The Impact of Dynamic Voltage and Frequency Scaling on Multicore DSP Algorithm Design

Erik G Larsson and Oscar Gustafsson

The Impact of Dynamic Voltage and Frequency Scaling on Multicore DSP Algorithm Design

Erik G. Larsson and Oscar Gustafsson

We connect to the two recent *IEEE Signal Processing Magazine* special issues on DSP on multicore processors (Nov 2009 & March 2010) [1] and address an issue that was not addressed in the papers there and which we believe has important consequences for DSP algorithm design in the future. The basic observation that we start out from is that in DSP algorithm design, there is very often a tradeoff between the computational effort spent (in terms of number of operations) and the quality/accuracy of the algorithm output. Think, for example, of an iterative decoder, that delivers more and more accurate decisions the more iterations it performs. This means that one can find the “optimal” number of operations to perform, for a given quality target. In the context of parallel (multicore) hardware architectures this becomes especially interesting, because emerging component technology makes it possible to operate different cores on the same chip at different speeds, and to adapt these speeds in real time. When a core runs at a lower speed it is possible to reduce its power supply voltage, drastically reducing the power consumption. If an algorithm consists of many independent components that can be run in parallel, then the quality of the overall result will depend on the quality of the partial results delivered by each component. As a consequence, the challenge of writing algorithms for parallel architectures becomes not only the “classical” one of parallelizing algorithms, but also that of selecting appropriate speed (voltage) parameters for the cores involved, so that the overall energy consumption is minimized subject to a constraint on the quality of the overall result.

Herein we discuss the problems that emerge when optimizing algorithms for circuits that support the operation of several parallel cores that operate at different speeds. The understanding, formulation and solution of these optimization problems require a cross-disciplinary approach that models the interplay between circuits, computer architecture, signal processing and optimization. While allocation of computational and transmission resources for the purpose of saving energy is an established research topic in other fields (sensor network lifetime maximization being a notable example), there appears to be relatively little open literature on the type of problems that we discuss here. Taken together, we believe that the challenges that we pose are important and that the signal processing algorithm design community is well positioned to tackle them.

I. Dynamic Voltage and Frequency Scaling (DVFS) in Digital Hardware

Energy consumption in digital electronics is a major limiting factor in portable devices where batteries are used to supply power. It is also becoming a problem in general purpose computing [2]. Fundamentally, the energy consumed by a circuit grows with the amount of computation performed and at fixed speed, the energy consumed is proportional to the number of operations carried out. In CMOS electronics, the gate delay of a circuit, and, hence the clock frequency it runs at, can be varied within some limits by adjusting the power supply voltage. This practice is called voltage and frequency scaling. The higher the supply voltage V is, the faster the circuit runs. Since each operation essentially amounts to charging and discharging a capacitor, the dynamic energy consumed per operation is related to V via

$$E_{\text{per operation}} \propto V^2. \tag{1}$$

Taken together, this means that the faster the circuit runs, the more energy it requires per operation.

If the voltage and frequency can be adjusted *at run-time*, then one speaks of “dynamic” voltage and frequency scaling (DVFS). DVFS is used in microprocessors to adjust the power supply voltage based on various factors. For example, in a laptop, the processor speed is dynamically adjusted depending on the instantaneous computational load (using e.g. Intel SpeedStep, TransMeta Crusoe, AMD PowerNow!, or Texas Instruments SmartReflex technology), or to control the processor temperature. State-of-the-art technology today can estimate the computational requirements in advance and adjust the power supply voltage and clock frequency accordingly.

To understand the energy saving benefits of DVFS we need a simple model for the dynamic circuit energy consumption. Throughout this discussion we intentionally neglect static energy consumption (i.e., power drawn when idle) due to leakage effects and other losses. Consider the completion of a task consisting of C operations and which must be finished within a fixed time of τ seconds, and suppose that V is chosen so that the processing finishes just on time. The speed with which the circuit can be operated, and therefore the energy required per operation, is a highly nonlinear function of V, and it depends on the specific technology used and on the regime in which the circuit is operated [3]. Commonly, as a first-order approximation, the energy required per operation $E_{\text{per op}}$ is modeled as a quadratic function of the circuit operating speed [4], [5]. This means that for fixed τ, $E_{\text{per op}}$ is a quadratic function of C:

$$E_{\text{per op}} \propto C^2 \tag{2}$$

A more accurate modeling of $E_{\text{per op}}$, such as those in [5], [6], will change the precise form of the equations used later in the paper but it does not change the basic implications of our discussion. We will use (2) to illustrate the key concepts. Note that (2) means that the total energy consumed by the C operations is

$$E_{C\text{ ops}} \propto C^3 \tag{3}$$

and that the *power* consumed by the circuit is

$$P = \frac{C \cdot E_{\text{per op}}}{\tau} \propto \frac{C^3}{\tau}.$$

An important insight when designing algorithms for circuitry that supports DVFS is that when there is a hard deadline at which the result must be available, and the quality of the computation result can be traded for a reduction in computational operations, then it is always better to run the circuit slower and finish just on time, than to run it fast and finish early so that it must spend time idling. The following calculation shows this quantitatively, and Figure 1 illustrates it pictorially. Consider again the task above comprising C operations. Suppose that the quality of the result may be compromised so that only $C - \Delta_C$ operations are needed. If the circuit is run slowly in order to finish just in time after τ seconds, then the energy required is

$$E_{\text{run slowly}} \propto (C - \Delta_C)^3$$

By contrast, if the circuit is run at nominal speed, it will finish at time $(C - \Delta_C)/C \cdot \tau$, hence requiring an energy of

$$E_{\text{finish early}} \propto (C - \Delta_C)C^2$$

Clearly,

$$\frac{E_{\text{finish early}}}{E_{\text{run slowly}}} \propto \frac{C^2}{(C - \Delta_C)^2} \approx 1 + 2\frac{\Delta_C}{C} > 1$$

so running at full speed and finishing early costs more energy than running slowly and finishing on time. This is similar to driving a car: the amount of fuel consumed per mile increases with the speed of driving.

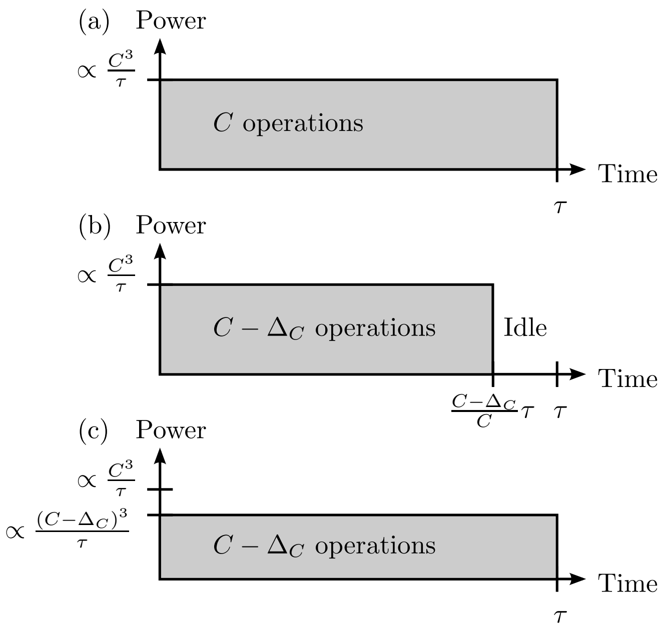
(a) Power
$\propto \frac{C^3}{\tau}$
C operations
Time
τ
(b) Power
$\propto \frac{C^3}{\tau}$
$C-\Delta_C$ operations
Idle
Time
$\frac{C-\Delta_C}{C}\tau$
τ
(c) Power
$\propto \frac{C^3}{\tau}$
$\propto \frac{(C-\Delta_C)^3}{\tau}$
$C-\Delta_C$ operations
Time
τ

Fig. 1. Power consumption for (a) performing C operations during the time τ, (b) performing $C - \Delta_C$ operations during the time $(C - \Delta_C)/C \cdot \tau$, and (c) performing $C - \Delta_C$ operations during time τ using DVFS.

II. Massively parallel computing

Parallelization of digital computing hardware is a major R&D trend today [1]. There are two drivers behind this. The first, most obvious one, is that the computational demands of some applications have increased so rapidly recently that single-core architectures are just not capable of handling the amount of computation needed.

