Haswell: The Fourth-Generation Intel Core Processor

Per Hammarlund
Alberto J. Martinez
Atiq A. Bajwa
David L. Hill
Erik Hallnor
Hong Jiang
Martin Dixon
Michael Derr
Mikal Hunsaker
Rajesh Kumar
Randy B. Osborne
Ravi Rajwar
Ronak Singhal
Reynold D'Sa
Robert Chappell
Shiv Kaushik
Srinivas Chennupaty
Stephan Jourdan
Steve Gunther
Tom Piazza
Ted Burton
Intel

HASWELL, THE FOURTH-GENERATION INTEL CORE PROCESSOR ARCHITECTURE, DELIVERS A RANGE OF CLIENT PARTS, A CONVERGED CORE FOR THE CLIENT AND SERVER, AND TECHNOLOGIES USED ACROSS MANY PRODUCTS. IT USES AN OPTIMIZED VERSION OF INTEL 22-NM PROCESS TECHNOLOGY. HASWELL PROVIDES ENHANCEMENTS IN POWER-PERFORMANCE EFFICIENCY, POWER MANAGEMENT, FORM FACTOR AND COST, CORE AND UNCORE MICROARCHITECTURE, AND THE CORE'S INSTRUCTION SET.

Haswell, the fourth-generation Intel Core Processor, delivers a family of processors with new innovations.[1,2] Haswell delivers a range of client parts, a converged microprocessor core for the client and server, and technologies used across many products.

Many of Haswell's innovations are in the areas of improving power-performance efficiency and power management. Power-performance efficiency has been enhanced to increase the processor's operating range and improve its inherent performance in power-limited scenarios and its battery life. Improvements in power management include additional idle states, specifically the new active idle state S0ix, which enables 20× reduction in idle power. One key enabler for power-performance improvements is the fully integrated voltage regulator (FIVR), which also improves board space and cost.

Performance improvements in the core and graphics come with corresponding improvements in cache hierarchies; the first two cache levels have twice the bandwidth. For the top graphics configurations, Intel Iris Pro Graphics, Haswell also introduces a new fourth-level, 128-Mbyte on-package cache that enables a new level of integrated graphics performance.

Haswell is a "tock"—a significant microarchitecture change over the previous-generation Ivy Bridge. Haswell is built with an SoC design approach that allows fast and easy creation of derivatives and variations on the baseline. Graphics and media come with more scalability that lets designers build efficient configurations from the lowest to highest end. The core comes with power-performance enhancements and a set of new instructions, such as floating-point fused multiply-add (FMA) and transactional synchronization extensions (TSX).

Haswell uses an enhanced version of Intel's 22-nm process technology, which has enhanced tri-gate transistors to reduce leakage current by a factor of 2× to 3× with the same frequency capability. Haswell's version of the 22-nm process has 11 metal interconnect layers, compared to nine for Ivy Bridge, to optimize for better performance, area, and cost.

Power efficiency and management

Current processors operate in power-constrained modes; they must maximize the performance they deliver inside a fixed power envelope. This power constraint is true for both server and mobile applications. One of

Published by the IEEE Computer Society

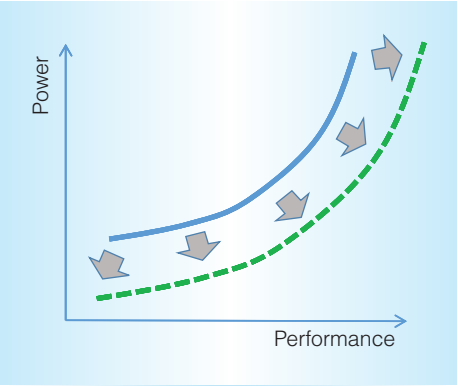

Figure 1. Power and performance voltage-frequency scaling improvements. The baseline (solid line) is improved (dashed line) by being lowered and by being extended for better burst and Turbo headroom.

the most important goals of a new processor generation is to dramatically improve power-performance efficiency. In Figure 1, the basic nonlinear relationship between power and performance is shown in the solid line. To improve power-performance efficiency across the voltage-frequency scaling range, we must achieve three goals, as shown in the dashed line:

- extending the operating range downward to allow the processor to go into smaller form factors that are even more power constrained,
- improving the basic power-performance efficiency of the processor by pushing each operating point to the right and down, and
- extending the operating range upward for more burst and Turbo headroom.

In Haswell, we employ multiple techniques to improve power-performance efficiency. We can describe them in three categories: low-level implementation, high-level architecture, and platform power management.

Examples of low-level implementation improvements include the following:

- Optimized manufacturing, process technology, and circuits help achieve all three goals just listed. These improvements are enabled by Intel's manufacturing capability and a deep collaboration across the different Intel teams.

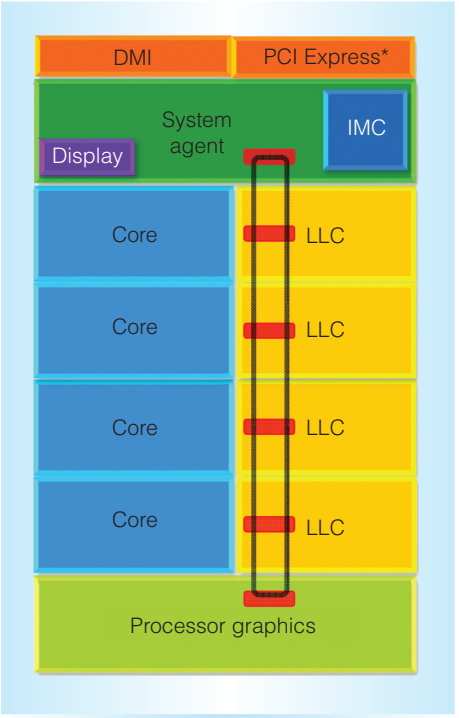

Figure 2. Conceptual block diagram of the Haswell processor showing the different independent voltage domains. The figure also shows Haswell's cache hierarchy and memory controller, which features bandwidth, load balancing, and DRAM efficiency improvements.

- Optimized microarchitecture and algorithms. In each generation, we evaluate for sufficient power-performance efficiency. Areas that fall below our goals will be reimplemented in ways that improve the power-performance efficiency.
- Optimization of design and implementation through continued focus on gating unused logic and using low-power modes.

An example of a high-level architecture improvement in Haswell is extending the use of independent voltage-frequency domains. Figure 2 shows a conceptual block diagram of the different voltage-frequency domains. Cores, caches, graphics, and the system agent are all running at dedicated, individually controlled voltage-frequency points. A power control unit (PCU) dynamically allocates the power budget among the domains to maximize performance. Prioritization based on

runtime characteristics select the domain with the highest-performance return. For example, for a graphics-focused workload, most of the processor power is allocated to the graphics domain. Sufficient power is allocated to the rest of the blocks that the graphics domain depends on for performance, such as the system agent to provide memory bandwidth.

At a platform level, we improved battery life to deliver "all-day experiences." To achieve this, we focused both on active workloads, such as media playback, and on idle power. Haswell achieves a 20× improvement in idle power. Haswell has evolutionary power-management improvements, such as improvements in C-states (CPU idle states). Haswell has both new, deeper C-states and improvements in the entry-exit latencies to C-states. These latency improvements let Haswell more aggressively enter deep C-states.

