THEME ARTICLE: HOT CHIPS 33

Intel Alder Lake CPU Architectures

Efraim Rotem, Adi Yoaz, Lihu Rappoport, Stephen J. Robinson, Julius Yuli Mandelblat, Arik Gihon, Eliezer Weissmann, Rajshree Chabukswar, Vadim Basin, Russell Fenger, Monica Gupta, and Ahmad Yasin, *Intel Corporation, Mountain View, CA, 94235, USA*

Alder Lake cores, system-on-a-chip, and hybrid architecture were designed to meet the challenging requirements for various traditional, emerging, and real-world computational tasks at extensive power performance points. Alder Lake introduces a revolutionary hybrid architecture with a whole new performance core and efficient core. The revolutionary thread director built into Alder Lake hardware and firmware guides the operating system in scheduling the right task to the right core type. Alder Lake hybrid architecture with Intel thread director delivers the highest power-performance dynamic range and computational density.

Alder Lake hybrid architecture is designed to deliver efficient high-compute performance in a large dynamic power and performance range. This is spanning from thin and light notebooks up to high-performance desktop computers. This also introduces the biggest strategic architecture change in the last decade.

Why are we driving this major shift? Single-threaded (ST) applications still dominate the client compute while most multithreaded (MT) applications are optimized for only a few cores. Highly parallel applications and workload importance is growing with the introduction of parallel data computation and AI algorithms. In addition, recent reality of collaborating work, education, and social interactions emphasizes the importance of multiprocessing for real-life user experience.

Single-thread performance is achieved by adding bigger and smarter architectural structures and adding new instructions and accelerators. All these come at the cost of size and power. Duplicating multiple instances of this bigger and more powerful core on a single system on a chip meets Moore's law power scaling recent reality. This resulted in an industry-wide challenge that requires a new architectural approach to the challenge.

Alder Lake architectural approach to the challenge is to break this trend by introducing "performance hybrid" architecture. We introduced a new performance core (P-core), allowing it to grow and deliver higher ST performance. The number of P-cores is optimized for the existing software ecosystem. To address the highly threaded applications, we have built a throughput machine around a new efficient core (E-core). It is optimized for computational density within the System-on-a-Chip (SoC) power and size constraints. Hybrid architectures already exist.[3,4] This type of hybrid solution is built mainly to address extended power and energy dynamic scaling with two core types that are very different in power and performance characteristics. They are managed as a performance state (P-state), and serve low computational demand and contribute very little to high computational demand. A key capability for managing performance hybrid is the new thread director architecture introduced on Alder Lake.

0272-1732 © 2022 IEEE
Digital Object Identifier 10.1109/MM.2022.3164338
Date of publication 1 April 2022; date of current version 18 May 2022.

P-CORE

Intel's new P-core, previously codenamed "Golden Cove," introduces major architectural and microarchitecture improvements, delivering a strong upgrade in instructions per cycle (IPC) along with a new set of instructions (ISA) to enable much greater performance for AI and scientific workloads.

To keep driving general-purpose performance, the P-core is architected to become wider, deeper, and smarter. It has a deeper out-of-order scheduler and buffers, more physical registers, a wider allocation window, and more execution ports. The P-core is enhanced with smart features that improve branch prediction and instructions supply, collapse dependency chains, and bring data closer to the time when it is needed.

The P-core adds dedicated features for workloads with large code footprints, features for workloads with large, irregular data sets, and features for machine learning.

P-core architecture introduces advanced matrix extensions, which define new registers and instructions that natively process matrix multiplication operations. Using the new instructions, P-core delivers 8× greater peak performance than when using the wide vector AVX-512 instructions.

ALDER LAKE HYBRID ARCHITECTURE IS DESIGNED TO DELIVER EFFICIENT HIGH-COMPUTE PERFORMANCE IN A LARGE DYNAMIC POWER AND PERFORMANCE RANGE.

