Parameter Sensitivity Analysis of the Energy/Frequency Convexity Rule for Nanometer-scale Application Processors

Karel de Vogeleer, Gerard Memmi, Pierre Jouvelot

Parameter Sensitivity Analysis of the Energy/Frequency Convexity Rule for Nanometer-scale Application Processors

Karel De Vogeleer, Gerard Memmi, and Pierre Jouvelot

Abstract—Both theoretical and experimental evidence are presented in this work in order to validate the existence of an Energy/Frequency Convexity Rule, which relates energy consumption and microprocessor frequency for nanometer-scale microprocessors. Data gathered during several month-long experimental acquisition campaigns, supported by several independent publications, suggest that energy consumed is indeed depending on the microprocessor's clock frequency, and, more interestingly, the curve exhibits a clear minimum over the processor's frequency range. An analytical model for this behavior is presented and motivated, which fits well with the experimental data. A parameter sensitivity analysis shows how parameters affect the energy minimum in the clock frequency space. The conditions are discussed under which this convexity rule can be exploited, and when other methods are more effective, with the aim of improving the computer system's energy management efficiency. We show that the power requirements of the computer system, besides the microprocessor, and the overhead affect the location of the energy minimum the most. The sensitivity analysis of the Energy/Frequency Convexity Rule puts forward a number of simple guidelines especially for by low-power systems, such as battery-powered and embedded systems, and less likely by high-performance computer systems.

Index Terms—DVFS, energy optimization, Energy/Frequency Convexity Rule, SoC.

1 Introduction

The execution time characteristics and power requirements of a code sequence are the main drivers that define its final energy consumption. This is a direct result of the definition of electrical energy consumption: the integral of electrical power over time. The execution time is influenced by the type and the amount of operations contained by the code sequence of concern. For example register-based operations will require less energy to execute compared to external memory-based instructions. As such, each functional unit within a microprocessor and, more generally, each component of the computer system have their own respective power and execution time profiles. As a result, every code sequence has different power and execution time demands. For example, Carroll and Heiser [1] showed that, for an embedded system running `equake`, `vpr`, and `gzip` from the SPEC CPU2000 benchmark suite, the microprocessor energy consumption exceeds the RAM memory consumption, whereas `crafty` and `mcf` from the same suite showed to be straining more energy from the device RAM memory.

A property of a code sequence's energy consumption is that, under certain assumptions, it shows convex properties, which is henceforth referred to as the Energy/Frequency Convexity Rule [2]. The rule states that there exists an optimum clock frequency for the execution of each sequence of code that minimizes the energy consumption of that code sequence. Under certain conditions this optimal clock frequency, minimizing energy consumption, lies between the minimum and maximum clock frequency. The existence of a minimum energy point results from the behavior of the microprocessor's power and the execution time w.r.t. the clock frequency. The microprocessor's power increases about linearly with clock frequency, meaning that more energy is consumed when the microprocessor's speed is increased. On the other hand, the slower the clock frequency, the longer execution time will increase the energy expenditure. As will be shown, running at the optimal clock frequency is a trade-off between performance, in terms of execution time, and energy savings. For applications requiring human interaction, it has been shown that the clock frequency can be scaled down considerably without affecting user's experience [3]. In this paper, experimental evidence is presented, supported by several independent publications, for the existence of an Energy/Frequency Convexity Rule that relates energy consumption and microprocessor clock frequency on mobile devices. This convexity property seems to ensure the existence of an optimal frequency where energy consumption is minimal. This existence claim is based on both theoretical and practical evidence on a Systems-on-Chip (SoC). Data gathered via acquisition campaigns on multiple platforms suggest that the energy consumed per input element is strongly correlated with microprocessor clock frequency and, more interestingly, that the corresponding curve exhibits a clear minimum over a frequency window specific to the computer system. An analytical model of this behavior is also motivated, which fits well with the experimental data. A parameter sensitivity analysis is carried out to assess the influence of the parameters on the optimal frequency minimizing energy consumption. This optimal frequency is shown to increase when the power requirements

- *K. De Vogeleer and Gerard Memmi are with TELECOM ParisTech – INFRES – CNRS LTCI - UMR 5141 – Paris, France, Email: {karel.devogeleer,gerard.memmi}@telecom-paristech.fr*
- *Pierre Jouvelot is with MINES ParisTech, PSL Research University, France. Email: pierre.jouvelot@mines-paristech.fr*

of the computer system, excluding the microprocessor's, increase. Clock cycles lost for routine maintenance of the system also force the optimal frequency up. The optimal frequency as derived from the theoretical framework is, however, independent of the number of instructions to be executed.

In addition to a deeper theoretical and practical understanding of a microprocessor's energy consumption and the Energy/Frequency Convexity Rule, this paper offers a new, in-depth, parameter sensitivity analysis compared to what was presented in De Vogeleer *et al.* [2]. The main contributions of this paper are thus:

- a theoretical framework for the Energy/Frequency Convexity Rule;
- a sensitivity analysis of the Energy/Frequency Convexity Rule to estimate the impact of multiple input parameters;
- an analysis of the Energy/Frequency Convexity Rule under special conditions, such as, out-of-order execution (OOE) and absence of slack time;
- supportive experimental data and a comprehensive survey of the state of the art.

The rest of the paper is organized as follows. Section 2 elaborates the Energy/Frequency Convexity Rule. Followed by the presentation of experimental results in Section 3, a parameter sensitivity analysis is carried out in Section 4. An overview of the related work is presented in the state-of-the-art Section 5. Finally, Section lists the main conclusions drawn from our analysis supporting a better usage of the energy especially for embedded systems.

2 Single-Core Convexity Model

The energy consumption of a computer system comprising a microprocessor, and possibly other components, over a time interval Δt, is equal to the integral of its system's power usage over time:

$$E_{\text{sys}}(\Delta t) = \int_0^{\Delta t} P_{\text{sys}}(t)\,\mathrm{d}t = \int_0^{\Delta t} I(t)\cdot V(t)\,\mathrm{d}t. \quad (1)$$

If the power is considered constant, the integral is equivalent to the product of the power consumption and the timespan of interest. $V(t)$ can often be considered constant by design; for example, portable devices such as smartphones are supplied by 3.7 V lithium-ion batteries, and microprocessors operate at very specific voltage levels. The current's time-dependent variance depends on the context, its history and the state of the microprocessor. However, at the time frame of an instruction execution, henceforth referred to as a *time quanta*, the energy consumption can be deemed quasi constant. Following this definition, the parameters that define the energy consumption during a time quanta are also constant. As such, similar to the rationale behind the *Riemann sum*, the total energy consumption of a code sequence can be thought of as the sum of the energy consumption during each time quanta Δt:

$$E_{\text{sys}} = \sum_{i=1}^{n} E_{\text{sys},i} = \sum_{i=1}^{n} P_{\text{sys},i}\cdot\Delta t_i, \quad (2)$$

where n is the number of time quanta. Δt_i is the time frame over which $P_{\text{sys},i}$ is constant. Δt_i could be the length of one instruction execution or, when the power variance is negligibly small, Δt_i can be the length of an arbitrary-sized code sequence. One has $\Delta t = \sum_{i=0}^{n}\Delta t_i$.

The models for the power and execution time are developed separately in the next two subsections . A more profound expound of the models can be found in De Vogeleer [4].

2.1 Power Model

A computer system's power usage P_{sys} is the sum of three power components:

1) P_{cpu}, the microprocessor's power,
2) P_{drop}, the system's power usage that is dependent or controllable by the microprocessor, and
3) P_{back}, the system's power that is independent of the microprocessor.

P_{drop} can be due to components that are put to sleep when the microprocessor doesn't need their functionality, e.g., audio codecs, camera circuits, or the radio interface. P_{back} constitutes components that require power independent from what the microprocessor is doing, e.g., memory refreshing in synchronous dynamic random access memory (SDRAM). P_{back} is however controllable. It is noted that the display of a hand-held device falls also under P_{back} as it is active when the user requires interaction with the device, not necessarily when the microprocessor is active.

For the formulation of the microprocessor's power P_{cpu}, we combined the well know expression for an electronic circuit's power dissipation $\frac{1}{2}\alpha V^2 f$ [5], referred to as the *dynamic power*, and the leakage current model of Skadron *et al.* [6] :

$$P_{\text{cpu}} = (1+\gamma V)\cdot\xi f V^2, \quad (3)$$

where γ is a parameter describing the magnitude of the leakage currents due to capacitor-based circuits, V is the supply voltage and ξ is a parameter defining the power requirements of the microprocessor. It is known that the leakage currents are temperature-dependent [7]. Henceforth, however, we deem the temperature constant throughout our analysis.

2.2 Execution Time Model

The execution time Δt of a code sequence, including slack time β and time thieves f_{k} (the time spend by the operating system), can be modeled as:

$$\Delta t = cc_{\text{b}}\left(\frac{1}{f-f_{\text{k}}}+\beta\right), \quad (4)$$

where cc_{b} is the number of clock cycles dedicated to the execution of the user program's statements, f_{k} the average number of clock cycles per time unit lost due to time thieves, and β the average amount of slack time per clock cycle. By definition $f \geq f_{\text{k}}$ since the system can't steal more clock cycles than what is available.