Haswell also has revolutionary power-management improvements—for example, the introduction of a new active idle-power state, S0ix. We leverage learnings from past phone and tablet development to deliver 20× improvements in idle power compared to the prior generation. This improvement enables significant improvements in realizable battery life. S0ix appears to software as an active state, while in actuality the hardware autonomously enters and exits deep idle states with low latency. The new power state is transparent to well-written software. Power management of platform components is continuous and fine grained; everything that is not needed is individually turned off.

Fully integrated voltage regulator

Power delivery to higher-performance processors comes with many conflicting requirements, such as the need for higher power for extended burst capability, a greater number of individually controlled voltage rails, and the need for a physically smaller footprint for new form factors. In response to these requirements, Haswell processors are powered by a 140-MHz, multiphase FIVR.[2-4] The industry's first large-scale deployment of high current switching regulators integrated into a VLSI die and package. FIVR is the enabling technology behind key Haswell improvements, including a 2× to 3× increase in peak available power (which converts into burst performance), a substantial battery life increase, and a 70 to 80 percent platform footprint reduction.

Figure 3 gives an overview of FIVR. A first-stage voltage regulator (VR), which is on the motherboard, converts from the power supply or battery voltage (12 to 20 V) to approximately 1.8 V, and the second conversion stage is provided by parallel FIVRs (one for each major architectural domain). As illustrated, FIVR eliminates four VRs from the prior platform. To support the new Intel Iris Pro Graphics variants of Haswell, those platform VRs would have grown in both size and number. With FIVR, a platform-size reduction opportunity was achieved instead of what would have been a substantial growth. That platform space can be used to add platform features, increase the battery size, and reduce the platform dimensions in many Haswell mobile products.

At the onset of the Haswell design, FIVR's expected benefits fell into half a dozen categories:

- Battery life increase. FIVR's 140-MHz switching frequency enables several orders of magnitude less output decoupling and much lower input decoupling than the prior generation's voltage rails, allowing input and output voltages to be quickly reduced or powered off to save power, and quickly ramped back up for brief high-performance bursts.
- Increased available power for increased burst performance, where FIVR can direct the entire package power to the unit that needs the most power, compared to separate VRs for separate units in the previous platform.
- Decreased power required for a given level of performance or, almost equivalently, increased performance for a given power consumed.
- Decreased platform cost and size from removal of components and external power rails.
- Improved product flexibility and scalability; for example, new units can be added with little impact on the platform power-delivery systems.

FIVR delivered benefits in every category, some larger than expected.

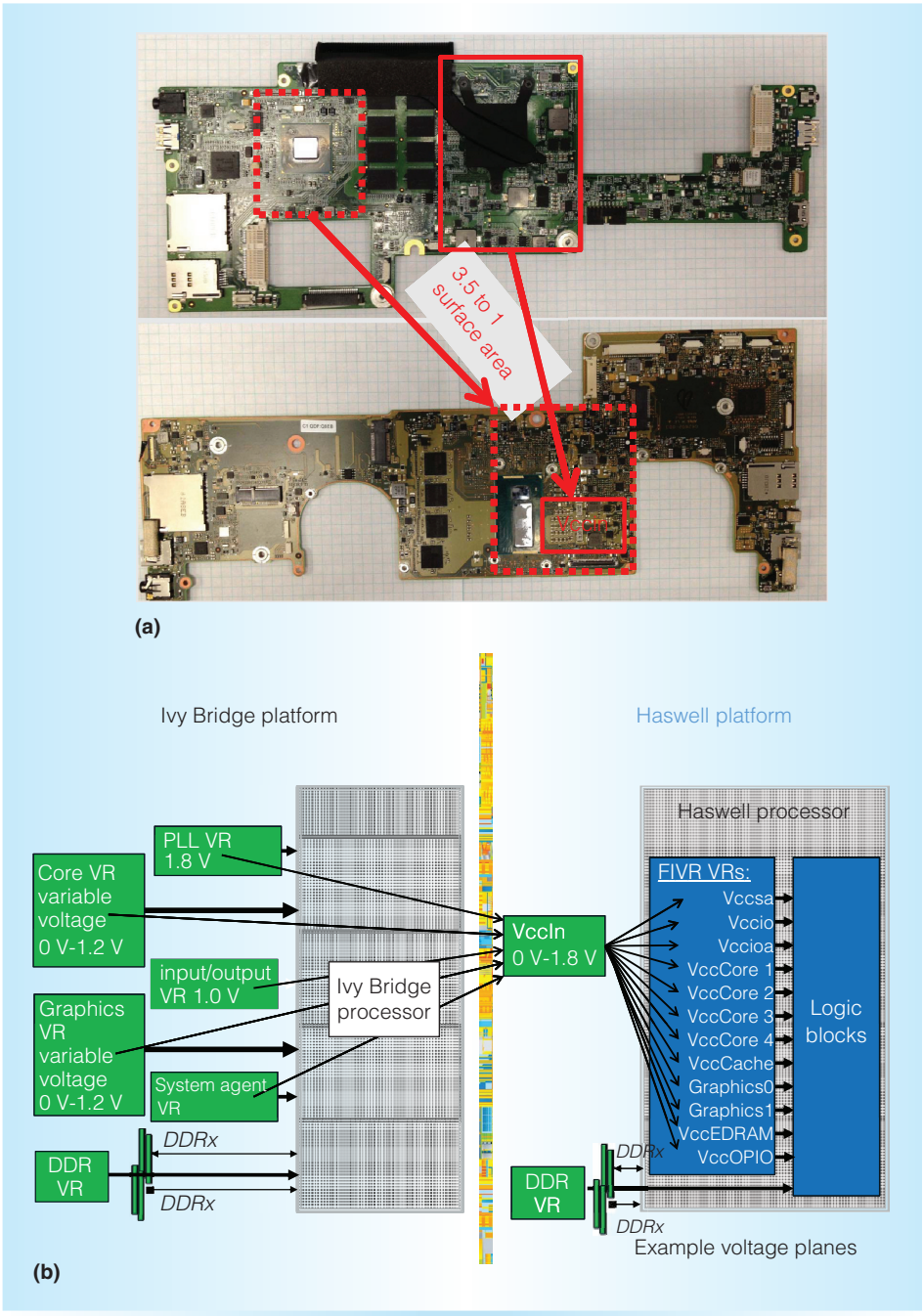

Figure 3. Example of possible platform improvements. Haswell's fully integrated voltage regulator (FIVR), shown at the bottom, enables substantial board space and cost savings compared to the Ivy Bridge platform, shown at the top (a). Multiple voltage regulators (VRs) in the previous platform are combined into one VR (b).

Haswell Peripheral Controller Hub

Significant power reduction was achieved by the power-management improvements in the Peripheral Controller Hub (PCH). The Haswell PCH is responsible for providing I/O management, and its activity depends on direct-memory-access traffic to and from peripheral devices in the PCI, USB, Serial ATA, Wi-Fi and audio-voice-speech subsystems.