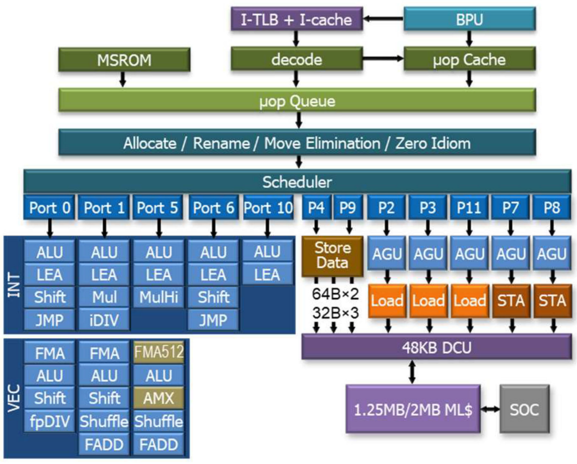


FIGURE 1. P-core block diagram.

P-core Frontend

The P-core frontend is built to feed the wider out-of-order core. The legacy decode pipeline fetch bandwidth is increased from 16 to 32 bytes/cycle. The number of decoders is increased from 4 to 6. This allows us to decode up to 6 IPCs. The micro-op cache size is increased to hold 4,000 micro-ops, and its bandwidth is increased to deliver up to 8 micro-ops per cycle. Branch prediction is improved.

To better support workloads with a large code footprint, the number of 4,000 pages and large pages stored in the iTLBs is doubled to hold 256 entries for 4-KB pages and 32 entries for 2/4 million pages. A smarter code prefetch mechanism hiding much of the instruction cache miss latency is added. Branch prediction structures have been increased.

P-core Backend

The backend side of the machine becomes wider, deeper, and smarter. Rename/allocation width grows from 5 to 6 wide. The number of execution ports goes from 10 to 12. The machine is turning significantly deeper with a 512-entry reorder buffer, more physical registers, and a deeper scheduling window. This is all tuned for performance and power efficiency. The new port 10 can handle common integer ALU operations as well as load effective address (LEA); LEA appears frequently in x86 code. Hence, the central processing unit (CPU) includes five LEA units as well as five integer ALUs. On the vector side, new fast adders were added on ports 1 and 5. These are three-cycle fast adders, with two cycles bypass between back-to-back floating-point ADD operations. Port 11 provides a third load port with a dedicated address-generation unit. This allows the L1 data cache to supply more data to the new wide-execution machine.

P-core Memory Subsystem

The P-core memory subsystem is designed to handle up to three 32-byte loads or two up to 64-byte loads per cycle, as well as up to two 64-byte stores per cycle, providing simultaneous 128 bytes of read bandwidth and 128 bytes of write bandwidth per cycle. The L1 load-to-use latency is five cycles.

Deeper load buffer and store buffer expose more parallelism for load and store operations.

Prediction of loads that are independent of previous stores is improved, and the penalty for wrong prediction is significantly reduced. Store-to-load forwarding is supported, as well as allowing cases where the first load bytes are forwarded from a store, and the rest are read from the data cache. In some cases, the store-to-load forwarding latency is greatly reduced.

Address translations are performed through the TLBs: 96-entry 6-way 4-KB-page TLB, 32-entry 4-way 2-MB/4-MB-page TLB, and 8-entry 1-GB-page TLB for loads. A 16-entry TLB for stores server all page sizes. The TLBs are backed by a 2,048-entry second level TLB (STLB), which is shared between code and data requests. STLB misses are sent to the page miss handler (PMH) that allows us to perform up to four page walks in parallel.

The L2 cache is 1.25 MB in size and delivers 64 bytes of data per cycle at a latency of 15 cycles.

This increased the number of outstanding misses for both the L1 data cache and L2 cache. Enhanced L1 and L2 cache prefetchers, including a new L2 cache pattern prefetcher is added.

P-core Power Management

With core-autonomous finer grain power management technology, the P-core integrates a new microcontroller in each core that can capture and account for events in

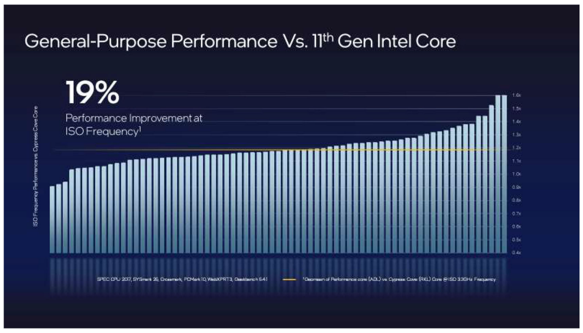


FIGURE 2. P-core performance improvement.

the granularity of microseconds and optimize the power budget utilization based on actual application behavior. The result is higher average frequency for any given application.

