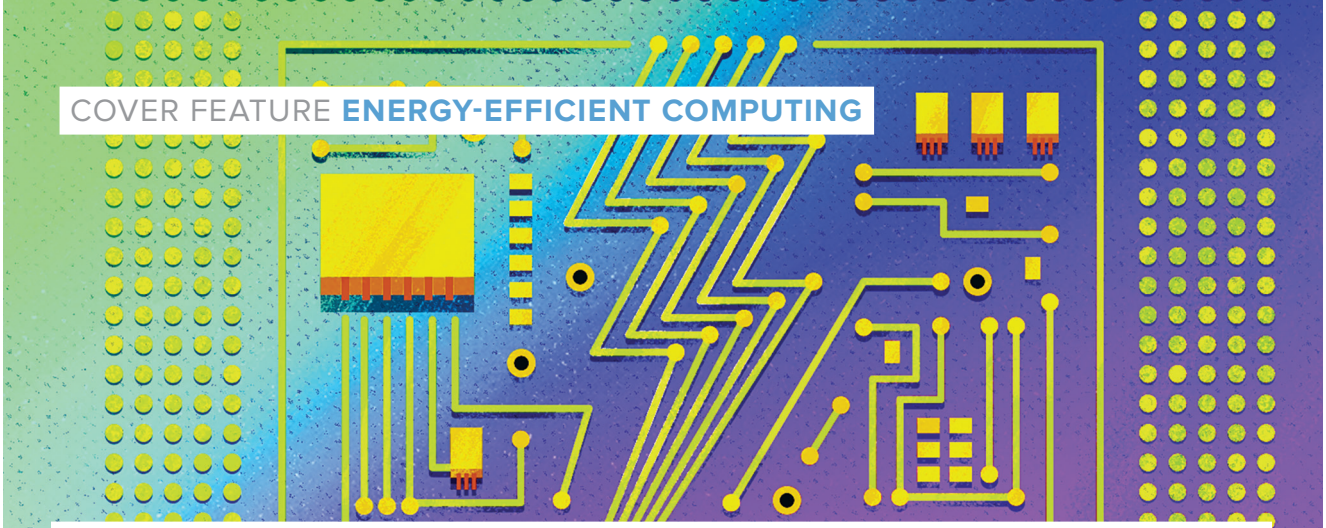

H-EARtH: Heterogeneous Multicore Platform Energy Management

Efraim Rotem, Intel and Technion—Israel Institute of Technology
Uri C. Weiser, Avi Mendelson, and Ran Ginosar, Technion—Israel Institute of Technology
Eliezer Weissmann and Yoni Aizik, Intel

By scheduling each workload according to its most advantageous core and managing voltage and frequency, the heterogeneous energy-aware race to halt (H-EARtH) algorithm optimizes CPU platform energy.

Modern computers' energy consumption is a major concern, given growing datacenter and client computer deployment. Mechanisms such as dynamic voltage and frequency scaling (DVFS) are commonly used to achieve energy-efficient computing while maintaining performance by controlling the CPU's voltage and frequency. Multicore heterogeneous processors have been introduced for use in computer platforms to further improve performance and energy efficiency.[1] Wide out-of-order cores, potentially at high voltage and frequency, are used when high performance is needed, while slow and lower power cores at lower voltage and frequency achieve better energy efficiency[2–4] when less performance is sufficient. Furthermore, because a "simple" core can be implemented using much less area than a "complex" core, it is possible to fit and activate many small cores within the same area as the complex one, resulting in superior multithreaded workload performance. Combining a heterogeneous processor with DVFS capability offers multidimensional energy and performance control.

The heterogeneous energy-aware race to halt (H-EARtH) algorithm offers a runtime scheduling policy for selecting the best core for a given application, and a power-management algorithm for selecting the voltage and frequency point on the selected core that achieves optimal platform energy.

PLATFORM ENERGY

Controlling CPU power and energy has a limited effect on a computer platform's overall energy efficiency because of the energy consumption of other platform