A significant innovation is the introduction of exit latency timers that are programmed with the latency demands of devices attached to the I/O links. The power management controller (PMC) in the PCH uses these latencies to calculate and gracefully adapt the I/O subsystem from "high performance with low latency" to "low power with higher latency" by stopping clocks, shutting down phase-locked loops and, finally, locally power-gating the I/O control modules of each system independently. These changes allowed for a 40× power reduction in the I/O subsystem from the previous-generation PCH.

Haswell graphics and media

Haswell graphics and media are built to scale across a wide range of processor configurations, from low to very high integrated graphics and media performance. To achieve this scale, Haswell graphics were built from the start with scale in mind (see Figure 4).

Haswell graphics and media are roughly split into six domains:

- Global assets, including geometry front-end up to setup.
- Slice common—shared functions, including a rasterizer, level-3 cache (internal to graphics), and pixel back-end.
- Subslice, including shaders (execution units [EUs]), instruction caches, and texture samplers. These subslices are scalable in number to achieve the desired performance.
- Multiformat video codec engine.
- Video quality enhancement engine.
- Display pipelines.

EUs are general-purpose programmable cores that support a rich instruction set that has been optimized to support various 3D API shader languages as well as media functions (primarily video) processing.

Shared functions are hardware units that provide specialized supplemental functionality for the EUs. A shared function is implemented where the demand for a given specialized function is insufficient to justify the costs on a per-EU basis. Instead, a single instantiation of that specialized function is implemented as a stand-alone entity outside the EUs and shared among the EUs.

The graphics and media configurations can vary by the number of subslices, video decoders, and samplers to vary power and performance profiles.

The generic and traditional rendering pipeline—fetch→shade (vertex, hull, domain, geometry), rasterize, pixel shade—maps onto the global assets. A global (across graphics) cache provides storage and coherence between shared elements and allows specific cache configurations for GPGPU (general-purpose computing on GPUs) computing.

The media functions are woven into the graphics architecture and provide a scalable and programmable option for video encoding, decoding, and postprocessing. The functions include the following:

- fixed function, such as multiformat video decoders sized for ultra high-power efficiency;
- scalable assets, such as motion estimation hardware; and
- programmable postprocessing filters.

Haswell graphics are designed to be latency tolerant, because they share the memory subsystem with latency-sensitive CPU cores. Graphics and media share last-level cache in a programmable fashion with the CPU, allowing for a software-tunable cache-sharing policy for graphics and CPU hardware for optimal performance. This is further enhanced in the presence of the larger embedded DRAM (eDRAM) cache, which lets graphics access a high bandwidth.

Cache hierarchy and the eDRAM cache

Haswell delivers substantial performance improvements in cores, media, and graphics, and needs a corresponding improvement in memory bandwidth. In addition to the traditional double-data-rate (DDR) memory-speed enhancements, Haswell has further improvements in cache hierarchy pipelines, single-thread streaming write bandwidth, load balancing, and memory-scheduling efficiency.

We optimized all of the pipelines of the cache and memory hierarchy for efficiency improvements. Microarchitecture work improved pipeline efficiency. For example,

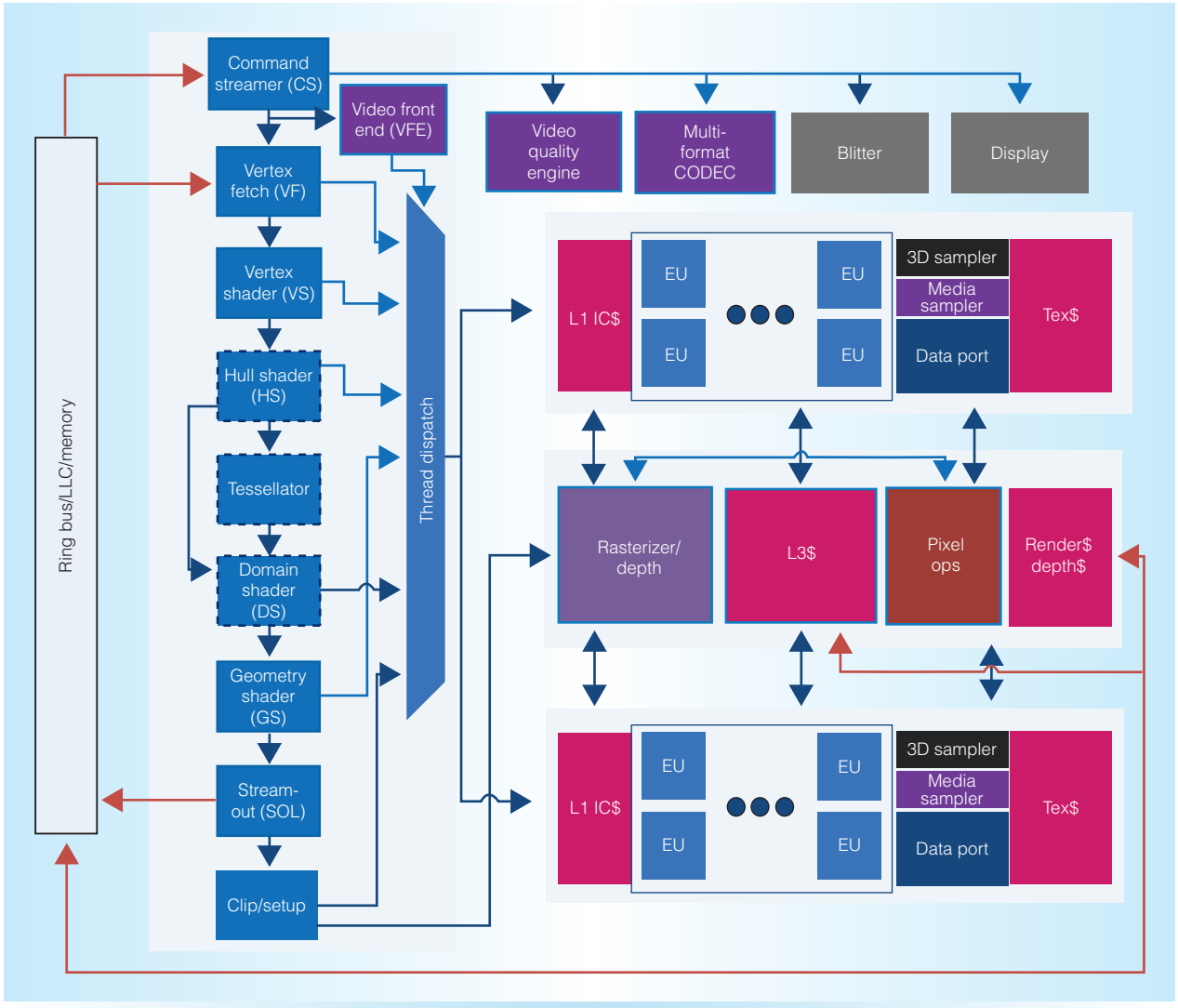

Figure 4. Haswell graphics block diagram. Haswell graphics and media are built with a modular approach, which enables scale across a range of system configurations.

optimizing the concurrency of requests—that is, handling different kinds of requests in separate pipelines—resulted in efficiency improvement of up to 40 percent. Furthermore, the cache hierarchy exploits the weakly ordered nature of write-combining stores to increase the maximum number of concurrent requests from a single Intel Architecture (IA) core from 10 to 40+. This change greatly increased the streaming write bandwidth to memory for single-thread workloads.