P-Core Performance

Figure 2 shows the P-core performance improvement over the 11th gen core across a wide range of workloads same frequency, with an average improvement of 19%.

For the new workloads that take the advantage of the P-core new ISA and architectural advancements, the performance improvement is higher.

E-CORE

Intel's new Alder Lake efficient CPU microarchitecture, formerly called Gracemont, is a step function in x86 efficiency, designed with the needs of computing scalability in mind. Both area and power were emphasized to complement the P-core.

Instruction Control

The ADL E-core features a frontend with 32-byte prediction. The first predictor is the next line predictor (NLP). This can predict a taken branch every cycle with no bubbles. The second level predictor is a 5,000 entry target array, a three-cycle predictor.

Predict and Fetch

The NLP prediction is sent down the fetch pipeline, look up the Instruction TLB (ITLB) and instruction cache tag to determine hit or miss. The ITLB and cache are built to understand relative offsets and are not typically enabled. Successful translations are stored in the instruction pointer (IP) queue. Instruction data are written into instruction data buffers. Each IP queue is in-order and delivers 32 bytes/cycle. The IP queues and decoders are referred to as clusters. Taken branches toggle between the clusters. This enables out-of-order fetch and decode allowing six variable length instructions fetch and decode per cycle.

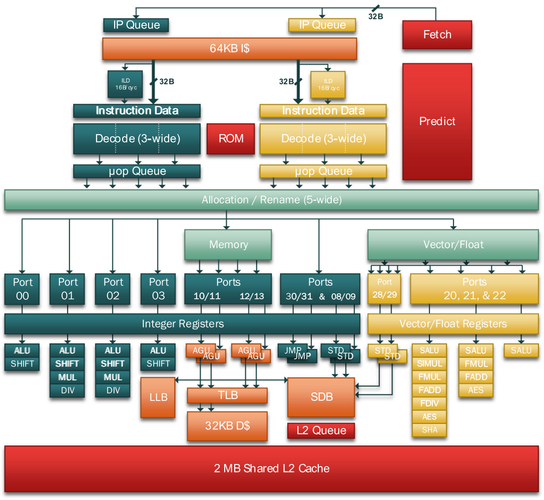


FIGURE 3. Alder Lake E-core block diagram.

Decode

The E-core stores a single bit for each byte in the instruction cache that marks an instruction boundary. This bit steers instruction bytes into decoder lanes. With predecode bits, no instruction decoding is required to steer instructions. The E-core introduces an "on-demand" instruction length decoder block. When used, two extra cycles are added to generate the predecode bits on the fly. This overall approach of instruction decoding scales variable length instruction decode to very wide designs and results in area and power efficient x86 decoder.

The six decoders (three within each cluster) are complex and symmetric. Load-op-stores, complicated addressing forms, most control enforcement technology instructions, and more, are generated in a single internal micro-operation (UOP) format. Each decoder can detect a microcode entry point. The most common microcode flows can be executed out of order between the clusters.

Allocation and Retire

The ADL E-core allocation is five instructions wide. The two UOP queues are read in order. For 256-bit AVX instructions, two allocation lanes are used to allocate two 128-bit halves of the instruction. Move elimination, NOP detection, and idiom reduce dependency chains and eliminate UOPs from execution.

Retirement is eight instructions wide and the out of order window is 256 entries. With wide retirement, the core has smaller structures as the operation lifetime is reduced.

Execution

UOPs are delivered to three independent structures. Integer operations are written into one of five reservation stations, which track their dependencies. There are four integer ALUs, two of which support longer latency operations. The fifth reservation station holds jumps and store data operations and is banked for scheduling two UOPs per cycle.

Allocation writes into a memory FIFO to enable a wider OOO window. It generates two loads and two store addresses per cycle.

The FIFO writes into either a reservation with three scheduling pipelines, or a store data queue, which generates two store data transfers per cycle. Combining the two enables a peak execution of 16 single precision FP operations per cycle. It executes up to three SIMD integer ALU and shuffle operations per cycle. There are also two advanced encryption standard units.

