Designing for Responsiveness with Computational Sprinting

This article explores computational sprinting, wherein a mobile device temporarily exceeds sustainable thermal limits to provide a brief, intense burst of computation in response to user input. By enabling tenfold more computation within the timescale of human patience, sprinting has the potential to fundamentally change the user experience that an interactive application can provide.

Arun Raghavan
University of Pennsylvania

Yixin Luo
Anuj Chandawalla
Marios Papaefthymiou
Kevin P. Pipe
Thomas F. Wenisch
University of Michigan

Milo M. K. Martin
University of Pennsylvania

Future technology generations are predicted to continue to provide more transistors, but unfortunately Dennard scaling—the concomitant reduction of threshold and supply voltages—has stalled. Without lower voltages, power density is projected to increase with each new process generation on a trajectory that far outstrips improvements in the ability to dissipate heat. In thermally constrained environments such as mobile devices, in particular, the confluence of these trends has led to a phenomenon referred to as the *utilization wall* or *dark silicon*—before the end of the decade, a mobile chip might have up to 10 times as many transistors than can be used on a sustained basis.[1-4]

Not all applications, however, call for peak performance to be sustained indefinitely. For rich, interactive mobile applications (such as image processing or voice recognition), *responsiveness*—how long a user must wait after initiating a command—likely captures user satisfaction better than sustained performance. Many such applications are characterized by short bursts of intense computation punctuated by long idle periods waiting for user input, especially in mobile settings. In this work, we pose the question, "What would a system look like if designed to provide responsiveness during bursts rather than with a singular focus on sustainable performance?" Our answer is *computational sprinting*: providing a brief, intense performance boost by far exceeding sustainable thermal limits to rapidly complete response-time-critical processing before the chip temperature rises too high. One way to sprint is to increase voltage and frequency beyond sustainable power levels. In fact, today's chips already use such strategies. For example, recent Intel chips opportunistically increase frequency and voltage by small margins (about 25 percent for tens of seconds) when thermal conditions allow.[5]

We see a more radical opportunity: activating tens of dark-silicon reserve cores to provide an order of magnitude performance improvement for bursty interactive applications—lighting up dark silicon for a brief instant when applications demand performance. Such intense sprinting has the potential to significantly change the user experience that an interactive application can provide. However, engineering a system for intense sprints raises a

Published by the IEEE Computer Society

host of research and design challenges, spanning from thermal and electrical engineering to application and system software design. This article explores thermal and electrical design considerations and demonstrates sprinting's potential through simulation studies of a sprint-enabled system. As a concrete objective, we target a design that can provide a more than tenfold improvement in responsiveness (that is, completing a 10-second computational task in less than 1 second) by sprinting with 16 1-W cores in a system that can sustain operation of only a single 1-W core.

Computational sprinting overview

We illustrate computational sprinting using an example multicore processor, which, under typical mobile-device thermal constraints, can sustain the operation of only a single core. We initiate sprints by activating additional on-chip cores when a response-time-critical task begins. Because the total chip power draw in this sprinting mode exceeds the system cooling capability, continued operation will cause the chip temperature to rise and eventually exceed safe operating temperatures. To prevent overheating, the runtime system detects when the temperature nears maximum and throttles operation to the sustainable single-core mode by migrating all application threads to a single core and powering down the other cores. Any remaining computation is completed in this nonsprinting mode, after which the last core is also powered down, letting the system cool. The potential responsiveness benefit of sprinting grows with the amount of computation that can be performed while sprinting.

The opportunity for sprinting arises from the time it takes for the chip to heat from its nominal temperature to the safe maximum. This sprint interval is a result of thermal capacitance—the ability of materials to store heat. The larger the thermal capacitance, the more slowly the temperature rises; conversely, the larger the input power, the more rapidly the temperature rises. Nominal operation is sustainable because the input power balances with dissipation to the environment at an equilibrium temperature below the safe limit. With sprinting, in contrast, the instantaneous total chip power greatly exceeds what a mobile system can dissipate. The thermal capacitance therefore determines the maximum safe duration for a sprint of a given intensity (see the "Computational Sprinting Operation" sidebar for illustration).