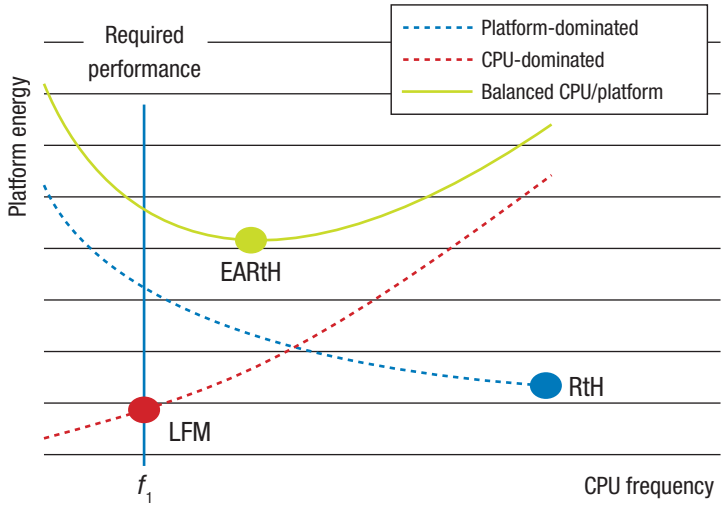

FIGURE 1. Conceptual total energy in three different scenarios. The minimum energy point depends on which portion of the platform dominates power consumption: low frequency mode (LFM) for CPU dominance, race to halt (RtH) for when the rest of the platform dominates, and energy-aware race to halt (EARtH) point for when power is balanced.

components. Although lowering the core's voltage and frequency decreases processor energy by ~(*freq***1**/*freq***2**)^2, computation time is lengthened by ~*freq***2**/*freq***1**, resulting in other platform components consuming more energy.[5,6] Using a small core also lengthens the execution runtime, resulting in higher platform energy.

Figure 1 shows a platform's total energy consumption as a function of CPU voltage and frequency. When CPU power dominates total power, the energy follows the red dotted curve in Figure 1, and minimum energy consumption is achieved when the CPU operates at its lowest voltage and frequency. Multicore processors further increase performance at the cost of higher power, making the processor power more dominant than the rest of the platform. This behavior is often observed in big-core client systems. In server computers and small-core CPUs, however, the platform power is high, making race to halt (RtH) the most energy-efficient policy (dashed blue line in Figure 1). Recent improvements in platform power management have significantly reduced platform idle power, balancing platform and CPU energy (solid green line in Figure 1).[5] Furthermore, single-threaded workloads activate a small number of cores and therefore consume a smaller portion of the system power than highly threaded applications.

RUNTIME EFFECT ON OPTIMAL FREQUENCY AND CORE TYPE

Finding each core's optimal execution point in a heterogeneous system requires runtime evaluation because it depends on both hardware and application characterizations;[7] for example, memory-bound workloads, running at a higher core frequency or a high-performance core, consume more CPU energy but gain very little performance and platform energy and therefore benefit from a lower performance state. Thus, using a fixed-core scheduling and frequency policy does not deliver platform energy efficiency.

In this article, we introduce the H-EARtH algorithm for scheduling and managing a heterogeneous processor's optimal energy-efficiency point. It implements a simplified runtime platform model using a small number of static and runtime parameters to select the core and frequency that achieve minimum platform energy. We also address other commonly used formal optimization matrices in the form of $E \cdot D^{\alpha}$ where E = energy and D = delay (application runtime).

We instrumented platforms with two types of Intel Core i7 processors, manufactured on 45-, 32-, 22-, and 14-nm processes; a standard voltage CPU used as a high-performance core; and an ultralow voltage (ULV) CPU for a more power-efficient core. We tested the algorithm using 37 different benchmarks at different temperatures. We also simulated a heterogeneous multicore constructed of Intel Atom cores and Intel Core processors, sharing the interconnect and memory hierarchy. Our results show that the H-EARtH algorithm achieves the actual minimum platform energy on a real system with an energy accuracy of 2.2 percent. We also demonstrate that the heterogeneous CPU, which operates at this optimal H-EARtH point, achieves an average of 21 percent energy savings, with up to 33 percent savings compared to a homogeneous CPU. The H-EARtH algorithm can save energy up to 44 percent compared to the commonly used fixed-frequency policies: RtH and low frequency mode (LFM).

The algorithm's homogeneous version[6] has been implemented as Intel Speed Shift Technology on the sixth-generation Intel Core processor's power-management firmware with an α value set by the OS power policy.[8]

THEORETICAL MODEL

We first review the platform energy model of the single active core (homogeneous core[6]) and then extend it to the heterogeneous core.

Homogeneous core

Our workload model, shown in Figure 2, is characterized as two interleaved phases, active and idle. The active phase

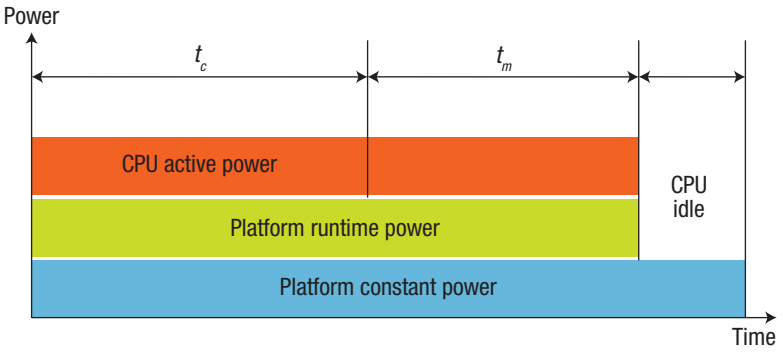

FIGURE 2. Conceptual platform power over time with CPU in active and idle states. Platform power is divided into continuous power and power that can be turned off when CPU execution ends.