Memory Subsystem

The memory subsystem is designed to handle two 16-byte loads and two 16-byte stores per cycle. This provides simultaneous 32 bytes of read plus 32 bytes of write bandwidth per cycle. The load latency for loads is 3–4 cycles. The L1 data cache is dual ported. The 2MiB L2 cache is shared among four cores. The L2 delivers 64 bytes per cycle at a latency of 17 cycles. The L2 can have 64 requests and 16 L2 data evictions outstanding on the fabric, which are competitively shared among the cores. Memory disambiguation is supported. Stores forward to loads in the same 3–4 cycle pipeline.

The first level TLB is fully associative and holds both 4KiB and 2MiB pages. The STLB is shared between code and data. This has 2,048 entries that supports both 4KiB and 2MiB pages. There is an eight entry STLB that holds GiB pages. The PMH can perform four walks in parallel.

Power Management

Each core can be individually clock gated (C1) or power gated (C6). The L2 can be kept powered, active, and coherent independent of the core power states. L2 flushing is also supported for deep sleep states (MC6).

Each core operates at the same frequency, which is independent from the fabric and other cores.

Performance

The relative performance of SPEC CPU 2017 benchmark, tested on prototype hardware is shown in Figure 4. The geomean performance of integer workloads is 0.8× that of a P-core running at the same frequency. The floating point geomean performance is 0.62×. Figure 5 compares two P-cores, each running two active threads, to a single E-core 4 core module, both running four copies. The score for integer is 13% higher, while the floating-point score is 17% lower at the same frequency. The single 4 E-core module is roughly half the area of the two P-cores used in this comparison. The P-core is capable of a higher peak operating frequency at the same operating voltage, delivering higher peak performance.

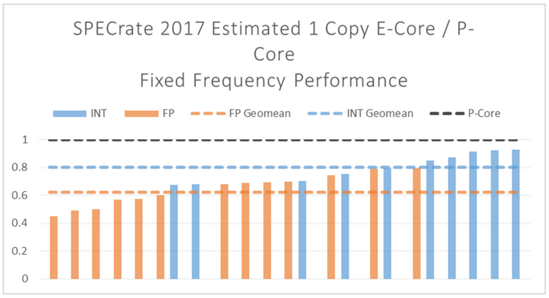


FIGURE 4. Comparing an E-core with a P-core, sorted by relative performance.

ALDER LAKE SoC ARCHITECTURE

Alder Lake SoC architecture is designed to deliver full client portfolio with a single, highly scalable architecture from high-performance desktop to ultramobile. A modular architecture is built by mix and match from a set of building blocks: P-cores and modules of 4 E-core that are interchangeable, two sizes of integrated graphics and a set I/O and accelerators.

The Alder Lake family is divided into following three architectural categories.

- *Desktop:* Up to 16C/24T, 30-MB LLC, discrete chipset, with LGA 1700 socket.

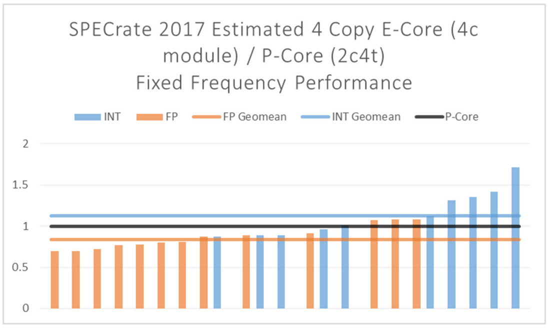
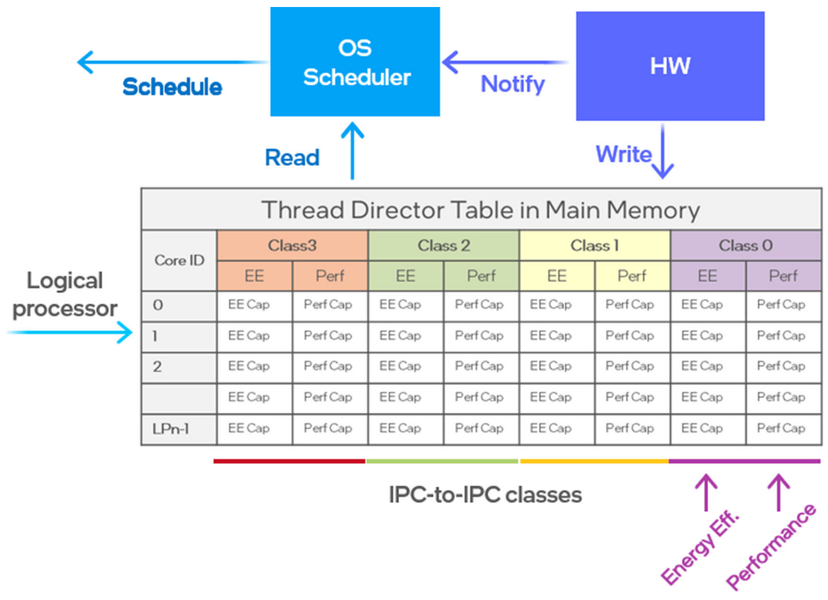