The second reason for parallelization is related to energy consumption. Hardware architectures that perform many operations slowly in parallel are more power-efficient than architectures that perform a single operation very fast. To understand why this is so, consider again the above task comprising C operations, and suppose that the computation is broken down into N parallel and equally large parts. Then each parallel circuit needs to perform C/N operations within the time τ, and hence it can be fed with a lower voltage V than supplied to the original circuit. The total amount of energy consumed by the N parallel parts is

$$E_{\text{parallel}} \propto N \cdot (C/N)^3 = \frac{C^3}{N^2} \tag{4}$$

which is N^2 times less than (3). Of course, this is an overoptimistic conclusion for two reasons. First, more accurate, models than (2) may result in a different value of E_{parallel}, even though (4) is always less than (2). Second, if N is large the static power consumption from leakage and the power from overhead in the circuit will be significant [4]–[6]. Moreover, the cores may then no longer operate in the super-threshold regime and the delay equations will drastically change [3].

III. Per-Core DVFS Parallel Hardware

DVFS for parallel hardware has been an active research topic during the last few years [5]–[7]. Much of this work has focused on multi-core general purpose processors, mainly motivated by energy saving considerations [2]. Today, the speed/voltage of all cores are typically adjusted simultaneously. This way, the voltage regulators can be located off-chip. The power supply voltage to the core for DVFS-enabled systems is either provided by selecting one of several available voltages through a switch (transistor) or by changing the voltage provided by the voltage regulator. The switch approach is only feasible for few voltages and for rather small systems, as the power consumption for the switch transistor will be significant for larger currents.

There are basically two types of voltage regulators: linear regulators and switched regulators [8]. Linear regulators roughly work by dividing the voltage range, as for two resistors in series. This provides a simple implementation, but if the ratio of the output voltage and input voltage is small, then the efficiency is low. Switched regulators have an efficiency that is more or less independent of the voltage ratio. However, they often rely on large inductors which are hard to integrate on a chip for large currents. Off-chip switched regulators typically operate at rather low switching frequencies (< 5 MHz) so that the time to change the voltage is in the order of micro-seconds.

Recently, there has been several examples of on-chip switched voltage regulators. These operate at a significantly higher switching frequency (in the order of hundreds of MHz) and can change voltage in tens of nanoseconds. They rely on on-chip capacitors, while the inductors are bounded directly to the chip. This makes it possible to provide different voltages to different cores. It has been shown that the faster the voltage adjustment can be done, the more power can be saved [9].

Multicore technology that offers individual DVFS for each core, is referred to as per-core DVFS here (PC-DVFS). PC-DVFS circuits with a small number of cores are just becoming available off-the-shelf, and prototypes of PC-DVFS chips with a large number of cores have been fabricated [10].

IV. Signal Processing on Per-Core DVFS Enabled Parallel Hardware

We will illustrate next that the design and optimization of parallel algorithms for hardware that supports PC-DVFS requires a fundamental change in the way of thinking as compared to algorithm design for classical computing architectures. As before, the main motivation is applications where there is an explicit tradeoff between the amount of computation performed and the accuracy/quality of the result obtained after a fixed time (τ). Specific application examples will be discussed in the next section.

Consider a signal processing algorithm which is composed of N independent but *not* necessarily equally large parts. Each part performs C_n operations, where $C_1 + \cdots + C_N = C$. Hence the algorithm can be *either* run serially on one computational core, or in parallel on N cores. In a parallel implementation, each computational core may be supplied with a different voltage V_n, adjusted to match C_n. The accuracy of the final result depends (in a non-trivial way) on the amount of work done in all N parts of the algorithm, so that it is meaningful to look for the choice of $\{C_n\}$ that minimizes the total energy consumption *subject to a constraint on the algorithm output accuracy*.

With a single-core architecture, the N parts of the computation are performed sequentially after one another. Once the nth part has finished, the processor starts with the $(n+1)$st. The total energy consumed is $E_{\text{serial}} \propto \left(\sum_{n=1}^{N} C_n\right)^3$ and the problem of interest is to solve

$$\min_{\{C_n\}} \left(\sum_{n=1}^{N} C_n\right)^3 \quad \Leftrightarrow \quad \min_{\{C_n\}} \sum_{n=1}^{N} C_n, \tag{5}$$

subject to a constraint on the accuracy of the algorithm output. By way of contrast, with N parallel PC-DVFS enabled cores, the energy consumed by the nth core is $\propto C_n^3$ and the total energy consumed is $E_{\text{parallel}} \propto \sum_{n=1}^{N} C_n^3$, so what is of interest here is to solve a different problem:

$$\min_{\{C_n\}} \sum_{n=1}^{N} C_n^3 \tag{6}$$

subject to a constraint on algorithm accuracy. As before, more accurate modeling of the tradeoff between speed and energy consumption in (2) would result in the cubic relation $f(\cdot) = (\cdot)^3$ in (5)–(6) being replaced with another function $f(\cdot)$. However, this function will be convex and lie below the straight line $f(x) = x$, and hence problems (5) and (6) will be fundamentally different. Of course, selecting $C_n = 0$ is fully possible, which corresponds to the special case of idling the nth core.

Harvesting the gains offered by PC-DVFS in practice requires appropriate hardware architectures and algorithms that can efficiently utilize this hardware. In the basic setting, we envision architectures consisting of a large number of computational cores where each core is run at the appropriate speed and this speed is determined by a parameter *before* the computation begins, as a function of factors that are known at that time. More generally, we envision architectures where the speed of each core is determined as a function of factors known before the algorithm starts *and* as a function of the output of previous steps of the computation. Naturally, combinations of parallel and serial processing may be conceived of.

V. Implications on Applications

A. Signal Processing in Communication Systems with Coding and Interleaving

In communication systems, transmitted messages are typically protected by an error-correcting code that effectively adds redundancy. Very often, the encoded message $\mathcal{M}$ is segmented into bursts, say $\mathcal{M}_k$, which are transmitted using different radio resources. As a result, different bursts will be subject to different channel conditions and will be received with different signal-to-noise ratios, say SNR_k. The spreading of $\mathcal{M}$ over time and frequency is a useful practice because if the channel changes sufficiently much between the transmissions of the different $\mathcal{M}_k$, then diversity against the channel fading is gained.

We are interested in the situation when different amounts of signal processing are required for each burst $\mathcal{M}_k$ at the receiver. In particular, we are interested in the case when the amount of processing per $\mathcal{M}_k$ can be adaptively chosen, as a function of SNR_k, for example. The constraints when making this choice consist of a total budget of energy that can be spent *for the entire codeword*, and an overall quality target in terms of packet-error-rate after decoding of $\mathcal{M}$. (The latter translates directly into quality-of-service in applications.) This problem can be modeled mathematically. By solving problems of the form (5)–(6), one can find the optimal amount of operations C_k (and hence the optimal supply voltage V_k) to spend on the processing of each burst $\mathcal{M}_k$. This principle is illustrated in Figure 2. Clearly, for the reasons discussed in Section IV, the optimal solution depends on the specific type of hardware architecture used.

Since coding and interleaving over time and frequency is a common practice, the type of problem outlined here occurs in many applications. For example, with OFDM technology, used in 4G (3GPP-LTE) mobile broadband systems for example, an encoded message $\mathcal{M}$ is segmented into parts $\mathcal{M}_k$ that are sent on different frequency subcarriers. When OFDM is combined with multiple-antenna (MIMO) technology, each $\mathcal{M}_k$ will propagate over a different matrix-valued channel. At the receiver, a MIMO detector must then be run for each subcarrier [11]. Since the propagation characteristics depend on frequency, some of the subcarriers will require heavier computation than others to achieve a given quality of the final result. MIMO detection is an excellent example of a situation where the accuracy of the algorithm output stands in proportion to the amount of computation that the algorithm is allowed to perform. Different detector algorithms have very different computational complexity and differ vastly in how often they find the correct result. It is therefore not surprising that substantial performance gains can be achieved by adaptively allocating computational resources over the subcarriers, subject to a constraint on the total complexity [12]. A similar problem of separating signal mixtures that result from crosstalk occurs in DSL systems.

