell (referred to as *Broadwell-D*) to meet the specific needs of Facebook.[14]

The technologies mentioned so far reduce design cycle time, but there is still significant overhead associated with bringing a chip to production. Post-silicon functional and performance debug is a formidable challenge for modern processors that may encompass multiple sockets, heterogeneous and/or multithreaded cores, many cores combined with multiple levels of memory hierarchy, complex memory coherence and consistency protocols, and extensive power and performance management via on-chip controllers. In addition, modern processors may operate under complex software stacks containing one or more nested virtual environments. For these reasons, even with mostly synthesized methodologies and reuse of existing components, the transition from first silicon to full production part can take up to a year or more.[15]

Functional verification and bug mitigation. Post-production bugs are commonplace, and fixing bugs in shipped products often involves errata, metal and full-layer spins, and/or replacing existing silicon. Infamous examples of such bugs are the Pentium FDIV bug,[16] the Haswell/Broadwell transactional memory bug,[17] and the AMD TLB bug.[18] These bugs cost the respective companies millions of dollars in lost revenue, and in AMD's case, contributed to its loss of momentum in the server market. All processors have a large list of errata. The table of known errata in Haswell, for instance, covers six pages.[19]

To meet market needs and address the complexity and cost of post-silicon debug, architects must focus on hardware and software solutions for exposing, analyzing, and mitigating functional and performance bugs. Processor vendors must provide tools that rapidly expose and identify bugs and have systems in place for mitigating these bugs without the need for extensive silicon changes. Efforts such as Arm's hardware debug architecture attempt to standardize the infrastructure so that common tools can be made available to the Arm hardware development ecosystem.[20]

Both software and hardware solutions should be explored for mitigating hardware bugs in the field. On the hardware front, microcode fixes on traditional CISC processors come to mind, as does the PAL (Privileged Architecture Library) code feature of DEC's Alpha processors. A similar technology that might help processor vendors mitigate bugs is virtual machine environments. Much software these days is compiled to an abstract machine. Two examples of such abstraction layers, one current and one historical, are Oracle's Java Virtual Machine (JVM)[21] and IBM's AS/400 Series.[22] If an entire processor is designed to execute only a JVM, then the JVM itself provides the instruction set architecture (ISA) of the machine, and the underlying physical machine may have bugs or features that are invisible to the JVM. The JVM addresses ISA-related bugs. Similarly, more fully specified virtual machine environments, such as VMware's vSphere and Microsoft's Hyper-V, virtualize system aspects of the machine, such as memory management and I/O. Machines such as IBM's AS/400 managed to maintain a stable abstract architecture through multiple generations of hardware. By expecting and architecting for bug discovery, analysis, and mitigation, processor vendors can reduce the number of bugs that reach production silicon, and respond to issues in post-production parts quickly and effectively. This shortens the designer-customer feedback loop and leads to a faster development cycle and improved successor products.

Performance verification and optimization. Another critical facet of bringing a processor to production is performance tuning. Processors are designed with dozens of control bits (also referred to as *chicken bits*) to manage system performance. Some chicken bits are exposed to the user (for example, disabling prefetching or simultaneous multithreading mode, or restricting power management), and others are known only by the manufacturer. Regardless, how these bits are set and tuned can have a significant impact on performance. Unfortunately, there are hundreds of these interdependent knobs, and tuning them by hand is impractical. However, self-tuning systems, either integrated into the operating system or as separate tools,[23]

that can dynamically adjust these bits according to application needs may be an innovative mechanism for achieving optimal performance. Best of all, these tuners can be deployed on-site, which means they do not gate product release to customers. Finally, the same techniques for fixing bugs via low-level software or implementing a virtual machine can also be used to adapt silicon to new applications. Hardware designers can enable and deploy new instructions and features through the same mechanisms used to patch around bugs. New versions of a JVM implementation, for example, may exploit optimizations that are relevant to new application areas.