FIGURE 5. Comparing an E-core with a P-core, sorted by relative performance.



Thread Director Table in Main Memory

Core ID	Class3		Class 2		Class 1		Class 0	
	EE	Perf	EE	Perf	EE	Perf	EE	Perf
0	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap
1	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap
2	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap
	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap
LPn-1	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap	EE Cap	Perf Cap

FIGURE 6. Thread director scheduling.

- *Mobile:* Up to 14C/20T and 30-MB LLC, 2 dies type3 BGA multichip package (MCP).
- *Ultramobile:* Up to 10C/12T, 18-MB LLC, 2 dies type3 BGA, and type4 HDI MCP.

Alder Lake integrates DDR4 and DDR5 in desktop adding LPDDR4 and LPDDR5 in mobile. High speed DDR supports smart runtime frequency gearing to scale with BW and power demands.

Alder Lake I/Os include PCIe Gen5 and Wi-Fi 6, high performance 2x4 Gen4 for fast SSDs and x16 Gen5 ready for the next gen discrete graphics.

THREAD DIRECTOR

Thread-aware scheduling is needed to extract the benefit of the hybrid architecture, i.e., the right workload needs to be scheduled on the best core type at the right time. It addresses activities such as: gaming, streaming, content creation, productivity, multitasking, and more. Real life workloads are asymmetric, comprising of high- and low-priority tasks with different runtime properties. This asymmetry changes as software applications go through phases of execution.

The knowledge about application priority or QoS resides in the operating system (OS) while the runtime characteristics of the workload resides in the hardware. Alder Lake introduces a novel architectural that works in collaboration with the OS to direct the right work to the right core type to maximize performance and efficiency. This architecture is called “thread director.”

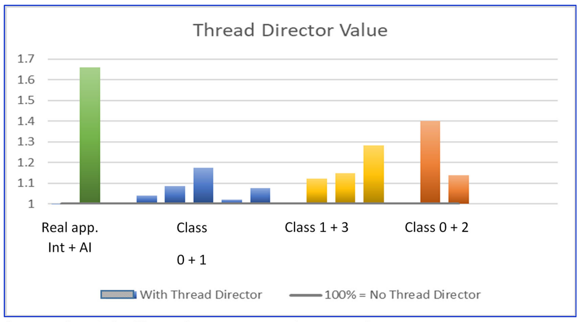
With thread director all the cores are exposed to the OS as logical processors and the SoC topology is fully enumerated with no preassumption on performance or efficiency. Existing SW is seamlessly supported by a single hybrid-enabled OS on all SoC topologies, including nonhybrid topologies.

Alder Lake processor implementation consists of two components, runtime characterizations of software threads, and thread directing interface table.

ALDER LAKE SoC ARCHITECTURE IS DESIGNED TO DELIVER FULL CLIENT PORTFOLIO WITH A SINGLE, HIGHLY SCALABLE ARCHITECTURE FROM HIGH-PERFORMANCE DESKTOP TO ULTRAMOBILE.

