A

Low Power High Efficiency Video Decoding using General Purpose Processors

CHI CHING CHI, Technishe Universität Berlin
MAURICIO ALVAREZ-MESA, Technishe Universität Berlin
BEN JUURLINK, Technishe Universität Berlin

In this paper we investigate how code optimization techniques and low power states of general purpose processors improve the power efficiency of HEVC decoding. The power and performance efficiency of the use of SIMD instructions, multicore architectures, and low power active and idle states are analyzed in detail for offline video decoding. In addition, the power efficiency of techniques such as "race to idle" and "exploiting slack" with DVFS are evaluated for real-time video decoding. Results show that "exploiting slack" is more power efficient than "race to idle" for all evaluated platforms representing smartphone, tablet, laptop and desktop computing systems.

Categories and Subject Descriptors: B [**Hardware**]: Power estimation and optimization; C.4 [**Computer Systems Organization**]: Performance of systems; C.1.4 [**Computer systems organization**]: Parallel architectures; D.4.8 [**Operating Systems**]: Performance

General Terms: HEVC, H.265, parallel, multicore, SIMD, DVFS, low power, UHD

Additional Key Words and Phrases: Video decoding, parallel processing, low power computing

ACM Reference Format:
Chi Ching Chi, Mauricio Alvarez-Mesa and Ben Juurlink. Low Power High Efficiency Video Decoding using General Purpose Processors *ACM Trans. Architec. Code Optim.* V, N, Article A (YYYY), 25 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Recent demands to support higher video resolutions such as 4k or Ultra HD (UHD) in consumer video devices have driven the video codec development towards higher compression rates. To meet these demands the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T and ISO/IEC has developed a new video coding standard, referred to as HEVC, aiming to reduce the bitrate of the H.264/AVC video codec by another 50% [Sullivan et al. 2012].

It has been demonstrated that by using SIMD and multithreading it is possible to achieve very high performance for HEVC decoding on recent computer architectures [Bossen et al. 2012; Chi et al. 2014; Bross et al. 2013]. As a result, much higher than real-time frame rates can be achieved even for UHD resolution videos on commodity desktop and mobile processors. While achieving high performance for video decoding can be beneficial for some applications, for instance, in offline transcoding and video analytics applications, video decoding is mostly utilized as a real-time ap-

This work is supported by the European Community's Seventh Framework Programme [FP7/2007-2013] under the LPGPU Project (www.lpgpu.org), grant agreement No. 288653.
Author's addresses: Technische Universität Berlin. Institut für Technische Informatik und Mikroelektronik (TIME). Sekretariat EN 12. Einsteinufer 17. D-10587 Berlin. Germany.

DOI: http://dx.doi.org/10.1145/0000000.0000000

plication in video playback. In video playback a steady frame decoding rate, measured in frames per second, is required as compared to decoding the full sequence as fast as possible in offline decoding scenarios. If the processor have more performance than required for real-time operation the unused capacity can be employed to improve power efficiency.

In the last years power consumption has become one of the main design considerations of computing platforms. To address this concern processor architectures and offchip memory have incorporated many low power states, which allow the processor to consume less energy when idle or at lower activity levels. On recent processors this has resulted primarily in so called P-States and C-states, which control the power consumption at lower processor activity. With these power states achieving the highest performance also improves the power efficiency. For real-time video decoding, increased performance allows the processor to go longer and more frequent in lower power C-states, resulting in overall less power consumption. Alternatively, lower power can be achieved when more than real-time performance can be reached at the nominal frequency by reducing the clock frequency, which allows the processor to run on a more efficient voltage-frequency operating point (P-state).

In research these strategies often called “race to idle” [Steigerwald 2011] and “exploiting slack” with dynamic frequency voltage scaling (DVFS) [Chandrakasan et al. 1992; Kaxiras and Martonosi 2008]. Several works have shown that DVFS benefits power efficiency [Simunic et al. 2001; Choi et al. 2002; Gu et al. 2006; Liang et al. 2013] for multimedia applications, while others claim that due to diminishing benefits in future process technologies DVFS will disappear in favor of idle states [Le Sueur and Heiser 2010; 2011] or that only specialized cores will deliver the required power efficiency [Esmaeilzadeh et al. 2012; Hardavellas 2012]. Clearly there are multiple ways towards higher power efficiency, but the applicability of the approaches depends on the particular platform and application.

The goal of the paper is to give more insight on which strategy achieves the best power efficiency for HEVC video decoding, both in real-time and offline decoding modes. This is performed through the following contributions:

— The power efficiency impact of code optimization techniques, such as SIMD and multithreading, and the use of low power states is investigated for offline and real-time HEVC decoding.
— The the two strategies “race to idle” and “exploiting slack” are compared for real-time video decoding.
— The effectiveness of the current software stack controlling the processor low power modes is investigated.
— The evaluation is performed on systems ranging from ultra mobile platforms to PCs, and focuses on the power consumption of the processor (cores and uncore) and offchip memory.

This paper is organized as follows: Section 2 provides an overview of architectural low power states available on modern processors, and present a description of how these low power states are exploited by the operating system. Section 3 presents related work in low power software video decoding. Section 4 discusses the opportunities for lower-power video decoding for both offline and real-time scenarios. In Section 5 the experimental setup is presented, detailing the hardware platforms, the software stack, and the test sequences utilized in the experiments. Section 6 presents the power characteristics of the platforms under idle and load, followed by the power and power efficiency results in Section 7 for both offline and realtime scenarios. Finally, conclusions are drawn in Section 8.

2. LOW POWER MODES AND OS SUPPORT

In this section we perform a review of the low power modes available in recent microprocessors, and also present a description of the current operating system support for using these low power modes.

2.1. Architectural Low Power States

Modern GPPs incorporate various low power states. These states can generally be classified in two kinds, states that reduce power consumption when the processor is executing instructions, and states that reduce the power when the processor components are idle. Techniques such as DVFS and asymmetric processing belong mainly to the former kind, while clock gating and power gating belong to the latter kind. Each of these techniques influences the total power consumption of the processor P_{cpu} which at a high level is given as

$$P_{cpu} = \underbrace{\alpha C V_{dd}^2 f}_{P_{dynamic}} + \underbrace{I_{sc} V_{dd}}_{P_{shortcircuit}} + \underbrace{I_{leak} V_{dd}}_{P_{leakage}}, \quad (1)$$

where α represents the switching activity, C is the circuit capacitance, V_{dd} is the supply voltage, f is the operating frequency, I_{sc} is the short circuit current, and I_{leak} is the leakage current [Chandrakasan et al. 1992; Li et al. 2013].

2.1.1. DVFS. A processor supporting DVFS has multiple voltage-frequency operating points and can dynamically change the operating point. The technique exploits the fact that the supply voltage can be lowered at lower frequencies [Burd et al. 2000]. Lowering the supply voltage reduces both the static and dynamic power, and in addition the power dissapation is proportional to the square of the supply voltage. Because the frequency is lowered a tradeoff with performance has to be made, but in many realtime applications, including video playback, rarely the maximum performance is required at all times. Deciding which voltage-frequency point to use is mostly controlled by the operating system. On recent Intel processors the control of the higher frequencies have moved to hardware under the feature called *Turboboost* [Intel 2008]. These processors enter the *Turboboost* frequencies when when the thermal and power threshold are not exceeded.

2.1.2. Clock Gating. Clock gating [Kathuria et al. 2011] is also widely used in modern processors. This technique lowers the dynamic power consumption when the processor is idle. Clock gating stops the propagation of the clock signals to the processor components, which essentially reduces the dynamic power ($P_{dynamic}$) to zero. This technique can be implemented by placing a latch and an AND-gate between the *clk_in* and the *clk_signal*. A *clk_enable* signal can then be used to propagate or stop the clocks. Clock gating can be applied on many levels of the processor, ranging from functional units to complete cores. In its basic form clock gating has a relatively short transition latency, as the clock generation is still active and simply stopped from reaching the components. In an extended form also the *phase-locked-loop* (PLL) that generates the clock is turned off to reduce power consumption further. The drawback is the longer wake-up latency, since the PLL must restabilize to the correct frequency.

2.1.3. Power Gating. Power gating reduces the power consumption even more than clock gating and also reduces the static power consumption to virtually zero. This is accomplished by placing a special low leakage power gate transistor between the supply voltage and the processor components. In a power gated state the core is turned off and consumes near zero power. Power gating is used in state-of-the-art processors, and with the more significant leakage power of smaller process technology, it has become

Table I. C-states for Intel Haswell processors. States are additive, meaning higher C-states contain the previous ones in their behavior

Core		Package	
C-State	Description	C-State	Description
C0	Core is execution code	C0	One or more cores are executing code
C1	Core is halted most clocks are stopped	C3	L3 may be flushed and power gated, memory in self-refresh, some uncore clocks stopped, some voltages reduced
C1E	Voltage reduced to Pn	C6	All uncore clocks stopped
C3	Core L1/L2 flushed to L3, PLL stopped	C7	L3 flushed and power gated, more uncore voltages reduced
C6-C10	Core state saved and power gated	C8	Most uncore power gated
		C9	FIVR in low power state
		C10	FIVR turned off

essential for achieving low idle power without sacrificing clock speed. The transition latency of power gating is larger than clock gating, because in addition the architectural state must be saved and restored. Also power gating must be performed gradually to control in-rush and out-rush currents [Kosonocky 2011].

2.1.4. Asymetric Cores. Using asymmetric cores, also known as single ISA heterogeneous multicore [Kumar et al. 2004], has shown good potential for improving power efficiency when complex cores cannot be utilized fully. The first actual implementation of asymmetric cores to save power in GPPs was introduced in 2011 by ARM under their *big.Little* [Greenhalgh 2011] design. In this approach several high performance cores are coupled with lower performance low power cores on the same die. Different from the original idea these approaches do not target micro-architectural slack, but are implemented to extend the effectiveness of DVFS at lower performance requirements.

DVFS is effective in improving energy efficiency until it reaches a voltage frequency point that is close to the threshold voltage, afterwhich the energy efficiency decreases again [Jain et al. 2012]. To improve energy efficiency beyond this point a processor that has been optimized for a lower performance level must be used instead. A drawback of asymetric cores is that operating systems must implement core migration and apply a cost model for each different type of processor.

2.2. Operating System Power Management

The usage of the architectural power states is mainly decided by the operating system. In this section we provide an overview of the most important interfaces and drivers controlling these states for operating systems using the Linux kernel.

2.2.1. ACPI. The Advanced Configuration and Power Interface (ACPI) standard [Brown 2005] specifies among others the power states for the device on several levels. In the *global states* the state of the device is specified such as working, hibernating, and standby. In the working state the processor can be in different power states referred to as C-states. In C0 the processor is executing code, C1 the processor is halted, C2 the clocks are stopped, and in C3 the cache is additionally flushed and does not have to respond to snoop requests. Higher C-states save more power but have a higher transition time.