Load balancing between request agents was improved. The credit-based bandwidth management system was optimized to efficiently share resources. The management system achieves fairness between reliable low-latency accesses to resources with the possibility of high-bandwidth access for, for example, the graphics engine.

Haswell also has improvements in the memory schedulers for better write throughput. The Haswell memory controllers have deeper pending queues, more decoupling, and better scheduling algorithms.

These cache hierarchy improvements are good for most of the Haswell configurations. For the Haswell top-end graphics configuration, Intel Iris Pro Graphics, we needed even more bandwidth and developed a dedicated solution.

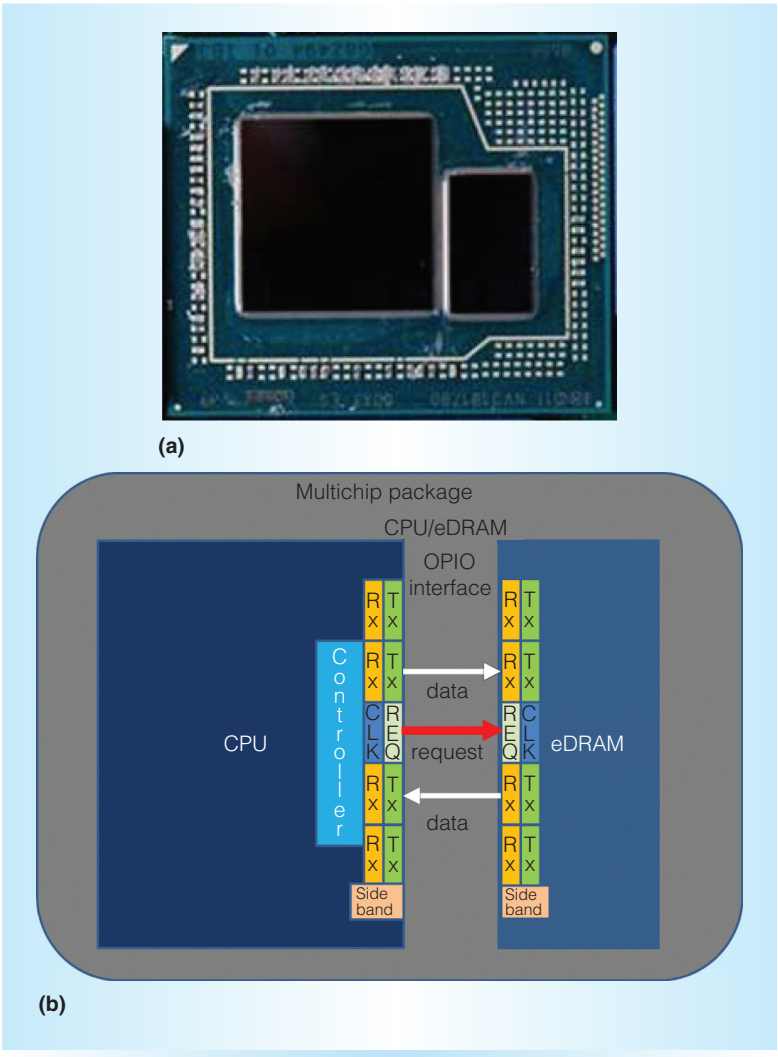

Figure 5. Intel Iris Pro Graphics multichip package ball grid array: photo of the package view (a) and block diagram (b).

eDRAM memory bandwidth solution

Meeting Intel Iris Pro Graphics performance targets required much more bandwidth than the two channels of DDRx for clients. Additional external memory channels would have been very costly and had adverse effects on the physical size in the platform. The bandwidth solution we implemented is a 128-Mbyte L4 (fourth-level) cache providing 102 Gbytes/second peak bandwidth (51 Gbytes/second read and simultaneously up to 51 Gbytes/second write). The L4 cache data store is a discrete die made using Intel eDRAM process technology, providing both high-density memory and high-speed logic for high-bandwidth I/Os.[5-7] The eDRAM die is on-package to exploit close proximity for low latency and low-power interconnect.

The L4 cache architecture with eDRAM gives the following benefits compared to other candidates:

- superior bandwidth per watt over DDRx and GDDRx,
- a unified memory bandwidth solution for both IA and graphics,
- minimized motherboard real estate for small form factors, and
- an in-package memory solution to enable package-based performance upgrade opportunities.

The Intel Iris Pro Graphics CPU and eDRAM are in a multichip package (MCP) connected using a full-duplex on-package I/O (OPIO), as shown in Figure 5. The CPU hosts the L4 cache controller, and the tags and the enhancements needed in the power control unit. A low-power, high-bandwidth link connects the CPU to the L4 and eDRAM cache die.

The eDRAM was architected to be integrated with the existing cache hierarchy with minimal impact. The eDRAM acts primarily as a victim cache for the L3, being filled by evictions. Unlike the on-die L3 cache, the eDRAM is not inclusive of the core caches, allowing graphics data to be read and written directly without the need to fill into the on-die L3, saving the L3 storage for more latency-critical IA core accesses.

The eDRAM tags are stored in traditional static RAM (SRAM) on the processor die. To save power and area, each tag entry, called a *superline,* represents a 1-Kbyte region made up of sixteen 64-byte cache lines. All incoming requests look up the on-die L3 and eDRAM in parallel.

Intel Iris Pro Graphics adds several cache controls that help graphics use the L3 and eDRAM caches. Either cache can be partitioned between IA or graphics traffic to provide quality of service. Graphics can prevent surfaces from allocating in the caches if they would not benefit from caching. Graphics is also able to use the eDRAM to cache displayable surfaces for the first time.

OPIO exploits the short trace lengths within the MCP to simplify I/O and clocking circuits to significantly reduce power,