Runtime Characterizations of Software Threads

The Alder Lake hardware track the runtime behavior of the SW threads at nanosecond granularity. It extracts the relevant information and dynamically adapts it as runtime behavior changes and communicates this to the OS. This information is not available otherwise to the OS. Alder Lake collects a set of telemetries that is comprised of architectural attributes, such as types of instructions, microarchitectural attributes, and frontend bound.[6] An



* Note: Chart is for illustrative purposes and not at scale

FIGURE 7. Thread director benefits over nonhybrid scheduling.

ML-based algorithm uses this telemetry to predict the IPC-to-IPC ratio (see Figure 4). It then categorizes the workloads into the following four classes.

- *Class 0* is where most applications are. Arithmetic, flow control, data transfer, etc. Figures 4 and 5 show that the E-core is efficient on these applications.
- *Classes 1–2* are emerging workloads with wide vectors, AI, and accelerated instructions. Here the P-core delivers up to 2.75× compared to the E-core.
- *Class 3* are workloads, such as spin loops that best run on the E-core.

The workload classification is communicated at every context switch, it is stored together with the thread context and is made available for the OS use for future scheduling decisions.

Thread Directing Table

The SoC hardware and firmware creates the routing and enumeration table described in Figure 6. It is built based on the SoC physical characteristics and topology. This table is normally generated much slower than the SW characterization and classification. It will typically change as a function of power or thermal constraints. This table is called EHFI table.[1]

The EHFI table has two columns for each class, one to enumerate performance (frequency*IPC) and one to enumerate efficiency. The architecture is built to allow extending to any number of classes beyond those described previously. A value of 0 in the table implies that it is not recommended to schedule a software thread of that specific core and class. This is a hint that may be ignored by the due to thread affinity. Each row in the table represents a single core type.

Note: There is no hard coding of E-core and P-core. Each core is described by its performance and efficiency characteristics and there can be multiple types of cores with different characteristics. Furthermore, E-core is not always more efficient. For example, energy equals Power*Time; therefore, Class 2 of the P-core with much higher performance may finish a task much faster than on E-core. This makes it more energy efficient for that class. Another example on an extremely power constraint condition, the frequency of the P-core may need to slow to a working point slower than the E-core.

Thread Director and OS Scheduling

The thread director table provides the OS core topology, performance, and efficiency properties. The individual software thread's class is communicated via the thread context. The OS generates the information about the software thread priority based on user experience.[2] When a thread is scheduled, the OS uses that thread's class information, and uses the columns of that class to picks the right core. High-priority threads are scheduled based on the performance column and vice versa.

Performance Benefits of Thread Director

On a symmetrically threaded application, such as Cinebench, hybrid architecture delivers computational density regardless the scheduling policy (baseline of 1 in Figure 7). Thread director is beneficial on real-life, asymmetrical workloads. Scheduling the right workload to the right core delivers up to 65% performance improvement over thread director disabled (see Figure 7). On a fully loaded system with a mix of high- and low-priority threads, thread director helps to guide the OS to migrate threads to the right core at runtime.

Thread director is fully supported in Windows 11 with partial support in previous Windows OS. Other OS support work is in progress.

REFERENCES

1. Intel 64 and IA-32 Architectures Software Developer's Manual. 2021. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html
2. J. Doweck *et al.*, "Inside 6th-generation Intel core: New microarchitecture code-named Skylake," *IEEE Micro*, vol. 37, no. 2, pp. 52–62, Mar./Apr. 2017, doi: 10.1109/MM.2017.38.
3. S. Khushu *et al.*, "Lakefield: Hybrid cores in 3D package," in *Proc. IEEE Hot Chips Symp.*, 2019, pp. 1–20.
4. "Processing architecture for power efficiency and performance." Accessed: Apr. 25, 2022. [Online]. Available: https://www.arm.com/technologies/big-little

5. E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the Intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar./Apr. 2012, doi: 10.1109/MM.2012.12.
6. E. Rotem, U. C. Weiser, A. Mendelson, R. Ginosar, E. Weissmann, and Y. Aizik, "H-EARtH: Heterogeneous multicore platform energy management," *Computer*, vol. 49, no. 10, pp. 47–55, Oct. 2016, doi: 10.1109/MC.2016.309.
7. A. Yasin, "A top-down method for performance analysis and counters architecture," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2014, pp. 35–44.