Time thieves, represented by f_{k} in Equation 4, are clock cycles lost due to low-level operations. These time thieves have higher priority than cc_{b}. Examples of f_{k} are pipeline

stalls due to branch miss-predictions, misaligned memory accesses, page faults, operation interventions, interrupt handling, operating system routine tasks, etc. The slack time represented by $cc_{\mathrm{b}}\beta$ is the time the microprocessor cannot continue execution as it is waiting for external data, e.g., in the main memory due to cache misses. Slack time can be addressed with out-of-order execution (OOE), which would scale down β (See Section 4.3).

2.3 System's Energy Consumption Model

Inserting the power model and execution time model from Equation 3 and 4, respectively, into the definition of the system's energy consumption in quanta time i:

$$\begin{aligned} E_{\mathrm{sys},i} &= P_{\mathrm{sys},i} \cdot \Delta t_i \\ &= (P_{\mathrm{cpu},i} + P_{\mathrm{drop},i} + P_{\mathrm{back}}) \cdot \Delta t_i \\ &= \left((1+\gamma_i V) \cdot \xi_i f V^2 + P_{\mathrm{drop},i} + P_{\mathrm{back}}\right) \\ &\qquad \cdot cc_{\mathrm{b},i} \left(\frac{1}{f - f_{\mathrm{k},i}} + \beta_i\right). \end{aligned} \tag{5}$$

Here, $P_{\mathrm{sys},i}$ is a monotonic increasing function of f, whereas Δt_i is a monotonic decreasing function of f, given that $\{P_{\mathrm{drop},i}, P_{\mathrm{back}}, V, \gamma, \xi_i, f_{\mathrm{k}}, \beta\} \in \mathbb{R}^+$. Note that P_{back} and cc_{b} are scaling factors of $E_{\mathrm{sys},i}$ and that this implies that the energy consumed during the execution of a piece of code is linearly dependent on its code complexity and background power demands. Moreover, this also implies that compiler optimization techniques that target code size optimization will directly also lead to an improved energy profile of the code [8]. On the other hand, a microprocessor can also reduce energy consumption by parallelizing code execution, increasing power demands but reducing execution time. Similar observations between the interaction of energy and power consumption were made by Valluri and John [9].

At this stage, we only apparently observe an hyperbolic relation between energy and frequency. We have to take into account the relationship between the voltage and the frequency to find a convex analytical relationship between E and f. Such convexity is of interest as there would exist a microprocessor configuration that minimizes the energy consumption for that particular combination of $\{P_{\mathrm{drop},i}, P_{\mathrm{back}}, \gamma, \xi_i, f_{\mathrm{k}}, \beta\}$.

2.4 Voltage/Frequency Relationship

The following derivation regarding the energy/frequency relationship is similar to Yuki and Rajopadhye [10]; however, different frequency and voltage relationships are used, mainly more contemporary, and the leakage current is scaled more realistically. Note that P_{back} and P_{drop} can be arbitrarily large; their values are inherent to the computer system and independent of the microprocessor. In the remainder of this work it is also assumed that the temperature of the microprocessor remains constant unless otherwise noted. In practice it was shown by De Vogeleer [7] that the microprocessor's power requirements show a strong exponential relation with the temperature. The non-linear temperature effects complicate the microprocessor's temporal power demands considerably. The temperature has, however, a small impact on the convex behavior of the Energy/Frequency Convexity Rule [4]. Therefore, we omit the temperature effects on the Energy/Frequency Convexity Rule further on.

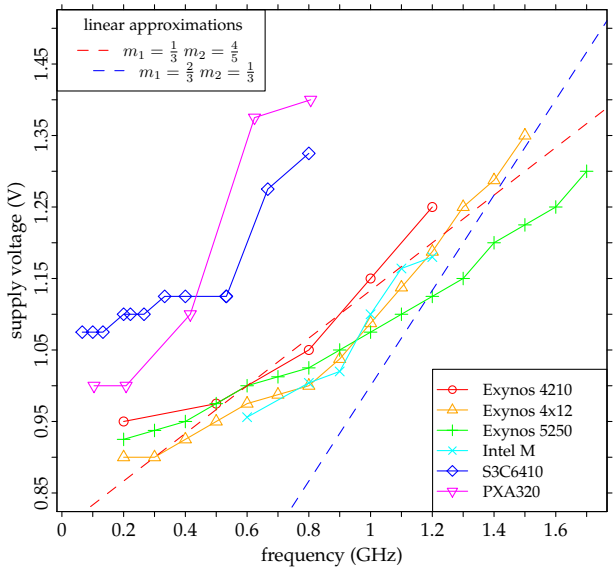


Fig. 1. Frequency/voltage relationships of multiple modern and vintage application microprocessors, as found in the Linux kernel. Two dashed linear curves are drawn: $V = m_1 f + m_2$; the red is fitted on the depicted data of the three Exynos microprocessors; the blue is borrowed from Yuki and Rajopadhyes [10].

For modern microprocessors, the frequency f and supply voltage V are approximately linearly related as shown in Figure 1. It is to be noted that the S3C6410 and the PXA320 are fairly outdated microprocessors and their low performance is visible; the Exynos series and the Intel M are more recent microprocessors designed for embedded multimedia applications, e.g., smartphones and tablets. The exact relationship between the voltage and frequency is dependent on the physical abilities of the microprocessor's internals, but also on the capability of the microprocessor's voltage and frequency regulator to scale the voltage and frequency on-demand. When the frequency of a microprocessor is ramped up, the transistors inside need to switch faster to meet timing and delay constraints. As subparts of transistors are essentially very small capacitors as well; a finite time is required to switch the transistor from one state to another. Thus if stringent timing delays need to be met, the microprocessor voltage needs be increased accordingly. The higher voltage supply will decrease the transistors' transition time and capacitors' charging time. This translates in a positive slope of the frequency/voltage relationship.

An affine transformation between voltage and frequency is expressed as follows:

$$V = m_1 f + m_2, \tag{6}$$

where m_1 and m_2 are positive regression coefficients. Figure 1 shows the voltage and frequency relationship for several microprocessors. The values $m_1 = \frac{2}{3}$ and $m_2 = \frac{1}{3}$, for the dashed blue line in Figure 1, are motivated to be adequate for high-performance microprocessors based on theoretical values [10]. Here, the values $m_1 = \frac{1}{3}$ and $m_2 = \frac{4}{5}$ are shown to better represent the voltage/frequency relationship for microprocessors for embedded applications.

These values are approximates of a linear fit on the combined data of the Exynos microprocessors.

Henceforth, the microprocessor's *default clock frequency window* ($\mathcal{F}_{\text{cpu}}$) is defined as the clock frequency range bounded by the minimum and maximum clock frequency of the microprocessor:

$$\mathcal{F}_{\text{cpu}} = f_{\min} \leq f_{\text{cpu}} \leq f_{\max}. \tag{7}$$

We have seen in Section 2.2 that $f \leq f_{\text{k}}$. Hence, the *exploitable clock frequency window* ($\mathcal{F}_{\text{epx}}$) is defined as the frequency range with an upper bound characterized by the microprocessor's maximum frequency $f_{\max}$, and the lower bound defined by the largest of the microprocessor's minimum frequency $f_{\min}$ and f_{k}:

$$\mathcal{F}_{\text{epx}} = \max(f_{\min}, f_{\text{k}}) \leq f_{\text{cpu}} \leq f_{\max}. \tag{8}$$

It is the exploitable clock frequency window that is open for energy optimization via clock frequency scaling.

2.5 Optimal Microprocessor Clock Frequency f_{opt}

The power model independent of V is obtained by inserting Equation 6 in the definition of P_{cpu}:

$$\begin{aligned} P_{\text{cpu}} &= (1+\gamma V)\xi f V^2 \\ &= af^4 + bf^3 + cf^2 + df, \end{aligned} \tag{9}$$

where $a = \gamma\xi m_1^3$, $b = m_1^2\xi(1+3\gamma m_2)$, $c = m_1 m_2 \xi(3\gamma m_2 + 2)$, and $d = m_2^2\xi(\gamma m_2 + 1)$. This power formulation can then be inserted in the energy consumption model E_{sys} of Equation 5.

For further analysis the normalized energy consumption E_{n} for code size and background power-independent analysis is introduced. The normalized energy consumption is defined as

$$E_{\text{n}} = \frac{E_{\text{sys}} - P_{\text{back}}\Delta t}{cc_{\text{b}}}. \tag{10}$$

Normalizing the energy consumption E_{sys} has no effect whatsoever on its tentative convex properties as cc_{b} and P_{back} merely induce an affine transformation of E_{n} without rotation. P_{back} has an effect on the convex properties. P_{back} should however not be part E_{n} as this power component will be present in the system regardless of what the microprocessor is doing. As a consequence, P_{back} should not influence optimal operating settings of the microprocessor.