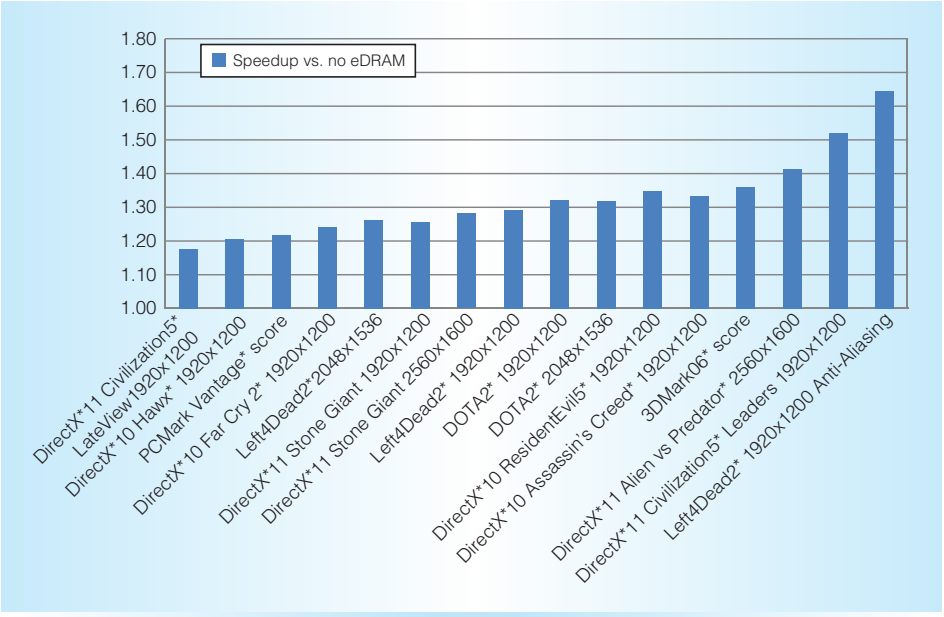

Figure 6. Performance for graphics workloads using the Intel Iris Pro Graphics eDRAM cache. (See the disclaimer in the Acknowledgments section.)

area, and latency while providing high bandwidth. OPIO uses only 1 W of total power to deliver 102 Gbytes/second. This is 3× bandwidth at 1/10 total power compared to the 32 Gbytes/second DDR3.

The Intel Iris Pro Graphics eDRAM die is Intel's first product instantiation of eDRAM technology. The ability to combine dense eDRAM memory technology and high-speed logic gave the capability to have a single die with both large capacity storage and high bandwidth.

The eDRAM die is implemented in Intel 22-nm technology. The eDRAM array is a total of 128 Mbytes and architected for high-bandwidth efficiency (also called *low-loaded latency*). There are 128 banks, each with a row cycle time of 4 ns, minimizing both the probability of a bank conflict and the penalty when it occurs. The array operates at 1.6 GHz, processing a command per clock. The entire datapath to the array and within the array is full duplex, enabling write data transfers simultaneous with read data transfers. Each data transfer takes two clocks.

The Intel Iris Pro Graphics Power Control Unit (PCU) dynamically manages the eDRAM device state depending on runtime evaluation of workload characteristics, performance goals, and energy efficiency considerations. The PCU maintains the eDRAM subsystem in one of three states: on, off, or self-refresh. In the "on" state, both eDRAM and OPIO are at their active voltage, and the eDRAM controller is sending refreshes to the eDRAM device across the OPIO link. In the "off" state, both eDRAM and OPIO clocks are stopped, and voltage is reduced to 0 V. In "self-refresh," EDRAM is kept at its active voltage, but clocks in the OPIO domain are stopped and voltage is dropped to 0 V. While the processor is active, the PCU can choose between on and off depending on the workload and power and performance goals. While the processor is idle, the PCU can choose to turn eDRAM off or put it into the self-refresh state. The PCU periodically evaluates the eDRAM's potential performance benefits and decides whether to power on the eDRAM subsystem.

eDRAM performance

The Haswell graphics performance improvements are the main bandwidth driver that motivates eDRAM. Figure 6 shows the performance gain for 128-Mbyte eDRAM over a baseline of two-channel DDR3-1600 only, for a broad set of graphics benchmarks and game titles (including a beta version of

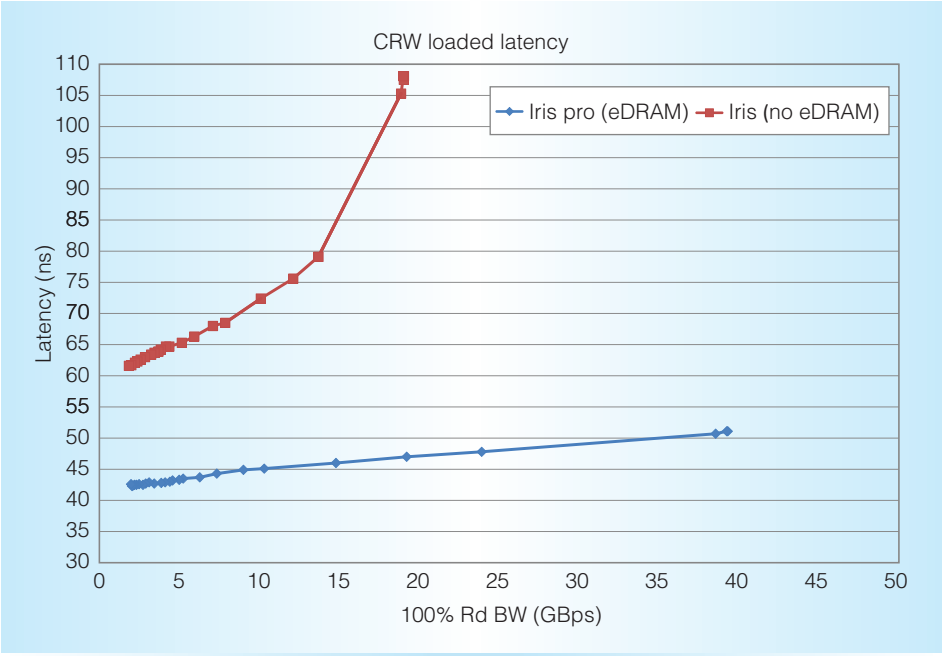

Figure 7. Loaded latency of the Intel Iris Pro Graphics eDRAM cache. (See the disclaimer in the Acknowledgments section.)

DOTA2). On average, eDRAM can increase performance by 30 to 40 percent.

The 128M eDRAM cache is large enough to exploit both inter- and intraframe data reuse. It's well known that graphics data, such as textures, are reused multiple times inside of a single frame. Extensive presilicon simulation, confirmed by silicon measurements, showed that data can also be reused between frames.

In addition to providing high-bandwidth power and energy efficiently to boost graphics performance, eDRAM is also carefully designed to minimize latency, as the outstandingly small latency sensitivity with load sustainable for random traffic shown in Figure 7, which is especially important for eDRAM to benefit not only graphics but also general CPU workloads.

Haswell core

Haswell's core is the next major evolution in general-purpose out-of-order microarchitecture, delivering both higher performance and improved power efficiency across a broad range of workloads and form factors.

Like its predecessor, Ivy Bridge, the Haswell front-end pipeline supplies micro-operations (μops) for execution from two primary sources. The first source, a traditional instruction cache/decoder pipeline, supplies μops by decoding up to 16 bytes per cycle of complex instructions into up to four compound μops. The second source, a μop cache, stores decoded μops natively and supplies them at a rate equivalent to 32 bytes per cycle.

Up to four compound μops per cycle allocate resources for out-of-order execution and are split into simple μops. Up to eight simple μops per cycle can be executed by heterogeneous execution ports. Once complete, up to four compound μops can be retired per cycle. Each Haswell core shares its execution resources between two threads of execution via Intel Hyperthreading.