Thermal design for sprinting

Although thermal capacitance provides the opportunity for sprinting, conventional systems are designed with a focus on reducing thermal resistance and not on providing thermal capacitance; any available capacitance arises incidentally from the specific heat of materials in the die and surrounding packaging. An integrated heat spreader commonly found within desktop-class processor packages can provide sufficient thermal capacitance for several seconds of sprinting at high intensity,[6] but mobile chips and packages are smaller, and often do not contain heat spreaders. Hence, thermal capacitance near the die can be quite limited, whereas other components of the system (such as the phone case) are too distant for heat to spread to them quickly (that is, such components are separated from the die by large thermal resistances that slow heat flow). Figures 1a and 1b illustrate such a system.

To extend sprint duration in mobile systems, we explore adding a small amount of phase-change material (PCM) close to the die (Figure 1c and C_{pcm} in Figure 1d). PCMs substantially increase thermal capacitance because of the latent heat of phase change (for example, solid-to-liquid fusion). When this material melts, the temperature is held constant at the melting point until all material has melted. Selecting a PCM with a melting point below the maximum safe chip temperature could substantially extend sprint duration. A PCM such as icosane (also known as candle wax), for example, has an appropriate melting point and large latent heat (241 J/g). Conservatively, for a PCM with a latent heat of only 100 J/g and density of 1 g/cm^3, about 150 milligrams of material (2.3-mm-thick block of PCM in contact with a 64 mm^2 die) absorbs approximately 16 J of heat during melting, which is enough to allow 16 cores to operate at 1 W each for 1 second at a constant temperature.

Computational Sprinting Operation

Figure A1 shows the thermal response of a conventional system starting from idle. The single core becomes active at time t_{on}, causing system temperature to rise asymptotically toward T_{max}. The system computes until time t_{done}, at which time the core becomes idle and the temperature begins to fall asymptotically toward $T_{ambient}$. For a given $T_{ambient}$ and T_{max}, the maximum sustainable thermal power is determined by total thermal resistance of the package and heatsink (but is independent of the thermal capacitance). In contrast, both the thermal resistance and capacitance determine the rate at which the temperature rises (and falls).

Figure A2 shows the sprinting mode operation. In this example, the system is initially idle and the system temperature matches that of the ambient environment ($T_{ambient}$). At time t_{on}, an external event (such as user input) triggers demand for a burst of computation, and it initiates a parallel sprint by activating all cores. Because the heat generation is greater than in sustained operation, the chip temperature rises more quickly. In this example, the temperature reaches T_{max}, and thus the computation exceeds the system's maximum sprint duration. When the temperature reaches T_{max}, the system terminates the sprint by disabling all but one core (at time t_{one}); any remaining work is completed by this core. During this interval (from time t_{one} to t_{done}), because the thermal system is designed to sustain single-core operation, temperature remains stable near T_{max}. When the computation is done (t_{done}), all cores become idle, so the system begins to cool.

Increasing sprint duration requires increasing the system's thermal capacitance. One way to increase the thermal capacitance is by placing a block of phase-change material close to the die. Adding such material increases the thermal capacitance both by its specific heat (the amount of energy to change the temperature of a gram of the material by one degree) and—more importantly—its latent heat of fusion (the amount of energy to melt a gram of the material). Figure A3 shows the same computation in sprint mode on such an augmented system. The temperature rises as before, but when the temperature reaches the melting point of the material (T_{melt}), the extra thermal energy injected into the system is absorbed by the melting process, letting the system continue to execute for a period without increasing the temperature. Only once the material has fully melted does the temperature begin to rise again. Similarly, when the system cools, its temperature is constant as the material returns to its solid phase. Overall, in this example the additional thermal capacitance lets the system perform significantly more computation during the sprint interval.