is further split into interleaved memory-bound intervals (t_m) and CPU-bound intervals (t_c).[9–11] CPU runtime is inversely proportional to the frequency; CPU frequency does not affect off-chip memory–bound intervals. Rather than measuring the time intervals directly, we used the method described subsequently. Big and small cores have different t_m and t_c ratios.

We address a full compute platform consisting of a processor system on chip (SoC), memory, power supplies, communication, disk drives, and so on. We reduce this complex platform to a simple model in which the platform's power dissipation can be categorized as follows:

- *CPU power* (orange bar in Figure 2), which is the sum of all active core power consumed at runtime, comprising both dynamic and leakage parts with nonlinear dependency on frequency and voltage. Leakage of nonactive cores is turned off by power gates.
- *Platform power*, which is dissipated by the platform and further divided into two subcategories:
  - *Fixed energy* (not shown in Figure 2): During workload execution, a fixed amount of data is transferred to and from memory, disk drives, and so on. This activity is a function of the application footprint and the devices' physical characteristics; it does not depend on CPU frequency and therefore translates to fixed energy. If these memory accesses are spread over a longer time, the power is lower but the energy is equal.
  - *Runtime power* (green bar in Figure 2): This device power can be turned off during platform idle times. The energy impact of this power is proportional to the runtime of the workload and inversely proportional to CPU frequency (for example, system memory idle power and I/O links).
- *Platform constant power* (blue bar in Figure 2), which is dissipated by the platform regardless of workload activity (such as display and DDR self-refresh). Unlike runtime power, it is not turned off and therefore not subject to runtime optimization. Note that the distinction between runtime power and constant power depends on the idle state in use (that is, C/S-state).

The research platform model is thus reduced to a heterogeneous processor capable of performing core selection and DVFS with nonlinear power to frequency dependency (for example, by f^3). Platform components consume fixed runtime power with energy proportional to runtime ($1/f$). All other components are ignored. The two opposite energy trends might have a global minimum (see Figure 1). We define the following variables:

- f_0: reference lowest frequency of the CPU;
- f_c: frequency ratio, relative to $f_0 \cdot f_c = f_{actual}/f_0$;
- t_c: CPU-bound runtime at $f_c \cdot t_{c0}$ is t_c at f_0;
- t_m: memory-bound runtime, fixed for all f_c;
- P_0: lowest CPU power consumed at f_0;
- P_c: CPU power at f_c. Power scales as a function of frequency $P_c = P_0 \cdot F(f_c)$, accounting for leakage, temperature dependency, and so on; and
- P_l: platform runtime power at f_c.

Given these notations, the frequency-dependent part E_f of platform energy is

$$E_f = (t_c + t_m) \cdot (P_c + P_1) = \left(\frac{t_{c0}}{f_c} + t_m\right) \cdot (P_0 \cdot F(f_c) + P_1). \quad (1)$$

For optimization, it is more convenient to consider energy relative to the platform energy E_{f0} at the reference point $f\mathbf{0}$. Dividing Equation 1 by the same equation with $t_c = t_{c0}$ yields

$$\frac{E_f}{E_{f0}} = \frac{\left(\frac{t_{c0}}{f_c} + t_m\right) \cdot \left(P_{c0} \cdot F(f_c) + P_1\right)}{(t_{c0} + t_m) * (P_{c0} + P_1)} = \left(\frac{t_{c0}}{(t_{c0} + t_m)} \cdot \frac{1}{f_c} + \frac{t_m}{(t_{c0} + t_m)}\right) \cdot \left(\frac{P_{c0}}{(P_{c0} + P_1)} \cdot F(f_c) + \frac{P_1}{(P_{c0} + P_1)}\right). \quad (2)$$

We define two platform and workload terms. One is the CPU to platform power ratio (*CPR*)—the ratio between CPU power at f_0 and total platform power:

$$CPR = \frac{P_{c0}}{(P_{c0} + P_1)}, \text{ and clearly } 1 - CPR = \frac{P_1}{(P_{c0} + P_1)}.$$

$CPR \to 1$ implies that the platform power is dominated by CPU power, while $CPR \to 0$ implies that the rest of the platform dominates the total platform power; in real platforms, CPR falls between these two extremes.