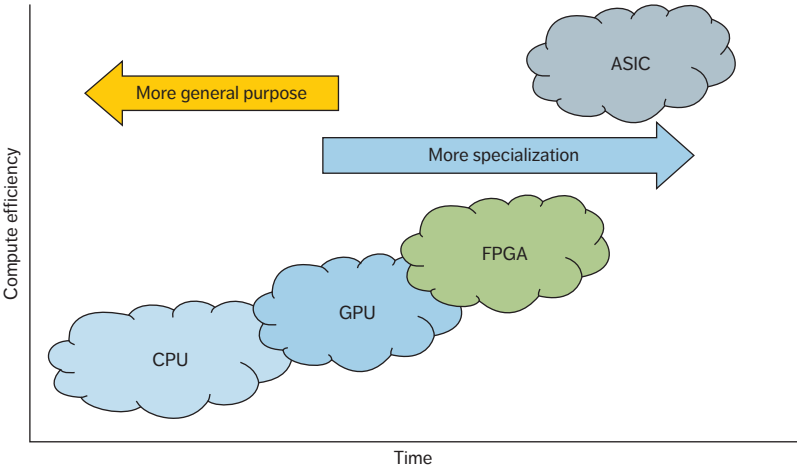

Figure 3. Specialization trend over time.

Computational Agility

So far, we have addressed agility at the processor level. However, with the advent of warehouse-scale systems driven by cloud computing, the processor becomes one piece of a larger computational problem. New companies entering the computing arena include numerous startups and large, established companies from outside the traditional chip design industry, such as Google, Microsoft, and Amazon. Few if any of these companies are choosing to go head-to-head in the general-purpose processor market with traditional designs such as Intel and AMD. Rather, they are achieving agility via specialized devices targeting narrower but highly relevant domains.

The need for specialization. The end of Dennard scaling and the slowdown and imminent demise of Moore's law drive the need for specialization, just as they demand agility in processor design. During the steep part of the S-curve for general-purpose processors, specialized architectures were quickly outpaced by these cheaper commodity devices. The slowing rate of improvement in general-purpose designs both creates opportunity for specialized architectures and drives demand, as customers can no longer rely on the commodity market to satisfy their computing needs.

A prerequisite for specialization is identifying an application or application domain narrow enough to benefit from specialization but large enough to justify a specialized device. Focusing on smaller and smaller domains (down to specific applications) increases the amount of potential performance uplift through specialization, while decreasing the potential market. To be successful, the total value created through specialization (roughly speaking, the value per device times the number of devices) must exceed the cost of developing the specialized device. By developing agile methodologies that reduce engineering costs, we can enable specialization for smaller domains and allow specialized devices to emerge sooner in growing markets.

Figure 3 shows the specialization trend over time, starting with CPUs and ending with custom ASICs. Cryptocurrency mining followed this trend,[24] and deep learning, one of the most prominent new markets attracting specialized architectures, is following suit. GPUs offer better performance than CPUs for certain tasks, such as training for AI, whereas state-of-the art field-programmable gate arrays (FPGAs) can outperform standard GPUs for certain computations such as low-precision arithmetic.[25] Finally, custom ASIC accelerators provide the highest performance efficiency.

Multiple startups such as Graphcore, Wave Computing, Nervana (now part of Intel), and Groq are developing or have developed customized deep learning accelerators that occupy the upper right corner of Figure 3. However, one of the earliest and most publicized deep learning accelerators is not from a startup but from an established company without a history of chip design. The Google Tensor Processing Unit (TPU) was developed in a short 15 months.[26] To achieve a rapid production cycle, Google used an older and more stable process technology (28 nm) and existing communication interfaces. The first-generation TPU was for internal use and had computational and memory bandwidth limitations. However, the TPU is now on its second iteration, and it not only supports higher computational capability and memory bandwidth, but will reportedly be made accessible to third parties.[27]

Even in an agile environment, the delay from the initial ASIC concept

to fully deployed device is measured in years. Once deployed, ASICs must continue to provide value for multiple additional years before replacement. Thus, an ASIC must accelerate a function that, from the point of conception, will still be valuable four to five years in the future. While some functions, such as compression and encryption algorithms, tend to be stable over these time frames, those in rapidly evolving fields such as deep learning may develop new and different requirements in the interval from design start to deployment. Stable, high-volume accelerators can easily justify an ASIC's higher nonrecurring engineering cost. Because an ASIC design needs larger markets and longer lifetimes, an ASIC accelerator typically includes as much flexibility as designers can afford in the form of configuration parameters, options, and software programmability.