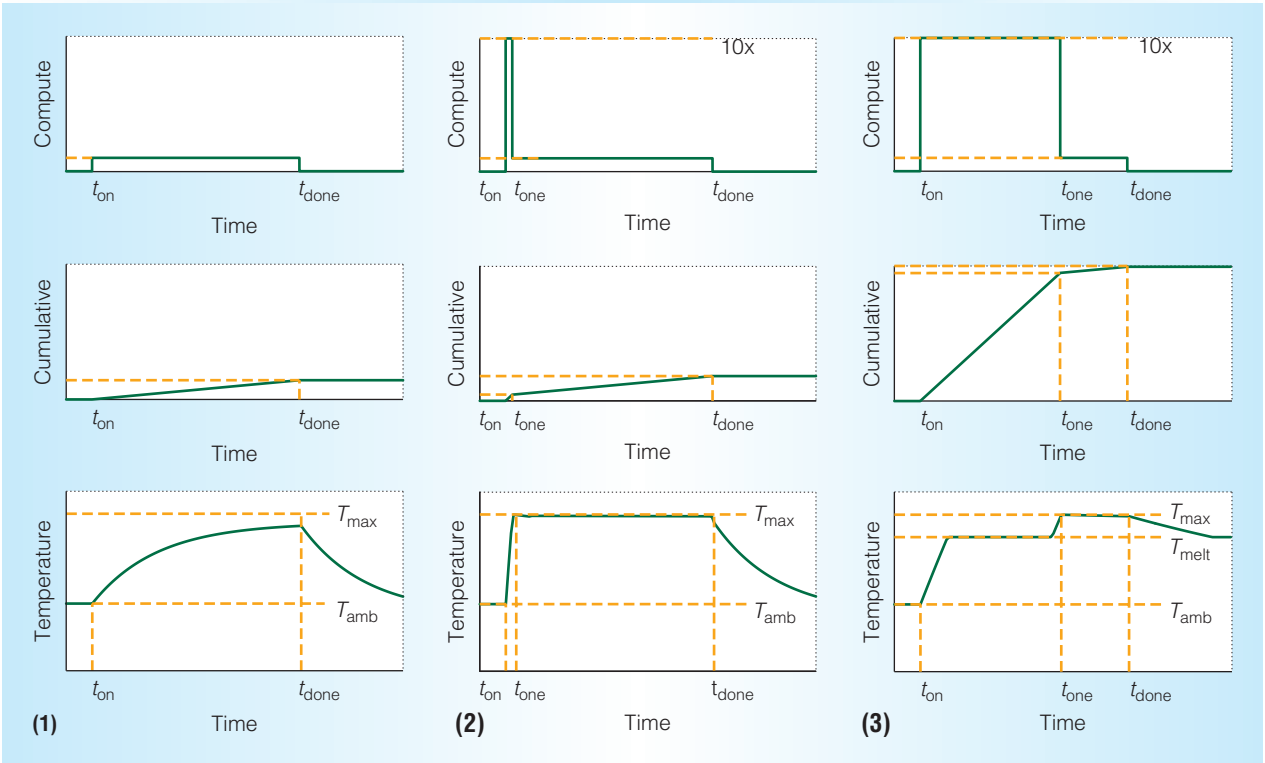

Figure A. Sprinting operation. Instantaneous compute throughput (top row), cumulative computation (middle row), and temperature (bottom row) over time for three execution modes: sustained (1), sprint (2), and sprint augmented with phase-change material (3).

Unfortunately, many PCMs (including icosane) have poor thermal conductivity, which can limit sprint intensity (due to slow heat spreading within the PCM) and reduce the maximum sustainable power of nonsprint operation (by increasing the thermal resistance to the environment). Integrating a heat-spreading network (such as a metal mesh) within the PCM can facilitate rapid heat transfer (R_{pcm} in Figure 1d). For even lower thermal resistance, recent work has examined the use of metal or diamond microchannels to achieve PCM-based heat sinking on a timescale of 100 µs.[7] In addition to enhancing thermal conductivity, such an integrated mesh can improve the PCM's mechanical robustness to reduce wear-out effects such as the formation of cracks or voids that might compromise the PCM's effectiveness.