In recent Intel processors more C-states are defined than in ACPI to further reduce power. Additionally, Intel also defines C-states for the package, which reduce the power of the components in the *uncore* (L3 cache,DRAM controller, system agent, etc). These package states can be entered if all processing cores are in the same or higher C-states. A short overview of the core and package C-states is listed in Table I

In C0, the active state, the cores can be in different DVFS points which in ACPI is referred to as P-states. In P0 the core is in the highest performance state, while in Pn the core is in the lowest active performance state with the lowest available frequency and voltage. Modern processors define many DVFS points, often one is available for each frequency multiplier.

In recent Intel processors also a feature called *Turboboost* is available. This feature allows, provided the core is P0 state, to dynamically overclock itself when there is power and thermal headroom. For big.Little ARM devices the low performance P-states also act as an interface for switching to the Little cores. The P-states of the Little cores are mapped to the low performance P-states, and when such a P-state is selected a switch from the big cores to the Little cores is made or vice versa.

Both C-states and P-states have to be controlled by the operating system. A so-called *governor* decides which C-state or P-state to enter. In the next sections an overview is given on how this is performed in Linux (3.11 and higher).

2.2.2. Cpuidle Governors. The *cpuidle* subsystem [Pallipadi et al. 2007] controls the usage of the available C-states. Of the two available idle governors, *menu* and *ladder*, the former is more commonly used. To determine which C-state to enter a balance must be struck with transition latency, energy cost, and wakeup latency. The *menu* governor predicts the length of the next idle based on previous idle periods and based on this prediction selects the best C-state to transit to.

For each processor the characteristics of the C-states are different. For this reason vendor specific cpuidle drivers are contributed to Linux. In addition to *acpi_idle*, for instance *intel_idle* and *exynos_idle* are available for Intel and Samsung ARM processors, respectively. For intel processors the C-state is selected if the idle governor predicts an idle period that is roughly $3\times$ longer than the wakeup latency. The processor itself can still demote and promote these requests based on its own internal control algorithms.

2.2.3. DVFS Governors. The *cpufreq* subsystem [Pallipadi and Starikovskiy 2006] controls the usage of the available P-states. Depending on the driver, several governors are available of which the dynamic ones commonly base their decision on the observed processor load.

Using the *acpi_freq* driver, the *powersave*, *conservative*, *ondemand*, *performance*, and *userspace* governors are available. Both *conservative* and *ondemand* are dynamic load-based governors, but *ondemand* responds to load changes more aggressively. The *exynos_cpufreq* driver provides in addition the *interactive* governor. The *interactive* governor acts like *ondemand*, and in addition also switches to the highest performance state when an interrupt from a user interface sensor is received, making devices more responsive.

Recently, Intel provides the *intel_pstate* driver, which only exposes the *powersave* and *performance* governors to generalize the P-state control. The *intel_pstate powersave* governor acts like the *acpi_freq ondemand* governor, and allows the user to map the usable frequency range to a relative performance interface. Based on the load, a PID control algorithm is used to select the target performance level.

3. RELATED WORK

Several techniques have been proposed to use DVFS for reducing energy in video decoding applications [Choi et al. 2002; Ma et al. 2011; Akyol and van der Schaar 2008; Mesarina and Turner 2003]. Most of them are based on a model that predicts the complexity of a decoding unit (usually a frame) and uses the prediction to adjust the frequency and voltage of the processor in order to meet the frame deadlines. In [Choi et al. 2002] the authors present a DVFS algorithm for MPEG-2 decoding using a prediction model based on frame history. The technique was implemented on an StrongARM pro-

cessor and allows to save more than 50% CPU energy. In [Ma et al. 2011] authors proposed a similar complexity model for H.264 decoding and used it to predict and adapt the processor frequency. The system was implemented for Intel mobile X86 and TI ARM Cortex A8 processors resulting in power savings of 73% and 55% respectively, when compared to *ondemand* governors.

Some other works have proposed a combination of DVFS and dynamic idle states exploitation for video decoding [Simunic et al. 2001; Akyol and van der Schaar 2008]. In this case, a complexity prediction model and processor energy model is extended to take into account also the power at idle states. In [Akyol and van der Schaar 2008] authors present a DVFS technique that also includes a video decoding buffer to reduce the total processing requirements, and takes into account the frame structure of the encoded video for making task scheduling and DVFS decisions.

Compared to our work most of the previous studies only take into account power consumed by the CPU, excluding memory and uncore modules, which can have a big impact in the total power consumption. In a similar way, most of the related works only take into account dynamic power, ignoring the static power component which has become more relevant in recent processor technologies. The main difference with the previous work is that in this paper we take an analysis of power consumption of a state-of-the-art video codec (HEVC) using on several recent processors and includes the CPU (or core) power as well as the memory and "uncore" power, giving a more complete picture of the total power consumption.

4. LOW POWER HEVC DECODING

In this section we describe the power saving techniques that can be applied to two HEVC decoding application scenarios. The first one is *offline decoding* where the decoding is unconstrained by timing, and the second one is *real-time decoding* where frames must be delivered at a constant rate. In both use cases increased performance offers more opportunities to the power saving techniques.

Performance improvements through general code optimizations and the use of SIMD instructions are likely to increase the power efficiency, because the performance improvements can be large and these code optimizations typically do not significantly contribute to the dynamic power consumption of the core. The effect of multithreading and using the low power states of a platform on the power efficiency are, however, less obvious and different for the two application scenarios. To maximize the power efficiency for offline decoding we only have to consider the power efficiency of the active states as their is no idle time. In offline decoding the processor can be fully loaded to decode frames as fast as possible, and the highest power efficiency can be achieved by using the most efficient voltage-frequency point and processor core count of a given processor/system.

Maximizing the power efficiency for real-time performance is more complex as the idle power consumption has to be considered in addition to the active power consumption. On a modern processor the idle power depends heavily on which C-state it resides in. A deeper C-state can be entered when a longer idle period is predicted, saving more power. To reduce the power consumption for real-time decoding one can either finish the decoding sooner using more cores and/or higher frequency in order to idle longer and deeper ("race to idle"), or try to exploit the idle time by running at a lower frequency ("exploiting slack").

Additionally, for real-time performance the frames of a video sequence have to be delivered at a constant rate. The complexity of decoding a frame, however, varies heavily from frame to frame. To compress a video sequence more efficiently not all coded pictures in a video sequence contain the same number of bits. For instance, frames that are used as reference frames are coded with higher quality and more bits to serve as

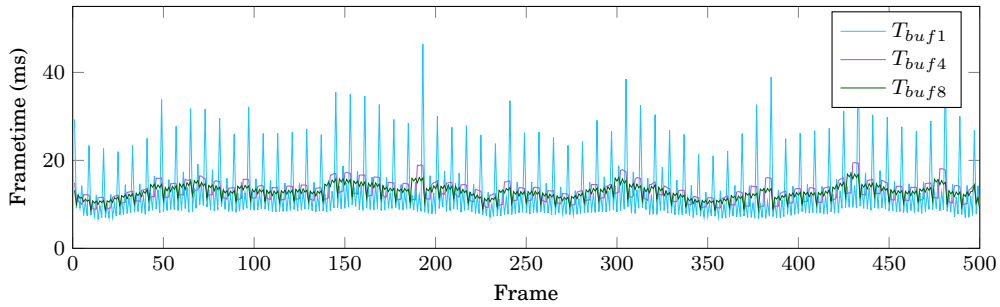


Fig. 1. Frame times of a 7Mbps 1080p50 sequence. Without playback buffers even high performance processors cannot guarantee real-time playback.

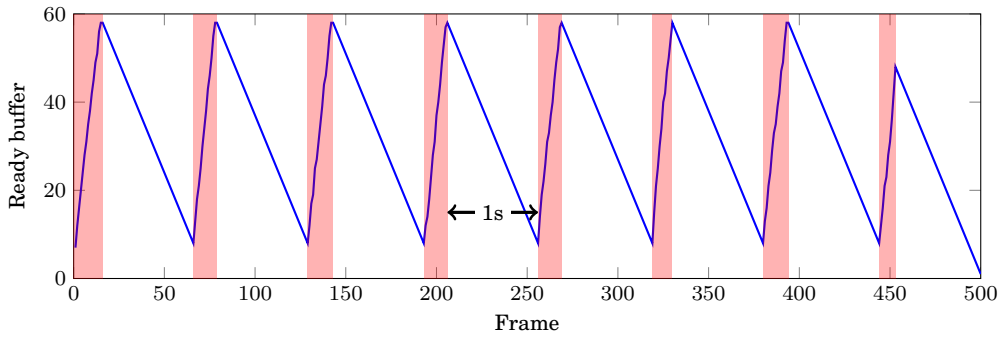


Fig. 2. Buffer occupancy over time when burst decoding a 7Mbps 1080p50 sequence. In the red periods the processor can run at full load, while between them frames only need to be released.

a good prediction. Moreover, even when frames have a similar coded size the decoding complexity can still be quite different due to a different use of coding tools. This leads to high variability in decoding frame time even within a video sequence.

In Figure 1 the frame times are plotted of decoding the 1080p 50Hz sequence BasketballDrive on an Intel Haswell@2.3GHz processor with a single core. The average frame time is 12.76 ms (78.4 fps). Periodic spikes can be observed for complex frames with a maximum of decoding time of 46.47 ms. To achieve a real-time performance of 50 fps, a new frame must be ready every 20 ms, which would not be possible in this case even on a high performance processor. Addressing this issue with more computing resources would be wasteful as the average frame rate is more than sufficient. Instead in video players a display buffer is used to account for this variability. Figure 1 shows that applying a display buffer of 4 (T_{buf4}) and 8 (T_{buf8}) pictures reduces the spikes considerably, providing more opportunity to exploit lower power active states of the processor.

Buffering also improves the effectiveness of the deeper idle states. Because frames have to be decoded periodically, finishing early might not allow the processor to enter a deep idle state, because the next wakeup is predicted to occur soon. To increase the deeper C-state residency, a *burst buffer* could be used to allow the processor to predecode a longer part of the sequence (e.g. 1-second) at full speed and then go to near idle until the buffer is almost empty. In the (mostly) idle time only periodically a frame is released from the buffer. When the number of frames in the buffer drops below some threshold, the next burst decode is triggered. Figure 2 illustrates the buffer occupancy over time for burst decoding.

Table II. Architecture parameters of the platforms used in the evaluation.

