An Energy Efficiency Feature Survey of the Intel Haswell Processor

Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, Robin Geyer
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden – 01062 Dresden, Germany
Email: {daniel.hackenberg, robert.schoene, thomas.ilsche, daniel.molka, joseph.schuchart, robin.geyer}@tu-dresden.de

***Abstract*—The recently introduced Intel Xeon E5-1600 v3 and E5-2600 v3 series processors–codenamed Haswell-EP–implement major changes compared to their predecessors. Among these changes are integrated voltage regulators that enable individual voltages and frequencies for every core. In this paper we analyze a number of consequences of this development that are of utmost importance for energy efficiency optimization strategies such as dynamic voltage and frequency scaling (DVFS) and dynamic concurrency throttling (DCT). This includes the enhanced RAPL implementation and its improved accuracy as it moves from modeling to actual measurement. Another fundamental change is that every clock speed above AVX frequency–including nominal frequency–is opportunistic and unreliable, which vastly decreases performance predictability with potential effects on scalability. Moreover, we characterize significantly changed p-state transition behavior, and determine crucial memory performance data.**

I. Introduction and Motivation

Intel processors code named "Haswell" are the first x86 processors that include on-die fully integrated voltage regulators (FIVR [1]). Moreover, the server processors (E5-1600 v3 and E5-2600 v3) include one voltage regulator per processor core, which enables fine-grained p-states. Also other features have been improved in core and uncore design for the Haswell processor generation. All these innovations have to be understood to make optimal use of the processors in terms of energy-efficient computing.

In this paper we cover a broad range of details and low-level benchmarks to provide researchers with fundamental understanding about the Intel Haswell processor generation. The paper is structured as follows: We describe micro-architectural improvements, new energy efficiency features, and innovative frequency specifications in Section II. Our specific test system is detailed in Section III. We validate the internal energy measurement mechanism in Section IV and analyze the transparent hardware settings for uncore and AVX frequencies ins Section V. In Section VI, we explain how fast Haswell processors can change between ACPI performance and power states. We describe how concurrency and frequency changes influence L3 and main memory bandwidths in Section VII. Finally, we describe how we maximize the power consumption of the processors under test.

II. Haswell Architecture and Features

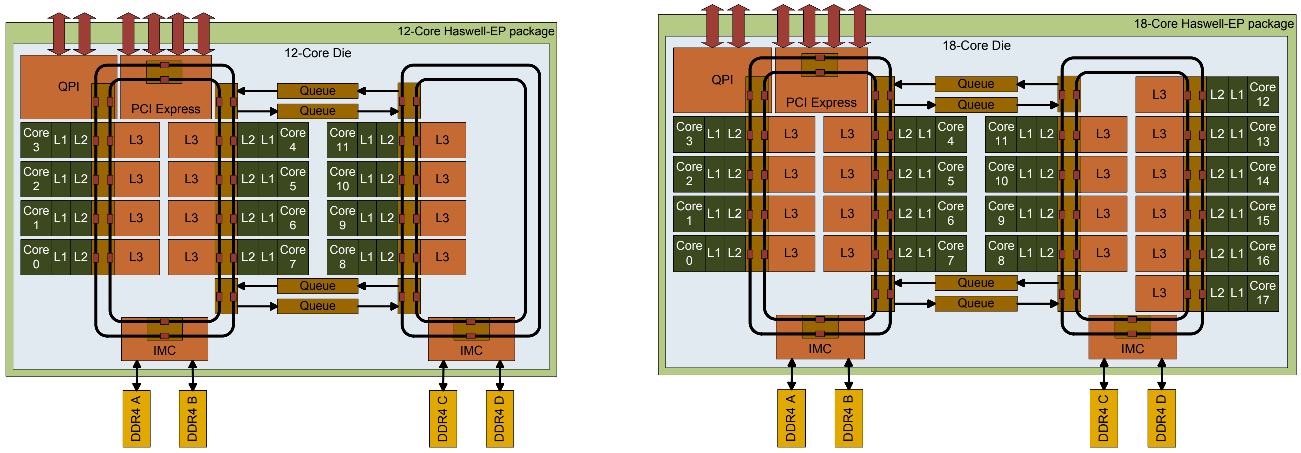
A. Microarchitecture

Compared to the previous Sandy/Ivy Bridge generation [2, Section 2.2], the Intel Haswell microarchitecture [2, Section 2.1] implements several enhancements in the core and uncore design. Table I details the major differences. Decode and retirement stay a 4-wide superscalar design. However, the execution and out-of-order resources have been increased significantly in order to extract more instruction parallelism. The ISA extensions AVX2 (256 bit integer SIMD instructions) and fused multiply-add (FMA) significantly increase the theoretical peak performance. Two AVX or FMA operations can be issued per cycle, except for AVX additions [2, Table 2-1], which creates a subtle optimization potential for addition-intense codes [3, Section 10.14]. To provide the execution units with enough data, the L1D and L2 cache bandwidths have also been doubled. Furthermore, the integrated memory controller (IMC) of the Haswell-EP line (Xeon E5-nnnn v3) supports DDR4 which increases the memory bandwidth as well.

Similar to the predecessors Nehalem-EX [7], Sandy Bridge-EP [8], and Ivy Bridge-EP/EX [9], a ring interconnect is used for on-chip communication. Haswell-EP [10] processors are available with 4 to 18 cores. Three different dies cover this range: The 8-core die (4/6/8 core units) uses a single bidirectional ring to connect the cores and uncore components. The 12-core die (10/12 core units) consists of an 8-core and a 4-core partition (see Figure 1a). The 18-core die (14/16/18 core units) consists of two bidirectional rings, one with 8 and one with 10 cores (see Figure 1b). Each partition has an integrated memory controller (IMC) for two memory channels. The rings are connected via queues to enable data transfers between the partitions. However, in the default configuration this complexity is not exposed to software.

Microarchitecture	Sandy Bridge-EP	Haswell-EP
References	[4];[2, Section 2.2]	[5];[2, Section 2.1]
Decode	4(+1) x86/cycle	
Allocation queue	28/thread	56
Execute	6 micro-ops/cycle	8 micro-ops/cycle
Retire	4 micro-ops/cycle	
Scheduler entries	54	60
ROB entries	168	192
INT/FP register file	160/144	168/168
SIMD ISA	AVX	AVX2
FPU width	2×256 Bit (1 add, 1 mul)	2×256 Bit FMA
FLOPS/cycle (double)	8	16
Load/store buffers	64/36	72/42
L1D accesses per cycle	2×16 byte load + 1×16 byte store	2×32 Byte load + 1×32 Byte store
L2 bytes/cycle	32	64
Supported memory	4×DDR3-1600	4×DDR4-2133
DRAM bandwidth	up to 51.2 GB/s	up to 68.2 GB/s
QPI speed	8 GT/s (32 GB/s)	9.6 GT/s (38.4 GB/s)

TABLE I. Comparison of Sandy Bridge and Haswell microarchitecture



(a) 12 core Haswell-EP

(b) 18 core Haswell-EP

Fig. 1. Haswell-EP implementations with partitioned ring interconnect [6]. The 12-core die (left) is used for 10-core and 12-core units. The 18-core die (right) is used for 14, 16, and 18-core units.