Using PCM to magnify available thermal capacitance increases sprint duration, but it also increases the post-sprint cooldown time by a similar factor. Until the PCM fully resolidifies, the system's ability to sprint will be limited. Ultimately, the cooldown rate—and the sustainable performance—remains firmly bound by the thermal resistance to the ambient ($R_{package}$ and $R_{convection}$ in Figure 1d). Therefore, the average power over the sprint and idle durations is still limited by the chip's sustainable power; sprinting merely shifts the energy budget from future idle periods to intensify and shorten active computation for bursty applications.

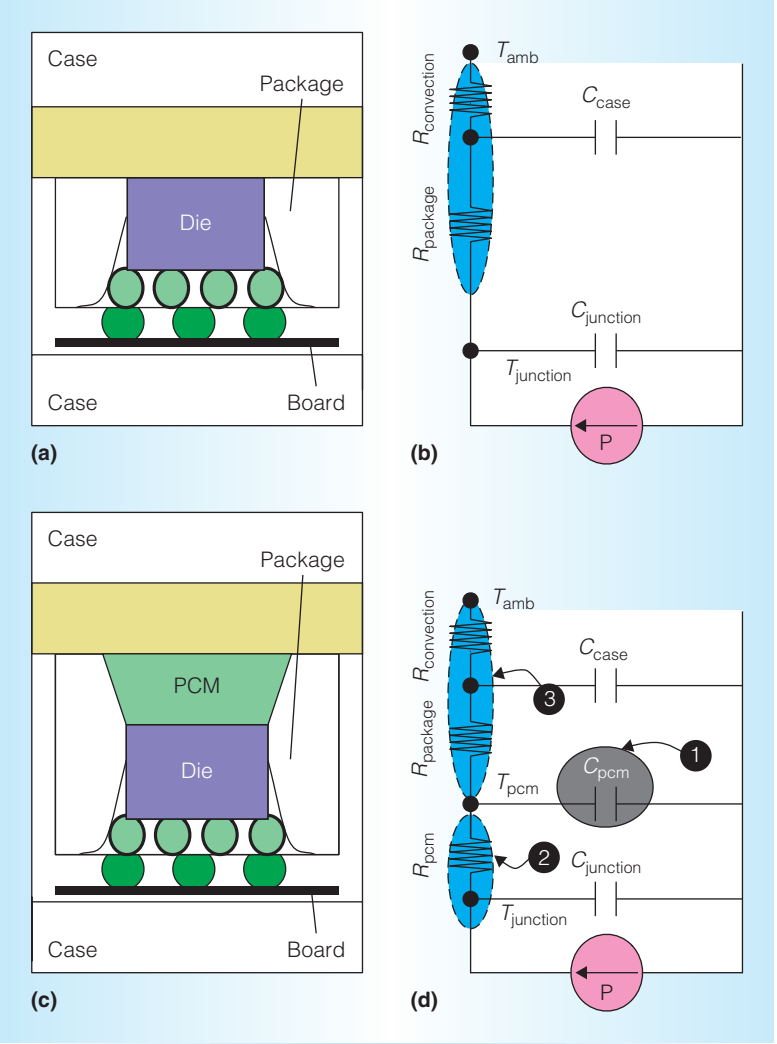

Figure 1. A mobile system's thermal components (a) and its thermal-equivalent circuit model (b). In (c) and (d), the system is augmented with a block of phase-change material (PCM). The amount of computation possible during a sprint is primarily determined by the PCM's thermal capacity (1). The maximum sprint power is determined by the thermal resistance into the PCM (2), whereas the sustained power is determined by the total resistance to the ambient (2 + 3). The thermal resistance from the PCM to the ambient (3) determines how quickly the system cools after a sprint.