Series	Model	Cores	SMT	Cache		Memory		
				Private	Shared	Type	Bus	Size
Haswell	i5-4670T	4	-	32kB/32kB/256kB	6MB	DDR3L-1600	2x64b	8GB
Haswell ULT	i5-4200U	2	2w	32kB/32kB/256kB	3MB	DDR3L-1600	2x64b	8GB
Baytrail-T	Z3740	4	-	24kB/32kB/-	2MB	LPDDR3-1066	2x64b	2GB
Exynos	5410-A7	4	-	32kB/32kB/-	512kB	LPDDR3-1600	2x32b	2GB
	5410-A15	4	-	32kB/32kB/-	2MB	LPDDR3-1600	2x32b	2GB

5. EXPERIMENTAL SETUP

In this section the experimental setup is described. First, we describe the used architectural parameters of hardware platforms and what power consumption can be measured on each platform. Second, we will detail software stack, such as the used operating systems and compilers. Finally, a description of the test sequences is provided, focused on the coding configuration and resulting bitrates in particular.

5.1. Platforms and Power Measurement

To investigate the power efficiency of HEVC decoding on general purpose processors, several state-of-the-art mobile as well as desktop platforms are used, on which power could be measured. In total three Intel platforms are used, two based on the Haswell microarchitecture [Hammarlund et al. 2014] and one based on the Silvermont microarchitecture. The ARM platform is based on the big.Little architecture.

The first Haswell-based platform is centered around a quad-core Haswell 4670T processor and with a a thermal design power (TDP) of 45 W it represent the low power desktop as well as high-performance notebook platforms. The second Haswell platform contains an ultra low TDP (ULT) dual-core processor with simultaneous multithreading (SMT) and has a TDP of 15 W. These processors are used in thin-and-light notebooks (ultrabooks), larger tablets, 2-in-1 devices, and convertibles. The platform based on the Silvermont microarchitecture is Intel's Baytrail platform which combines 4 Silvermont cores in a SoC designed for low cost tablets and has a scenario design power (SDP) of 2 W.

The three Intel platforms allow power and energy to be measured using their Running Average Power Limit (RAPL) architectural power counters [Rotem et al. 2012]. RAPL is a feature included in Intel processors since the Sandy Bridge generation and estimates with an accurate architectural power model the power consumed by the x86 cores, GPU, and processor package. On some processors a power model of the off-chip DRAM is also included. The power and energy of the x86 cores and the total package power are available on all platforms.

The ARM platform, the Odroid XU-E development board, uses the Exynos 5410 System on Chip (SoC) with 4 A7 cores and 4 A15 cores on the same die. The Exynos 5410 SoC is most notably used in Android tablets and smartphones. Unlike the Intel platforms, the Exynos 5410 does not expose architectural power counters. Power measurements have to be performed by the platform, for which the Odroid XU-E board provides 4 external power sensors. These four sensors measure the voltage and current supplied to the A15 cores, the A7 cores, the PowerVR GPU, and the on-package DRAM.

An overview of the architectural and technology parameters, as well as the components for which the power can be measured are provided in Tables II and III. Not every platform can measure all power statistics. In the results, unless specified otherwise, the sum of all measurable power and energy statistics for each platform is presented.

Table III. Technology parameters of the platforms used in the evaluation.

Series	Model	Process	Die size	Nominal Frequency[a]	TDP	Measurable power		
						Cores	Uncore	DRAM
Haswell	i5-4670T	22nm Tri-gate	177mm^2	2300MHz	45W	✓	✓	✓
Haswell ULT	i5-4200U	22nm Tri-gate	134mm^2	1600MHz	15W	✓	✓	✓
Baytrail-T	Z3740	22nm SoC	102mm^2	1866MHz	-	✓	✓	✗
Exynos	5410	28nm HKMG	122mm^2	1.6/1.2 GHz	-	✓	(GPU)	✓

[a]Sustained maximum frequency at which the processor can run at full load in video.

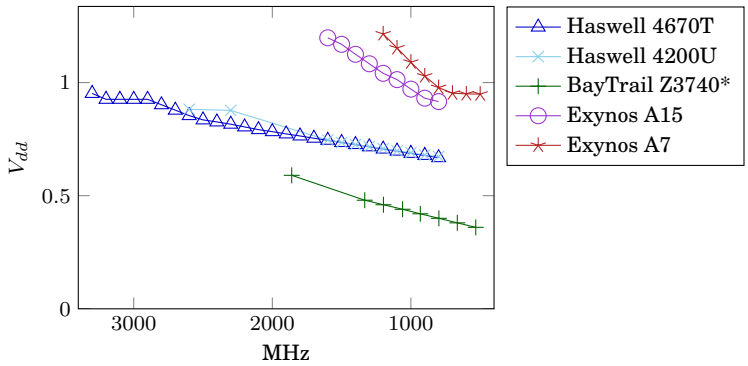


Fig. 3. Available core voltage-frequency points for each platform. *Baytrail based on VID readout HWinfo32.

Each platform can operate at different voltage-frequency points. These points are plotted in Figure 3. It shows that the frequency span of each processor is quite wide, with a 2 to 4× difference between the lowest and highest frequency. On all platforms a P-state is defined for every frequency multiplier, which depending on the base clock results in a P-state every 100MHz to 133MHz. The observed frequency jumps, for example from 1.33GHz to 1.86GHz, are due to the Turboboost feature, which puts the hardware in control of the higher frequency bins. Finally, the figure also shows the advantage of Intel's 22nm Tri-gate process technology: the Intel chips can run at a higher clock frequency with lower supply voltage than the Exynos chips. When comparing the power efficiency the process technology advantage should considered as well.

5.2. Software Environment

The HEVC decoder was compiled using GCC 4.8.1 with -O3, but without auto vectorization. Instead, all the vectorizable kernels are hand-optimized using SIMD: AVX2 is used for Haswell, SSE4.1 for Baytrail, and NEON for ARM. More information can be found in [Chi et al. 2014].

All platforms use a derivative of the (K)Ubuntu 13.10 Linux distributions. The x86 platforms run the 64-bit version and the ARM platform runs a 32-bit version of the operating system. The Linux kernels used are, however, different from the standard one. For the ARM platform a modified kernel of Linux 3.4 is used, which is maintained by the platform manufacturer and includes the necessary drivers. The x86 platforms use Linux 3.12, 3.13rc2, and a modified version of Linux 3.13rc2, for Haswell ULT, Haswell, and Baytrail, respectively. The standard kernel (3.11) could not or only to a lesser extend exploit the C-states available on the platform. For Baytrail-T the support for deeper sleep states was not enabled in the *intel_idle* driver, but has been enabled with a small kernel modification (only for the cores).

Table IV. Kernel and power state drivers.

Series	Model	Kernel	DVFS driver	Idle driver	Core C-state	Pkg C-state
Haswell	4670T	3.13rc2	intel_pstate	intel_idle	C7	C3
Haswell ULT	4200U	3.12	intel_pstate	intel_idle	C7	C7
Baytrail-T	Z3740	3.13rc2(m)	acpi_cpufreq	intel_idle	C6	C1
Exynos	5410	3.4.67	exynos_cpufreq	exynos_idle	C2	n/a

In addition to employing an appropriate Linux kernel, the platform intrinsic devices are configured to use their lower power states. This is needed to reach deeper package C-states on the x86 platforms. All kernels were booted with forcing the use of Active State Power Management (ASPM) for PCI-Express and Active Link Power Management (ALPM) was enabled for each SATA link to reduce the interrupt rate to the processor. While these settings allow longer residences in deeper C-states, not all package C-states could be reached, except for the Haswell ULT platform. The Haswell platform could enter only C3 for the package, while Baytrail could not use a deeper package C-state at all. The C-state support for these platforms are still maturing and it is expected that in future Linux kernel editions this improves. The possible additional power savings, however, are relatively small and will not influence the presented results significantly. An overview of the kernel, DVFS driver, idle driver and the reachable sleep states are listed in Table IV.

5.3. Test sequences

For the experiments the Joint Collaborative Team on Video Coding (JCT-VC) class B (1080p) and EBU UHD1 [Hoffman et al. 2012] (2160p) test sets are used. From both test sets 5 videos are selected. The 1080p sequences are encoded using the HEVC reference encoder HM-12.1 with the random access main configuration (8-bit), and the 2160p sequences are encoded using the random access main10 configuration (10-bit) [Bossen 2013]. All videos sequences are encoded for each quantization pararameter (QP) value between 22 and 38. With the random access configuration a repeating constant QP cascading pattern of 8 pictures is employed, In this pattern the QP of pictures are increased with respect to the base QP for pictures 0 to 8 as follows: +0 (I-frame) , +4, +3, +4, +2, +4, +3, +4, +1. The QP increment pattern for pictures 1 to 8 is repeated for the next group of pictures.

For the experiments only the encoded sequences with QPs with bitrates that fit into a certain bits/frame range are selected. For the 1080p sequences this range is between 40kb/frame to 200kb/frame. For the 2160p sequences the range is between 100kb/frame to 500kb/frame. The characteristics of the encoded test sequences are listed in Table V.

6. PLATFORM POWER CHARACTERISTICS

To provide more insight on the effect of the different power states on the actual power consumed by the processors and memory, we have measured the power during several idle states and active states with different core counts. The results are shown in Figures 4a to 4d. On the processors different idle states have been measured by manipulating the maximum allowed wake-up latency of the system. If the maximum allowed wake-up latency is lower than a particular C-state's wake-up latency, that particular C-state is not entered by the operating system. The active states are measured using multiple instances of *burnP6* or *burnCortexA9* program from the *cpuburn* package to load the processor. These test programs are constructed to have high (integer) instruction level paralellism, but does consume as much dynamic power as, for instance, a

Table V. The selected video sequences from the JCT-VC and EBU-UHD1 test sets

Video	Resolution	Bitdepth	Hz	Frames	QP range
BasketballDrive			50	500	25-34
BQTerrace			60	600	26-31
Cactus	1920×1080	8-bit	50	500	25-34
Kimono			24	241	23-32
ParkScene			24	240	25-35
FountainLady			50	500	26-35
LuppoConfeti			50	500	24-36
RainFruits	3840×2160	10-bit	50	500	22-31
StudioDancer			50	500	22-32
WaterfallPan			50	500	27-35

SIMD floating point based test program, but for the measured power is close to the power consumed by HEVC decoding under similar circumstances.

Figure 4a shows that during idle periods On the Haswell systems it can be observed that during idle most of the energy is consumed by the memory and uncore. The power savings of the *cores* in C-states above C1E is small, but the savings on the package idle power is significant, especially for the Haswell ULT 4200U ultrabook processor. For the 4-core Haswell the C3 and C7 states are not as gainful as on the ULT variant. This is because the package never enters the C7 state of the package even when all the cores are in C7. Additionally, the processor resides only upto 60% of the time in the C3 package state. As stated earlier the support for Haswell is still maturing. On Baytrail the uncore is using much less power. Reasons for this are likely a combination of process technology, less area, and a design more tuned for this performance level.

When active, the cores obviously start using more power. On Haswell this only becomes more significant at the higher frequencies. On Baytrail and Exynos 5410 the power difference between the low frequency active states and the high frequency states are more significant than on the Haswell due to a lower uncore power. The difference between the big A15 and the Little A7 cores is also relatively large, with the A15 consuming around 5× more power at the same frequency.