B. *Voltage Regulators*

The Intel Haswell-EP processors include multiple fully integrated voltage regulators (FIVR [1]). A mainboard voltage regulator (MBVR) is still needed, but only three[1] voltage lanes are attached to the processor [11, Section 2.1]: processor V_{CCin}, DRAM channels 0 and 1 V_{CCD_01}, and DRAM channels 2 and 3 V_{CCD_23}. The processor controls the input voltage by sending serial voltage ID (SVID) signals to the MBVR, which then regulates V_{CCin} accordingly. As with previous processors, the MBVR supports three different power states which are activated by the processor according to the estimated power consumption [11, Section 2.2.9].

C. *Performance and Energy Bias Hint*

The *Performance and Energy Bias Hint* (also referred to as energy performance bias – EPB) influences the selection of the processor's operating frequencies. It can be set to optimal performance (*performance*), low power (*energy saving*), or a balanced approach which considers both (*balanced*). The EPB setting can be changed by writing the configuration into 4 bits of a model-specific register (MSR). However only 3 of the possible 16 settings are defined. A setting of 0, 6, and 15 can be used for *performance*, *balanced*, and *energy saving*, respectively. According to our measurements, other settings are mapped to *balanced* (1-7) and *energy saving* (8-14). In [12, Table 2], Intel describes different mappings with more categories.

Even though EPB has been introduced with earlier processors, it plays a more important role in the Haswell-EP microarchitecture. It is used by several energy efficiency features. Both uncore frequency scaling (UFS, see II-D) and energy-efficient turbo (EET, see II-E) partly base their frequency decisions on the EPB setting, which can be selected in the BIOS and via software tools [13, Section 14.3.4],[12]. When setting EPB to *performance*, turbo mode will be active even when the base frequency is selected.

[1]compared to five voltage lanes on previous products

D. *Per-Core P-States and Uncore Frequency*

The FIVRs (see Section II-B) in Haswell-EP provide individual voltages for every core. This enables per-core p-states (PCPS) [14] instead of one p-state for all cores as in previous products. The finer granularity of voltage and frequency domains enables energy-aware runtimes and operating systems to lower the power consumption of single cores while keeping the performance of other cores at a high level.

Another new feature, uncore frequency scaling (UFS), enables the processor to control the frequency of the uncore components (e.g., last level caches) independently of the core frequencies. Previous Intel processor generations used either a fixed uncore frequency (Nehalem-EP and Westmere-EP) or a common frequency for cores and uncore (Sandy Bridge-EP and Ivy Bridge-EP). The uncore frequency has a significant impact on on-die cache-line transfer rates as well as on memory bandwidth. According to the respective patent [15], the uncore frequency depends on the stall cycles of the cores, the EPB of the cores, and c-states. By default it is set by hardware and can be specified via the MSR `UNCORE_RATIO_LIMIT` [16]. However, neither the actual number of this MSR nor the encoded information is available.

E. *Energy-Efficient Turbo*

High turbo frequencies—typically only limited by power or thermal constraints—tend to hurt energy efficiency, especially if the performance increase is negligible. The energy-efficient turbo (EET) feature [17] attempts to reduce the usage of turbo frequencies that do not significantly increase the performance. EET monitors the number of stall cycles and uses this information as well as the EPB setting (see II-C) to select a frequency that is predicted to be optimal. However, the monitoring mechanism polls the stall data only sporadically (the patent lists a period of 1 ms [17, Table 4.8]). Therefore, EET may impair performance and energy efficiency of workloads that change their characteristics at an unfavorable rate.

F. AVX Frequencies

Another feature that has been added to provide system stability are AVX frequencies, which are activated when 256-bit wide AVX instructions are used. The Haswell CPU family uses a lower clock frequency for workloads with substantial amounts of AVX instructions. According to [18], the workflow of AVX frequency transitions is as follows:

- AVX instructions draw more current and higher voltage is needed to sustain operating conditions
- The core signals the Power Control Unit (PCU) to provide additional voltage and slows the execution of AVX instructions
- To maintain the limits of the thermal design power (TDP), the increasing voltage may cause a drop in clock frequency
- The PCU signals that the voltage has been adjusted and core returns to full execution throughput
- The PCU returns to regular (non-AVX) operating mode 1 ms after AVX instructions are completed

Multiple new frequencies are defined, e.g., *AVX base* and *AVX max all core turbo*. The former defines the minimal frequency that is guaranteed when running AVX workloads. The latter is the maximal frequency that will be provided for AVX workloads that use all cores. Further, *AVX turbo* frequencies are defined for various core counts [10, Table 3]. Every frequency above *AVX base*, (even the base frequency) can be considered turbo and is potentially limited by the TDP.

On our test system, *AVX base* is defined as 2.1 GHz. The *AVX turbo* frequencies are between 2.8 and 3.1 GHz, depending on the number of active cores on a processor.

III. Test System Setup

For our experimental evaluation we use a bullx R421 E4 compute node with a Supermico X10DRG-H mainboard and two Intel Xeon E5-2680 v3 processors (see Table II). We experimentally discovered that the cores' voltages for a given p-state differ on the two processors. On average, the cores of the second processor have a higher voltage than the cores of the first processor. The first processor also appears to use lower sustained turbo frequencies, possibly due to thermal reasons.

We use a calibrated LMG450 power meter [19] by ZES ZIMMER for our reference power measurements. It provides AC power consumption data for the full node at 20 Sa/s. Internally it samples the voltage and current at a substantially higher rate to achieve its accuracy.

IV. Quality of RAPL energy measurements

The RAPL (running average power limiting) interface describes model specific registers (MSRs) that can be read for obtaining energy consumption information for different components such as the processor package and DRAM. The raw energy values from these registers have to be multiplied by an *energy unit* (listed in another MSR) to compute the correct information. With the introduction of FIVRs, the RAPL measurements are becoming more reliable.