B. Iterative Decoding of Error-Correcting Codes

The problem of recovering a transmitted message from a noisy received version of the coded signal is called decoding and in terms of computational complexity, this is a major task in communication receivers. Decoding algorithms for conventional codes (convolutional codes, for example), are deterministic in the sense that they perform a given fixed number of operations and then terminate. State-of-the-art codes such as LDPC and turbo codes, by contrast, work in an iterative fashion where the number of iterations required to converge is random and depends on factors such as the signal-to-noise-ratio. Sometimes, the decoder may not converge at all. This opens up several possibilities. One is to estimate the number of iterations required before the decoder begins, and adapt the circuit speed accordingly [13]. Another more

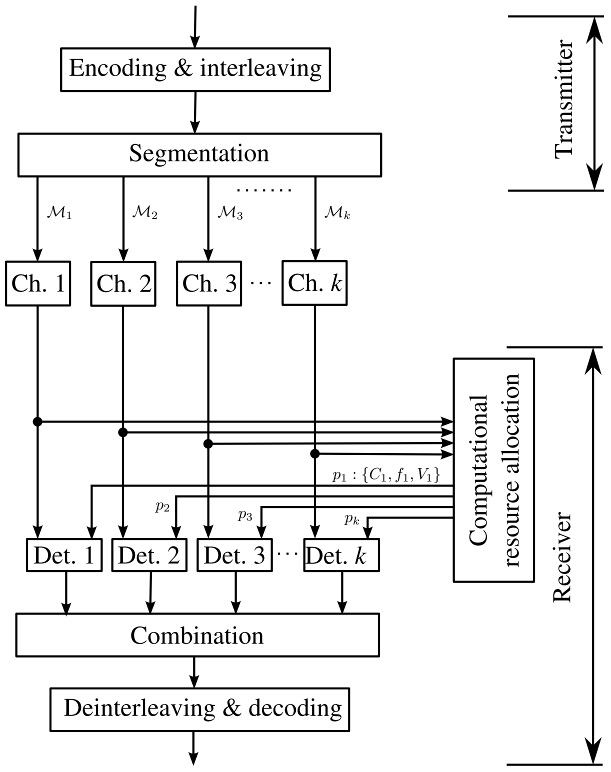

Fig. 2. Communication system with coding and interleaving across multiple bursts that propagate through independent channels. The receiver is PC-DVFS enabled and sets the operating parameters for each computational core individually.

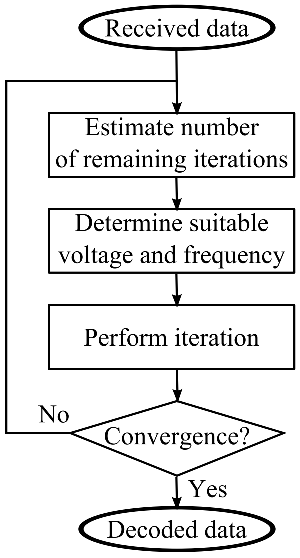
Received data
Estimate number of remaining iterations
Determine suitable voltage and frequency
Perform iteration
Convergence?
No
Yes
Decoded data

Fig. 3. Flowchart of DVFS-enabled iterative decoding.

sophisticated strategy is to look at the partial result after each iteration, and based on this make decisions on if and how to proceed with further iterations, and at what speed, conceptually illustrated in Figure 3 [14].

C. Frequency-Selective Digital Filters

The approximation error of frequency-selective digital filters is proportional to the filter order, and, hence to the number of arithmetic operations required. While the actual number of arithmetic operations required depends on the filter structure, the increase of arithmetic complexity with decreasing approximation error is general. In the context of our discussion, it is possible to adapt the filter length to the currently targeted approximation error [15]. For example, in a channel-selection filter in a communication receiver, the interference attenuation (which increases with increasing filter complexity) can be adapted, both to the actual interfering signals and to the actual SNR of the received signal. This scenario also introduces the additional problem of either designing the filters on-line, using a set of previously designed filters, or using a filtering algorithm whose complexity can be dynamically set as a function of the tolerated approximation error.

VI. Conclusions: Challenges for the DSP Community in the Next Decade

We have seen that PC-DVFS technology may fundamentally change the way of thinking when designing and optimizing algorithms. This leads to several new, interesting and challenging questions that should stimulate innovation within the DSP community. We summarize a few of them as follows:

- The design of signal processing algorithms needs to consider the capabilities of state-of-the-art hardware technology and of hardware that will become available off-the-shelf in a decade or two. In particular, *accurate models for circuit power/energy consumption* must be found which have an analytical form that yields tractable optimization problems. Ideally, these models should include transition overheads and leakage power consumption. This requires careful consideration of several intertwined issues. For example, the leakage power can be optimized by adjusting the threshold voltage using a technique analogous to DVFS called adaptive body biasing [7]. Some models for different scenarios are given in [5], [6].
- *Models for algorithm output accuracy.* Formulation of problems like (5)–(6) requires that one can quantify "accuracy" of an algorithm result or partial result in a more precise manner. This is a non-trivial task and goes well beyond that of current practice of quantifying the bit-error-rate (BER) or frame-error-rate (FER). For example, if the processing consists of signal detection or decoding, one needs measures of accuracy that quantify the chances that later steps of the processing will succeed. This may involve approximations of standard (but difficult to evaluate) measures such as mutual information. In an iterative decoding application, one would need to predict "how far from convergence" that the decoder after a given iteration. For a frequency-selective digital filter the attenuation can be adapted to the signal conditions. Of course, obtaining these accuracy measures themselves must not be too computationally demanding, relative to the actual processing.
- *Real-time aspects.* Is it better to have (i) several computational cores that are designed to run at different speed and switch between them, or to have (ii) several identical computational cores and select their operating parameters (speed/voltage) dynamically? To answer this we need precise models both for the hardware and for the communication system. If the number of parallel units K is very large, then the number of computational cores that should be operated with a specific set of parameters will be nearly constant, so that it is sufficient to have a number of cores designed for fixed parameters and switch between them.
- The computational resource *optimization problems* of the type outlined in (5)–(6) are combinatorial in nature. Care must be taken so that the cost of solving them does not outweigh the benefit obtained by the optimal allocation of computational resources.

Authors

Erik G. Larsson (erik.g.larsson@liu.se) is a professor and head of division for communication systems in the EE department (ISY) of Linköping University, Sweden. For more information, visit www.commsys.isy.liu.se.

Oscar Gustafsson (oscar.gustafsson@liu.se) is an associate professor and head of division for electronics systems in the EE department (ISY) of Linköping University, Sweden. For more information, visit www.es.isy.liu.se.

References

[1] Special issues on *Signal Processing on Platforms with Multiple Cores, IEEE Signal Processing Magazine*, vol. 27, November 2009 and March 2010.

[2] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[3] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "The limit of dynamic voltage scaling and insomniac dynamic voltage scaling," *IEEE Trans. VLSI Systems*, vol. 13, no. 11, pp. 1239-1252, Nov. 2005.

[4] V. Gutnik and A. P. Chandrakasan, "Embedded power supply for low-power DSP," *IEEE Trans. VLSI Syst.*, vol. 5, no. 4, pp. 425–435, Dec. 1997.

[5] E. Seo, J. Jeong, S. Park, and J. Lee, "Energy efficient scheduling of real-time tasks on multicore processors," *IEEE Trans. Parallel Distributed Syst.*, vol. 19, no. 11, pp. 1540–1552, Nov. 2008.

[6] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. Al Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Trans. VLSI Syst.*, vol. 15, no. 3, pp. 262–275, Mar. 2007.

[7] A. Milutinovic, A. M. Molnos, K. G. W. Goossens, and G. J. M. Smit, "Dynamic voltage and frequency scaling and adaptive body biasing for active and leakage power reduction in MPSoC: A literature overview," in *Proc. Program for Research on Integrated systems and Circuits*, 2009, pp. 488–495.

[8] D. Ma and R. Bondade, "Enabling power-efficient DVFS operations on silicon," *IEEE Circuits Syst. Mag.*, vol. 10, no. 1, pp. 14–30, 2010.

[9] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. IEEE Int. Symp. High Perf. Comp. Arch.*, 2008, pp. 123–134.

[10] D. N. Truong et al., "A 167-processor computational platform in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, April 2009.

[11] E. G. Larsson, "MIMO detection methods: How they work," *IEEE Signal Processing Magazine*, vol. 26, pp. 91–95, May 2009.

[12] M. Čirkić, D. Persson and E. G. Larsson, "New results on adaptive computational resource allocation in soft MIMO detection," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011.

[13] W. Wang, G. Choi, and K. Gunnam, "Low-power VLSI design of LDPC decoder using DVFS for AWGN Channels," in *Proc. Int. Conf. VLSI Design*, 2009, pp. 51–56.

[14] E. Amador, R. Knopp, V. Rezard, and R. Pacalet, "Dynamic power management on LDPC decoders," in *Proc. IEEE Symp. VLSI*, 2010, pp. 416-421.

[15] J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE J. Solid-State Circuits*, vol. 31, no. 3, pp. 395–400, Mar. 1996.