Because only the GPU part of the Exynos 5410 uncore power can be measured directly, the uncore power in Figure 4d has been derived from the total platform power. Similar to the other platforms, the GPU is not active and uses negligible power. The uncore power is derived using the following model,

$$P_{exynos\_uncore} = 0.9 * (P_{platform} - 1.2W) - P_{sensors} \quad (2)$$

where $P_{platform}$ is the platform power measured after AC-DC conversion, and $P_{sensors}$ is the total power measured by the sensors. In the model, 90% efficiency is estimated for the on-package voltage conversion, and 1.2W is estimated for the peripheral power. These numbers result from the assumption that the SoC uses 0.5W at idle. The absolute value of the uncore value highly depends on the actual power consumed by the board peripherals, and for this reason this model is not used in other results in this paper. It serves more as an indication of the power characteristics of the uncore.

The Exynos 5410 DRAM consumes around 10× less power than the DRAM on Haswell. The memory is mostly idle, because the working set of the load program is very small. The same relative difference has also been observed for a memory intensive benchmark, that stresses the memory subsystem more. For this benchmark the memory consumes around 4× more power on both platforms. The memory of the Haswell platform is of type DDR3L and 8 GB versus the 2 GB LPDDR3 on Exynos 5410. LPDDR3 has a lower supply voltage and is designed for mobile devices, which

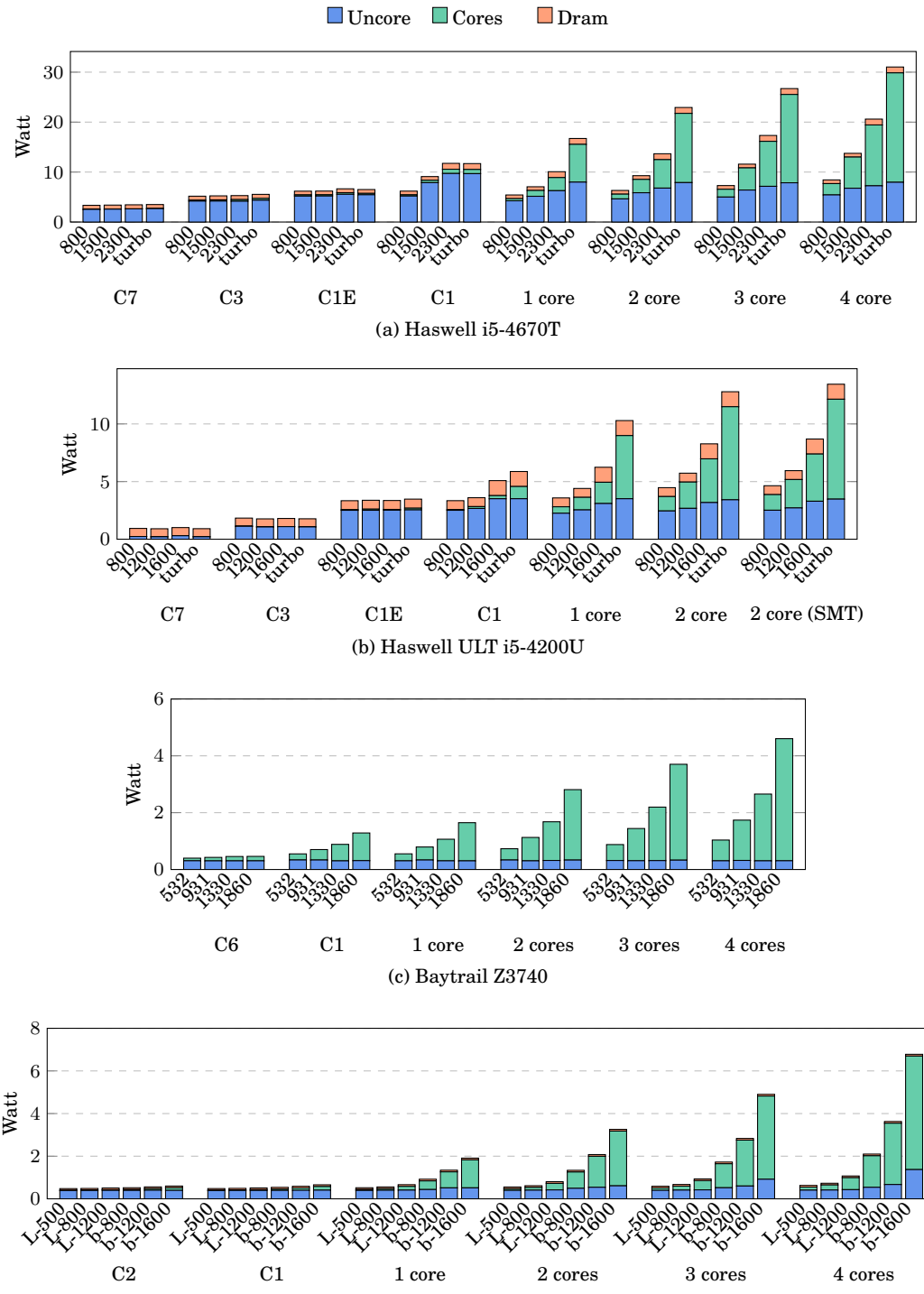


Fig. 4. Idle and load power of the Cores, Uncore, and the offchip Dram. The power is measured for different active cores counts and core frequencies.

typically use only one chip per memory channel. DDR3(L) typically has many chips per channel (commonly 16), and is designed to support larger memory capacities.

Although the power consumption of the different platforms covers a wide span, one common observation can be made. At low frequencies, the power of the cores is relatively low compared to when the cores are running at high frequencies. While the power consumed by the cores scale with the aggregate performance (frequency$\times$cores), the same cannot be said for the uncore and the memory (small spike uncore Exynos 5410 with 4 cores@1.6GHz is caused by the increased temperature). In the next section we will investigate how these power characteristics relate to the power efficiency.

7. POWER AND ENERGY RESULTS FOR HEVC DECODING

In this section the experimental results are organized in four parts. In Section 7.1, we first present how code optimizations (SIMD and multithreading) affect the energy efficiency. Then in Section 7.2 we focus on how DVFS changes the energy efficiency of offline decoding. This is followed in Section 7.3 by an analysis of the power consumption of real-time decoding. There the experiments will focus on the comparison of "exploiting slack" with DVFS and "race to idle". Finally, in Section 7.4, we investigate how effective the current dynamic DVFS governors are in controlling the P-states.

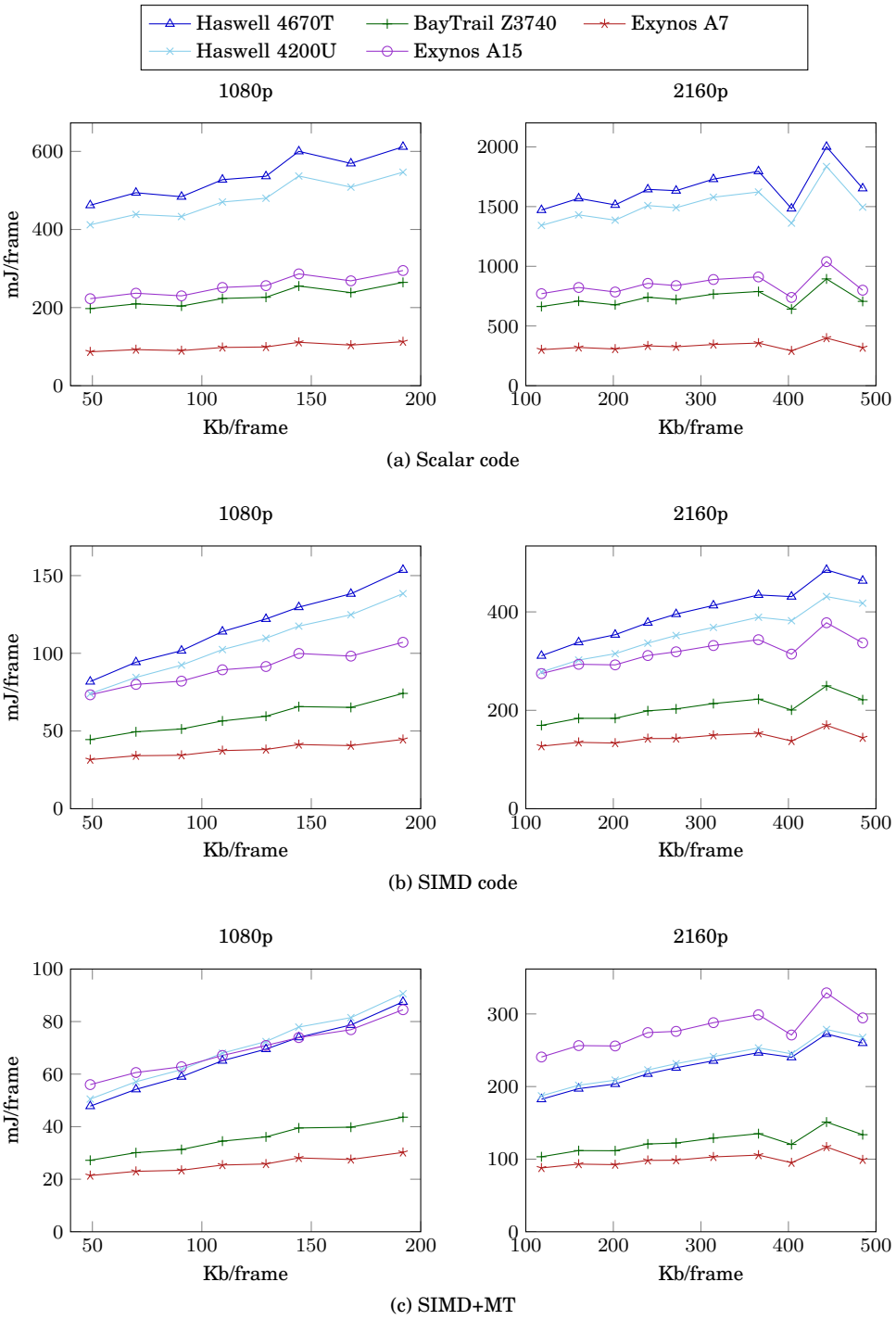
7.1. Impact of Code Optimizations (SIMD and multithreading) on Energy Efficiency

In video decoding the performance, as well as the power consumption, depend on the decoder implementation and the input sequences. Therefore, in this section we present results for the scalar implementation of the decoder, the SIMD implementation of the decoder, and the SIMD implementation of the decoder combined with multithreading. Figures 5a, 5b, and 5c depicts results for the scalar, SIMD, and SIMD with multithreading implementation, respectively. For these experiments the frequency of the platforms was fixed to their nominal frequency, and all multithreading results are using 4 threads. In the figures the energy per frame is plotted against the average kbits per frame (kbpf). A single point represents the average result of the sequences that have similar kbpf values.