Processor	2x Intel Xeon E5-2680 v3
Frequency range (selectable p-states)	1.2 – 2.5 GHz
Turbo frequency	up to 3.3 GHz
AVX base frequency	2.1 GHz
Energy perf. bias	balanced
Energy-efficient turbo (EET)	enabled
Uncore frequency scaling (UFS)	enabled
Per-core p-states (PCPS)	enabled
Idle power (fan speed set to maximum)	261.5 Watt
Power meter	ZES LMG 450 [19]
Accuracy	0.07 % + 0.23 W

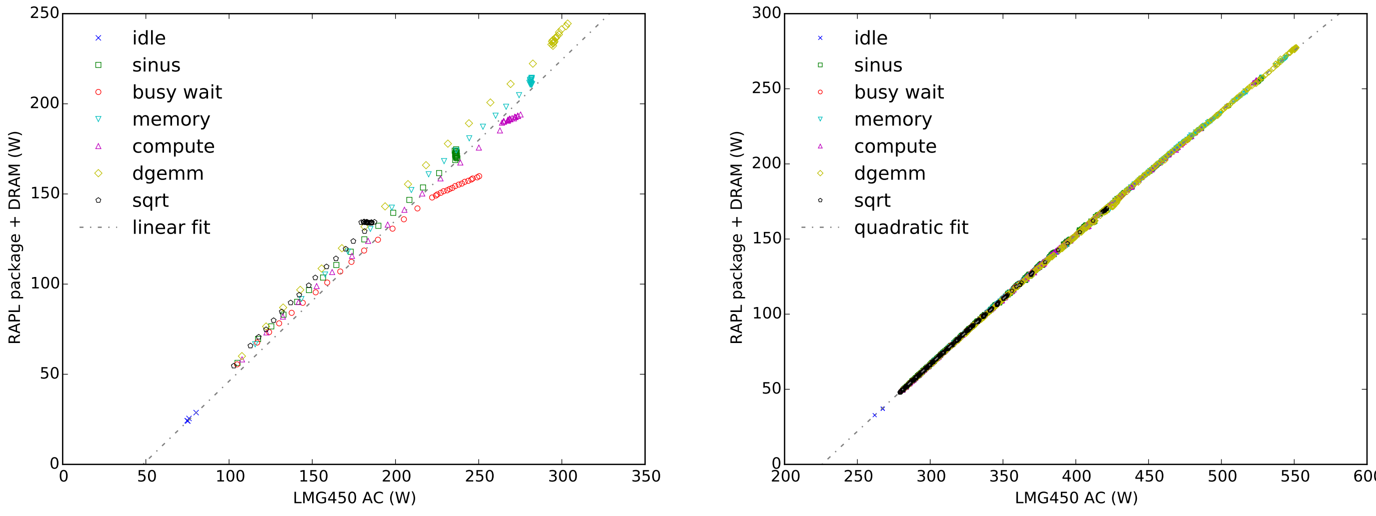
TABLE II. Test system details

On Haswell-EP, RAPL covers power domains for package and DRAM. The power domain for core consumption (PP0) is not supported on Haswell-EP. It is important to note that the DRAM energy consumption for Haswell-EP should *not* be calculated based on the information given in [13, Section 14.9], but with the energy unit given in [21, Section 5.3.3] instead: "ENERGY_UNIT for DRAM domain is 15.3 µJ." Although this reference only describes measuring DRAM RAPL via the PCI uncore registers, the energy unit is apparently also valid for the RAPL MSRs. Using the information provided in [13] would result in unreasonable high values for DRAM power consumption. While RAPL DRAM mode 0 may still be available in a system's BIOS, only RAPL DRAM mode 1 is supported in Haswell-EP. Using DRAM mode 0 will result in unspecified behavior. Our experiments show a high precision of the readings obtained in DRAM mode 1.

To verify the energy measurements, we run several micro-benchmarks in different threading configurations. To avoid interference effects, e.g., due to time synchronization, we use the average power consumption of a constant load during four seconds. Our reference measurement is done at a different domain (AC power) and therefore also includes other consumers such as fans, power supply losses, and mainboard voltage regulators. The setup is designed to keep the power consumption of other components in the system constant, e.g., by using constant fan speeds and avoiding any I/O during the test. The power supply losses are likely to be nonlinear.

If RAPL measurements were perfect, it could be expected that the reference measurement values are a single continuous function of RAPL package + DRAM values. For previous generations of modeled RAPL values, we have observed different functions depending on the benchmark type [20]. On Sandy Bridge-EP, a bias towards certain workloads can be noted (see Figure 2a). In contrast, the results for the Haswell-EP system with RAPL DRAM mode 1 (see Figure 2b) shows an almost perfect correlation to the AC reference measurement that can be approximated with a quadratic fit.[2] This is a significant improvement compared to RAPL on previous processor generations. Due to the different reference domain, the absolute calibration of RAPL cannot be confirmed. The remaining deviation of measurement samples from the quadratic fit is below 3 W and well within the influences of our measurement setup. We observed similarly good results on a Haswell-HE platform, also benefiting from the availability of the DRAM domain in contrast to previous generation desktop platforms.

[2] $P_{AC} = 0.0003 P_{RAPL}^2/W + 1.097 * P_{RAPL} + 225.7W$, $R^2 > 0.9998$.



(a) Sandy Bridge-EP

(b) Haswell-EP

Fig. 2. Comparison of power measurements with RAPL (package + DRAM, sum of two sockets) on Sandy-Bridge EP [20] and Haswell-EP versus total system power consumption (AC) measured with a high-accuracy power meter. Note that the different x-axis ranges are due to the full speed fan settings on the Haswell-EP system which does not affect RAPL accuracy.

V. Transparent Frequency Scaling Mechanisms

A. Uncore Frequency

As described in Section II-D, the uncore frequency is set transparently for operating system and user. We therefore determine the uncore frequency bounds of our test system using benchmarks. To get a lower bound for uncore frequencies, we execute a benchmark that does not access any memory. We run a `while(1)`-loop on one core of the system and measure the uncore frequency[3] on both uncores using LIKWID [22] for 10 seconds. Additionally, we change the core frequencies to examine whether these influence the uncore frequencies as well. The results are presented in Table III. They indicate that uncore frequencies–in addition to EPB and stall cycles–depend on the core frequency of the fastest active core on the system. Moreover, the uncore frequency is also a target of power limitations.

The upper bound for the uncore frequency in memory-stall scenarios is 3.0 GHz on our system, also for lower core frequencies.

Our analysis also shows that the uncore clock is halted when a processor goes into deep package sleep state (PC-3/PC-6). However, these states are not used when there is still any core active in the system—even if this core is located on the other processor.