Electrical design for sprinting

In addition to thermal considerations, a system designed for sprinting must ensure that transitioning into or out of sprint mode does not compromise on-chip voltage rail stability (to preserve state and prevent timing errors) despite massive in-rush current when activating (or deactivating) many cores. Using electrical models of board-, chip-, and package-level RLC components, we found that activating 16 1-W cores at once caused fluctuations in excess of 2 percent of nominal supply voltages, which would likely result in incorrect operation. By simply activating cores gradually one at a time over a 128 µs ramp-up interval, voltages stayed within tolerance. As this interval is much smaller than our target sprint duration, the impact on attainable sprint speedup is negligible.

The electrical supply must also deliver the peak currents demanded by sprinting. Conventional phone batteries and associated voltage regulators supply at most a few amps of power at peak, with higher currents precluded by internal thermal constraints. Because intense sprints require

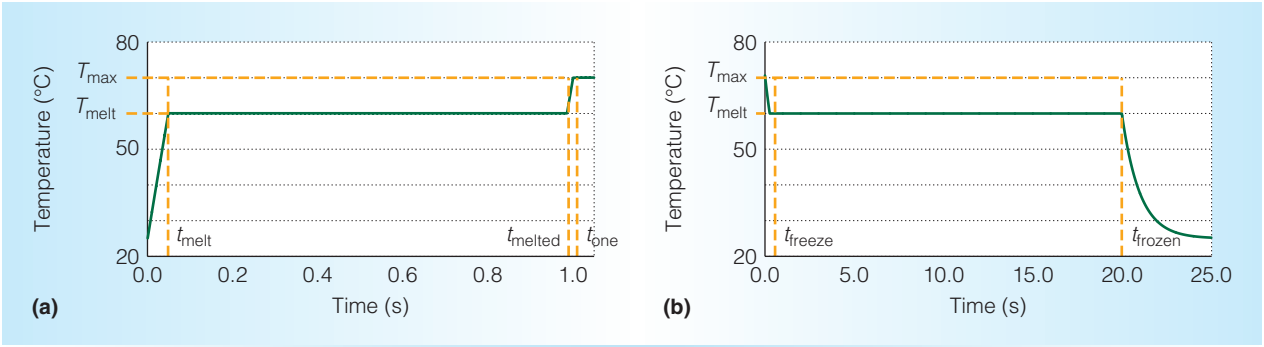

Figure 2. Transient behavior of initiation and termination of sprinting. Sprint initiation (a); post-sprint cooldown (b). While sprinting (a), die temperature rises rapidly to the melting point of the phase change material (T_{melt}) and remains at T_{melt} until all the phase change has occurred (time t_{melted}), after which it rapidly rises once again. When the temperature reaches T_{max}, the system throttles down to sustainable operation at time t_{one}.

much higher peak current, we surveyed alternative energy supplies capable of high discharge rates. One potential approach is to employ hybrid battery–ultracapacitor power sources to meet the peak current demands of sprinting.[8] Ultracapacitor technology is an area of active research,[9] and the latest ultracapacitors show the potential of meeting the requirements for energy capacity, peak current, and form factor with negligible energy losses due to leakage. For example, a 25F Nesscap ultracapacitor weighs 6.5 g, can store 182 joules, and provides a peak current of 20 A with a total leakage current below 0.1 mA.

A further design challenge lies in delivering the necessary peak currents from the off-chip power source over the chip pins. Whereas 100-A peak currents are commonly sustained in desktop and server package and socket designs, peak currents of 16 A exceed the norm for mobile devices. Providing such peak currents will likely require more power and ground pins, which can increase the package cost. On-chip voltage regulators combined with higher pin voltages could help satisfy peak current demands.[10]

Evaluating the potential of sprinting

To better understand the performance of sprinting, we experimentally investigated how much sprint duration can be extended by integrating PCM, and we analyzed the responsiveness benefits achievable with sprinting by activating cores on a PCM-augmented system.