The second parameter we define is *scalability*, the ratio of CPU bound time to total execution time, computed at f_0. We define workload scalability (SCA) as

$$SCA = \frac{t_{c0}}{(t_{c0} + t_m)}, \text{ and clearly } 1 - SCA = \frac{t_m}{(t_{c0} + t_m)}.$$

SCA is a workload characteristic that represents the performance dependency on CPU frequency. High scalability ($SCA \to 1$) indicates that performance is CPU bound and tightly related to frequency, while low scalability ($SCA \to 0$) indicates that the performance is memory bound and not affected by frequency. It is not possible to measure workload time intervals t_c and t_m directly because they are tightly interleaved. However, we can extract SCA at runtime by collecting execution parameters.

The platform energy can now be expressed as

$$\frac{E_f}{E_{f0}} = (SCA \cdot \frac{1}{f_c} + 1 - SCA) \cdot (CPR \cdot F(f_c) + 1 - CPR). \quad (3)$$

This equation implies that the relative platform energy is a function of frequency, CPU power, and SCA and CPR, characteristics of the platform and the workload.

To minimize energy, we must find the frequency that minimizes Equation 3. Note that using the term $D^{\alpha+1} = (t_{c0} + t_m)^{\alpha+1}$ in Equation 2 yields Equation 3.1:

$$\frac{E \cdot D^{\alpha}_f}{E \cdot D^{\alpha}_{f0}} = (SCA \cdot \frac{1}{f_c} + 1 - SCA)^{\alpha+1} \cdot (CPR \cdot F(f_c) + 1 - CPR. \quad (3.1)$$

Equation 3.1 is an analytical model that can be calculated at runtime and allows minimization of $E{\cdot}D^{\alpha}$ of a particular workload's total platform.

Heterogeneous core

Our heterogeneous core processor model consists of two core types sharing a single interconnect. At any given time, only one type of core might be running. Building the full heterogeneous core energy model at runtime therefore requires us to cross-predict the model parameters from the active core to the nonactive core. The interconnect and the memory architecture of our heterogeneous CPU are shared, so t_m is the same for both big and small cores. We approximate the runtime of the CPU-bounded portion on the big versus small core as a fixed ratio: $k \times t_{c\_big} = t_{c\_small}$. Using a fixed k proved to be a good approximation in our architecture. We obtained k through offline characterization using a training set of workloads out of SPEC2000 (186.crafty, 164.gzip, 181.mcf, 256.bzip2, 171.swim, 177.mesa, 179.art, and 188.ammp).

We define scalability as

$$SCA = \frac{t_{c0}}{(t_{c0} + t_m)}, \text{ and clearly } 1 - SCA = \frac{t_m}{(t_{c0} + t_m)}.$$

Dividing the two equations (with indices b for the big core and s for the small one):

$$\frac{t_{cb}}{t_m} = \frac{SCA_b}{1 - SCA_b}, \text{ and } \frac{k * t_{cb}}{t_m} = \frac{SCA_s}{1 - SCA_s}.$$

Finally,

$$\frac{SCA_s}{1 - SCA_s} = \frac{k * SCA_b}{1 - SCA_b}. \quad (4)$$

Equation 4 provides a function to calculate the scalability of a nonactive core based on the measured SCA of the active core at runtime (with a known k).

Equation 3 expresses the energy in relative terms. Note that the small core's E_0 is lower than the big core's at the same reference frequency. To compare the energy, we need to place it on a common scale:

$$\frac{E_{0b}}{E_{0s}} = \frac{P_{0b} * \text{runtime}_b}{P_{0s} * \text{runtime}_s}. \quad (5)$$

The power at the reference point P_0 is measured at system configuration. To calculate a workload's runtime, we now divide $1 - SCA$ of the big core by that of the small core:

$$\frac{t_{cs} + t_m}{t_{cb} + t_m} = \frac{\text{runtime}_s}{\text{runtime}_b} = \frac{1 - SCA_b}{1 - SCA_s}. \quad (6)$$

```
// Parameter initialization. Offline characterization at
// system design. Parameters stored in, or loaded by BIOS at power up

Get Pl  // Get Platform Run Time Power
Get α   // Characterize F(fc); Function can be polynomial, table or other.

// Run time optimization control
Every time interval {
 For each core {
  Pc = CPU power         // Sample CPU power meter; internal power meter or calculated
  CPR = Pc/(Pc+Pl)
  Get SCA                // Read CPU monitor or use collected statistics.
  Fopt=min((SCA · 1/f_c + 1−SCA) · (CPR · f^α_c + 1 − CPR)) // over valid frequencies
  Freq = Get Operating System frequency request
  F(resolved) = max(F_opt, F_req)
 }
Scale energy E_f/E_f0 to a common reference using Equations 5 and 6.
Select the core with minimum energy
}
```

FIGURE 3. H-EARtH algorithm.