[3]`UNCORE_CLOCK:UBOXFIX`

B. AVX Frequencies and TDP Limitations

With the introduction of the *AVX base* frequency, the turbo mode concept has been extended significantly. While the turbo mode of previous processor generations has been the only setting that provided an opportunistic performance gain under certain thermal and power consumption conditions, now all frequencies above AVX base–including the nominal frequency–must be considered opportunistic. To evaluate the practical performance impact of this novelty we use the stress test FIRESTARTER [23] with active turbo mode and Hyper-Threading (2 threads per core). The advantages of FIRESTARTER for this test is that it reliably reaches the TDP limit, needs no thread synchronization, and provides a highly constant workload. We sample core and uncore cycles, instructions, and RAPL values for both processors once per second via LIKWID [22] on one core per processor (the other cores of the processor use the same frequency). We use 50 samples to calculate a median for uncore frequency, core frequency and instructions per second. The resulting performance characteristics are listed in Table IV.

The RAPL package consumption (not listed) indicates that both processors are limited by their TDP for all frequency settings at or above 2.2 GHz. As stated in Section III, processor 0 is less efficient than processor 1 in our test system. Therefore, frequencies and instructions per second (IPS) are higher on processor 1. When disabling turbo mode, core and uncore frequency increase slightly. When reducing the core frequencies setting from 2.4 to 2.3, 2.2, and 2.1 GHz, both processor slightly decrease their core frequency and use the available

Core frequency setting [GHz]	Turbo	2.5	2.4	2.3	2.2	2.1	2.0	1.9	1.8	1.7	1.6	1.5	1.4	1.3	1.2
Active processor uncore frequency [GHz]	3.0	2.2*	2.1	2.0	1.9	1.8	1.75	1.65	1.6	1.5	1.4	1.3	1.2	1.2	1.2
Passive processor uncore frequency [GHz]	2.9-3.0*	2.1*	2.0	1.9	1.8	1.7	1.65	1.55	1.5	1.4	1.2	1.2	1.2	1.2	1.2

TABLE III. Uncore frequencies in a single threaded no memory stalls scenario where the thread runs on processor 0 (*): 3.0 GHz if EPB is set to performance

Core frequency setting [GHz]	Turbo	2.5	2.4	2.3	2.2	2.1
Measured core frequency processor 0 [GHz]	2.30	2.31	2.31	2.27	2.19	2.09
Measured core frequency processor 1 [GHz]	2.32	2.35	2.35	2.28	2.18	2.09
Measured uncore frequency processor 0 [GHz]	2.33	2.34	2.34	2.46	2.80	3.00
Measured uncore frequency processor 1 [GHz]	2.35	2.37	2.37	2.58	2.86	3.00
Measured GIPS processor 0	3.55	3.56	3.56	3.58	3.58	3.51
Measured GIPS processor 1	3.58	3.60	3.60	3.62	3.59	3.52

TABLE IV. PERFORMANCE OF FIRESTARTER WITH DIFFERENT FREQUENCIES. WHEN REDUCING THE CORE FREQUENCY TO 2.3 GHZ, THE AVAILABLE HEADROOM IS USED TO INCREASE THE UNCORE FREQUENCIES WHICH ENHANCES THE PERFORMANCE IN TERMS OF IPS.

thermal budget to increase the uncore frequencies. For 2.1 GHz and slower, both processors use less than 120 W (according to RAPL) which prevents all throttling: the measured core frequency then equals the set core frequency, and the uncore frequency is at 3.0 GHz (max. turbo).

The performance (in terms of IPS) depends on core and uncore frequencies. A lower core frequency may be overcompensated by a higher uncore frequency (e.g., for the 2.3 and 2.4 GHz results). A performance gain of 1 % can be seen when reducing the frequency setting from turbo to 2.3 GHz. The different power characteristics of the processors can lead to performance imbalances as described in [24].

VI. P-State and C-State Transition Latencies

A. P-State Transition Latencies

The introduction of integrated voltage regulators, per core frequency domains, and improvements in the power control unit (PCU) have a direct influence on the latency and duration of ACPI processor state [25] transitions. To examine the new architecture, we use FTaLaT [26] for p-states and the tools developed by Schöne et al. [27] for c-states. We modified FTaLaT in the following ways:

- The original FTaLaT reads `scaling_cur_freq` from the Linux *cpufreq* subsystem to verify frequency settings. However, these readings are not reliable indicator for an actual frequency switch in hardware. We therefore add a verification by reading the `PERF_COUNT_HW_CPU_CYCLES` performance counters via the Linux *perf events* subsystem for a 20 µs busy-wait loop.
- We change the confidence level from 95% to 99 %.
- We parallelize FTaLaT to be able to capture transition times from two different cores in parallel.
- We recalculate the confidence intervals when the presumed performance level of the target frequency doesn't match the measured performance in a later stage.

Despite these changes, initial measurements still indicate a wide range and high fluctuation of transition latencies. We therefore take 1,000 measurements for a single pair of start and target frequencies. We chose 1.2 and 1.3 GHz, but other frequency pairs yield similar results.

Figure 3 depicts the results of four experiments with 1,000 results each as a histogram. With frequency change requested at random times, the resulting latency is evenly distributed between a minimum of 21 µs and a maximum of 524 µs. Requesting a frequency transition instantly after a frequency change has been detected leads to around 500 µs in the majority of the results. If we introduce a 400 µs delay after the last frequency change, the transition time is typically about 100 µs. If the delay is in the order of 500 µs, the transition latencies can be split into two different classes–some yield an immediate frequency change while others require over 500 µs.

These results indicate that frequency changes only occur in regular intervals of about 500 µs. The distance between the start

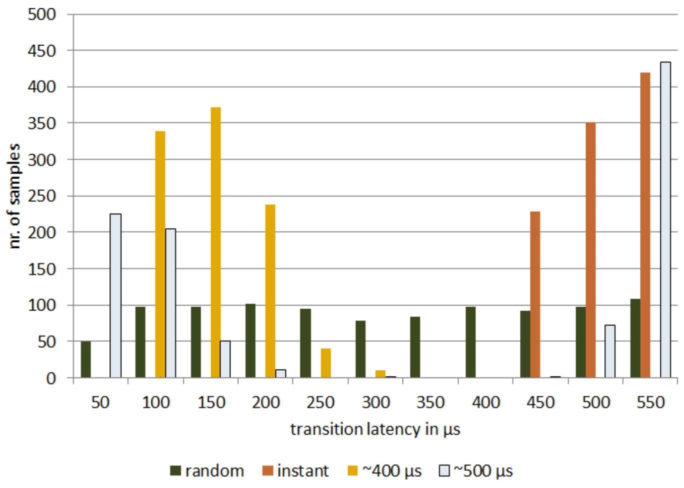


Fig. 3. Histogram of frequency transition latencies for switching between 1.2 and 1.3 GHz, depending on the time since the last frequency change.