The energy function in Equation 10 is called strictly convex over the exploitable clock frequency window if and only if (iff)

$$\begin{aligned} &\forall f_1 \neq f_2 \in \mathcal{F}_{\text{cpu}}, \forall t \in (0,1): \\ &\quad E_{\text{n}}(tf_1 + (1-t)f_2) < tE_{\text{n}}(f_1) + (1-t)E_{\text{n}}(f_2). \end{aligned} \tag{11}$$

In other words, if E_{sys} is strictly convex, then E_{sys} possesses no more than one minimum in the exploitable frequency window. If the minimum of E_{sys} is not within the microprocessor's boundaries, then the minimum f_{opt} can be found via the first derivative of E_{n}, while its second derivative must remain positive:

$$\left(\frac{\partial E_{\text{n}}}{\partial f}\right)_{f=f_{\text{opt}}} = 0 \quad \text{and} \quad \frac{\partial^2 E_{\text{n}}}{\partial f^2} > 0. \tag{12}$$

To simplify the derivative calculation for Equation 5, E_{n} is split into a polynomial and non-polynomial part, namely E_{n}^{A} and E_{n}^{B}:

$$E_{\text{n}} = E_{\text{n}}^{\text{A}} + E_{\text{n}}^{\text{B}}$$

$$E_{\text{n}}^{\text{A}} = (af^4 + bf^3 + cf^2 + df + P_{\text{drop},i}) \cdot \beta \tag{13a}$$

$$E_{\text{n}}^{\text{B}} = (af^4 + bf^3 + cf^2 + df + P_{\text{drop},i}) \cdot \frac{1}{f - f_{\text{k}}}, \tag{13b}$$

The respective derivatives are then as follows:

$$\frac{\partial E_{\text{n}}^{\text{A}}}{\partial f} = (4af^3 + 3bf^2 + 2cf + d) \cdot \beta \tag{14a}$$

$$\frac{\partial E_{\text{n}}^{\text{B}}}{\partial f} = \frac{3af^4 + (2b - 4af_{\text{k}})f^3 + (c - 3bf_{\text{k}})f^2}{(f - f_{\text{k}})^2} - \frac{2cf_{\text{k}}f + P_{\text{back}} + df_{\text{k}}}{(f - f_{\text{k}})^2} \tag{14b}$$

$$\frac{\partial^2 E_{\text{n}}^{\text{A}}}{\partial f^2} = (12af^2 + 6bf + 2c) \cdot \beta \tag{14c}$$

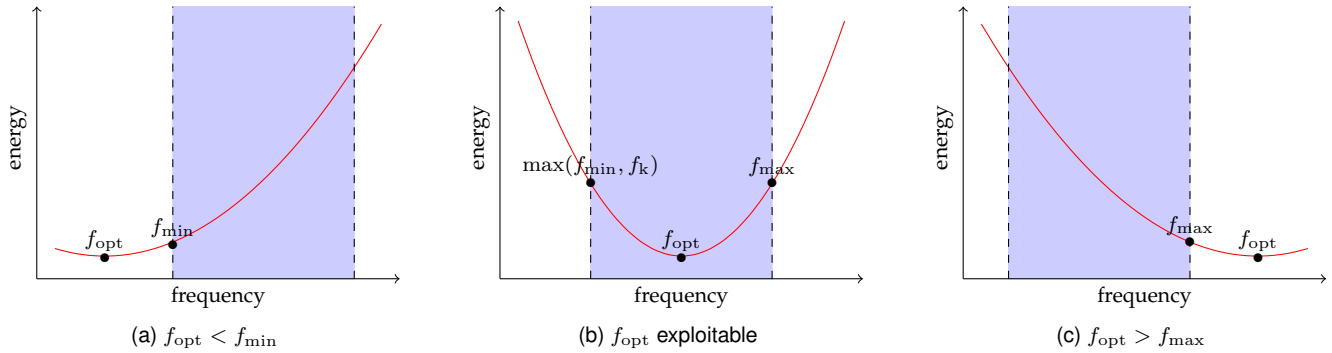
$$\frac{\partial^2 E_{\text{n}}^{\text{B}}}{\partial f^2} = \frac{6af^4 + (2b - 16af_{\text{k}})f^3 + \left(12af_{\text{k}}^2 - 6bf_{\text{k}}\right)f^2}{(f - f_{\text{k}})^3} + \frac{6bf_{\text{k}}^2 f + 2(P_{\text{back}} + cf_{\text{k}}^2 + df_{\text{k}})}{(f - f_{\text{k}})^3}. \tag{14d}$$

These equations will be used further on in Section 4 on parameters sensitivity analyses and are also the base for the next section's approximate solutions.

Convex properties can be observed for E_{n}. For $f \to^+ f_{\text{k}}$, E_{n}^{A} will approach $\beta P_{\text{drop},i}$, whereas E_{n}^{B} is amplified, and tends to positive infinity because of the presence of $f - f_{\text{k}}$ in the denominator. When $\frac{f}{2} < f_{\text{k}}$, the system is spending more energy in overhead than in the actual program, as the overhead has priority over the program. In the limit, E_{n} goes to infinity at f_{k}. At this point the system is overloaded and is not reactive anymore from the point of view of cc_{b}. For $f \to \infty$, it is E_{n}^{A} that inflates whereas E_{n}^{B} approaches zero. In other words, for the smaller clock frequencies, by virtue of the increased execution time, more energy due to leakage currents needs to be accounted for. The execution time for large frequencies are dramatically lower, but the dynamic power consumption of the microprocessor increases cubically and the leakage currents increase quartically with clock frequency. As a result, the convex minimum of the energy function, at the optimal frequency f_{opt}, is the point where a balance is found between the consequences of the inflated execution time and the total power demands of the microprocessor.

Given an energy/frequency convex behavior, three classes of microprocessor configurations can be distinguished, as shown in Figure 2. When the optimal clock frequency f_{opt} is left of the default clock frequency window ($f_{\text{opt}} < f_{\min}$), setting the clock frequency at $f_{\min}$ yields the best energy gains; if $\max(f_{\min}, f_{\text{k}}) < f_{\text{opt}} < f_{\max}$ then chasing f_{opt} will earn the best energy efficiency; and when $f_{\text{opt}} > f_{\max}$, then the race-to-halt[1] energy optimization technique is shown to be most effective. It was noted by Rizvandi [11] that under certain circumstances

1. The energy optimization technique *race-to-halt* runs the microprocessor at full speed until all tasks are completed; then the microprocessor is put in a low-power mode.



(a) $f_{\text{opt}} < f_{\text{min}}$ (b) f_{opt} exploitable (c) $f_{\text{opt}} > f_{\text{max}}$

Fig. 2. The location of the optimal frequency f_{opt} w.r.t. default clock frequency window (blue) is an indication of which energy optimization technique is most effective: (a) when f_{opt} is left of the exploitable clock frequency window ($f_{\text{opt}} < f_{\text{min}}$), one should set the clock frequency as low as possible; (b) if $\max(f_{\text{min}}, f_{\text{k}}) < f_{\text{opt}} < f_{\text{max}}$ then chasing f_{opt} will yield the best energy efficiency; (c) when $f_{\text{opt}} > f_{\text{max}}$, then the race-to-halt energy optimization technique is most effective. Powerful microprocessors are most likely to fall in the category (c), e.g. DGEMM 8C in Figure 7c, whereas low-power microcomputers are more likely to be in category (b), e.g. TI C62 in Figure 7f.

it can be more efficient, in terms of energy consumption, to have a binary frequency scheme, including the maximum and minimum clock frequency, rather than scaling the clock frequency through the whole frequency space. The presented performance-oriented work, and also the user-oriented work of Seeker *et al.* [3], suggest that this is, in fact, not the case. f_{opt} may assume any frequency within the default clock frequency window, and may fluctuate throughout the code execution depending on the kind of operations scheduled.