We investigated the benefits of using PCM for sprinting with the thermal models shown in Figure 1. Figures 1a and 1b illustrate the thermal network of a conventional system in which heat flows from the die junction to the ambient through parallel paths that include a number of components. We based our model—in particular, the values we use for the thermal resistances and capacitances of various components (such as the circuit board, battery, and case)—on a physically validated model of a mobile phone.[11] The model assumes ideal heat transfer within the PCM, and that the melting point of the PCM (for example, 60°C) is sufficiently high such that it melts during sprints but not during sustained single-core operation.

Figure 2a illustrates the transient thermal behavior of a 16 W sprint on a 1 W thermal design power (TDP) system under the thermal model with 150 milligrams of PCM. The chip temperature initially rises rapidly, then plateaus for approximately 0.94 seconds during the phase change, subsequently rising to the maximum safe temperature (set at 70°C for these simulations). Factoring in all thermal capacitance sources, the sprint duration is a little over 1 second. Between sprints, the PCM resolidifies as the system cools. Figure 2b shows the projected cooldown behavior. The exponential nature of heat transfer results in the chip temperature nearing its initial temperature after about 24 seconds.

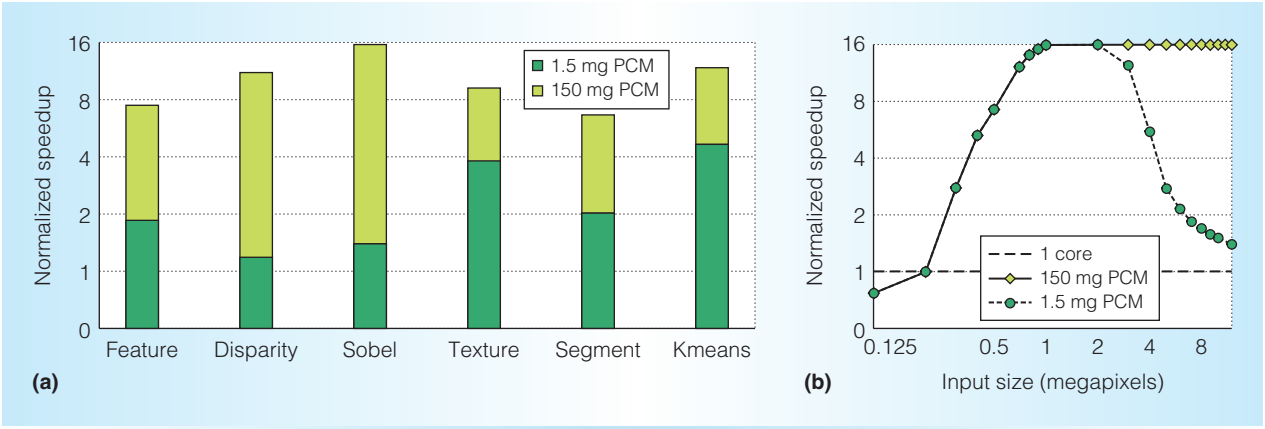

Figure 3. Comparison of responsiveness when sprinting with 16 cores over a 1-core baseline. Speedup with sprinting over all workloads (a); sensitivity of speedup (*y*-axis) to the computation length (*x*-axis) for the `sobel` workload (b). In both graphs, when configured with 150 mg of PCM, all computations complete before reaching maximum temperatures. In contrast, when configured with 1.5 mg of PCM, the speedups are lower because the system reaches maximum temperature and thus the computation must be completed in the sustainable (nonsprinting) 1-core baseline mode.

To study the performance potential of sprinting by activating otherwise idle cores, we performed simulations of a many-core system running a suite of vision and image-analysis kernels. The workloads are intended to represent the processing performed by applications such as visual search.[12] We augmented this performance model with a dynamic energy model that associates energy with the type of instruction being executed. During execution, we sampled the energy consumed by each core every 1,000 cycles to drive the thermal network model of the PCM-augmented system described earlier.

To demonstrate the potential for improvement in responsiveness (that is, reduction in time to complete a short task), we simulated the parallel workloads on 16 cores with 150 milligrams of PCM. The total height of the bars in Figure 3a shows an average parallel speedup of 10.2× over a single-core nonsprinting baseline. The sprinting and nonsprinting configurations both have the same last-level cache and memory bandwidth constraints, but the sprinting configuration can use the additional cores (dark silicon) to improve responsiveness.