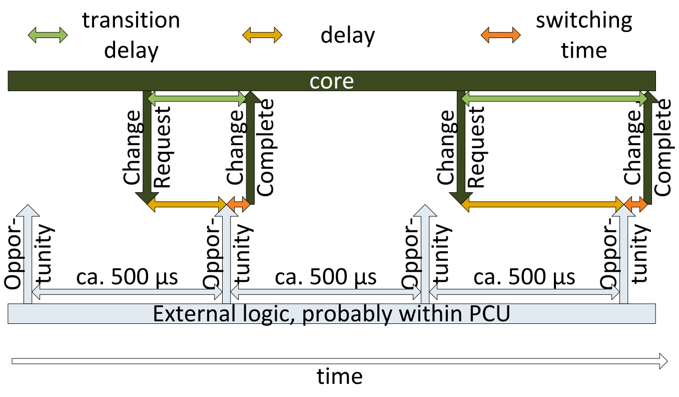


Fig. 4. Presumed mechanism for p-state changes on Intel Haswell-EP processors.

and the target frequency has negligible influence compared to the 500 µs delay. The assumed frequency changing mechanism is depicted in Figure 4.

In another experiment we measure two parallel threads changing the frequency of two different cores simultaneously. The results show that cores on the same processor change their frequency at the same time, while cores on different processors transition independently. This means the 500 µs update interval is driven by an external source, presumably the PCU, as it already handles turbo frequencies according to [17].

It should be noted that the ACPI tables report an estimated 10 µs for p-state transition latencies. This is not supported by the measurements and can hence be considered inapplicable. It is also important to note that on previous processors (including Haswell-HE), p-state transition request are always carried out immediately (requiring only the switching time).

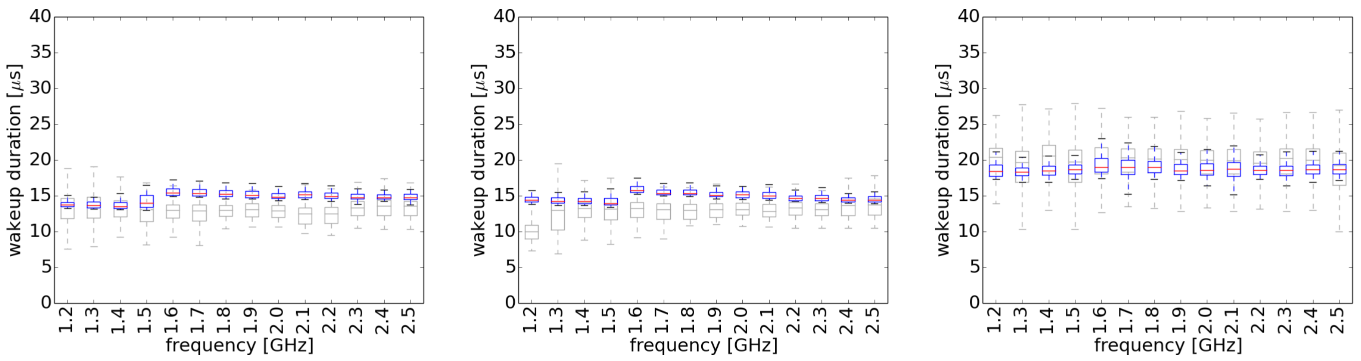
B. C-State Transition Latencies

C-state transition latencies reflect the time a processor core needs to return from an idle state to a functional mode (C0). In [27], we present results for several x86 processors, showing that transition latencies depend on the processor frequency, the relationship between the involved processor cores, and the idle state of the other cores on a processor.

In Figure 5, *local* measurements represent scenarios with both cores (waker and wakee) located on the same processor while for *remote* measurements they are located on different processors. In the *local* and *remote idle* scenario, one core wakes up another core in an idle system. In the *remote active* scenario, a third core is preventing the remote processor from going to a package c-state. The transition times for C3 states are mostly independent of the core frequencies. However, the latency is 1.5 µs higher when frequencies are greater than 1.5 GHz. As depicted in Figure 5c, the package C3 state increases the latency by another two to four microseconds.

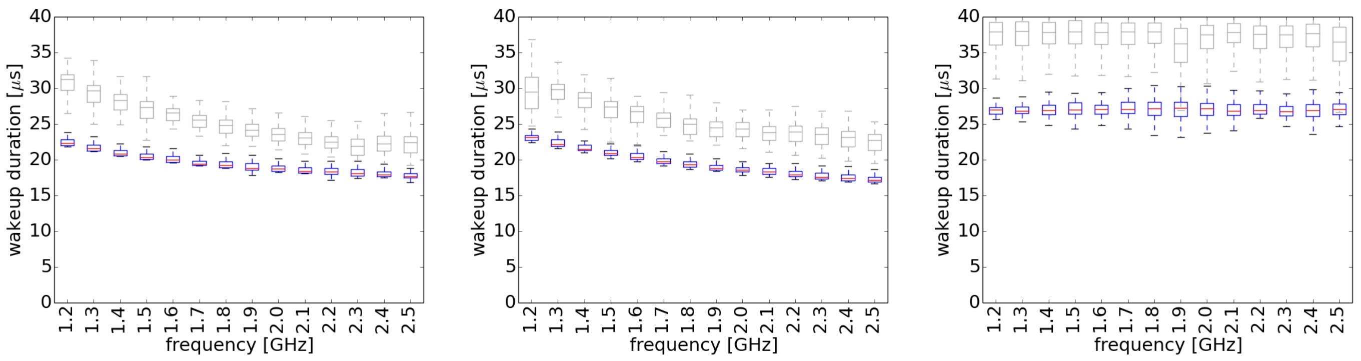
Transition times from C6 states depend strongly on the processor frequency, as depicted in Figure 6. Compared to C3, the latency is increased by 2 to 8 µs in the local C6 case. However, the package C6 state increases latency by 8 µs, compared to package C3. Transitions from C1 (not depicted) are below 1.6 µs for local measurement and up to 2.1 µs for remote measurement (at 1.2 GHz core frequency). Even when assuming that caches are flushed and re-read from DRAM when changing c-states, the c-state transitions happen faster than p-state (core frequency) transitions.

It is interesting to note that the measured transition times for C3 and C6 are lower than the definitions in the respective ACPI tables (33 and 133 µs). These tables are used by the operating system to decide which c-state to use for an assumed idle interval. The discrepancy between the measured and defined latencies underlines the need for an interface to change these tables at runtime.