2.6 Approximate Optimal Clock Frequency f_{opt}

The power model (Equation 3) in the energy consumption formulation of Equation 5 is of the fourth order. When the fourth-order power equation can be adequately approximated with a quadratic polynomial, the derivations can be simplified somewhat. The power consumption P_{sys} of the system can then be represented as:

$$P_{\text{sys}} = af^4 + bf^3 + cf^2 + df \approx kf^2 + lf + m, \tag{15}$$

and accordingly the energy consumption of the system becomes

$$E_{\text{n}} = (kf^2 + lf + m + P_{\text{drop},i}) \cdot \Delta t. \tag{16}$$

$k \in \mathbb{R}_0^+$, though $\{l, m\} \in \mathbb{R}$. The first and second derivatives of the normalized energy consumption are then as follows:

$$\frac{\partial E_{\text{n}}}{\partial f} = \beta(2kf + l) - \frac{f_{\text{k}}(2kf + l) - kf^2 + m + P_{\text{drop},i}}{(f - f_{\text{k}})^2}, \tag{17a}$$

$$\frac{\partial^2 E_{\text{n}}}{\partial f^2} = 2k\beta + \frac{2(f_{\text{k}}^2 k + f_{\text{k}} l + m + P_{\text{drop},i})}{(f - f_{\text{k}})^3}. \tag{17b}$$

There exists a convex minimum if $\frac{\partial E_{\text{n}}}{\partial f}$ has a root and $\frac{\partial^2 E_{\text{n}}}{\partial f^2}$ is a monotonous increasing function. In other words:

$$\begin{aligned} 0 &= 2k\beta f^3 + (k + \beta(l - 4f_{\text{k}}k))f^2 + 2f_{\text{k}}\beta(f_{\text{k}}k - l)f \\ &\quad -2f_{\text{k}}kf - (m + P_{\text{drop},i} + f_{\text{k}}l(1 - \beta f_{\text{k}})) \\ 2k\beta &\geq -2\frac{f_{\text{k}}^2 k + f_{\text{k}} l + m + P_{\text{drop},i}}{(f - f_{\text{k}})^3}. \end{aligned} \tag{18}$$

The solution to Equation 18 is the frequency that minimizes energy consumption. Via Ferarri's solution [12] for the calculation of the roots of a third order polynomial, the optimal frequency can be determined analytically. Yet, the analytical formulation to calculate the roots of a cubic polynomial is still elaborate. Let's assume some further simplifications. For $\beta = 0$, one gets that

$$\begin{aligned} f_{\text{opt}} &= f_{\text{k}} + \frac{\sqrt{2k^2 f_{\text{k}}^2 + 2k(m + P_{\text{drop},i} + f_{\text{k}} l)}}{k} \\ 0 &\leq 2\frac{f_{\text{k}}^2 k + f_{\text{k}} l + m + P_{\text{drop},i}}{(f - f_{\text{k}})^3}. \end{aligned} \tag{19}$$

If all parameters are elements of $\mathbb{R}^+$, the latter inequality holds whenever $f_{\text{k}} < f$. Additionally, for $f_{\text{k}} = 0$, one obtains

$$\begin{aligned} f_{\text{opt}} &= \sqrt{\frac{m + P_{\text{drop},i}}{k}} \\ 0 &\leq \frac{2(m + P_{\text{drop},i})}{f^3}, \end{aligned} \tag{20}$$

which is only valid for $-P_{\text{drop},i} < m$. These simplified models for $\beta = f_{\text{k}} = 0$ may be used when the context allows for, i.e., when cc_{b} is executed without any interruption. For example, from practical experience and in the literature, f_{k} is often observed to be close to zero in a multi-core context. β may vary considerably for different applications and should be assessed before deeming insignificant.

3 Experimental Results

In this section experimentally-obtained power and execution time measurement traces are presented and used as a reference to study the Energy/Frequency Convexity Rule in the next section.

3.1 Platform and Benchmark Description

A Samsung Galaxy S2, sporting an ARM Cortex A9 dual-core microprocessor, was used as testbed. The A9 uses clock frequency ranges from 0.2 GHz to 1.6 GHz in steps of 100 MHz. The Gold-Rader implementation of the bit-reverse algorithm was used as benchmark; it is part of the ubiquitous Fast Fourier Transformation (FFT) algorithm, in which

TABLE 1
Benchmark execution time model parameters: ξ, γ and P_{sys} as per Equation 3, and cc_{b}, f_{k}, β as per Equation 4 for running the Gold-Rader algorithm on the A9 microprocessor. These values were used for the fitted models in Figure 3b.

	GOLD-RADER – INPUT SIZE 2^N – A9					
N	6	8	10	12	14	16
cc_{b}	1.943	8.596	31.1	144.359	670.8	2918.837
f_{k}	0.134	0.129	0.137	0.13	0.13	0.129
β	-0.166	-0.167	-0.152	-0.202	-0.183	-0.182
ξ	0.101	0.108	0.134	0.137	0.44	0.011
γ	5.578	5.127	4.030	4.36	1.035	65.985
P_{sys}	0.480	0.480	0.477	0.469	0.394	0.407

it rearranges deterministically elements in an array. Besides the Gold-Rader algorithm, the BEEBS benchmark [13] was also run on an ODROID XU+E, featuring an Exynos 5240, while the execution time and power was measured. The measurement data of the Gold-Rader algorithm and BEEBS show large similarities. The Gold-Rader algorithm is chosen as a base for the expound in the sequel. More info on the BEEBS measurements can be found in De Vogeleer [4].

3.2 Execution Time and Power Consumption

Figure 3a shows the execution time of the Gold-Rader algorithm on the A9 microprocessor. Table 1 shows the fitted execution time parameters as per Equation 4. The fitted execution time model has a relative error such that 90 % of the errors are between 0.18 % and 7.36 % and shows a median of 3.12 % for the execution time traces.

Figure 3b shows the power profile of the Gold-Rader algorithm on the A9. All traces were recorded while the temperature of the hardware fluctuated. During the recording of the power traces the temperature of the testbed was artificially oscillated around 37°C and then the power samples at a temperature of 37°C were selected.

Table 1 also shows the fitted values for ξ, γ and P_{sys} as per Equation 3 for the A9. Discrete voltage/frequency pairs were used to fit the measured data as reported in Figure 1 for the Exynos 4210.

The fitted model parameters in Table 1 seem to be consistent for an input size up to 2^{12}. The fitted model parameters for larger input sizes seem to be much different. Note that array sizes up to 2^9 fit in the L1 cache, while sizes over 2^{18} are too big to fit in the L2 cache. Therefore external memory accesses and microprocessor slack time may influence the power of the microprocessor. Overall, the power variation of the different input sizes are not as large as what was observed for the case of the execution time. The magnitude of the power of all traces are all of the same order, whereas for the execution time it may differ by multiple orders.

As observed from Figure 3b the power model fits well on the experimental data. The fitting errors for the A9 are between 0.07 % and 3.18 % with a median of 0.86 %. The fitted model for the A9 in Figure 3b for $f = 1.5\,\text{GHz}$ seems to deviate persistently from the measured data. This could be due to a slightly higher supply voltage at 1.5 GHz than reported in Figure 1 for the Exynos 4210 microprocessor.

3.3 Energy Consumption

The estimated experimental energy consumptions are obtained by multiplying the power traces with the execution time traces for each frequency. This was done for both the experimental traces and the fitted power and execution time models. Figure 3c shows the energy consumption of the Gold-Rader algorithm on the A9 microprocessor. The fitted errors are the sum of the errors of the power and execution time traces separately. For the A9 traces a clear minimum energy consumption is observed between 500 MHz and 800 MHz.

4 SENSITIVITY OF THE CONVEXITY MODEL

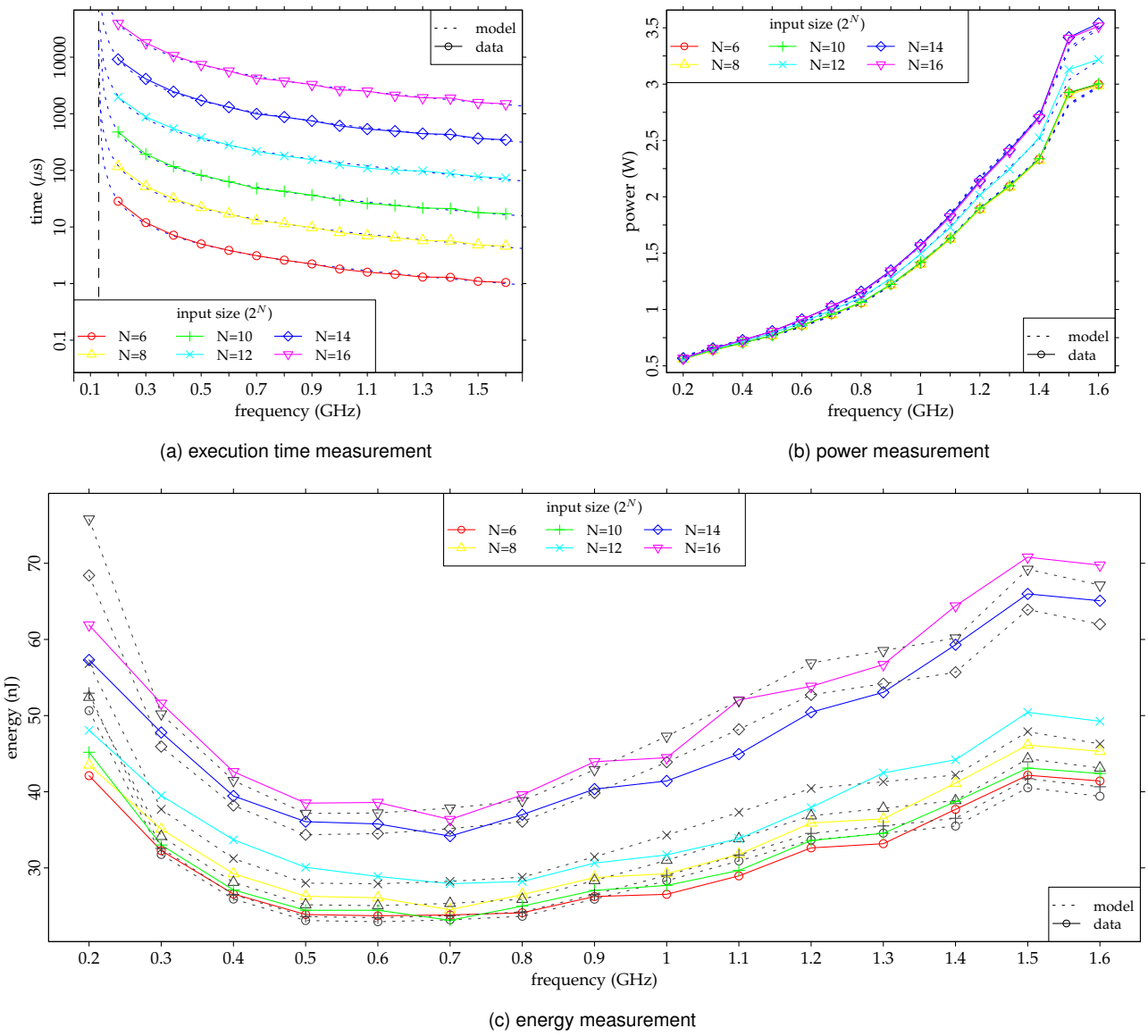
To analyze the behavior and parameter sensitivity of the convexity model of Equation 5, the Cortex A9 processor of the Exynos 4210 is used as reference use case, representative for embedded multimedia applications, e.g., smartphones [2]. The following values were used, based on the measurements presented in the previous section: m_1 = 0.330 [V/f], m_2 = 0.808 [V], β = 0 [s], γ = 3.137 [V^{-1}], f_{k} = 0.130 [GHz], $\xi_{\max}$ = 0.181 [W/(GHz·V^2)], $\xi_{\min}$= 0.155 [W/(GHz·V^2)], P_{drop} = 0 [W]. The microprocessor's clock frequency starts at 200 MHz and goes to 1.6 GHz and ξ is a parameter that describes the power profile of an application. The values for β, f_{k}, γ and ξ were defined via fitting as presented in the previous sections. The microprocessor's clock frequency is also considered a continuous variable from here on. In reality the clock frequency is limited to a discrete set of values. However, for analytical purposes, not to mention the aesthetics of the graphs, the clock frequency is deemed continuous.