Haswell's core contains several innovations for performance and power, including the following:

- *Intelligent speculation.* Haswell decouples branch prediction, instruction tag accesses, and instruction translation look-aside buffer (TLB) accesses from the supply of μops to execution. State-of-the-art advances in branch prediction algorithms enable accurate fetch requests to "run ahead" of μop

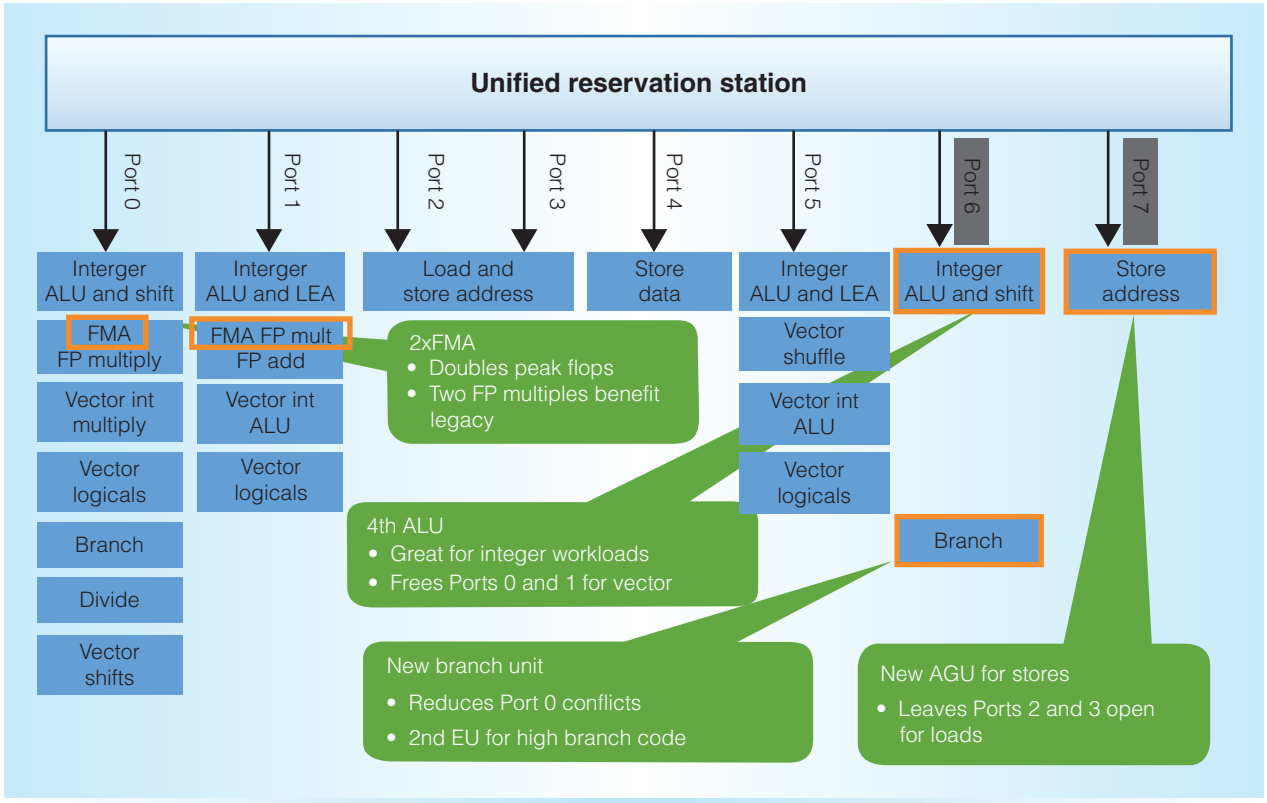

Figure 8. Haswell's execution ports and execution units. The figure shows which functional units are mapped onto which port, and which new units and ports have been added. (LEA: load effective address.)

supply to hide instruction TLB and cache misses. A new data TLB prefetcher keeps page walk latency from delaying many memory accesses.

- *A large out-of-order window.* Haswell maintains 192 μops in flight in its reorder buffer and the supporting structures, such as load buffers (72), store buffers (42), reservation stations (60), and physical register files (168). This is approximately a 15 percent increase over Ivy Bridge to extract more parallelism.
- *Raw execution horsepower.* Haswell provides eight heterogeneous execution ports, an increase over the six in Ivy Bridge. The two new ports and additions to existing ports combine to provide a fourth integer ALU, a second branch unit, a second floating-point multiplier, and a third store address-generation unit. The additional resources provide higher peak throughput and fewer false resource conflicts. See Figure 8 for details.
- *Aggressive clock and data gating, and new dynamic power-saving modes.* The Haswell core achieves power efficiency by aggressively gating idle logic. New power-saving modes adapt to workload phases to save power without sacrificing performance.

With these improvements, the Haswell core delivers improvements across a range of workloads.

Instruction set enhancements

Haswell delivers performance improvements on legacy and unchanged codes, and also adds new instructions for even more performance on suitable workloads.

Advanced Vector Extensions 2

With Advanced Vector Extensions 2 (AVX2), Haswell adds instructions for FMA, 256-bit integer vector computation, full-width element permute, and vector gather to benefit high-performance computing, audio and video processing, and games.

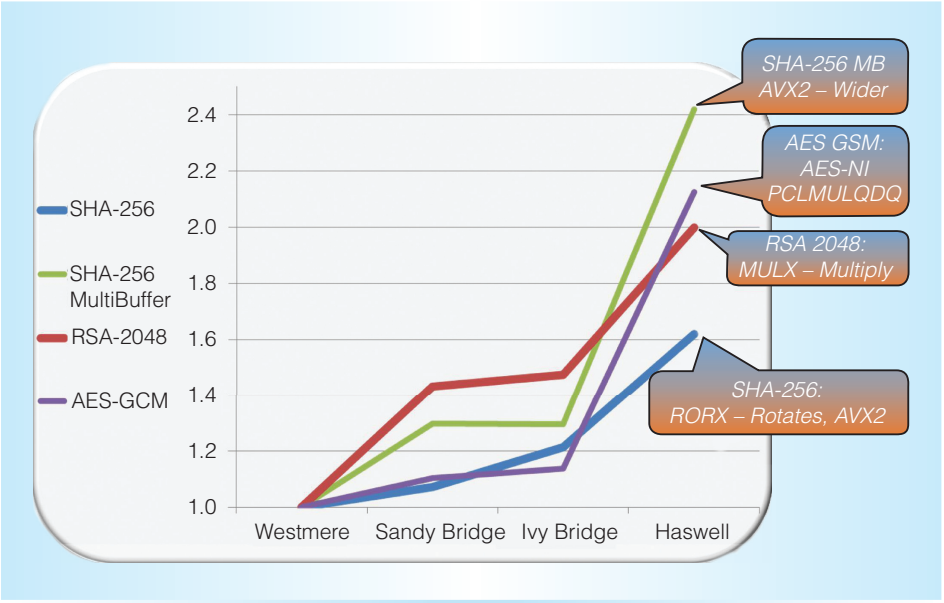

Figure 9. Examples of performance improvements with new instructions in the Haswell core. (See the disclaimer in the Acknowledgments section.)

Each Haswell core provides up to 32 single-precision or 16 double-precision floating-point operations per cycle using AVX2's FMA instructions and Haswell's two FMA hardware units. FMA operations can be done with the same latency as a floating-point multiply (five cycles) and are fully pipelined. This achieves a 1.6× latency reduction versus the previous generation. Haswell's FMA can be used for both computational throughput enhancement and for latency reductions.