The figures show that the energy required per frame is higher for higher bitrates, and also for higher resolutions at the same bitrate. With SIMD optimizations the effect of bitrate on the performance becomes more relevant as the non-vectorizable entropy decoding stage becomes more dominant. What cannot be seen directly from the figure is that sequences with the same bitrate can have quite different decoding complexities, with variations of up to $\pm$20%. This also causes the irregularities in the 2160p results at the 400 and 480 kbpf points, for which some more complex sequences are not represented (no QP value generated a bitstream for that range).

The use of SIMD instructions increases the energy efficiency significantly for all sequences. The increase is largely proportional to the speedup SIMD provides (2.3 - 4.9$\times$). Multithreading also improves the energy efficiency, but to a lesser extent (1.14 - 1.74$\times$) as more power is required to accelerate the decoding.

When comparing the platforms it can be observed that ARM A7 and A15 processors have a relatively high energy efficiency compared to the Intel processors when running scalar code, but lose ground when using the SIMD and multithreading optimizations. The Exynos 5410 using the A15 cores moves from being twice as efficient as the Haswell platforms when executing the scalar code to becoming around 15% less efficient with 2160p sequences using SIMD+MT. The Baytrail platform is at any point more energy efficient than the Exynos 5410 when using the A15 cores, and comes close to matching energy efficiency of the A7 cores when executing with SIMD and multithreading. This is the case even without including the uncore power of the Exynos 5410.



(a) Scalar code

(b) SIMD code

(c) SIMD+MT

Fig. 5. Energy per frame with different levels of decoder optimizations.

Table VI. Performance, energy-efficiency, and power of SIMD and multithreading (MT) optimizations on different platforms

1080p 8-bit	Scalar			SIMD			SIMD + MT		
	fps	mJ/f	W	fps	mJ/f	W	fps	mJ/f	W
Haswell 4670T	20.6	507	10.12	107.1	104	10.70	388.4	60	22.30
Haswell 4200U	14.3	452	6.26	74.7	94	6.74	168.5	63	10.19
Baytrail Z3740	7.4	215	1.55	25.4	53	1.31	93.2	32	2.92
Exynos 5410 A15	7.5	242	1.75	24.1	84	1.99	77.7	64	4.90
Exynos 5410 A7	3.4	94	0.31	9.0	35	0.31	31.2	24	0.74
2160p 10-bit	Scalar			SIMD			SIMD + MT		
	fps	mJ/f	W	fps	mJ/f	W	fps	mJ/f	W
Haswell 4670T	6.6	1609	10.09	29.9	373	10.87	110.6	214	23.14
Haswell 4200U	4.6	1469	6.33	21.1	333	6.86	49.3	220	10.58
Baytrail Z3740	2.2	718	1.49	6.7	195	1.26	25.3	118	2.89
Exynos 5410 A15	2.2	832	1.71	6.1	307	1.82	19.8	268	5.20
Exynos 5410 A7	1.0	325	0.31	2.3	139	0.31	7.9	96	0.74

In Table VI the results are summarized (averaged over all test sequences) and supplemented with the average performance and power results, shown in the metrics frames per second (fps), millijoules per frame (mJ/f), and watts (W). Table VII list application characteristics, such as the average number of instructions per frame (in million instruction per frame (minsn/f)), the fraction of SIMD instructions, as well as the instructions per cycle (IPC). As mentioned before, for the multithreading (MT) results 4 threads have been used (i.e. all the cores of each processor are used). It can be seen that part of the reason why the A15 and A7 based processors consume less power for scalar code is that they executes around 10% fewer instructions. The ARM NEON SIMD instruction set, however, is less effective in reducing the number of instructions than SSE4.1 and AVX2 used on Baytrail and Haswell, respectively. Compared to SSE4.1, NEON requires between 10% and 30% more instructions. The mixed 64/128-bit SIMD ISA is less effective than the full 128-bit Intel ISA. With the AVX2 ISA extension most SIMD operations are widened to 256-bit with 3 operands, allowing the instruction count to be reduced further. We have observed with instruction profiling that around 66% of the executed SIMD instructions are 256-bit AVX2 instruction on the Haswell processors, meaning that around 80% of the 128-bit SIMD instructions could be replaced with 256-bit SIMD.

When using SIMD, the power consumption increases slightly compared to the scalar code on most platforms. This is mainly due to the increased power consumed by the off-chip memory and not the cores themselves. Since around 50% of the executed instructions are SIMD, one could expect that the power consumption would increase due to the increased datapath width and wider memory accesses. The extra power usage is, however, balanced out with an IPC reduction of around 40%. For Baytrail the power usage even decreases when using SIMD. Mainly this is due the partial out-of-order design to Baytrail in which the integer instructions are processed out-of-order and the floating-point/SIMD μops remain processed in-order. This results in less activation of the out-of-order logic when using SIMD instructions.

Multithreading increases the power consumption because more cores are active. The increase in power, however, is relatively lower than the increase of the number of cores, leading to a higher power efficiency when using multiple threads. Furthermore, the performance of the platforms varies widely between, for instance, there is around a factor 13$\times$ performance difference between the Exynos 5410 A7 and Haswell 4670T. The power consumption difference is even larger with more than 30$\times$ difference. The power efficiency differences between a high performance architecture, such

Table VII. Application instruction characterization

1080p 8-bit	50 kbpf			170 kbpf			Average IPC	
	minsn/f scalar	minsn/f SIMD	fraction SIMD	minsn/f scalar	minsn/f SIMD	fraction SIMD	scalar	SIMD
Haswell	319.2	33.8	44.3%	376.6	54.0	34.6%	2.97	1.84
Baytrail Z3740	319.3	52.6	57.9%	376.7	76.0	47.9%	1.32	0.81
Exynos 5410 A15	288.4	66.5	N/A	345.8	87.0	N/A	1.41	1.09
Exynos-5410-A7	288.4	66.5	N/A	345.8	87.0	N/A	0.85	0.50
2160p 10-bit	120 kbpf			440 kbpf			Average IPC	
	minsn/f scalar	minsn/f SIMD	fraction SIMD	minsn/f scalar	minsn/f SIMD	fraction SIMD	scalar	SIMD
Haswell	1027.7	115.0	50.2%	1378.2	143.7	44.8%	2.98	1.72
Baytrail Z3740	1027.7	196.1	64.8%	1378.3	231.2	59.7%	1.23	0.78
Exynos 5410 A15	958.3	260.2	N/A	1277.9	296	N/A	1.35	1.06
Exynos 5410 A7	958.3	260.2	N/A	1277.9	296	N/A	0.81	0.53

as the Haswell 4670T, and a energy efficient architecture employed in the Exynos 5410 are much smaller, however, with a 2.2-2.5$\times$ in favor of the slower Exynos 5410 A7 cores. In reality, the efficiency difference is slightly lower as the uncore power of the Exynos 5410 is not taken into account.

7.2. Impact of DVFS in Engergy Efficiency in Offline Video Decoding Scenario

Offline decoding, used for example in video analytics and transcoding, often run in throughput/cloud computing environments. In this use case the videos are decoded at full speed and the average energy consumed per frame is used as the efficiency metric. As described earlier, each platform can run at different voltage-frequency points (P-states). When the frequency is reduced the supply voltage is lowered as well, which increases the energy efficiency. The energy consumption for all P-states have been measured and the energy efficiency results are depicted in Figure 6 for each platform. Each point represents the average joules per frame. In addition to the total measurable energy (E_{tot}) also the energy contributed by the cores only is plotted (E_{core}), for the sequential decoder using 1 thread (T1), and multithreading decoder results with 2 threads and 4 threads (T2 and T4). For space reasons, only for Haswell 4670T 2160p results are provided in addtion to the 1080p results.

A main observation is that the joules per frame consumed exhibit a similar trend on all platforms, and for both resolutions (the not presented 2160p figures depict a near identical shape as their 1080p counterparts). First, it can be seen that employing more threads lowers the total energy consumption in all cases. Second, on most platforms the most energy efficient P-state is not the one with the lowest frequency. Mostly a moderate frequency in the middle of the frequency span is the most efficient one. Finally, the energy consumed by the cores exhibits a different trend than that of the total energy consumption. The number of threads does not change the energy efficiency of the cores much, and the most energy efficient P-state is the one with the lowest frequency. Thus, lower frequencies result in less energy spent in the cores, with up to an order of magnitude difference between the highest and the lowest frequencies.

Because on current platforms voltage and frequency is mainly applied to the cores, the total energy consumption does not always improve with lower frequencies. When decreasing the frequency the runtime increases proportionally, but off-chip memory and uncore structures, such as the memory controller, I/O hubs, and L3 caches, either do not perform DVFS at all, perform DVFS in a coarser grained manner, or have less headroom for voltage scaling. As a result, the power usage of these uncore structures does not scale down as much as the cores when less performance is required.