In the next sections we will look at how time thieves and OOE impacts the convexity model. Time thieves are basically clock cycles lost to overhead, whereas OOE is an intelligent instruction execution scheme to minimize execution slack time.

4.1 What About Those Time Thieves?

When considering the execution time of a code sequence, f_{k} was previously defined as the number of clock cycles per time unit not available to the execution of the user code. These clock cycles are spent, for example, to handle microprocessor exceptions, or to execute operating system routine tasks. f_{k} can therefore be regarded as little *time thieves*. From a mathematical point of view, the presence of f_{k} in Equation 5 also introduces some complexity for derivations such as Equation 14. Bear in mind that the microprocessor's clock frequency f is always larger than f_{k}; otherwise the execution time is not defined. Consequently, $f_{\text{k}} < f_{\max}$ must be satisfied.

Figure 4 shows the sensitivity of f_{k} with regards to the optimal frequency f_{opt}, the microprocessor power ($P_{\text{cpu}} \propto \xi$), and the background power P_{back}. In the bottom plot it is seen that $f_{\text{opt}}(f_{\text{k}} = 0, P_{\text{back}} = 0.5) \approx 0.8\,\text{GHz}$. The optimal frequency increases for increasing values of f_{k} and hits the microprocessor's maximum frequency $f_{\max} = 1.6\,\text{GHz}$ around $f_{\text{k}} = 0.7\,\text{GHz}$. At this point, about 45 % ($\approx 0.7/1.6$) of the clock cycles would not be available to the code sequence. Furthermore, it is observed that $f_{\text{opt}} > f_{\text{k}}$ always



(a) execution time measurement

(b) power measurement

(c) energy measurement

Fig. 3. Experimental data for the Cortex A9 microprocessor. The energy consumption of the benchmarks with different input sizes is shown for the Gold-Rader algorithm. The solid lines represent the measured data whereas the dotted lines is the product of the fitted power and execution time models from Figure 3a and Figure 3b, respectively.

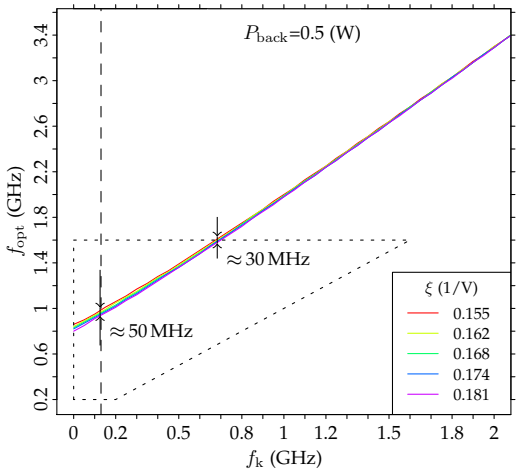
holds. The effect of the microprocessor's power demands on f_{opt} is fairly small, expressed by the ξ parameter. A 30 MHz to 50 MHz difference in f_{opt} is observed between the minimum and maximum microprocessor's power usage as ξ varies between 0.155 V^{-1} and 0.181 V^{-1} (see Figure 4a).

The background power usage P_{back} has a bigger impact on f_{opt} than ξ. For $P_{\text{back}} = 0$, f_{opt} even drops below the minimum operation frequency of the microprocessor. Increasing P_{back} inflates f_{opt}. For $f_{\text{k}} = 0$ and $P_{\text{back}} \approx 2.5\,\text{W}$ the optimal frequency already surpasses $f_{\max}$. For a typical value of f_{k} (130 MHz), an increase in f_{opt} is observed for increasing values of P_{back}; yet, the increase becomes smaller for larger values of P_{back}. The average difference between $f_{\text{opt}}(f_{\text{k}} = 0)$ and $f_{\text{opt}}(f_{\text{k}} = 0.13)$, within the microprocessor's clock frequency range, is approximately 100 MHz.

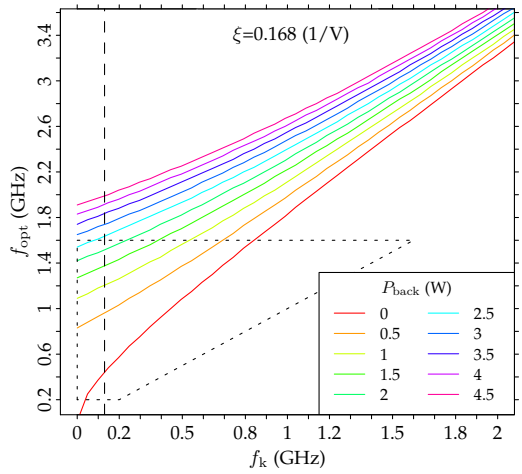
In the rest of this section it will be assumed for simplicity that $f_{\text{k}} \ll f$ unless otherwise stated. For a more realistic estimate of f_{opt}, in case f_{k} is not negligible, it was observed from the graphs that adding 100 MHz to f_{opt} is a reasonable assumption.

4.2 Absence of Time Thieves

It is not unthinkable that, in particular contexts, f_{k} is indeed negligibly small compared to f: $f_{\text{k}} \ll f$. For example, such occasions may occur when the clock frequency microprocessor is reasonably fast, or the code sequence of concern is running only on one of the available cores of a multi-core microprocessor without interruption. Assuming f_{k} negligible considerably simplifies Equation 14. For $\max(f_{\min}, f_{\text{k}}) < f_{\text{opt}} < f_{\max}$, E_{n} was said to be strictly convex iff there exists only one point in the exploitable clock frequency window for which $\frac{\partial E_{\text{n}}}{\partial f} = 0$ and $\frac{\partial^2 E_{\text{n}}}{\partial f^2} > 0$. Given the system of Equations 14, these two requirements

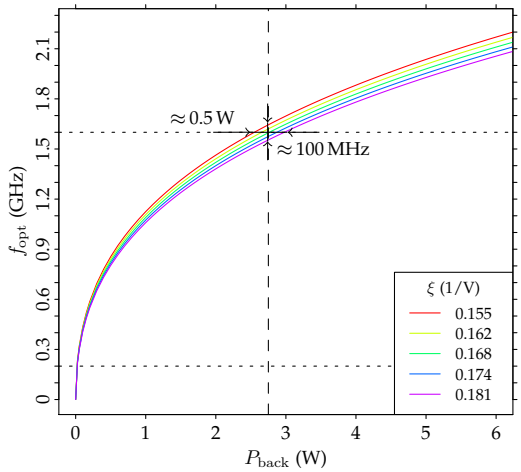


(a) $f_{\text{opt}}(f_{\text{k}},\xi)$

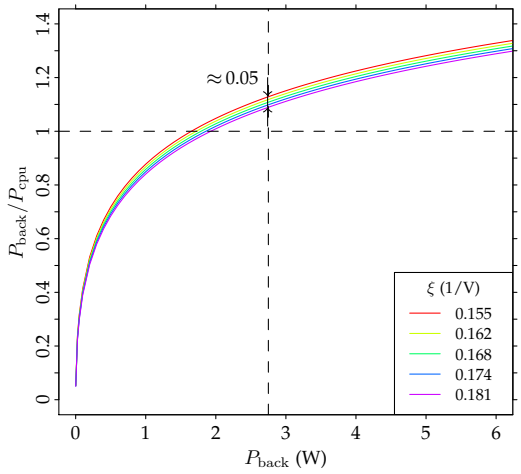


(b) $f_{\text{opt}}(f_{\text{k}},P_{\text{back}})$

Fig. 4. Optimal microprocessor frequency f_{opt} for variable levels of f_{k} in function of ξ, on the top, and P_{back}, on the bottom. A typical value for f_{k} is drawn at 0.13 GHz (dashed vertical line). The area encapsulated by the dotted line signals the microprocessor's exploitable clock frequency window: $\max(0.2\,\text{GHz}, f_{\text{k}}) \leq f \leq 1.6\,\text{GHz}$.