Using Equations 5 and 6, we can compare the big and small cores' energy on a common scale and minimize overall energy. Equation 3 allows us to calculate the energy global minimum work point. The H-EARtH algorithm is shown in Figure 3. Initial P_1, $P(f_c)$ and k settings are stored in nonvolatile memory. At runtime, the H-EARtH algorithm activates once every time interval, calculates *CPR* and *SCA*, and determines the core and frequency that minimize Equation 3.1, with n as a user-defined parameter.

MEASUREMENTS AND SIMULATIONS

We evaluated a homogeneous core processor and two single-instruction-set architecture heterogeneous multicores:

- *Homogeneous processor*: We evaluated SkyLake, a sixth-generation Intel Core processor code. The H-EARtH algorithm is implemented in the processor's power-management firmware. We measured the power, energy, and performance of various benchmarks of a single unit and platform at the DC-balanced setting.
- *Single-ISA same microarchitecture*: This model contains four high-power high-frequency cores and four low-power slow cores. All cores share the Intel Core processor's single architecture and logic design at two different design targets. This topology is often referred to as asymmetric multicore.
- *Single-ISA different microarchitecture*: This model contains a combination of four big Intel Core processors (two-thread simultaneous multithreading each) and eight Intel ATOM cores.

Sixth-generation Intel Core processor measurements

We instrumented a mobile system with an Intel m7-6Y75 processor for power measurements and tested a set of common mobile benchmarks, capturing power performance as measured by the benchmark score. We calculated energy as the power integral over the entire benchmark runtime. Results are listed in Table 1. The baseline reference is Win-10 frequency control, as compared to the H-EARtH algorithm (homogeneous-core version). Frequency was dynamically controlled by the algorithm during workload runtime. As Table 1 shows, we saw

TABLE 1. Heterogenous energy-aware race to halt (H-EARtH) algorithm's energy savings and performance gains (higher percentages reflect better savings and gains).

			Average frequency (GHz)	
Workload	**Energy savings (%)**	**Performance gain (%)**	**Baseline**	**H-EARtH**
WebXPRT 2015 Chrome—sales graphs	24	29	1.38	1.41
WebXPRT 2015 Chrome—local notes	13	20	0.91	1.11
PCMark Vantage—communications	5	11	2.13	2.33
WebXPRT 2015 Chrome—stock-option pricing	5	9	1.18	1.21
WebXPRT 2015 Chrome—organize album	3	8	1.12	1.15
WebXPRT 2015 Chrome—photo enhancement	5	8	0.91	0.93
TouchXPRT 2014—create slideshow from photos	3	3	0.99	1.16
TouchXPRT 2014—blend photos	–1	3	1.25	1.45
PCMark Vantage—memories	0	2	2.21	2.30
TouchXPRT 2014—beautify photos	2	2	1.43	1.67
PCMark Vantage—HDD	2	2	2.45	2.54
PCMark Vantage—productivity	–1	2	2.13	2.20
TouchXPRT 2014—create music podcast	1	2	1.04	1.21
PCMark Vantage—gaming	1	0	2.36	2.42
Media Playback—4k.mp4	2	0	0.81	0.94
Media Playback—HD.mp4	2	0	0.74	0.87
WebXPRT 2015— explore DNA sequencing	–2	0	1.26	1.28
PCMark Vantage—music	–4	–1	1.94	2.20

performance gains due to the higher frequency, and overall energy savings due to both the shortened runtime and the system energy savings. We achieved up to 29 percent performance gain by running at a higher frequency, with up to 24 percent SoC energy savings. Two benchmarks lost 2–4 percent energy.

The sixth-generation Intel Core power-management algorithm also implemented the H-EARtH algorithm with energy optimization with $E{\cdot}D^{\alpha}$ as described in Equation 3.1. Higher α provides better performance at a cost of increased power and energy. We measured power and performance of SPEC 2006 of a high-end desktop part (Intel Core i7 6700K) with $\alpha = 2$ and $\alpha = 3$ (see Figure 4).

Asymmetric CPU: real-system implementation

We implemented the H-EARtH algorithm on platforms employing state-of-the-art 45-nm (Intel Core 2 Duo T9900), 32-nm (2860QM), and 22-nm (3840QM) processors. We measured two types of processors: a standard voltage Intel Core 2 Duo 2860QM and a ULV Intel Core 2 Duo 2677M intended for Ultrabook computers. We used these two processors to construct an asymmetric processor model. The platforms were instrumented to measure CPU and total platform power. We used Spec CPU 2000 and 2006 and SYSmark benchmarks at case temperatures of 45°C and 60°C on the two CPUs at eight frequencies. We evaluated both single and multithreaded workloads. We implemented the H-EARtH algorithm in the power-management firmware of the processor. The following algorithm parameters were set:

- *CPR*: We characterized the platform power P_1 offline once and stored it in nonvolatile memory. We used the built-in power meter[12] to calculate *CPR* at runtime.
- *Power function* $P(f_c)$: We characterized this once at setup by measuring power at eight frequencies. We identified a polynomial dependency, $P_c \propto f_c^{\alpha}$, with each core type having a different α value.