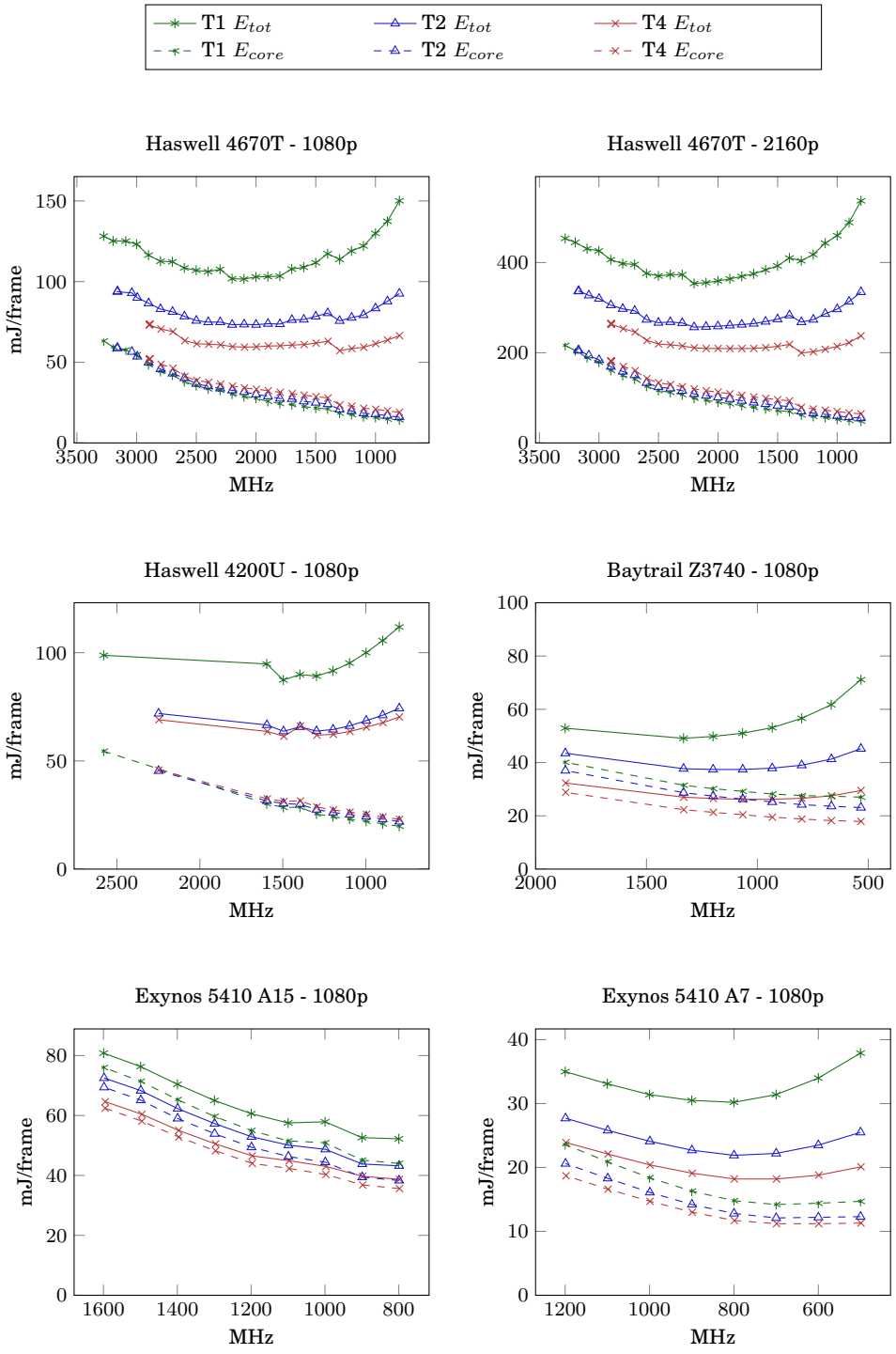


Fig. 6. Energy consumed per frame at different voltage-frequency points (P-states).

The energy consumption of the Cortex-A7 cores show a slightly different behavior than the observed trend at low frequencies. The Cortex A7 cores do not benefit from reducing the frequency below 800MHz. While the frequency can be reduced further down to 500 MHz, the voltage remains constant from 800 MHz on. The energy efficiency results are, therefore, in line with the expectation as scaling only the frequency does not improve energy efficiency. Frequency-only scaling P-states can only improve the power consumption during idle periods when the core is not power gated to reduce clocking power, but in general should not be considered when the core is active.

For offline decoding, DVFS provides limited energy efficiency benefits for the tested platforms. If we would also consider the power consumed by the components outside of the processor and memory, such as hard disks and I/O chips, the energy efficiency benefits would decrease further and the most efficient point would move closer to the maximum frequency. DVFS can improve the energy efficiency for offline decoding only on platforms where the processor consumes most of the energy.

7.3. Power Consumption in Real-time Scenario: "Race to Idle" vs "Exploiting Slack"

In the second scenario we focus on real-time video decoding, as is required for video playback. In contrast to offline decoding, using a faster or higher clocked processor does not reduce the decoding time of the video in real-time decoding. Because a certain frame rate must be achieved the decoding time is always fixed to the length of the video sequence. Therefore, for real-time decoding the power consumption is equivalent to power efficiency because the runtime is the same. Not all videos, however, can be decoded at the required frame rate for all platform, core count, and frequency combinations. Therefore, only a selection of test sequences is used to have comparable results. In Figures 7 to 9 the power consumption of all platforms is plotted for real-time decoding of 1080p24 2.1 Mbps sequences (ParkScene@qp30, Kimono@qp27), and 1080p50 4.5 Mbps sequences (BasketballDrive@qp29, Cactus@qp29). The power consumption for the 2160p50 12 Mbps sequences (FountainLady@qp30, LupoConfetti@qp29, RainFruits@qp26, StudioDancers@qp26, WaterfallPan@qp30) is only presented for the Haswell 4670T platform, since this platform is the only that is fast enough.

The power consumption is plotted against the frequencies associated with each the P-state that were capable of decoding the sequences in real-time. Each experiment is performed with a frame buffer of 8 frames to smooth the frame-to-frame decoding time variations. Additionally, the same experiments have been performed with an additional burst buffer able to hold 1-second of decoded video frames. The results of these experiments are shown on the right side of the figures.

The figures show that the power efficiency behavior of real-time decoding responds very differently to DVFS than offline decoding. On all platform less power is consumed when decoding the same sequences at lower frequencies. Furthermore, using more threads than required for real-time decoding mostly increases the power consumption. Compared to offline decoding, the power consumption behavior of only the cores is similar, though, as it reduces with lower frequencies.

The main difference between offline decoding and real-time decoding is that faster processors and higher frequencies do not translate in shorter runtimes, but instead more idle time is introduced with a non-zero power consumption. The figures show that for all platforms, decoding faster at higher frequencies and/or by using more cores ("race to idle") never yields the lowest power consumption for real-time decoding. Instead lower power consumption is achieved by spreading out the computation over more active time running at a lower frequency ("exploiting slack").

Even when performing burst decoding, in order to stay in deeper C-states longer, the power consumption behavior does not change much. Additional experiments showed

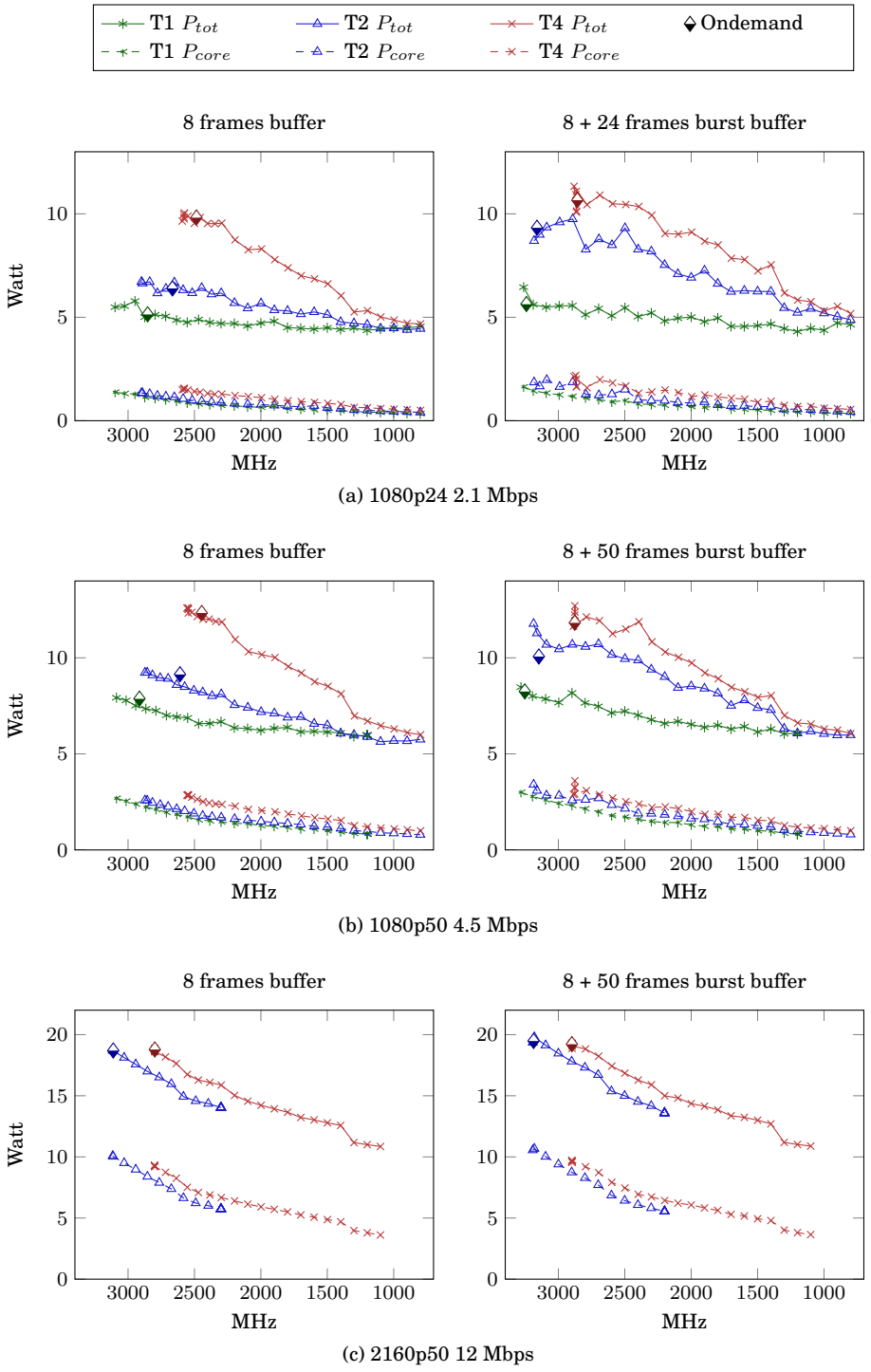
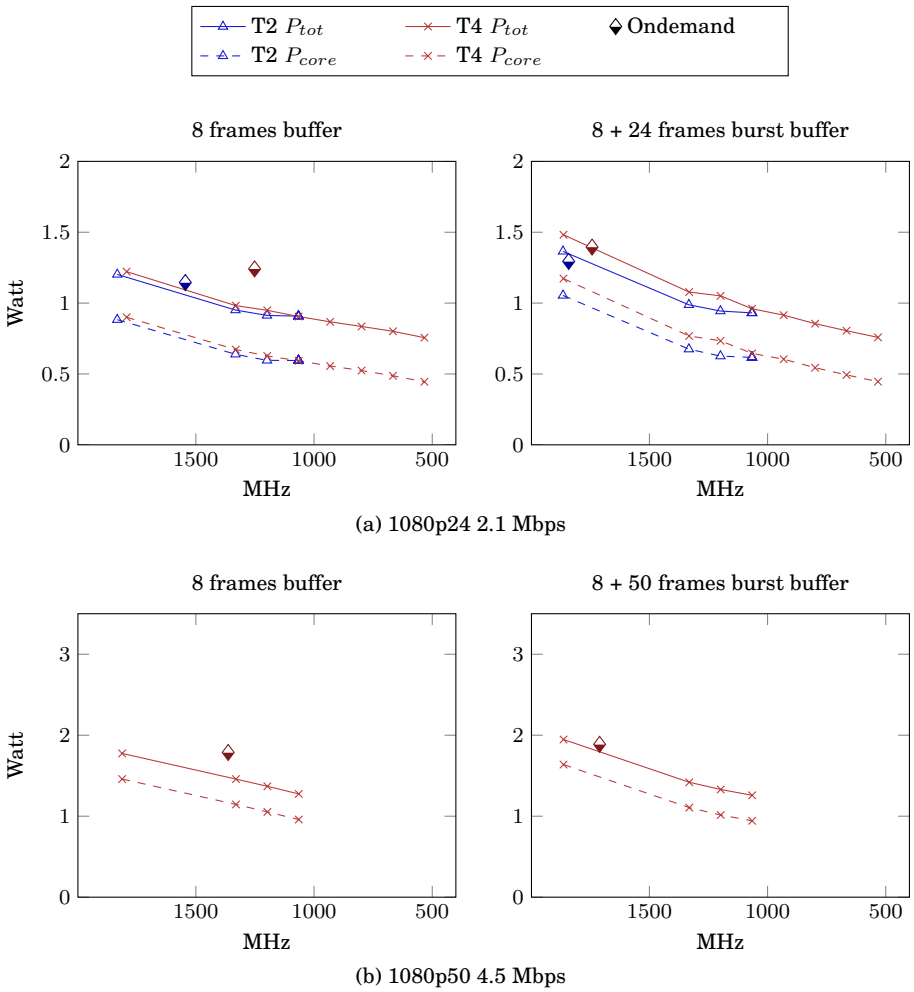


Fig. 7. Power of real-time decoding video at different voltage-frequency points for Haswell 4670T.