To achieve a more agile acceleration framework, Microsoft took an unusual approach to specialization by focusing on FPGAs rather than ASICs for datacenter acceleration.[28] For a given accelerator design, an FPGA implementation could be several times slower and less energy efficient than an ASIC implementation. However, by using hardware devices that can be reprogrammed after deployment, Microsoft gains agility at the expense of computational efficiency. FPGA-based accelerators not only are tolerant to the changing requirements of a given application, but can be completely retargeted as new applications emerge or demand shifts. An FPGA accelerator design can afford to be less configurable and more customized to specific situations, as the design itself can be incrementally modified after initial deployment to address new circumstances. In this fashion, the FPGA's agility as a platform can be used to recover a portion of the efficiency that it sacrifices to an equivalent ASIC-based design.

FPGAs can also close the gap with ASICs by incorporating larger and more complex hard logic blocks on chip. Current FPGAs include multiply-accumulate units and even full microprocessor cores as hard logic. Researchers have also proposed devices that are mostly hard logic, but with configurable interconnect, referred to as *coarse-grained reconfigurable accelerators* (CGRAs).[29] The line between FPGAs and ASICs is further blurred by integrated multichip packages that incorporate both an FPGA and ASIC die.[30] The ability for customers to specify which ASICs are included in the package provides yet another dimension of flexibility.

The computational marketplace. Amazon has also developed hardware for internal consumption from custom routers to chipsets used in its servers.[31] This enables Amazon to optimize the hardware for its specific needs with full control of both the hardware and software stack. Amazon also provides hardware agility to its customers by offering platforms for custom programmable hardware as part of the AWS services plan.[32] The goal is to encourage companies to develop accelerators using Amazon's FPGA framework for internal use and/or sell the resulting computational capability to end customers on the AWS Marketplace. Amazon's EC F1 instances with FPGAs offer two significant benefits for custom solution developers. First, Amazon provides the FPGA hardware, tools, and infrastructure, significantly lowering the cost and convenience threshold for developing customized hardware. Second, Amazon provides a deployment model (via AWS) and a ready marketplace of potential customers for the final product. No longer are hardware developers restricted to products with a large Tier One customer base. They can rapidly develop and deploy niche hardware and test its viability in the AWS computational marketplace with many small customers across the country and the world. The computational marketplace scenario comes closest to achieving the rapid deployment model highlighted in *The Lean Startup*.[7] Finally, if any of these customized solutions become pervasive, they can eventually be reimplemented as an ASIC, as noted by Khazraee,[3] or integrated into a general-purpose processor architecture.

Standardized ecosystem. A successful computational marketplace requires standardized interfaces for interacting with accelerators. On the hardware side, current solutions from Amazon, Microsoft, Google, and others rely on PCIe for accelerator integration. PCIe has been the de facto standard for peripherals for many years, and a part of its success can be attributed to having an open standard. However, for processor designers wanting to create specialized accelerators, PCIe may not offer the tightly coupled memory system integration desired or required by the application. Proprietary coherent processor interconnects such as Intel's QPI and AMD's Infinity Fabric offer the memory system integration that a specialized accelerator might require, while Nvidia's NVLink is a proprietary interconnect for GPUs. Nonproprietary standards from different consortia such as OpenCAPI (www.opencapi.org), Gen-Z (www.genzconsortium.org), and CCIX (www.ccixconsortium.com) might also supplement PCIe as these standards evolve. What is clear, from the PCIe example, is that the new standard should be easily licensable and controlled by an open standards organization to enable a level playing field.