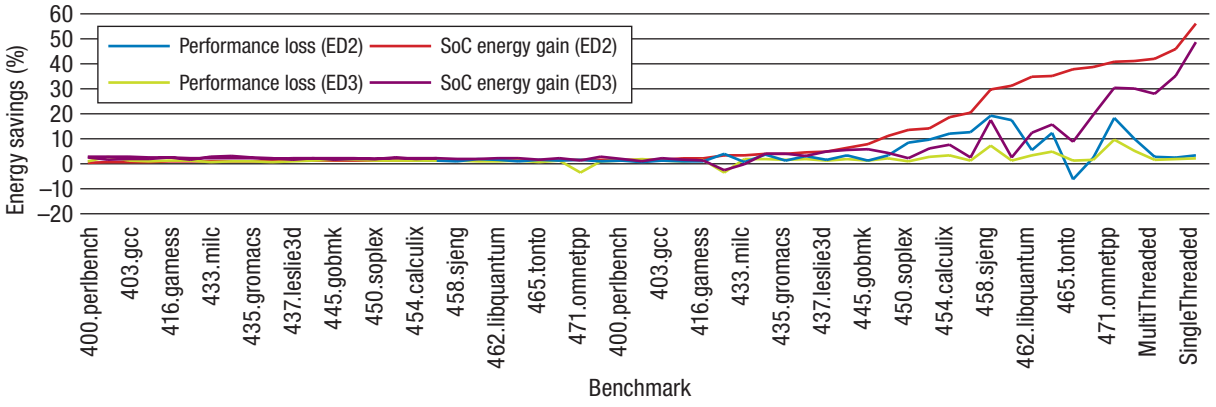

FIGURE 4. Energy performance tradeoff for $\alpha = 2$ and $\alpha = 3$. An α value of 2 achieves higher energy savings at the cost of higher performance loss. Performance loss is better than 1:2 for $\alpha = 2$. SoC: system on chip.

- *SCA:* This was calculated using the internal architectural memory stall counters.[12]
- *Sampling rate:* The H-EARtH algorithm used a 1-ms sampling rate, performing frequency decisions every 10 ms.

The asymmetric processor model consisted of four big and four small cores. Not having a real CPU with both core types, we ran the H-EARtH algorithm independently on each core type, selecting the lowest-energy core (or "optimal point") offline. Because this study was limited to a single core for the entire run, we missed out on the potential benefits of migrating the workload on the fly from small to big core and vice versa, as well as on the cost of migration. Figure 5 plots the energy savings of the optimal point versus fixed policies of LFM and RtH on each core type. The energy savings of the workloads is sorted individually for each policy from low to high.

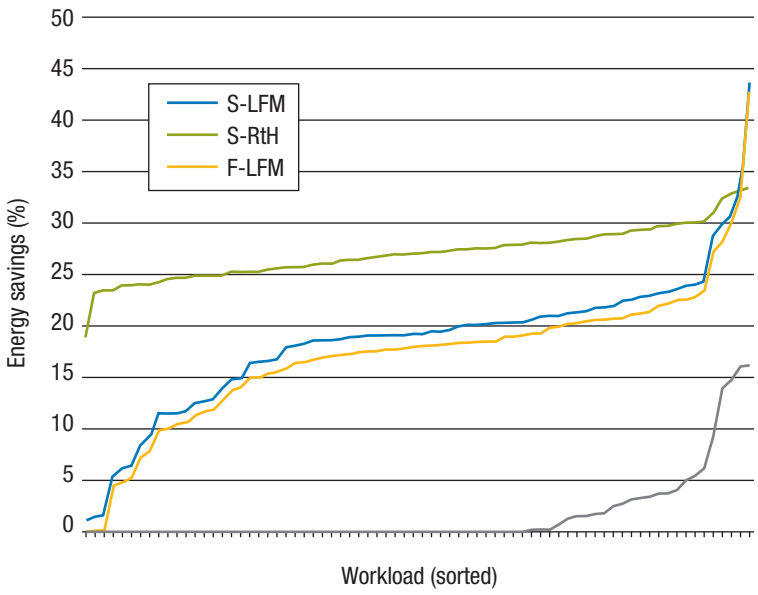

FIGURE 5. Energy savings of the optimal point versus fixed policies LFM and RtH on each core type. F: fast-core; S: slow-core.