To explore the effect of limited sprint duration with tractable simulation times, we reduced the system's thermal capacitance by reducing the amount of PCM by 100×. The bottom segment of the bars in Figure 3a represents the speedup for this design point. These simulations show smaller speedups, because under this more constrained thermal configuration all workloads exhaust the sprint duration and are forced to execute some of the computation in the post-sprint single-core mode.

Figure 3b shows the impact of the thermal design on speedup (*y*-axis) for the `sobel` kernel as the amount of computation per sprint is increased by increasing the image resolution (*x*-axis). Except for the lowest-resolution images, where the overheads of creating threads outweigh parallel speedup due to limited computational demand, `sobel` scales linearly up to 16 cores. For the fully-sized PCM (150 mg of PCM), the system can sustain the sprint for the entire computation at all image resolutions. However, for the artificially limited design point (1.5 mg of PCM), the graph shows that speedup drops off as the fixed-sized sprint can handle less of the total computation.

Energy efficiency is, of course, a paramount concern for battery-powered mobile devices, so we investigated the impact of sprinting on energy efficiency. Whereas sprinting by increasing voltage and frequency can significantly degrade energy efficiency, results from our simple dynamic energy

model indicate that sprinting by activating cores is energy-neutral whenever a workload exhibits reasonable parallel speedup (see our HPCA 2012 paper for more details).[13] This model ignores static energy and background power. In fact, most chips exhibit a fixed background power to operate any cores owing to "uncore" components, such as the interconnect and large last-level cache. Hence, the marginal power required to activate additional cores is much lower than the single-core power draw. In systems with a substantial background power, we have found that by "racing to idle"—that is, using more cores to complete the computation faster and enter an energy-conserving sleep mode sooner—sprinting can actually reduce the total energy needed to perform a computation.[6]

Computational sprinting offers a new paradigm for designing digital systems for responsiveness rather than for sustained operation. Pushing chips past sustainable power limits has the potential to provide far greater computational power for interactive mobile applications. Whereas users, application developers, and system designers have traditionally reasoned solely about the maximum sustainable performance, this conventional wisdom is challenged by the observation that peak performance can vary drastically over time and might depend on temperature conditions or recent user behavior. Further investigation is needed to determine how sprinting impacts user experience and how systems might best use this new capability. MICRO

Acknowledgments

We thank Andrew Hilton for use of his simulation infrastructure, Jason Clemons for providing the feature workload, and Dan Sorin, Karu Sankaralingam, Laurel Emurian, Lei Shao, Yatin Manerkar, and the anonymous reviewers for their comments on this work. This material is based on work supported by the US National Science Foundation under grants CCF-0644197, CCF-0811320, CCF-0815457, and CCF-0916714.