While we have thus far emphasized agility in hardware development and deployment, software agility is also a critical requirement. An environment in which hardware capabilities change and evolve rapidly is impossible to use unless low-level software can adapt equally rapidly, while providing stable

APIs to higher-level services so that the bulk of the code base can remain independent of the underlying implementation's details. Software stacks can provide additional agility when they help to automate the mapping of applications to accelerators, and enable hardware bug workarounds to cope with issues that may slip through an accelerated development and testing schedule.

Processor architecture has changed significantly over the past few decades with the advent of multicore designs, design for low power, heterogeneous systems, and many-core processors that can run a hundred or more threads. With cloud computing and the emerging customizable marketplace of products, we are once again witnessing a sea change in the way computing takes place.

In this article, we have made a case for agility because we cannot predict the future with any level of accuracy. We need agility not only for rapid evolution of conventional architecture, but also for lowering the barrier for specialized architectures. As Bill Gates once noted, "We always overestimate the change that will occur in the next two years and underestimate the change that will occur in the next ten. Don't let yourself be lulled into inaction."[33] As architects, we must develop the infrastructure and mindset that enable us to be agile and take risks in order to evolve with a rapidly changing environment and create the next disruptive technology.

References

1. C.M. Christensen, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Harvard Business School Press, 1997.
2. "A Look Back at Single-Threaded CPU Performance," blog, 8 Feb. 2012; http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance.
3. M. Khazraee et al., "Moonwalk: NRE Optimization in ASIC Clouds," *Proc. 22nd Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 511–526.
4. J. Hruska, "Intel Formally Kills its Tick-Tock Approach to Processor Development," blog, 23 Mar. 2016; www.extremetech.com/extreme/225353-intel-formally-kills-its-tick-tock-approach-to-processor-development.
5. T.P. Morgan, "Qualcomm Fires ARM Server Salvo, Broadcom Silences Guns," 7 Dec. 2016; www.nextplatform.com/2016/12/07/qualcomm-fires-arm-server-salvo-broadcom-silences-guns.
6. R. Brueckner, "Cavium ThunderX2 Processors Power New Baymax HyperScale Server Platforms," blog, 29 May 2017; http://insidehpc.com/2017/05/cavium-thunderx2-processors-power-new-baymax-hyperscale-server-platforms.
7. E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Business,* Crown Publishing Group, 2011.
8. C. Murphy, "Facebook Guru and Agile Pioneer Kent Beck Reveals the Mind of the Modern Programmer," *Forbes*, 9 Jan. 2017; www.forbes.com/sites/oracle/2017/01/09/facebook-guru-and-agile-pioneer-kent-beck-reveals-the-mind-of-the-modern-programmer.
9. J. Bird, "This Is How Facebook Develops and Deploys Software. Should You Care?" blog, 4 Sept. 2013; http://dzone.com/articles/how-facebook-develops-and.
10. M. Humrick, "Exploring DynamIQ and ARM's New CPUs: Cortex-A75, Cortex-A55," blog, 29 May 2017; www.anandtech.com/show/11441/dynamiq-and-arms-new-cpus-cortex-a75-a55.
11. P. Gelsinger et al., "Such a CAD! Coping with the Complexity of Microprocessor Design at Intel," *IEEE Solid-State Circuits*, vol. 2, no. 3, 2010, pp. 32–43.
12. M. Ziegler, R. Puri, and B. Philhower, "POWER8 Design Methodology Innovations for Improving Productivity and Reducing Power," *Proc. IEEE Custom Integrated Circuits Conf.*, 2014, pp. 1–9.
13. A. Patrizio, "Intel Shakes Up Its Chip Design Process," blog, 23 May 2014; www.itworld.com/article/2699164/hardware/intel-shakes-up-its-chip-design-process.html.
14. V. Rao and E. Smith, "Facebook's New Front-End Server Design Delivers on Performance without Sucking Up Power," blog, 9 Mar. 2016; http://code.facebook.com/posts/1711485769063510/facebook-s-new-front-end-server-design-delivers-on-performance-without-sucking-up-power.
15. M. Abramovici and P. Bradley, "A New Approach to In-System Silicon Validation and Debug," *EE Times*, 16 Sept. 2007; www.eetimes.com/document.asp?doc_id=1276099.
16. "Pentium FDIV Bug," blog; www.cs.earlham.edu/~dusko/cs63/fdiv.html.
17. S. Wasson, "Errata Prompts Intel to Disable TSX in Haswell, Early Broadwell CPUs," blog, 12 Aug. 2014; http://techreport.com/news/26911/errata-prompts-intel-to-disable-tsx-in-haswell-early-broadwell-cpus.
18. K. Kubicki, "Understanding AMD's 'TLB' Processor Bug," blog, 5 Dec. 2007; www.dailytech.com/Understanding++AMDs+TLB+Processor+Bug/article9915.htm.