The Haswell core complements the addition of AVX2 with doubled L1 and L2 cache bandwidth. Each L1 data cache port, two loads and one store, natively supports full-width AVX2 operations at 32 bytes. The L2 cache can return a full cache line of data, 64 bytes, to the L1 data cache. Despite these bandwidth increases, the L1 and L2 cache sizes and latencies remain the same as Ivy Bridge.

New instructions for hashing and cryptography

Haswell provides several packages of new instructions to benefit high-value workloads that rely on bit-field manipulation and arbitrary precision arithmetic. These new instructions benefit many algorithms ranging from fast indexing, to cyclical redundancy checking (CRC), to widespread encryption algorithms. Figure 9 summarizes the performance.

Intel Transactional Synchronization Extensions

With Intel Transactional Synchronization Extensions (Intel TSX), Haswell adds hardware support to improve the performance of lock-based synchronization commonly used in multithreaded applications. These applications take advantage of the increasing number of cores to improve performance. However, when writing such applications, programmers must use concurrency-control protocols to ensure threads properly coordinate access to shared data. Otherwise, threads might observe unexpected data values, resulting in failures. These protocols ensure that threads serialize access to shared data, often by using a critical section. A software construct, referred to as a *lock,* protects access to the critical section.

Because serialized access to a critical section limits parallelism, programmers try to reduce the impact of critical sections, either by minimizing synchronization or by using fine-granularity locks, where multiple locks protect different data. Unfortunately, this is a difficult and error-prone process and a common source of bugs. Furthermore, programmers must use information available only at the time of program development to decide whether synchronization is required

and when to serialize. This often leads programmers to use synchronization conservatively, thus limiting parallelism.

To address this challenge, Intel TSX provides hardware support for the processor to determine dynamically if it should serialize critical section execution and expose any hidden concurrency.

Overview

With Intel TSX, the processor executes critical sections, also referred to as *transactional regions,* transactionally using a technique known as *lock elision.* Such an execution only reads the lock, and does not acquire it. This makes the critical section available to other threads executing at the same time.

However, because the lock no longer serializes access, hardware now ensures that threads access shared data correctly by maintaining the illusion of exclusive access to the data. It does so by tracking memory addresses accessed and by buffering any memory updates performed within the transactional execution. The processor also checks these accesses against conflicting accesses from other threads. A conflicting access occurs if another thread reads a location that this transactional thread wrote, or another thread writes a location that was accessed (either using a read or a write) by the transactional thread. When such an access occurs, the hardware alone cannot maintain the illusion of exclusive access.

To commit a transactional execution, the hardware ensures that all memory operations performed during the execution appear to occur instantaneously when viewed from other threads. Furthermore, any memory updates during execution become visible to other threads only after a successful commit.

Not all transactional executions can be successfully committed. For example, the execution could encounter conflicting data accesses from other threads. In that event, the processor performs a transactional abort. This process discards all updates, memory, and registers during the execution and makes it appear as if the transactional execution never occurred. The subsequent re-execution can retry lock elision or fall back to acquiring the lock.

Because a successful transactional execution ensures an atomic commit, the processor can execute the programmer-specified code section optimistically without synchronizing through the lock. If synchronization was unnecessary, the execution can commit without any cross-thread serialization.

Programming interface

Intel TSX provides two programming interfaces to specify transactional regions. The first interface, called Hardware Lock Elision (HLE), is a pair of legacy-compatible prefixes called XACQUIRE and XRELEASE. These prefixes appear as NOPs to previous-generation cores. The second interface, called Restricted Transactional Memory (RTM), is a pair of new instructions called XBEGIN and XEND. Programmers who also want to run Intel TSX-enabled software on hardware without Intel TSX support would use the HLE interface to implement lock elision. Programmers who do not have legacy hardware requirements and who deal with more complex locking primitives would use the RTM interface to implement lock elision.

Programmers can use the HLE interface by adding prefixes to the instructions that perform the lock acquire and release. Programmers can use the RTM interface by providing an additional code path to the existing synchronization routines. In this path, instead of acquiring the lock, the routine uses the XBEGIN instruction and provides a fallback handler to execute if a transactional abort occurs. The code also tests the lock variable inside the transactional region to ensure that it's free and to enable the hardware to look for subsequent conflicts. These changes to enable lock elision are localized to synchronization routines; the application itself does not need to be changed. This substantially eases enabling of applications to take advantage of Intel TSX.

Intel TSX provides two additional instructions—XTEST and XABORT. The XTEST instruction tests if a logical processor is executing transactionally, whereas the XABORT instruction can explicitly abort a transactional region.

Implementation on Haswell

The first implementation of Intel TSX on the fourth-generation core processor uses the first-level 32-Kbyte data cache (L1) to track

the memory addresses accessed (both read and written) during a transactional execution and to buffer any transactional updates performed to memory. The implementation makes these updates visible to other threads only on a successful commit. The implementation uses the cache coherence protocol to detect conflicting accesses from other threads. Because hardware is finite, transactional regions that access excessive state can exceed hardware buffering. Evicting a transactionally written line from the data cache will cause a transactional abort. However, evicting a transactionally read line does not immediately cause an abort. The hardware moves the line to a secondary structure for subsequent tracking.

The *Intel 64 Architecture Software Developer Manual* has a detailed specification for Intel TSX,[8] and the *Intel 64 Architecture Optimization Reference Manual* provides detailed guidelines for program optimization with Intel TSX.[9] The Intel TSX web resources site (www.intel.com/software/tsx) presents information on various tools and practical guidelines.

Haswell uses an optimized version of Intel 22-nm process technology to provide comprehensive enhancements in power-performance efficiency, power management, form factor and cost, core and uncore microarchitecture enhancements, and new instructions in the core. For example, the core delivers performance enhancements for high-performance computing with the new FMA instructions and for parallel workloads with the new Intel TSX synchronization primitives.[10,11]

During Haswell's design, we focused on performance, power, and form factors to ensure that Haswell delivers a compelling user experience in new form factors. MICRO

Acknowledgments

We thank Kevin Zhang, Fatih Hamzaoglu, Eric Wang, Ruth Brain and the Intel TMG Organization, Dave Dimarco, Manoj Lal, Steve Kulick, the Haswell architecture team, and the entire Intel CCDO team, and Patty Kummrow and the SDG Org for their significant contributions to the success of eDRAM and the Intel Iris Pro Graphics product.

Disclaimer for Figures 6, 7, and 9: Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Results have been simulated and are provided for informational purposes only. Results were derived using simulations run on an architecture simulator or model. Any difference in system hardware or software design or configuration may affect actual performance. Requires a system with Intel Turbo Boost Technology. Intel Turbo Boost Technology and Intel Turbo Boost Technology 2.0 are only available on select Intel processors. Consult your system manufacturer. Performance varies depending on hardware, software, and system configuration. For more information, visit http://www.intel.com/go/turbo. Iris graphics is available on select systems. Consult your system manufacturer.