(a) $f_{\text{opt}}(P_{\text{back}},\xi)$



(b) $P_{\text{cpu}}/P_{\text{back}}$ ratio at f_{opt}

Fig. 5. Optimal microprocessor frequency f_{opt} for variable background power consumption P_{back}. On the top f_{opt}, is shown for various microprocessor loads ξ. On the bottom, the ratio between the background power and the microprocessor power P_{cpu} at f_{opt} is shown. The area between the dotted lines signals the effective clock frequency window: $0.2\,\text{GHz} \leq f \leq 1.6\,\text{GHz}$.

translate, respectively, into:

$$\frac{P_{\text{back}}}{f_{\text{opt}}^2} = 4a\beta f_{\text{opt}}^3 + 3(a+b\beta)f_{\text{opt}}^2 + 2(b+c\beta)f_{\text{opt}} + (d\beta+c), \quad (21)$$

$$0 < 12a\beta f_{\text{opt}}^2 + 6(a+b\beta)f_{\text{opt}} + 2(b+c\beta) + 2\frac{P_{\text{back}}}{f_{\text{opt}}^3}. \quad (22)$$

Recall that for all constants in this system of equations: $\{a,b,c,d,\beta\} \in \mathbb{R}^+$. Thus the requirement in Equation 22 is satisfied by default as the right-hand side will never be negative. Accordingly, the root requirement of Equation 21 is also satisfiable. It is immediately clear that the background power demands P_{back} directly controls the optimal frequency f_{opt}. The constants $\{a,b,c,d\}$ describe the microprocessor's power usage whereas P_{back} describes the power demands of everything in the computer system besides the microprocessor. For systems with a large P_{back}, e.g., servers or desktop computers, f_{opt} will therefore be higher than for systems with a low P_{back}, e.g., wireless sensors. Moreover, f_{opt} may be so high that it is larger than the maximum microprocessor's clock frequency.

Figure 5 shows the optimal frequency for a variable background power consumption P_{back} and microprocessor loads ξ. Also, the ratio between the microprocessor P_{cpu} and the background P_{back} power consumption is given. The area encapsulated by the dotted line signals the operating range of the microprocessor. For the microprocessor to be able to exploit the minimum-energy operation frequency, the background power consumption needs to be between 0.02 W and about 2.75 W, depending on the exact microprocessor load. The influence of the different microprocessor loads on P_{back} is not significant; at 1.6 GHz there is a 0.5 W difference between P_{back} for ξ_{min} and ξ_{max}. If P_{back} is larger than 2.75 W, it is advised to run the microprocessor at the maximum clock frequency to minimize energy

consumption. Under such conditions, the energy optimization technique known as *race-to-halt* is a good strategy. This was also Yuki and Rajopadhye's [10] main conclusion while studying high-performance computers. The optimal frequency f_{opt} surpasses the microprocessor's maximum frequency roughly around the point where the background power demands become larger than the microprocessor's power usage. Battery-powered electronic systems such as embedded systems, wireless sensors or smartphones aim at minimizing their background power demands, which thus increases the feasibility of f_{opt} exploitation. For more powerful computers, however, such as servers, the optimal frequency will be very likely out of reach of the microprocessor's capabilities: $f_{\text{opt}} > f_{\max}$. For example, Seo *et al.* [14] claim that Dynamic Voltage and Frequency Scaling (DVFS) in general hardly improves the energy efficiency of mobile multimedia electronics. The testbed power measurements of their embedded system show, however, that their P_{cpu} to P_{back} ratio is smaller than 1 to 18, and their m_1 is very small. For their specific testbed, f_{opt} is very likely larger than $f_{\max}$, and race-to-halt should indeed be most benificial when aiming for energy savings.

4.3 Out-of-Order Execution

Out-of-order execution (OOE) is parametrized via $\beta \in [0, \infty[$ in Equation 4: $\beta = 0$ when OOE is perfectly able to cover the time during external memory accesses with data-independent code execution; otherwise β is larger than 0. The system's normalized energy consumption, assuming $f_{\text{k}} \approx 0$, is given by:

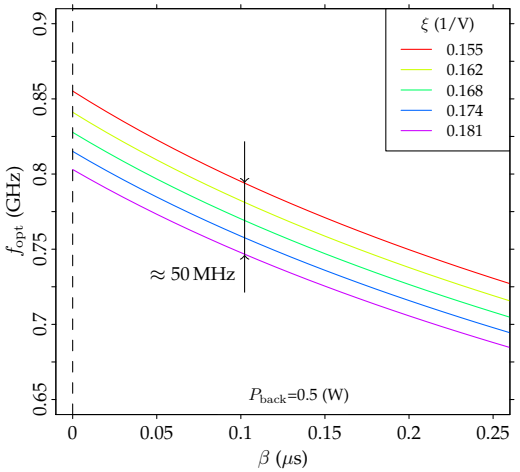
$$E_{\text{n}} = (af^4 + bf^3 + cf^2 + df + P_{\text{back}}) \cdot \left(\frac{1}{f} + \beta\right).$$

Its requirements for convexity are defined the same as for the case where time thieves are absent, given by Equation 21 and 22. It can be observed that for $\beta = 0$ the most left-hand term in Equation 21 becomes zero, resulting in an increased f_{opt} for the equality to be satisfied. Similarly, the larger β, the more f_{opt} needs to decrease for the inequality of Equation 22 to hold. Figure 6 shows the sensitivity of the β parameter on the optimal frequency f_{opt}. Indeed, from the figure, it is observed that f_{opt} decreases for increasing β. Moreover, f_{opt} changes about 100 MHz over a 0 to 0.25 μs β range for medium levels of P_{back}. The larger P_{back}, the larger the spread in f_{opt} for variable β. For P_{back} over 4 W, the f_{opt} spread between $\beta = 0$ and $\beta = 0.25$ increases to more than 200 MHz.

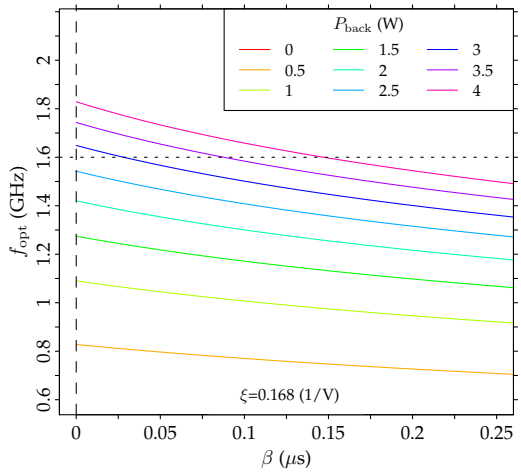
In theory, β can be frequency-dependent as well. That is, the memory clock frequency can be scaled along with the microprocessor's frequency, this to ensure the timely delivery of data in the microprocessor registries and caches. β in such a case would not be constant over f. Here, it was assumed that the microprocessor's clock frequency, once set at f_{opt}, doesn't change over time. Another common approach to save energy is to have a variable clock frequency to minimize OOE slack-time and also energy consumption.

5 State of the Art

In the previous sections, it is shown that the energy consumption of a microprocessor shows convex properties with regard to its clock frequency. The convex energy consumption curve has been mentioned before several times in the literature. A sensitivity study of the convexity model, as presented here, has not been reported before. A series of papers, approaching the problem from a chip point of view, without the consideration of software, have shown the energy consumption with respect to Dynamic Voltage and Frequency Scaling (DVFS) [15], [16], [17], [18]. The literature puts forward some motivation for the energy consumption's convexity, but rarely provides analytical frameworks based on physical explanations. For example, Senn *et al.* [19] and Austin and Wright [20] provide a heuristic model. Other studies, e.g., Hager *et al.* [21] and Freeh *et al.* [22], discuss what the consequences are of said behavior and how to exploit them, from a high-level point of view. Other researchers have also shown energy measurements under DVFS processes but no convexity is shown by the measurements, e.g., Sinha and Chandrakasan [23], and Šimunić *et al.* [24], who are not running their benchmarks on top of an Operating System (OS). Authors, such as Austin and



(a) $f_{\text{opt}}(\beta,\xi)$



(b) $f_{\text{opt}}(\beta,P_{\text{back}})$

Fig. 6. Optimal microprocessor frequency f_{opt} for variable levels of β in function of ξ, on the top, and P_{back}, on the bottom. The area below the horizontal dotted line signals the microprocessor's default clock frequency window ($0.2\,\text{GHz} \leq f \leq 1.6\,\text{GHz}$).

