CONVEX ENERGY OPTIMIZATION OF STREAMING APPLICATIONS FOR MPSOCS

E. Nogues†, A. Mercat‡, F. Arrestier†, M. Pelcat†, D. Menard†

†Univ. Rennes, INSA Rennes, CNRS IETR UMR 6164, Rennes, France
‡ Tampere University, Korkeakoulunkatu 10, Tampere, 33720, Finland
firstname.lastname@insa-rennes.fr, alexandre.mercat@tuni.fi

ABSTRACT

The energy efficiency of modern MPSoCs is enhanced by complex hardware features such as Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM). This paper introduces a new method, based on convex problem solving, that determines the most energy efficient operating point in terms of frequency and number of active cores in an MPSoC. The solution can challenge the popular approaches based on never-idle (or As-Slow-As-Possible (ASAP)) and race-to-idle (or As-Fast-As-Possible (AFAP)) principles. Experimental data are reported using a Samsung Exynos 5410 MPSoC and show a reduction in energy of up to 27 % when compared to ASAP and AFAP.

1. INTRODUCTION

Stream processing has become one of the major classes of applications processed by embedded systems. Portable consumer electronics devices incorporate a wide range of signal processing applications, ranging from telecommunications to multimedia services. These applications are increasingly complex and manipulate a large amount of data at high data rates. When stream processing applications are implemented in battery powered systems, a main challenge is to minimize their energy consumption.

In modern systems, multi-core architectures are mandatory to manage the large amount of data to process. Recent Multiprocessor SoCs (MPSoCs) based on multi-core processors offer high processing capabilities and programmable architectures. The availability of embedded operating systems and the support of high-level languages ease the application implementation and reduce the time to market. Nevertheless, compared to dedicated or configurable architectures, the control of the energy consumption in an MPSoC is a challenge.

Two main power management techniques are provided by modern MPSoCs that minimize the energy consumption. Dynamic Power Management (DPM) [1] combines clock gating and power gating to turn a non-used processing core into a low power state. Dynamic Voltage Frequency Scaling (DVFS) [2] reduces the influence of dynamic power by adapting both the clock frequency and the supply voltage to real-time constraints. The energy efficiency of these MPSoCs comes primarily from parallel processing combined with these two power management techniques. Thus, in order to obtain the most energy efficient implementation of an application over an MPSoC, DPM and DVFS should be jointly optimized while taking into account the parallelism level of the application and the number of active processing cores. Different techniques have been proposed to exploit DVFS and DPM. Some approaches focus on dynamic power and do not take into account static power consumption [3]. Many approaches consider only mono-core execution [4–6]. In [7], an optimal DVFS and DPM combination is proposed to minimize the total energy consumption in an MPSoC. However, the parallelism level of the application is not taken into account in the energy minimization formulation. In [8], the speed-up due to application parallelism is integrated into the problem formulation with a simplistic application model based on Amdahl's law. Consequently, the parallelism level can not be adjusted for each task composing the application. In [4, 6, 7] analytical energy models [9] are constructed from technological parameters. Such technological parameters are usually not available for off-the-shelf MPSoCs, limiting the usability of these models.

This paper proposes a new method, based on convex problem solving, that determines the most energy efficient operating set point in terms of processing frequency and number of active cores of the MPSoC. The modeling can be considered either at fine grain with a *per*-task optimization or at a coarse grain with a top-level optimization of the considered application. The energy modeling is based on platform measurements that can be easily accessed. A framework that preserves the convex properties of the model throughout the optimization process is detailed. The solution challenges approaches based on the never-idle principle or the race-to-idle principle. An High Efficiency Video Coding (HEVC) decoder is analyzed as a use case. To the best of our knowledge, the method proposed in this paper is the first one to consider together DVFS, DPM and precise application parallelism. The energy gains obtained with the method call for new research in joint DVFS and DPM optimization.

The rest of this paper is organized as follows. Section 2 describes the proposed framework to optimize the energy at compile-time. The experimental results on a real platform are presented in Section 3 before concluding in Section 4.

2. PROPOSED APPROACH

2.1. Optimization Framework

Finding the best match between architecture features and applications characteristics is one of the greatest system design challenges. In this paper, we propose to formulate into one equation all design key parameters, ranging from platform features to application characteristics. Figure 1 depicts the elements of the framework.