Table 2 summarizes the best policy occurrences, that is, the ratio of workloads that achieved minimum platform energy at each policy. H-EARtH indicates an intermediate frequency other than RtH or LFM.

Running the small core at its maximum frequency usually delivers the lowest energy (63 percent of workloads). This fixed policy, however, results in more than one-third of workloads running at suboptimal frequency, losing up to 16 percent energy (S-RtH in Figure 5). Furthermore, the H-EARtH algorithm can save up to 44 percent of platform energy (F-RtH in Figure 5). A slow core is usually more energy efficient; however, in our platform, the fast core is more energy efficient in 9 percent of the workloads.

Heterogeneous CPU

In this part of the study, we used a tested and validated cycle accurate simulator, with two third-generation Intel Core processors as the big cores and four Intel Atom processors as the small cores sharing the Intel Core processor interconnect. We extracted the power and performance of multithreaded SPEC components at the eight frequencies using our simulator. We used the H-EARtH algorithm offline to find the frequency that optimally minimized the entire

TABLE 2. Frequency of achieving minimum platform energy at each policy.

Policy	Occurrence (%)
Slow-core H-EARtH	28
Slow-core race to halt	63
Fast-core low frequency mode	4
Fast-core H-EARtH	5

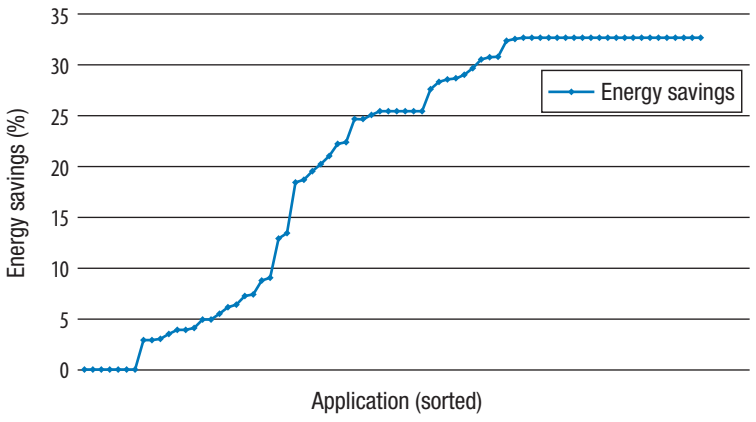

FIGURE 6. Energy savings of heterogeneous CPU versus big core homogeneous CPU running the H-EARtH algorithm.

platform's energy consumption for each workload and for each core type independently. We again chose a fixed core type for the entire workload. CPU power (used for *CPR* calculation) and *SCA* were obtained from the simulator, while P_1 for the rest of the platform was adopted from the real-system study described earlier. The cross-prediction k ratio was extracted from a 100 percent scalable application (SPEC CPU 2000 gzip).

Figure 6 sorts the energy savings of the heterogeneous CPU versus the homogeneous CPU for all 37 workloads at the two temperatures in ascending order. The left-most 9 percent of the applications achieve the lowest energy by using the big core (yielding no energy savings on the heterogeneous CPU). The remaining 91 percent benefit from the heterogeneous architecture; 31 percent achieve the maximum 33 percent energy savings by using the small cores at RtH frequency. The heterogeneous CPU saves an average 21 percent of energy compared to the big core CPU.

Multicore heterogeneous CPUs can perform computational tasks at lower platform energies than CPUs with only big cores: we demonstrated average energy savings of 21 percent on all workloads, with up to 33 percent savings in some cases. Using small cores, however, is not always energy efficient. Optimal core use depends on platform and workload characteristics. Operating an ill-suited core at a fixed frequency can lead to 44 percent platform energy loss. Our heterogeneous H-EARtH algorithm allows scheduling a workload to the most advantageous core while managing that core's voltage and frequency. Such a model aligns with existing OSs but does not fully utilize all available cores. A practical implementation will obviously perform periodic workload adjustments, with better workload adjustments at a cost of workload change overhead. Overall, H-EARtH offers a runtime scheduling and energy management algorithm for multicore heterogeneous CPUs to optimize the total platform $E{\cdot}D^{\alpha}$ metric.

ACKNOWLEDGMENTS

This work was supported by ICRI-CI: Intel Collaborative Research Institute—Computational Intelligence.