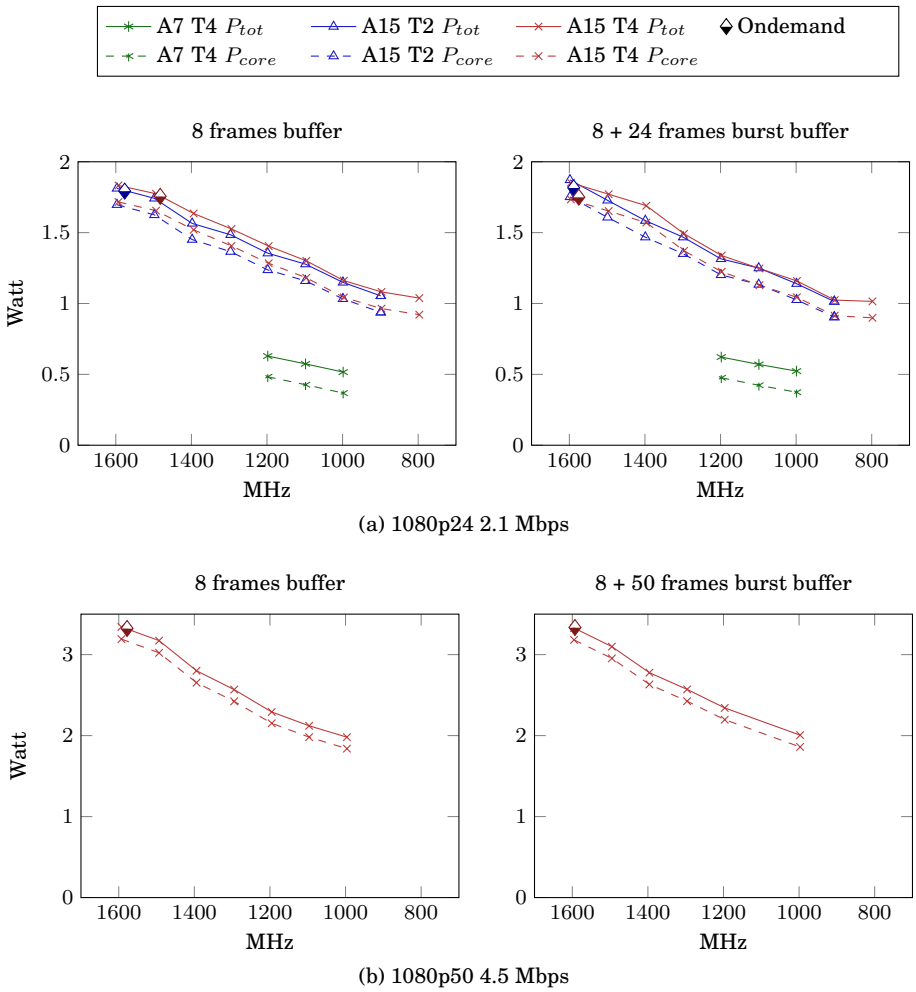
(a) 1080p24 2.1 Mbps

(b) 1080p50 4.5 Mbps

Fig. 8. Power of real-time decoding video at different voltage-frequency points for Baytrail Z3740.

that, when using more threads, the deeper C-state residency did not increase as much as should have been possible. Mostly even the time spent in the deeper C-state residency is reduced, which is the cause of the observed higher power consumption when using multiple threads. For instance, the C-state residency of encoding a 1080p24 2.1 Mbps sequence on Haswell 4670T changes from CO(22%) - C1(18%) - C7(370%) when running 1 thread, to CO(33%) - C1(205%) - C7(162%) when running 4 threads. Also the package C-state residency is affected, and changes from PC0(32%) - PC7(68%) to PC0(90%) - PC7(10%), respectively, for 1 thread and 4 threads. While this could originate from hardware limitations to put multiple cores to a deeper C-states simultaneously, it is more likely that this is caused by adverse effects to the idle time predictor in the kernel idle driver. We observed that the idle periods are predicted more conservatively when multiple threads were running.

From the results can be concluded that using lower frequencies and fewer threads for real-time decoding is more advantageous than “race to idle”. The technique itself, however, is not always inferior for real-time decoding and other real-time applications. On platform-application combinations for which running at a higher frequency increases



(a) 1080p24 2.1 Mbps

(b) 1080p50 4.5 Mbps

Fig. 9. Power of real-time decoding video at different voltage-frequency points for Exynos 5410.

the power consumption relatively little, and the deep idle states consume significantly less power than the lowest possible active state, and these deep idle states can have high residency, “race to idle” can be more effective. These cases, however, are the exception rather than the rule. This situation could arise for instance in a hypothetical leakage dominated design. Some have predicted that leakage power would dominate the total power consumption when left unattended [Kim et al. 2003; Agarwal et al. 2004], but architectural and process enhancements have been able keep the leakage power in check so far and are likely able to continue this in the foreseeable future [Kaul et al. 2012].

What can be observed further is that the power consumption of the high performance (Haswell) platforms does not differ significantly at low activity levels. For instance, on the Haswell 4670T, the 1080p50 sequences requiree roughly twice as much processing compared to the 1080p24 sequences, but the lowest achievable power consumption is 5.6 W and 4.5 W, respectively. The reason for this is that most of the power at these points is consumed by the uncore and memory, and, as in the offline scenario, it is very

important to consider the power consumption of the entire processor, especially in the low power modes.

In addition to the hardware not being able to scale down power consumption at low activity, the software stack controlling the low power modes also limits the effectiveness of DVFS. This is evident by the fact that the OS-controlled *ondemand* DVFS governor is not able to select the most efficient P-state. The *ondemand* DVFS governor actually tends to select the P-states that result in the highest possible power consumption. In the next section we investigate these hardware and software inefficiency further.

7.4. Effectiveness of Current Dynamic DFVS Governors

Current OS DVFS drivers (at least the ones in Linux) change frequencies based on the observed load, and respond quite rapidly to prevent performance regressions for latency sensitive applications. In the previous section we found that ondemand DVFS does not achieve the lowest power consumption with real-time video decoding and actually is closer to worst case power consumption. The analysis was limited to a small selection of video sequences. A more detailed and focused analysis is performed covering the complete test set in this section.

Figure 10 shows the power consumption of the OS ondemand DVFS governor and best manual DVFS configuration for each sequence that could be decoded in real-time on the different platforms. The power consumption is plotted against the required activity level (required active cycles per second) to decode the sequence in real-time. For the ondemand DVFS experiments the number of decoding threads is set equal to the core count (no SMT), and for all experiments a frame buffer of size 8 is used.

The figures show that for all platforms and nearly all sequences, ondemand DVFS does not achieve lower power consumption than using the optimal manually selected static threads-frequency combination. At high and very low activity levels the two converge. At high activity levels the cores have to operate at/near the maximum to be able to decode the sequences in real-time. At low loads the power consumption of the cores is relatively low and, independent of the method, DVFS is not able to reduce the power consumption much further. For the other sequences, corresponding to mid-activity levels, more than 50% higher power consumption is required by ondemand DVFS.

The static DVFS approach is infeasible for real-world scenarios, however, since the complexity of video sequences change heavily in full length videos. Due to complexity variations in a sequence a static approach would also not be able to achieve the minimum power consumption. The frequency must be high enough to decode the most complex 8 consecutive frames in real-time, since there are that many frame buffers. The ondemand DVFS approach, on the other hand, is not currently bot able to use the P-states of a modern processor effectively.

8. CONCLUSIONS

The fact that modern multicore CPUs can achieve higher performance than is required for video decoding can, to a certain degree, be exploited to lower the power consumption. To reduce power, modern CPUs provide various low power states that reduce the active as well as the idle power. In this paper we have investigated how both code optimization of the application and employing the exposed low power states of the processor can improve power efficiency. This investigation has been performed for both offline as well as real-time video decoding scenarios.

In the offline scenario, the exploitation of SIMD and multithreading improves the performance and (hence) the power efficiency on all platforms. The energy savings SIMD provides are, in most cases, proportional to the speedup it provides, since the

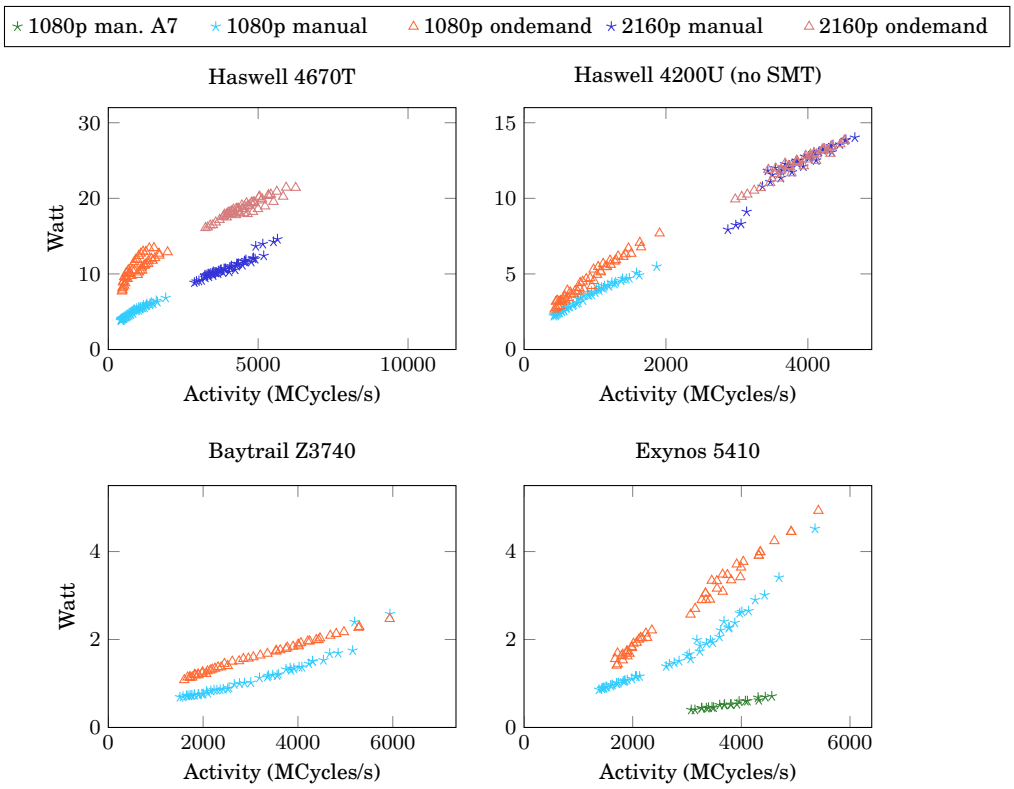


Fig. 10. Power consumption of real-time decoding with best manual configuration and ondemand DVFS.