19. *Desktop 4th Generation Intel Core Processor Family, Desktop Intel Pentium Processor Family, and Desktop Intel Celeron Processor Family*, report 328899-037US, Mar. 2017.
20. "Debug Architecture Overview," *ARM*, 2017; http://developer.arm.com/products/architecture/debug-architecture
21. T. Lindholm et al., *The Java Virtual Machine Specification: Java SE 7 Edition*, 28 Feb. 2013.
22. F.G. Soltis, *Inside the AS/400: Featuring the AS/400e Series*, 2nd ed., 29th Street Press, 1997.
23. T. Morad, *The Era of Self-Tuning Servers*, 7 Feb. 2017; www.hpcadvisorycouncil.com/events/2017/stanford-workshop/pdf/Morad_TheEraOfSelfTuningServers.pdf.
24. P. Jama, "The Future of Machine Learning Hardware," blog, 10 Sept. 2016; http://hackernoon.com/the-future-of-machine-learning-hardware-c872a0448be8.
25. L. Barney, "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Learning?" blog, 21 Mar. 2017; www.nextplatform.com/2017/03/21/can-fpgas-beat-gpus-accelerating-next-generation-deep-learning.
26. K. Sato, C. Young, and D. Patterson, "An In-Depth Look at Google's First Tensor Processing Unit (TPU)," blog, 12 May 2017; http://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu.
27. P. Teich, "Under the Hood of Google's TPU2 Machine Learning Clusters," blog, 22 May 2017; www.nextplatform.com/2017/05/22/hood-googles-tpu2-machine-learning-clusters.
28. A. Putnam et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," blog, 1 June 2014; www.microsoft.com/en-us/research/publication/a-reconfigurable-fabric-for-accelerating-large-scale-datacenter-services.
29. M. Gao and C. Kozyrakis, "HRL: Efficient and Flexible Reconfigurable Logic for Near-Data Processing," *Proc. IEEE Int'l Symp. High Performance Computer Architecture*, 2016, doi: 10.1109/HPCA.2016.7446059.
30. M. Deo, *Enabling Next-Generation Platforms Using Intel's 3D System-in-Package Technology*, white paper WP-01251-1.5, Intel, Aug. 2017.
31. D. Richman, "Amazon Web Services' Secret Weapon: Its Custom-Made Hardware and Network," blog, 19 Jan. 2017; www.geekwire.com/2017/amazon-web-services-secret-weapon-custom-made-hardware-network.
32. "Amazon EC2 F1 Instances, Customizable FPGAs for Hardware Acceleration Are Now Generally Available," blog, 19 Apr. 2017; http://aws.amazon.com/about-aws/whats-new/2017/04/amazon-ec2-f1-instances-customizable-fpgas-for-hardware-acceleration-are-now-generally-available.
33. B. Gates, *The Road Ahead*, Viking Penguin, 1996.

Srilatha Manne is a principal hardware architect at Cavium. Contact her at bobbiemanne12@gmail.com.

Bryan Chin is a lecturer in the Computer Science and Engineering Department at the University of California, San Diego. Contact him at b5chin@ucsd.edu.

Steven K. Reinhardt is a partner hardware engineering manager at Microsoft. Contact him at stever@microsoft.com.