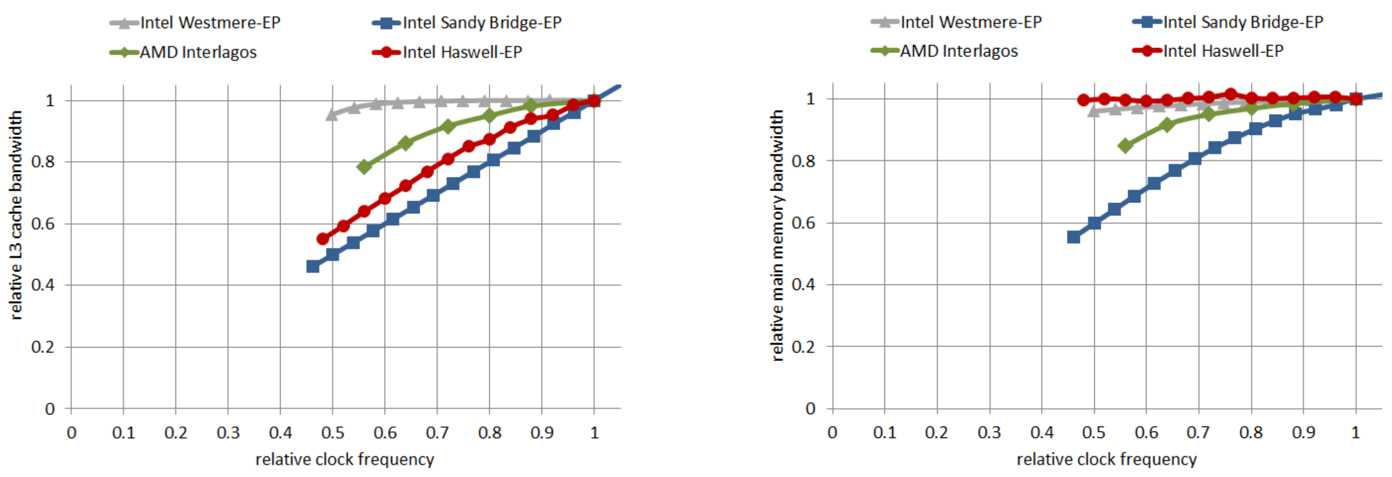
(a) local results (C3) (b) remote active results (remote C3) (c) remote idle results (package C3)

Fig. 5. Idle transition latencies for different C3 scenarios in comparison to Sandy Bridge EP processors (grey)



(a) local results (C6) (b) remote active results (remote C6) (c) remote idle results (package C6)

Fig. 6. Idle transition latencies for different C6 scenarios in comparison to Sandy Bridge EP processors (grey)



(a) relative L3 cache read bandwidth (b) relative main memory read bandwidth

Fig. 7. Scaling of shared main memory and L3 cache bandwidth at maximum thread concurrency, normalized to the bandwidth at base frequency.

VII. Memory Bandwidth at Reduced Clock Speeds and Concurrency

In this section we examine the dependency of the L3 cache and local DRAM read bandwidth of our Haswell-EP test system with regards to frequency and concurrency. Due to the erratic behavior of uncore frequencies (see Section II-D), performance variations cannot be avoided. We arbitrarily choose to measure on processor 1 of our test system. It performs equally or better than processor 0 which is idle during the measurements. The bandwidth benchmarks from [28] have been extended for the new architecture. We consecutively access 17 MB of data for the L3-cache and 350 MB of data for the DRAM measurement. Hardware prefetchers are enabled.

Figure 7 compares the Haswell-EP results with previous systems [29]. On the Haswell-EP architecture, DRAM performance at maximal concurrency does not depend on the core frequency (see Figure 7b). Thus, the core frequency can be reduced to save energy in memory-bound applications. The behavior of the Westmere-EP generation with its constant uncore frequency was similar, while the generation in between behaves completely different: On Sandy Bridge-EP, the uncore frequency reflects the core frequency, making DRAM bandwidth highly dependent on core frequency. Furthermore, the memory bandwidth on Sandy Bridge-EP depends on the package c-state of the other socket [29]. This is no longer the case on Haswell-EP, presumably due to the interlocked uncore frequecies (see Table III). Figure 7a shows that the L3 bandwidth of Haswell-EP strongly correlates with the core frequency. This is surprising since other processors with dedicated uncore/northbridge frequencies are less influenced by lower core frequencies in terms of L3 bandwidth.

The L3 read bandwidth strongly depends on both concurrency and frequency as depicted in Figure 8. The scaling is not exactly linear with these factors, compared to the linear scaling on Sandy Bridge processors [29]. The L3 read bandwidth scales slightly better than linear with the number of cores at low levels of concurrency and approximately linearly otherwise. In contrast, it scales linearly with frequency for lower frequencies but flattens at higher frequency levels without converging to a specific plateau. The main memory read bandwidth saturates at 8 cores (see Figure 8) and becomes independent of the core frequency if ten cores are active. This allows DCT and DVFS optimizations for memory bound codes. Using multiple threads per core only is beneficial for low-concurrency scenarios.

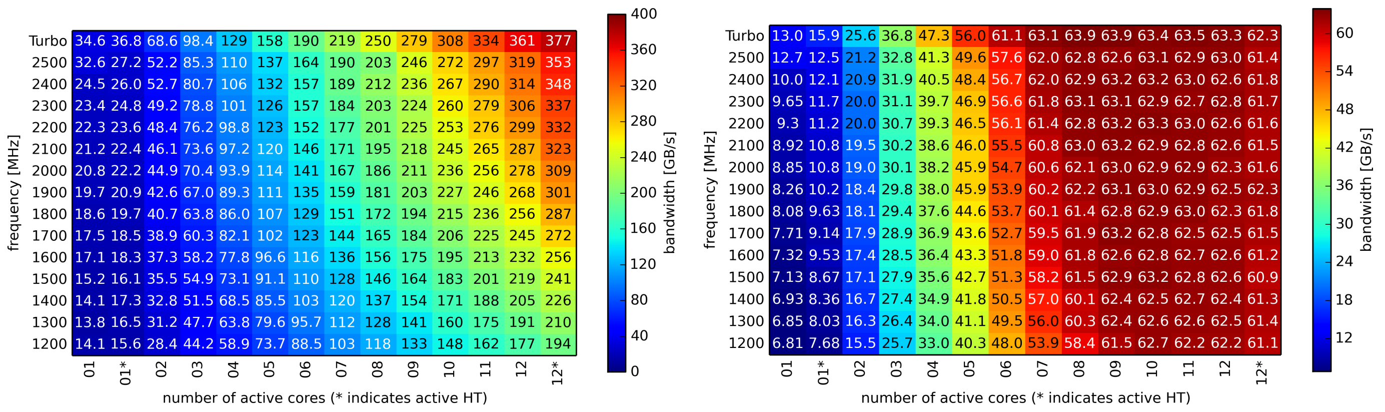


Fig. 8. Read bandwidths from Level 3 cache (left) and DRAM (right) depending on concurrency and frequency

VIII. Maximizing Power Consumption

Stress-tests that maximize the power consumption of a compute node are an important tool for infrastructure tests for cooling and power distribution as well as experiments regarding architectural and power management features. *FIRESTARTER* is specifically designed to generate very high power consumption, equaling or outperforming other commonly used tools, while causing extremely constant power consumption patterns and without requiring any manual configuration. Haswell-EP is supported since version 1.2.