References

1. H. Esmaeilzadeh et al., "Dark Silicon and the End of Multicore Scaling," *Proc. 38th Ann. Int'l Symp. Computer Architecture* (ISCA 11), ACM, 2011, pp. 365-376.
2. R. Merritt, "ARM CTO: Power Surge Could Create 'Dark Silicon,'' *EE Times,* 22 Oct. 2009; http://www.eetimes.com/electronics-news/4085396/ARM-CTO-power-surge-could-create-dark-silicon.
3. M.B. Taylor, "Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse," *Proc. 49th ACM/EDAC/IEEE Design Automation Conf.* (DAC 12), IEEE CS, 2012, pp. 1131-1136.
4. G. Venkatesh et al., "Conservation Cores: Reducing the Energy of Mature Computations," *Proc. 15th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 10), ACM, 2010, pp. 205-218.
5. Intel, *Turbo Boost Technology 2.0,* http://www.intel.com/technology/turboboost/index.htm.
6. A. Raghavan et al., "Computational Sprinting on a Hardware/Software Testbed," *Proc. 18th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 13), 2013, ACM, pp. 155-166.
7. S.P. Gurrum, Y.K. Joshi, and J. Kim, "Thermal Management of High Temperature Pulsed Electronics Using Metallic Phase Change Materials," *Numerical Heat Transfer, Part A: Applications,* vol. 8, no. 42, 2002, pp. 777-790.
8. M. Pedram et al., "Hybrid Electrical Energy Storage Systems," *Proc. 16th ACM/IEEE Int'l Symp. Low Power Electronics and Design* (ISLPED 10), ACM, 2010, pp. 363-368.
9. J. Schindall, "The Charge of the Ultracapacitors," *IEEE Spectrum,* vol. 44, no. 11, 2007, pp. 42-46.
10. W. Kim et al., "System Level Analysis of Fast, Per-Core DVFS Using On-Chip Switching Regulators," *Proc. 14th IEEE Int'l Symp. High-Performance Computer Architecture* (HPCA 08), IEEE CS, 2008, pp. 123-134.
11. Z. Luo et al., "System Thermal Analysis for Mobile Phone," *Applied Thermal Eng.,* vol. 28, nos. 14-15, 2008, pp. 1889-1895.

12. B. Girod et al., ''Mobile Visual Search,'' *IEEE Signal Processing Magazine,* vol. 28, no. 4, 2011, pp. 61-76.
13. A. Raghavan et al., ''Computational Sprinting,'' *Proc. IEEE 18th Symp. High-Performance Computer Architecture* (HPCA 12), IEEE CS, 2012, doi:10.1109/HPCA.2012.6169031.

Arun Raghavan is a PhD candidate in the Department of Computer and Information Science at the University of Pennsylvania. His research interests are in parallel computer architectures and programming models. Raghavan has a BEng in electronics and communications engineering from R.V. College of Engineering, India.

Yixin Luo is a PhD candidate in the Computer Science Department at Carnegie Mellon University. His research interests include parallel computer architecture and efficient storage and memory systems. Luo has a BS in computer engineering from the University of Michigan, where he performed the research for this article.

Anuj Chandawalla is an engineer at Qualcomm Technologies. His research interests include low-power circuit design and high-performance applications. Chandawalla has an MS in electrical engineering from the University of Michigan, where he performed the research for this article.

Marios Papaefthymiou is a professor in the Department of Electrical Engineering and Computer Science and Chair of Computer Science and Engineering at the University of Michigan. He is also a cofounder and chief scientist of Cyclos Semiconductor, a start-up company specializing in energy-efficient chips for power-critical applications. Papaefthymiou has a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology.

Kevin P. Pipe is an associate professor in the Department of Mechanical Engineering and holds joint appointments with the Department of Electrical Engineering and Computer Science and the Applied Physics Program at the University of Michigan. His research interests include microscale heat transfer, optoelectronic devices, thermoelectric energy conversion, scanning probe techniques, photovoltaic energy conversion, and organic and hybrid organic/inorganic devices. Pipe has a PhD in electrical engineering from the Massachusetts Institute of Technology.

Thomas F. Wenisch is the Morris Wellman Faculty Development Assistant Professor of Electrical Engineering and Computer Science at the University of Michigan and a member of the Advanced Computer Architecture Lab. His research interests focus on computer architecture with emphasis on multiprocessor and multicore systems, multicore programmability, smartphone architecture, datacenter architecture, and performance evaluation methodology. Wenisch has a PhD in electrical and computer engineering from Carnegie Mellon University.

Milo M. K. Martin is an associate professor in the Department of Computer and Information Science at the University of Pennsylvania. He coleads Penn's Computer Architecture and Compilers Group. His research interests include multiprocessor and multicore computer architecture, compiler and hardware support for security, and programming models for next-generation architectures. Martin has a PhD in computer science from the University of Wisconsin–Madison.

Direct questions and comments about this article to Arun Raghavan, LVN 302, 3330 Walnut Street, Philadelphia PA 19104; arraghav@cis.upenn.edu.