power consumption of the SIMD code is hardly higher than that of the scalar code. Multithreading also improves the performance and the power efficiency, but since the power consumption increases when using multiple cores the power efficiency gain are lower. Employing DVFS in the offline decoding scenario reduces the energy consumption of the processor even further, and the most energy efficient frequency is often found in the middle of the frequency range of the processor. Utilizing these frequencies for offline decoding, however, should only be considered if the energy consumed by the cores is a large fraction of the total energy consumed by the entire platform. On systems where the processor consumes only a small fraction of the energy, reducing the frequency would actually increase the total energy consumption due to longer execution times.

The optimal strategy for the real-time decoding scenario is different from the optimal strategy for offline decoding. For real-time decoding running at a higher frequency and employing a faster processor does not reduce the execution time, but instead introduces more idle time during which the platform consumes the idle power of the system. As for offline decoding, SIMD also provides power savings for real-time video decoding. Multithreading, however, often increases power consumption when less threads would have been also sufficient for reaching real-time performance. Often the deeper C-state residency decreases when more threads are used to perform the same work and as a result it increases the average idle power. Instead of finishing faster in order to idle longer, referred to as “race-to-idle”, we found that for real-time decoding “exploiting

slack" by lowering the frequency and using more cores provided the highest power efficiency.

Current operating and runtime systems, however, do not exploit the power saving modes of the processor effectively. Deep C-states are not entered as often as possible and the *ondemand* DVFS strategy the OS applies runs the processor most of the time at the highest frequency, providing sub-optimal power savings. Significant power reductions ($\geq 30\%$) are still possible for real-time HEVC decoding by improving the effectiveness of the exploitation of the power saving modes, which is future work.

Additionally, our results indicate that architectural improvements are required to reduce the power consumption of the platform at low loads. At loads that demand little processing power, the high performance Haswell platform consume about $10\times$ as much power as the mobile SoC platforms, in particular the Exynos 5410 with A7 cores. At full load, on the other hand, the Exynos 5410 is only $3\times$ as power efficient as the Haswell 4670T. Improving the power-performance characteristics at low loads would lengthen the battery life more than improving them at high loads. For exactly this reason ARM introduced the big.LITTLE heterogeneous architecture consisting of so-called big high performance A15 cores and little power efficient A7 cores. Although this technology improves the power efficiency of the processor, our results show that at low loads more power is consumed by the uncore and the memory than by the cores.

REFERENCES

A. Agarwal, C.H. Kim, S. Mukhopadhyay, and K. Roy. 2004. Leakage in nano-scale technologies: mechanisms, impact and design considerations. In *Design Automation Conference, 2004. Proceedings. 41st*. 6–11.

E. Akyol and M. van der Schaar. 2008. Compression-Aware Energy Optimization for Video Decoding Systems With Passive Power. *IEEE Transactions on Circuits and Systems for Video Technology* 18, 9 (Sept. 2008), 1300–1306.

F. Bossen. 2013. *Common test conditions and software reference configurations*. Technical Report L1100. JCTVC.

F. Bossen, B. Bross, K. Suhring, and D. Flynn. 2012. HEVC Complexity and Implementation Analysis. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1685–1696.

Benjamin Bross, Valeri George, Mauricio Alvarez-Mesa, Tobias Mayer, Chi Ching Chi, Jens Brandenburg, Thomas Schierl, Detlev Marpe, and Ben Juurlink. 2013. HEVC Performance and Complexity for 4K Video. In *IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*.

Len Brown. 2005. ACPI in Linux. In *Linux Symposium*. 51.

T.D. Burd, T.A. Pering, A.J. Stratakos, and R.W. Brodersen. 2000. A Dynamic Voltage Scaled Microprocessor System. *IEEE Journal of Solid-State Circuits* 35, 11 (Nov. 2000), 1571–1580.

AP. Chandrakasan, S. Sheng, and R.W. Brodersen. 1992. Low-Power CMOS Digital Design. *IEEE Journal of Solid-State Circuits* 27, 4 (Apr 1992), 473–484. DOI:http://dx.doi.org/10.1109/4.126534

C. C. Chi, M. Alvarez-Mesa, B. Bross, B. Juurlink, and T. Schierl. 2014. SIMD Acceleration for HEVC Decoding. *IEEE Transactions on Circuits and Systems for Video Technology* PP, 99 (2014), 1–1. DOI:http://dx.doi.org/10.1109/TCSVT.2014.2364413

Kihwan Choi, Karthik Dantu, Wei-Chung Cheng, and Massoud Pedram. 2002. Frame-based Dynamic Voltage and Frequency Scaling for a MPEG Decoder. In *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design (ICCAD '02)*. ACM, New York, NY, USA, 732–737.

Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2012. Dark Silicon and the End of Multicore Scaling. *IEEE Micro* 32, 3 (May 2012), 122–134.

Peter Greenhalgh. 2011. Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7. (Sept. 2011). http://www.arm.com/files/downloads/big.LITTLE_Final.pdf

Yan Gu, Samarjit Chakraborty, and Wei Tsang Ooi. 2006. Games Are Up for DVFS. In *Proceedings of the 43rd Annual Design Automation Conference*. 598–603.

P. Hammarlund, A.J. Martinez, A.A. Bajwa, D.L. Hill, E. Hallnor, Hong Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, R.B. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, S. Gunther, T. Piazza, and T. Burton. 2014. Haswell: The Fourth-Generation Intel Core Processor. *Micro, IEEE* 34, 2 (Mar 2014), 6–20. DOI:http://dx.doi.org/10.1109/MM.2014.10

Nikos Hardavellas. 2012. The Rise and Fall of Dark Silicon. *USENIX* 37, 2 (April 2012).

Han Hoffman, Adi Kouadio, Yvonne Thomas, and Massimo Visca. 2012. The Turin Shoots. In *EBU Tech-i*. Number 13. European Broadcasting Union (EBU), 8–9. http://tech.ebu.ch/docs/tech-i/ebu_tech-i_013.pdf

Intel. 2008. Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors. (2008).

S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S.K. Gb, R. Ramanarayanan, V. Erraguntla, J. Howard, S. Vangal, S. Dighe, G. Ruhl, P. Aseron, H. Wilson, N. Borkar, V. De, and S. Borkar. 2012. A 280mV-to-1.2V Wide-Operating-Range IA-32 Processor in 32nm CMOS. In *2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 66–68.

Jagrit Kathuria, M Ayoubkhan, and Arti Noor. 2011. A Review of Clock Gating Techniques. *MIT International Journal of Electronics and Communication Engineering* 1, 2 (2011).

H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar. 2012. Near-Threshold Voltage (NTV)–Opportunities and Challenges. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*. 1149–1154.

Stefanos Kaxiras and Margaret Martonosi. 2008. *Computer Architecture Techniques for Power-Efficiency* (1st ed.). Morgan and Claypool Publishers.

N.S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. 2003. Leakage Current: Moore's Law Meets Static Power. *Computer* 36, 12 (Dec 2003), 68–75.

Stephen Kosonocky. 2011. Practical Power Gating and Dynamic Voltage/Frequency Scaling. (August 2011). http://www.hotchips.org/wp-content/uploads/hc_archives/hc23/HC23.17.1-tutorial1/HC23.17.111.Practical_PGandDV-Kosonocky-AMD.pdf Hot Chips: A Symposium on High Performance Chips.

Rakesh Kumar, Dean M. Tullsen, Parthasarathy Ranganathan, Norman P. Jouppi, and Keith I. Farkas. 2004. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. *SIGARCH Compututer Architecture News* 32, 2 (March 2004), 64–.

Etienne Le Sueur and Gernot Heiser. 2010. Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems (HotPower'10)*. USENIX Association, Berkeley, CA, USA, 1–8.

Etienne Le Sueur and Gernot Heiser. 2011. Slow Down or Sleep, That is the Question. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'11)*. USENIX Association, Berkeley, CA, USA, 16–16.

Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2013. The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing. *ACM Transactions on Architecture and Code Optimization* 10, 1, Article 5 (April 2013), 29 pages. DOI:http://dx.doi.org/10.1145/2445572.2445577

Wen-Yew Liang, Ming-Feng Chang, Yen-Lin Chen, and Chin-Feng Lai. 2013. Energy Efficient Video Decoding for the Android Operating System. In *IEEE International Conference on Consumer Electronics (ICCE) 2013*. 344–345.

Zhan Ma, Hao Hu, and Yao Wang. 2011. On Complexity Modeling of H.264/AVC Video Decoding and Its Application for Energy Efficient Decoding. *IEEE Trans. on Multimedia* 13, 6 (Dec. 2011), 1240–1255.

Malena Mesarina and Yoshio Turner. 2003. Reduced Energy Decoding of MPEG Streams. *Multimedia Systems* 9, 2 (2003), 202–213.

Venkatesh Pallipadi, Shaohua Li, and Adam Belay. 2007. cpuidle–Do nothing, efficiently.... In *Proceedings of the Linux Symposium*.

Venkatesh Pallipadi and Alexey Starikovskiy. 2006. The Ondemand Governor. In *Proceedings of the Linux Symposium*, Vol. 2. sn, 215–230.

E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann. 2012. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE Micro* 32, 2 (March 2012), 20–27.

T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. De Micheli. 2001. Dynamic voltage scaling and power management for portable systems. In *Proceedings of the Design Automation Conference*. 524–529.

Bob Steigerwald. 2011. *Energy Aware Computing. Powerful Approaches for Green System Design*. Intel Press, Hillsboro, Or.

Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (Dec. 2012), 1649–1668.

Received March 2014; revised October 2014; accepted November 2014