Wright [20] and Snowdon [15], [16], have shown more specifically that for applications with certain behavioral patterns no energy convexity is observed. However, the energy consumption model presented in our work can explain such behavior.

In the Very-Large-Scale Integration (VLSI) design domain, voltage scaling has also been discussed but usually for a fixed frequency [25], [26], [27]. The aim of the voltage scaling is to find a minimum energy operation point where the digital circuit yields the correct output. The major trade-off is between increased circuit latency and leakage power, and decreasing dynamic power. This trade-off also yields a convex energy consumption curve, but for a fixed frequency. In this paper, however, the combined effect of voltage/frequency scaling is of interest.

There are some works that cover the energy/frequency convexity properties in a more analytical framework. Figure 7 shows excerpts of convex energy graphs provided by the cited works. Yuki and Rajopadhye [10] explored the particular case of energy consumption of high-performance computers in the context of compiler optimization and optimal frequency conditions of the microprocessor. One of their conclusions is that for power-hungry systems the *race-to-halt* energy optimization technique is more effective than DVFS. Hager *et al.* [21], on the other hand, showed that race-to-halt is not always the most effective strategy in a multi-core context with bandwidth-bound codes. The authors studied the energy consumption of modern multi-core chips via simple machine models and showed how to minimize the energy consumption with respect to the number of cores, serial code performance, and clock frequency. Austin and Wright [20] examined the energy consumption of micro-benchmarks and applications on a Cray CX30 super computer system. The authors developed a simple linear heuristic energy model. They also stressed that the frequency/energy minimum is application-specific. Cho and Chang [28] assessed the optimal frequency conditions for a microprocessor in conjunction with a memory. Their resulting model is fairly complex; yet the authors show the feasibility of a microprocessor's optimal frequency conditions in conjecture with a memory system. Cho and Melhelm [29] produced a convex model derived from Amdahl's law and extended with the notion of energy. The authors use a simplifying assumption for the representation of power and execution time. They show via their model that there is a certain clock frequency range that yields both energy and speed improvements. Similarly, Rizvandi *et al.* [30] devised a convex model but, just as Cho and Melhelm, simplified representations of power and execution time were assumed. Vasilaki [31] showed experimental evidence for a convex energy curve in relation to the microprocessor's clock frequency for almost all individual instructions of the ARM Cortex A7. No theoretical framework is provided by Vasilaki, however, to backup these findings analytically.

From an experimental perspective, Halimi *et al.* [32] claim to save up to 39 % of energy, and Qiu *et al.* [33] advertise an energy gain of 25 %, by adjusting the microprocessor's clock frequency via an experimental algorithm with predefined user or application constraints. Although no theoretical framework was provided by the authors about the energy/frequency convexity, their algorithm is essentially chasing the convex minimum. Senn *et al.* [19] showed also convex energy/frequency curves, based on a simplified system model, for their TI C55, C62, C64, and C67 platforms.

Applications of the work presented in this paper focuses on embedded systems, in contrast with Yuki and Rajopadhye's, Hager *et al.* and Austin and Wright's work, which is dedicated to more powerful computer architectures. The sensitivity of the parameters that constitute the energy consumption equation are also analyzed via both an analytical approach and via experimental data, the former fitted with data from the latter. The convex energy model presented here is, in contrast with the mentioned works, more extensive, which allows for a more realistic modeling. For example, temperature has not been a subject of interest and a sensitivity analysis of parameters has also not been carried out in any of the referenced works.

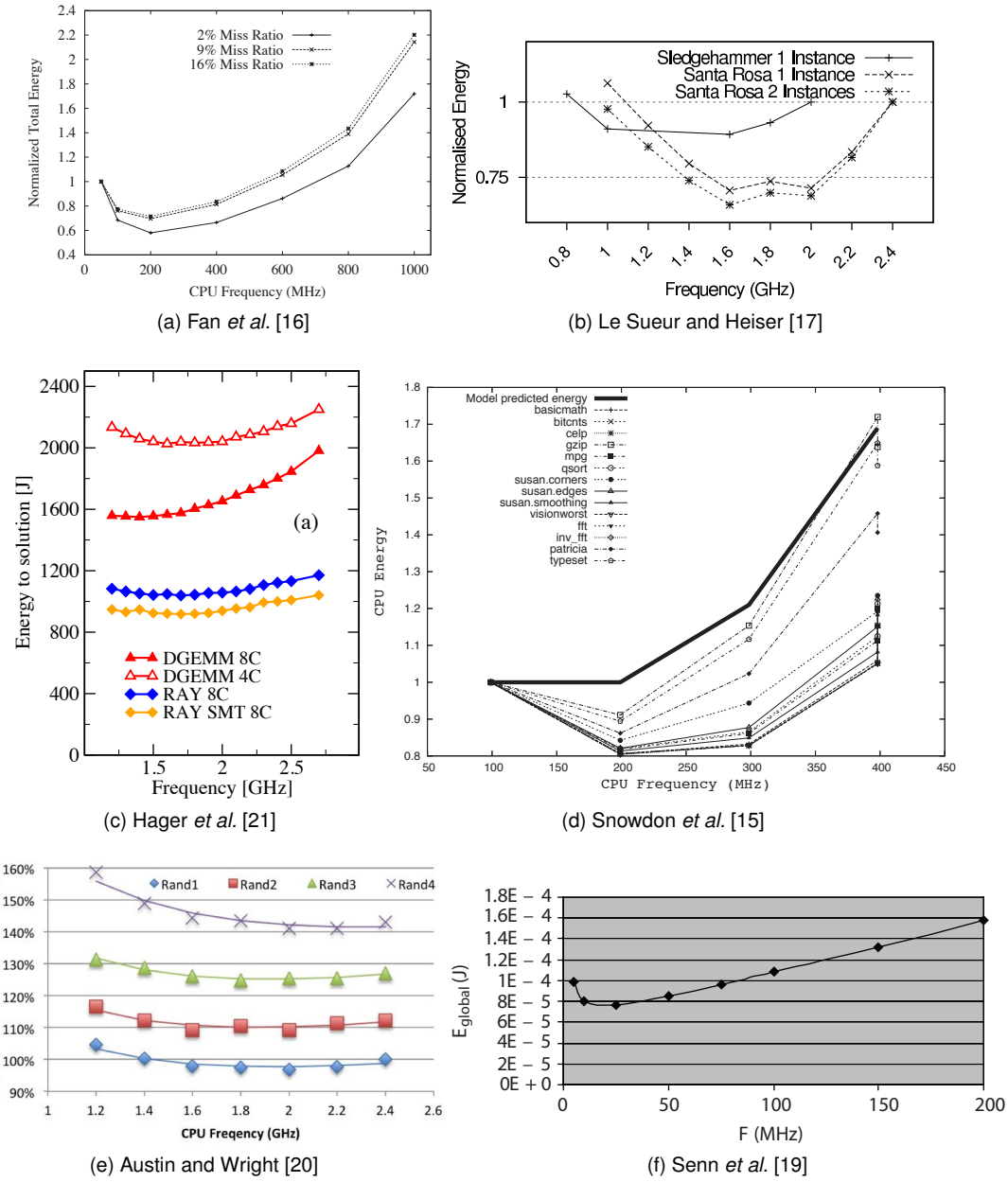
6 Conclusion

In this paper we developed and analyzed the energy consumption equation of a microprocessor operating in a computer system with other components. An analytical analysis, along with numerical simulation and measurement data, was used to study the behavior and sensitivity of its parameters. It was shown through an analytical framework, measurements, and literature review that the energy consumption curve shows convex properties with regard to the clock frequency of the microprocessor. The convex energy minimum is the point with a given clock frequency f_{opt} where the computer system consumes the minimum amount of energy while executing a code sequence.

The energy saving gained by running at the optimal clock frequency is a trade-off with the performance of the system, in terms of execution time. For applications requiring human interaction, it has been shown by Seeker *et al.* [3], however, that the clock frequency can be scaled down considerably without affecting the user's experience. More generally, this kind of energy savings can be obtained for code sequences where a limited slowdown can be tolerated and time is not critical. For example, such slowdowns could be applied to code sequences, in multithreaded programs that are not on the critical path [34].