The CPU power consumption is influenced by the utilization of the execution units as well as data transfers [30]. Furthermore, the decoders need to be utilized as much as possible. Therefore, the stresstest loop has to be larger than the micro-op cache but small enough for the L1 instruction cache. The code is structured in groups of four instructions (*I1*, *I2*, *I3*, *I4*) that fit into the 16 byte fetch window. Ideally, one such group is executed per cycle. There are seperate instruction groups for each level in the memory hierarchy (*reg*, *L1*, *L2*, *L3*, and *DRAM*). *I1* is a packed double FMA instruction working on registers (*reg*, *mem*) or a store to the respective cache level (*L1*, *L2*, *L3*). *I2* is an FMA instruction which can be combined with a load operation (*L1*, *L2*, *L3*, and *mem*). *I3* performs a right shift. *I4* is a xor (*reg*) or an add instruction (*L1*, *L2*, *L3*, and *mem*) that increments a pointer. The groups are executed at the ratio of 27.8 % *reg*, 62.7 % *L1*, 7.1 % *L2*, 0.8 % *L3*, and 1.6 % *mem*. The resulting sequence combines a high ratio of floating point operations with frequent loads and stores. However, memory accesses also stall the execution and thereby limit the IPC. We achieve 3.1 executed instructions per cycle with Hyper-Threading enabled and 2.8 without.

In Table V we compare the power consumption of FIRESTARTER 1.2 against LINPACK (as provided with the Intel Composer XE 2015.1.133, problem size 80,000) and mprime (version 28.5, runtime 1000 s). Our experiments cover different scenarios, varying the settings for turbo mode, Hyper-Threading, and the EPB. We extract the 1 minute interval with the highest average power consumption for each test result. This favors LINPACK and mprime, as their power consumption is not as constant over time. We also measure the actual core frequency during the 1 minute power measurement interval in order to evaluate frequency reductions in the presence of AVX operations.

LINPACK causes a notably lower power consumption than the other two benchmarks. It also runs with the lowest frequency, meaning that some kind of frequency reduction due to TDP restrictions appears to be active. The power consumption of FIRESTARTER and mprime is almost on par. However, FIRESTARTER uses a lower frequency and causes a much more static power consumption than mprime. EPB, turbo mode, and Hyper-Threading settings (not depicted) have very little impact on the core frequency and the power consumption.

Selected frequency		**2500 MHz**			**Turbo**	
EPB	power	bal	perf	power	bal	perf
						power in W
FIRESTARTER	561.0	560.4	560.1	559.8	560.0	560.0
LINPACK	548.6	547.9	547.7	547.4	547.6	548.1
mprime	561.3	558.6	560.6	560.2	560.1	560.5
						measured frequency in GHz
FIRESTARTER	2.46	2.45	2.44	2.44	2.44	2.44
LINPACK	2.28	2.28	2.28	2.27	2.28	2.27
mprime	2.45	2.49	2.59	2.61	2.60	2.62

TABLE V. Average power consumption and measured core frequency over 1 minute (Hyper-Threading not active)

IX. Conclusions

The Intel Haswell processor architecture incorporates significant enhancements compared to its predecessors. Most notable for the energy efficiency community are the voltage regulators that move from the mainboard to the CPU die. We have discussed a number of relevant consequences of this development, such as the new per-core p-states that allow a much finer-grained control of voltages and frequencies. Our experiments disclose that p-state transitions are now handled differently, resulting in significantly increased transition latencies compared to previous processors. In contrast, transition latencies from deep c-states have slightly improved. Depending on the workload, this can indicate a reduced effectiveness for DVFS on Haswell-EP in very dynamic scenarios, while DCT becomes a more viable approach for energy efficiency optimizations.

Besides per-core frequencies, the uncore frequency settings also provide unprecedented flexibility which is unfortunately only controlled by hardware. A major implication of this is that DRAM performance at reduced clock speeds has improved significantly compared to the Sandy Bridge-EP generation and is back at the level of Westmere-EP, thereby making well-known efficiency optimizations for memory-bound workloads viable again.

Another major change affects the RAPL energy consumption values that were based on a modeling approach in previous processor generations and that reflect actual measurements in the Haswell architecture, resulting in new level of accuracy and thereby tremendously increasing the value of this interface.

In this paper we also present the improvements to the processor stress test utility FIRESTARTER that were required for the Haswell processor generation along with first measurement results in comparison with LINPACK and mprime.

Probably the most momentous novelty comes as a consequence of two aspects: (1) the processor cannot continuously operate at full speed due to TDP limitations, and (2) the RAPL mechanism to enforce the TDP limitation has changed from a modeling to a measurement approach. The transparent controls that set core and uncore frequencies according to TDP limitations complicate performance analysis tasks significantly. The significance of this change is tremendous: Starting with the Haswell-EP processor generation, the previously observed power consumption variations will likely be replaced by uncontrollable and unpredictable performance variations, with a potentially large impact on the performance of tightly synchronized, large scale parallel applications. The newly introduced AVX frequency makes every clock speed above this level–including nominal frequency–opportunistic and unreliable. The performance of the uncore can change depending on the previous memory access patterns, making performance significantly less predictable, including increased performance at reduced clock speeds for certain workloads.

Acknowledgment This work has been funded by the Bundesministerium für Bildung und Forschung via the research project Score-E (BMBF 01IH13001) and the Deutsche Forschungsgemeinschaft (DFG) via the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing" (HAEC, SFB 921/1 2011). The authors would like to thank Thomas Röhl and Mario Bielert for their support.

References

[1] E. Burton, G. Schrom, F. Paillet, J. Douglas, W. Lambert, K. Radhakrishnan, and M. Hill, "FIVR – Fully integrated voltage regulators on 4th generation Intel Core SoCs," in *Applied Power Electronics Conference and Exposition (APEC), 2014 Twenty-Ninth Annual IEEE*, March 2014, pp. 432–439.