EFRAIM ROTEM is an Intel fellow at the Intel Client Architecture Group, Mountain View, CA, 94235, USA. His research interests include power and energy efficiency, and next generation power management of modern system on a chip, CPUs, and platform. Rotem received a Ph.D. degree in electrical engineering from Technion—Israeli Institute of Technology. Contact him at efraim.rotem@intel.com.

ADI YOAZ is an Intel fellow and the director of Intel Core CPU Architecture at Intel Corporation, Mountain View, CA, 94235, USA. As the Core CPU chief architect, he defines next-generation architecture and microarchitecture for the Intel Core processor family. Yoaz received an MBA degree from the Faculty of Industrial Engineering and Management, Technion—Israeli Institute of Technology. Contact him at adi.yoaz@intel.com.

LIHU RAPPOPORT is a senior principal engineer in the Design Engineering Group, Intel, Mountain View, CA, 94235, USA. He is a CPU core architect, focused on core performance. Rappoport received a Ph.D. degree in computer science from Technion—Israeli Institute of Technology. Contact him at lihu.rappoport@intel.com.

STEPHEN J. ROBINSON is an Intel fellow in the Design Engineering Group, Intel, Mountain View, CA, 94235, USA. He is part of the E-core architecture team. Robinson graduated from the University of Texas at Austin. Contact him at stephen.j.robinson@intel.com.

JULIUS YULI MANDELBLAT is an Intel fellow at the Intel Client Architecture Group, Mountain View, CA, 94235, USA. His research interests include SoC architecture, memory and caching solutions, CPU performance, and on-die interconnect. Mandelblat received an M.Sc. degree in electrical engineering from Moscow Transport University. Contact him at julius.mandelblat@intel.com.

ARIK GIHON is senior principal engineer leading the SoC Hardware Architecture team in the Intel Client Architecture group, Mountain View, CA, 94235, USA. His research interests include power management and power delivery architecture, microarchitecture, and interaction with the SW stack. He is holds 18 patents. Gihon received a B.S.E.E degree from Technion—Israel institute of technology. Contact him at arik.gihon@intel.com.

ELIEZER WEISSMANN is a senior principal engineer at Intel in the Artificial Intelligence and Analytics Group, Mountain View, CA, 94235, USA. His research interests include Core and SoC power management, operating system, and hardware architectures and hardware–software co-design. Weissmann received an M.Sc. degree in computer science from Technion—Israeli Institute of Technology. Contact him at eliezer.weissmann@intel.com.

RAJSHREE CHABUKSWAR is an Intel fellow in the Intel Client Architecture Group, Mountain View, CA, 94235, USA. She works on software stack optimizations and co-design for Intel client platforms. Chabukswar received a master's degree in computer engineering from Syracuse University. Contact her at rajshree.a.chabukswar@intel.com.

VADIM BASIN is a principal engineer in the Intel Client Architecture Group, Mountain View, CA, 94235, USA. His main professional interests include the area of performance and power efficiency of modern processors, and also on SW design and architecture area. Basin received the M.Sc. degree from the Belarusian State University of Informatics and Radioelectronics, Minsk, Belarus. Contact him at vadim.bassin@intel.com.

RUSSELL FENGER is a senior principal engineer at Intel in the Software & Advanced Technology Group, Mountain View, CA, 94235, USA. His research interests include operating system scheduler and power management optimizations. Fenger graduated from Iowa State University. Contact him at russell.j.fenger@intel.com.

MONICA GUPTA is a technical lead at Intel Software and Advanced Technology Group, Mountain View, CA, 94235, USA. Her interests include system software, windows operating system, next generation processor power management, and power and energy efficiency of modern system-on-a-chip. Gupta received a master's degree in computer science from the University of Florida, Gainesville, FL, USA. Contact her at monica.gupta@intel.com.

AHMAD YASIN is a principal engineer in the Design Engineering Group, Intel, Mountain View, CA, 94235, USA. His research interests include performance tuning, platform analysis methods and technologies to optimize CPU performance. Yasin received a Ph.D. degree in computer science from the University of Haifa. Contact him at ahmad.yasin@intel.com.