The existence of the energy/frequency convexity property was further confirmed via experimental measurement traces of multimedia microprocessors commonly used for embedded system applications. The main conclusions of the analysis are:

- Energy/frequency convexity occurs always, but, to exploit the convex minimum, f_{opt} should be within the exploitable clock frequency window;
- The background power requirement (P_{back}) is the parameter that influences the optimal frequency the most; the larger the background power demands, the larger the optimal clock frequency: when P_{back} equals P_{cpu}, f_{opt} will be close to the maximum microprocessor clock frequency;
- An application's power profile (ξ) has a minimal effect on the optimal frequency, mostly because the variations in power profiles are fairly small in the



(a) Fan *et al.* [16]

(b) Le Sueur and Heiser [17]

(c) Hager *et al.* [21]

(d) Snowdon *et al.* [15]

(e) Austin and Wright [20]

(f) Senn *et al.* [19]

Fig. 7. Excerpts of energy/frequency measurements as found in the literature. Convex minimums are observable for the energy at a certain microprocessor clock frequency, depending on the microprocessor and architecture. In the sequel the behavior of this convex minimum is analyzed. All figures were originally published in the papers referenced in their respective captions.

experiments we ran, an average of 50 MHz in f_{opt} between the power profile's extremities;

- The number of instructions of a code sequence has no influence on the optimal clock frequency, following the energy consumption model, but does scale the energy consumption linearly on the premise that ξ has minimal effect at constant temperature;
- Application concurrency and clock cycle thieves (f_{k}) significantly affect the optimal frequency; the less clock cycles available to the applications, the larger the optimal clock frequency: on average for a 1 GHz increase in f_{k}, f_{opt} increases by 2 GHz;
- Microprocessor slack time (β), during off-chip operations, forces the optimal clock frequency down: 300 MHz for $0 < \beta < 0.25$ in the extreme case;
- The *race-to-halt* strategy is justified only when the optimal clock frequency is larger than the microprocessor's maximum frequency.

Given that P_{back} has a large effect on the optimal frequency f_{opt}, it was shown that a system with a P_{back} of the order of P_{cpu} and larger will have a f_{opt} likely outside the reach of the microprocessor's clock frequency range. Thus chasing the optimal clock frequency f_{opt} is especially beneficial for low-power systems, such as for embedded applications, as their P_{back} is much smaller than what would be expected for high-performance computer systems.

REFERENCES

[1] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the USENIX conference on USENIX*, Berkeley, CA, USA, 2010.

[2] K. De Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "The Energy/Frequency Convexity Rule: modeling and experimental validation on mobile devices," in *Proceedings of the 10th Conference on Parallel Processing and Applied Mathematics*. Springer Verlag, Sep. 2013.

[3] V. Seeker, P. Petoumenos, H. Leather, and B. Franke, "Measuring qoe of interactive workloads and characterising frequency governors on mobile devices," in *2014 IEEE International Symposium on Workload Characterization, IISWC 2014, Raleigh, NC, USA, October 26-28, 2014*, 2014, pp. 61–70.

[4] K. De Vogeleer, "The energy/frequency convexity rule of a program's energy consumption: Modeling, thermosensitivity and applications," Ph.D. dissertation, TELECOM ParisTech, 46 rue Barrault, 75013 Paris, France, 2015.

[5] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI design: a systems perspective*. Boston, USA: Addison-Wesley Longman Publishing Co., Inc., 1985.

[6] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[7] K. De Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "Modeling the temperature bias of power consumption for nanometer-scale cpus in application processors," in *XIVth International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, Jul. 2014, pp. 172–180.

[8] J. Pallister, S. Hollis, and J. Bennett, "The impact of different compiler options on energy consumption," in *First LPGPU Workshop on Power-Efficient GPU and Many-core Computing*, ser. PEGPUM '13. New York, NY, USA: ACM, 2013.

[9] M. Valluri and L. John, "Is compiling for performance == compiling for power?" in *Interaction between Compilers and Computer Architectures*, ser. The Springer International Series in Engineering and Computer Science, G. Lee and P.-C. Yew, Eds. Springer US, 2001, vol. 613, pp. 101–115.

[10] T. Yuki and S. Rajopadhye, "Folklore confirmed: Compiling for speed = compiling for energy," in *Proceedings of the 26th International Workshop on Languages and Compilers for Parallel Computing*, 2013.

[11] N. Rizvandi, J. Taheri, A. Zomaya, and Y. C. Lee, "Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010, pp. 388–397.

[12] G. Candido, "Le risoluzioni della equazione di quarto grado (Ferrari-Eulero-Lagrange)," *Period. Mat.*, vol. 21, no. 4, pp. 88–106, 1941.

[13] J. Pallister, S. Hollis, and J. Bennett, "BEEBS: Open benchmarks for energy measurements on embedded platforms," *CoRR*, vol. abs/1308.5174, 2013.

[14] Y. Seo, J. Kim, and E. Seo, "Effectiveness analysis of dvfs and dpm in mobile devices," *Journal of Computer Science and Technology*, vol. 27, no. 4, pp. 781–790, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11390-012-1264-6

[15] D. C. Snowdon, S. Ruocco, and G. Heiser, "Power management and dynamic voltage scaling: Myths and facts," in *2005 WS Power Aware Real-time Comput.*, New Jersey, USA, Sep. 2005.

[16] X. Fan, C. S. Ellis, and A. R. Lebeck, "The synergy between power-aware memory systems and processor voltage scaling," in *Proceedings of the Third international conference on Power - Aware Computer Systems*. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 164–179.

[17] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: the laws of diminishing returns," in *Proceedings of the 2010 international conference on Power aware computing and systems*, ser. HotPower'10, Berkeley, CA, USA, 2010, pp. 1–8.

[18] M. Seok, D. Jeon, C. Chakrabarti, D. Blaauw, and D. Sylvester, "Pipeline strategy for improving optimal energy efficiency in ultra-low voltage design," in *Proceedings of the 48th Design Automation Conference*, ser. DAC '11. New York, NY, USA: ACM, 2011, pp. 990–995.

[19] E. Senn, J. Laurent, N. Julien, and E. Martin, "Softexplorer: Estimating and optimizing the power and energy consumption of a C program for DSP applications," *EURASIP J. Adv. Sig. Proc.*, vol. 2005, no. 16, pp. 2641–2654, 2005.

[20] B. Austin and N. J. Wright, "Measurement and interpretation of microbenchmark and application energy use on the Cray XC30," in *Proceedings of the 2Nd International Workshop on Energy Efficient Supercomputing*, ser. E2SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 51–59.

[21] G. Hager, J. Treibig, J. Habich, and G. Wellein, "Exploring performance and power properties of modern multi-core chips via simple machine models," *Concurrency and Computation: Practice and Experience*, pp. n/a–n/a, 2013. [Online]. Available: http://dx.doi.org/10.1002/cpe.3180

[22] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 835–848, Jun. 2007.

[23] A. Sinha and A. P. Chandrakasan, "Jouletrack: a web based tool for software energy profiling," in *Proceedings of the 38th annual Design Automation Conference*, ser. DAC '01. New York, NY, USA: ACM, 2001, pp. 220–225.

[24] T. Simunic, L. Benini, and G. De Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," in *Design Automation Conference, 1999. Proceedings. 36th*, 1999, pp. 867 –872.

[25] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proceedings of the 41st annual Design Automation Conference*, ser. DAC '04. New York, NY, USA: ACM, 2004, pp. 868–873.

[26] B. Calhoun, A. Wang, N. Verma, and A. Chandrakasan, "Sub-threshold design: The challenges of minimizing circuit energy," in *Low Power Electronics and Design, 2006. Proceedings of the International Symposium on*, 2006, pp. 366–368.

[27] T. Kuroda, "Optimization and control of vdd and vth for low-power, high-speed CMOS design," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, Nov 2002, pp. 28–34.

[28] Y. Cho and N. Chang, "Energy-aware clock-frequency assignment in microprocessors and memory devices for dynamic voltage scaling," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 6, pp. 1030–1040, Jun. 2006.

[29] S. Cho and R. G. Melhem, "On the interplay of parallelization, program performance, and energy consumption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 342–353, 2010.

[30] N. B. Rizvandi, J. Taheri, and A. Y. Zomaya, "Some observations on optimal frequency selection in dvfs-based energy consumption minimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1154 – 1164, 2011.

[31] E. Vasilaki, "An instruction level energy characterization of arm processors," Institute of Computer Science (ICS), Foundation of Research and Technology Hellas (FORTH), Tech. Rep. FORTH-ICS/TR-450, 2015.

[32] J.-P. Halimi, B. Pradelle, A. Guermouche, N. Triquenaux, A. Laurent, J. Beyler, and W. Jalby, "Reactive dvfs control for multicore processors," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 102–109.

[33] M. Qiu, Z. Ming, J. Li, S. Liu, B. Wang, and Z. Lu, "Three-phase time-aware energy minimization with {DVFS} and unrolling for chip multiprocessors," *Journal of Systems Architecture*, vol. 58, no. 10, pp. 439 – 445, 2012.

[34] J. Halimi, B. Pradelle, A. Guermouche, and W. Jalby, "Forestmn: Runtime DVFS beyond communication slack," in *International Green Computing Conference, IGCC 2014, Dallas, TX, USA, November 3-5, 2014*. IEEE, 2014, pp. 1–6.