Optimization notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

References

1. P. Hammarlund, "Intel 4th Generation Core Processor (Haswell)," Hot Chips 25, 2013.
2. N. Siddique et al., "Haswell: A Family of IA 22nm Processors," to be published in *Proc. IEEE Int'l Solid-State Circuits Conf.*, 2014.

3. E. Burton et al., "FIVR—Fully Integrated Voltage Regulators on 4th Generation Intel Core SoCs," to be published in *Proc. IEEE Applied Power Electronics Conf. and Exposition*, 2014.
4. D. Kanter, "Haswell's FIVR Extends Battery Life," *Microprocessor Report*, 30 July 2013.
5. R. Brain et al., "eDRAM process in 22nm Technology," *Proc. Symp. VLSI Circuits*, 2013, pp. T16-T17.
6. Y. Wang et al., "Retention Time Optimization for eDRAM in 22nm Tri-Gate CMOS Technology," *Proc. IEEE Int'l Electron Devices Meeting*, 2013, pp. 240-243.
7. F. Hamzaoglu et al., "A 1Gb 2GHz Embedded DRAM in 22nm Tri-Gate CMOS Technology," to be published in *Proc. IEEE Int'l Solid-State Circuits Conf.*, 2014.
8. *Intel 64 and IA-32 Architectures Software Developer Manual*, Intel, 2013.
9. *Intel 64 and IA-32 Architectures Optimization Reference Manual*, Intel, 2013.
10. R. Yoo et al., "Performance Evaluation of Intel Transactional Synchronization Extensions for High Performance Computing," *Proc. Int'l Conf. High Performance Computing, Networking, Storage and Analysis*, 2013, doi:10.1145/2503210.2503232.
11. T. Karnagel et al., "Improving In-Memory Database Index Performance with Intel Transactional Synchronization Extensions," to be published in *Proc. 20th Int'l Symp. High-Performance Computer Architecture*, 2014.

Per Hammarlund is an Intel fellow. His research interests include performance modeling, microarchitecture, SMT, power-performance efficiency and modeling, integration, and SoC design. Hammarlund has a PhD in computer science from the Royal Institute of Technology (KTH), Stockholm.

Alberto J. Martinez is a senior principal engineer at Intel and the chief architect for the Embedded Subsystems and IP Group. His research focuses on embedded subsystems software and hardware and PC I/O architectures. Martinez has an MS in electrical engineering from Sacramento State University.

Atiq A. Bajwa is the director of microprocessor architecture at Intel. He is responsible for the architectural definition and development of microprocessors for Intel's mobile, desktop, workstation, and server computing segments. Bajwa has an MS in electrical engineering from Yale University.

David L. Hill is a senior principal engineer at Intel. His research interests include modular high-performance caches, coherent interconnects, and system memory technologies. His team was responsible for the modular uncore architecture sections of the Haswell and Broadwell product families. Hill has a BS in electrical engineering from the University of Minnesota.

Erik Hallnor is a cache and coherent fabric architect, focusing on client and SoC products at Intel. He was the lead architect for the Haswell coherent fabric, consisting of the ring interconnect, LLC, and eDRAM cache integration. Hallnor has a PhD in computer science and engineering from the University of Michigan.

Hong Jiang is an Intel Fellow, the chief media architect for the Platform Engineering Group, and the director of the Visual and Parallel Computing Group's Media Architecture Team at Intel. He leads the media architecture of processor graphics and its derivatives. Jiang has a PhD in electrical engineering from the University of Illinois at Urbana-Champaign.

Martin Dixon is a principal engineer in the Intel Product Development Group, where he's working to develop and enhance the overall instruction set and SoC architecture. Dixon has a BS in electrical and computer engineering from Carnegie Mellon University.

Michael Derr is a principal engineer at Intel. His work focuses on power managing PC I/O architectures. Derr has an MS in electrical engineering from the Georgia Institute of Technology.

Mikal Hunsaker is a senior principal engineer at Intel. His research interests focus on chipset high-speed serial I/O design, including PCI Express, SATA, and USB3. Hunsaker has an MS in electrical engineering from Utah State University.

Rajesh Kumar is a senior fellow, director of circuit and power technologies, and the lead interface to process technology at Intel. For Haswell, he guided the development of power delivery integration (FIVR), on-package I/O, and the novel process-technology needs. Kumar has a master's degree in electrical engineering from the California Institute of Technology.

Randy B. Osborne is a principal engineer working to improve performance of memory hierarchies at Intel. His research interests include memory controllers, memory interconnects, memory devices, and large caches, including the introduction of eDRAM into Intel products. Osborne has a PhD in electrical engineering from the Massachusetts Institute of Technology.

Ravi Rajwar is a principal engineer in the Intel Product Development Group, working on various aspects of SoC architecture and development. His research interests include the IA synchronization architecture. Rajwar has a PhD in computer science from the University of Wisconsin–Madison.

Ronak Singhal is a senior principal engineer at Intel. His research interests include server architecture development, ISA development, and performance analysis and modeling. Singhal has an MS in electrical and computer engineering from Carnegie Mellon University.

Reynold D'Sa is vice president of the Platform Engineering Group and general manager of the Devices Development Group at Intel. He leads the design engineering teams responsible for designing and developing SoC products for Intel's next-generation client and mobile platforms, including tablets and smartphones. D'Sa has an MS in electrical engineering from Cornell University.

Robert Chappell is a CPU architect at Intel, where he works to define the performance, power, reliability, and ISA features of the CPU cores used in various products ranging from phones to servers. He was the lead architect for the Haswell memory execution cluster, the Haswell core in its later stages, and the Haswell follow-on core (Broadwell). Chappell has a PhD in computer science from the University of Michigan.

Shiv Kaushik is a fellow at Intel, where he leads the Windows OS Division in Intel's Software and Services Group. His research interests include the design of platform hardware and firmware interfaces to operating systems and virtualization software for power management, scaling, performance, and reliability. Kaushik has a PhD in computer science and engineering from Ohio State University.

Srinivas Chennupaty is a CPU and SoC architect at Intel. His research interests include the Intel microprocessor architecture and instruction set development. Chennupaty has an MS in computer engineering from the University of Texas at Austin.

Stephan Jourdan is a senior principal engineer at Intel, where he leads the architecture engineering teams responsible for defining and developing SoCs for device products. He was the chief architect on HSW ULT. Jourdan has a PhD in computer science from the University of Toulouse.

Steve Gunther is a senior principal engineer and a lead power architect for Intel, where he leads a team responsible for defining the power management architecture for Intel's microprocessor product line. His research interests include power analysis, power management, and power reduction. Gunther has a BS in electrical engineering from Oregon State University.

Tom Piazza is a senior fellow and director of graphics architecture at Intel. His research interests include computer graphics. Piazza has a BS in electrical engineering from the Pratt Institute.

Ted Burton is a senior principal engineer working on advanced technologies in the Devices Development Group at Intel. His research interests include power delivery. Burton has a BS in physics from Brigham Young University.

Direct questions and comments about this article to Per Hammarlund, Intel, 2111 NE 25th Ave., Hillsboro, OR 97124; per.hammarlund@intel.com.