REFERENCES

1. N. Rajovic et al., "Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC?," *Proc. Int'l Conf. High Performance Computing, Networking, Storage, and Analysis* (SC 13), 2013, article no. 40.
2. *Variable SMP–A Multi-Core CPU Architecture for Low Power and High Performance*, white paper, NVIDIA, 2011; www.nvidia.com/content/pdf/tegra_white_papers/tegra-whitepaper-0911b.pdf.
3. A. Moore, *Hybrid-SMP*, white paper, Marvell, June 2012; www.marvell.com/application-processors/armada/pxa2128/assets/Marvell-Hybrid-SMP-WP.pdf.
4. *big.LITTLE Technology: The Future of Mobile*, white paper, ARM, 2013; www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf.
5. D. Meisner, B.T. Gold, and T.F. Wenisch, "PowerNap: Eliminating Server Idle Power," *Proc. 14th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 09), 2009, pp. 205–216.

6. E. Rotem et al., "Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management," *IEEE Computer Architecture Letters*, vol. 13, no. 1, 2014, pp. 25–28.
7. R. Ge, X. Feng, and K.W. Cameron, "Modeling and Evaluating Energy-Performance Efficiency of Parallel Processing on Multicore Based Power Aware Systems," *Proc. IEEE Int'l Symp. Parallel & Distributed Processing* (IPDPS 09), 2009; doi:10.1109/IPDPS.2009.5160979.
8. E. Rotem, "ARCS0001—Intel® Architecture, Code Name Skylake Deep Dive: A New Architecture to Manage Power Performance and Energy Efficiency," *Proc. Intel Developer Forum* (IDF 15), 2015; http://myeventagenda.com/sessions/0B9F4191-1C29-408A-8B61-65D7520025A8/7/5#sessionID=155.
9. C. Kihwan, R. Soma, and M. Pedram, "Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Tradeoff Based on the Ratio of Off-Chip Access to On-Chip Computation Times," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, 2005, pp. 18–28.
10. C.-H. Hsu and W.-C. Feng, "Effective Dynamic Voltage Scaling through CPU Boundedness Detection," *Proc. 4th Int'l. Conf. Power-Aware Computer Systems* (PACS 04), 2004, pp. 135–149.
11. C. Isci, G. Contreras, and M. Martonosi, "Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management," *Proc. 39th IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 39), 2006, pp. 359–370.
12. E. Rotem et al., "Power Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," *IEEE Micro*, vol. 32, no. 2, 2012, pp. 20–27.

ABOUT THE AUTHORS

EFRAIM ROTEM is a senior principal engineer at Intel and lead power architect of Intel's Mobile Computing Group in Haifa. He is responsible for defining and developing the power and energy management architecture of Intel Core client products. Rotem received a BSc and a PhD in electrical engineering from Technion–Israel Institute of Technology (IIT). Contact him at efraim.rotem@intel.com.

URI C. WEISER is a professor (emeritus) of electrical engineering at Technion—IIT. His research focus is computer system architecture. Weiser previously worked at Intel, where he was instrumental in defining the Pentium processor and MMX technology, invented the Trace Cache, co-managed the Intel Microprocessor Design Center, and researched advanced media applications. He received a BSc and an MSc in electrical engineering from Technion—IIT, and a PhD in computer science from the University of Utah. Weiser is an Intel, IEEE, and ACM Fellow and an IEEE/ACM Eckert-Mauchly award recipient. Contact him at uri.weiser@ee.technion.ac.il.

AVI MENDELSON is a professor of computer science at Technion—IIT. His research interests include computer architecture, OSs, power management, reliability, and high-performance computing. As a senior researcher and principal engineer, Mendelson was chief architect of the CMP (multicore-on-chip) feature of Intel's first dual-core processors. He received a PhD in computer engineering from the University of Massachusetts Amherst. Mendelson was previously an associate editor for *IEEE Computer Architecture Letters* and is now an associate editor of *IEEE Transactions on Computers*. Contact him at avi.mendelson@tce.technion.ac.il.

RAN GINOSAR is a professor of electrical engineering and head of the VLSI Systems Research Center at Technion—IIT. His research interests include VLSI architecture, many-core computers, asynchronous logic and synchronization, networks on chip, and biologic implant chips. Ginosar received a PhD in electrical and computer engineering from Princeton University. Contact him at ran@ee.technion.ac.il.

ELIEZER WEISSMANN is a senior principal engineer with Intel's Software Solutions Group in Haifa. His work focuses on power management of Intel's next-generation client, including Speed Shift technology, enhanced platform suspend mode, and other deep idle states. Weissmann received a BSc in computer science and an MS in computer databases from Technion—IIT. Contact him at eliezer.weissmann@intel.com.

YONI AIZIK is a power management architect at Intel Corporation. His research interests include power management, power and performance tradeoffs in digital circuits, and low power design. Aizik received a BSc and an MSc in electrical engineering from Technion—IIT. Contact him atyoni.aizik@intel.com.