[2] *Intel 64 and IA-32 Architectures Optimization Reference Manual*, Intel, Sep 2014, order Number: 248966-030. [Online]: http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf

[3] A. Fog, "The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers," online, Copenhagen University College of Engineering, August 2014. [Online]: http://agner.org/optimize/microarchitecture.pdf

[4] M. Yuffe, M. Mehalel, E. Knoll, J. Shor, T. Kurts, E. Altshuler, E. Fayneh, K. Luria, and M. Zelikson, "A Fully Integrated Multi-CPU, Processor Graphics, and Memory Controller 32-nm Processor," *Solid-State Circuits, IEEE Journal of*, vol. 47, no. 1, pp. 194–205, 1 2012.

[5] N. Kurd, M. Chowdhury, E. Burton, T. Thomas, C. Mozak, B. Boswell, M. Lal, A. Deval, J. Douglas, M. Elassal, A. Nalamalpu, T. Wilson, M. Merten, S. Chennupaty, W. Gomes, and R. Kumar, "Haswell: A family of IA 22nm processors," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, 2 2014, pp. 112–113.

[6] *Intel Xeon Processor E5 v3 Family Uncore Performance Monitoring Reference Manual*, Intel, 9 2014. [Online]: http://www.intel.com/content/dam/www/public/us/en/zip/xeon-e5-v3-uncore-performance-monitoring.zip

[7] C. Park, R. Badeau, L. Biro, J. Chang, T. Singh, J. Vash, B. Wang, and T. Wang, "A 1.2 TB/s on-chip ring interconnect for 45nm 8-core enterprise Xeon® processor." in *ISSCC*, 2010, pp. 180–181.

[8] *Intel Xeon Processor E5-2600 Product Family Uncore Performance Monitoring Guide*, Intel, 3 2012. [Online]: http://www.intel.com/content/dam/www/public/us/en/documents/design-guides/xeon-e5-2600-uncore-guide.pdf

[9] S. Rusu, H. Muljono, D. Ayers, S. Tam, W. Chen, A. Martin, S. Li, S. Vora, R. Varada, and E. Wang, "5.4 Ivytown: A 22nm 15-core enterprise Xeon processor family," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb 2014, pp. 102–103.

[10] *Intel® Xeon® Processor E5 v3 Product Family - Processor Specification Update*, Intel, 1 2015. [Online]: http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-e5-v3-spec-update.pdf

[11] *Intel® Xeon® Processor E5-1600 and E5-2600 v3 Product Families, Volume 1 of 2, Electrical Datasheet*, Intel Corporation, Sep 2014.

[12] K. Sistla, J. Shrall, S. Gunther, E. Rotem, A. Naveh, E. Weissmann, A. Aggarwal, M. Rowland, A. Varma, I. Steiner *et al.*, "User Level Control Of Power Management Policies," Aug. 9 2012, uS Patent App. 13/326,586. [Online]: http://www.google.com/patents/US20120204042

[13] *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A, 3B, and 3C: System Programming Guide*, Intel, Sep 2014.

[14] M. Bhandaru, E. Dehaemer, S. Ho, S. Bobholz, and C. POIRIER, "Method and apparatus for per core performance states," Sep. 19 2013, patent App. PCT/US2012/028,923. [Online]: http://www.google.com/patents/WO2013137865A1?cl=en

[15] M. Bhandaru, A. Varma, J. Vash, M. Wong-Chan, E. Dehaemer, S. POIRIER, and S. Bobholz, "Dynamically controlling interconnect frequency in a processor," Sep. 19 2013, patent App. PCT/US2012/028,902. [Online]: http://www.google.com/patents/WO2013137862A1?cl=en

[16] G. Lento, "Optimizing Performance with Intel Advanced Vector Extensions," online, September 2014. [Online]: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/performance-xeon-e5-v3-advanced-vector-extensions-paper.pdf

[17] M. Bhandaru and E. Dehaemer, "Providing energy efficient turbo operation of a processor," Sep. 19 2013, patent App. PCT/US2012/028,865. [Online]: http://www.google.com/patents/WO2013137859A1?cl=en

[18] C. Gianos, "Intel Xeon Processor E5-2600 v3 Product Family Architectural Overview," in *IHPCC*, 2014. [Online]: http://ihpcc2014.com/pdf/IntelR%20XeonR%20Processor%20E5-2600%20v3%20Overview%20for%20SC14.pdf

[19] *4 Channel Power Meter LMG450, User manual*, ZES ZIMMER Electronic Systems, 2009.

[20] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, 2013. [Online]: http://dx.doi.org/10.1109/ISPASS.2013.6557170

[21] *Intel® Xeon® Processor E5-1600 and E5-2600 v3 Product Families, Volume 2 of 2, Registers Datasheet*, Intel Corporation, Sep 2014.

[22] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments," in *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ser. ICPPW '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 207–216. [Online]: http://dx.doi.org/10.1109/ICPPW.2010.38

[23] D. Hackenberg, R. Oldenburg, D. Molka, and R. Schöne, "Introducing FIRESTARTER: A processor stress test utility," in *International Green Computing Conference (IGCC)*, 2013. [Online]: http://dx.doi.org/10.1109/IGCC.2013.6604507

[24] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, "Beyond DVFS: A first look at performance under a hardware-enforced power bound," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 947–953.

[25] "Advanced Configuration and Power Interface (ACPI) specification, revision 5.0 Errata A," Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation, Nov. 2013. [Online]: http://www.acpi.info/

[26] A. Mazouz, A. Laurent, B. Pradelle, and W. Jalby, "Evaluation of CPU frequency transition latency," *Computer Science - Research and Development*, 2014. [Online]: http://dx.doi.org/10.1007/s00450-013-0240-x

[27] R. Schöne, D. Molka, and M. Werner, "Wake-up latencies for processor idle states on current x86 processors," *Computer Science - Research and Development*, 2014. [Online]: http://dx.doi.org/10.1007/s00450-014-0270-z

[28] D. Molka, D. Hackenberg, R. Schöne, and M. S. Müller, "Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System," in *PACT '09: Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 261–270.

[29] R. Schöne, D. Hackenberg, and D. Molka, "Memory performance at reduced CPU clock speeds: an analysis of current x86_64 processors," in *Proceedings of the 2012 USENIX conference on Power-Aware Computing and Systems*, ser. HotPower'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 9–9. [Online]: http://dl.acm.org/citation.cfm?id=2387869.2387878

[30] D. Molka, D. Hackenberg, R. Schöne, and M. S. Müller, "Characterizing the Energy Consumption of Data Transfers and Arithmetic Operations on x86-64 Processors," in *Proceedings of the 1st International Green Computing Conference*. IEEE, 2010, pp. 123–133.