The first issue that our framework addresses is to compute a power model when the detailed parameters of the underlying MPSoC are not available. Contrary to previous studies where the key parameters are known in advance, we propose to generate a compact formulation with the processing frequency f and the number of active cores or more precisely the number of threads c as only parameters. The next input of the framework is the requirements step. In this work, real-time streaming applications are considered. The application is made-up of a sequence of N tasks t_i applied on the input stream. Let $\mathbf{f}$ be a N-length vector defining the normalized

processing frequency $\mathbf{f}_i$ of each task t_i. The processing frequency is normalized by f_{max} the maximal processing frequency of the platform. The minimal processing frequency of the platform is f_{min}. One iteration of the application on the input stream is supposed to be finished before a deadline D. The application level is then characterized by two parameters: the number of operations to execute and the level of parallelism that the application inherently supports. However, the capability to scale efficiently onto the available cores is depending on the application itself. Let $\mathbf{w}$, be a N-length vector defining the complexity in cycle, $\mathbf{w}_i$, of each task t_i of the application in the case of mono-core execution. Let $\mathbf{c}$, be a N-length vector defining the normalized parallelism parameter $\mathbf{c}_i$ used for each task t_i. The parallelism parameter of a task corresponds to the number of threads used to parallelize this task. This parallelism parameter is equal to the number of cores used to execute the task when only one thread is bound to each core. c_{min} and c_{max} are respectively the minimum and maximum number of threads. The parallelism parameter is normalized by c_{max} to be lower or equal to one. The proposed framework uses parallelism and energy information to formulate the optimization. The goal is to find the optimal values for the vectors $\mathbf{c}$ and $\mathbf{f}$ which minimize the total energy consumed by the system E_{tot} such as the real-time constraint (deadline D) is fulfilled

$$\begin{aligned} \min_{\mathbf{f},\mathbf{c}} \quad & E_{tot}(\mathbf{f},\mathbf{c}) \\ \text{subject to} \quad & T_{tot}(\mathbf{f},\mathbf{c}) \leq D \\ & \mathbf{f_i} \geq \frac{f_{min}}{f_{max}},\ \mathbf{f_i} \leq 1,\ \mathbf{c_i} \geq \frac{c_{min}}{c_{max}},\ \mathbf{c_i} \leq 1 \end{aligned} \tag{1}$$

where T_{tot} is the total execution time of one iteration of the application. This execution time depends on the processing frequency $\mathbf{f}$ and the parallelism parameter $\mathbf{c}$ and is the sum of the execution time of each task t_i as expressed with the following expression

$$T_{tot}(\mathbf{f},\mathbf{c}) = \sum_{i=1}^{N} \frac{W(\mathbf{c}_i)}{\mathbf{f_i}} = \sum_{i=1}^{N} \frac{\mathbf{w}_i}{S_i(\mathbf{c_i}).\mathbf{f_i}} \tag{2}$$

where $W(\mathbf{c}_i)$ is a function providing the workload in cycles of the task t_i after parallelization on $\mathbf{c}_i$ threads. This workload is computed from the task complexity in the case of mono-core execution, $\mathbf{w}_i$, and the speed-up factor $S_i(\mathbf{c}_i)$ when the task t_i is computed with $\mathbf{c_i}$ threads. S_i is the speed-up function associated with task t_i and the different considered models are discussed in section 2.3.2.

Even if the problem is simple to express, its resolution is complex. As shown in [10], the power function, is however convex and this convexity can be used to solve the optimization problem in polynomial time.

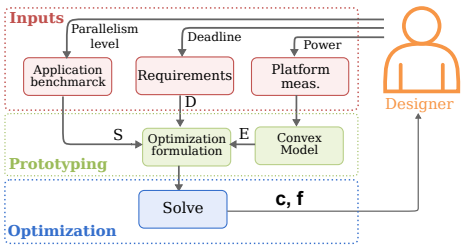

Fig. 1. Framework overview computing the energy efficient operating point

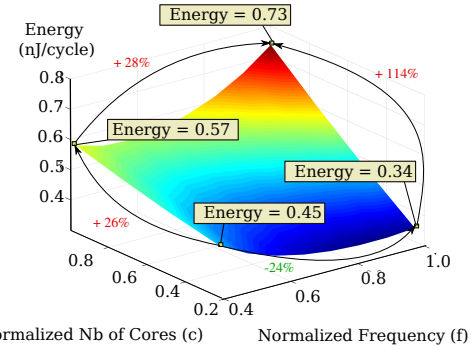

Fig. 2. Energy efficiency model of with DPM and DVFS

2.2. DVFS and DPM in an MPSoC

The Hardkernel XU3 platform integrating the Samsung Exynos 5410 processor is considered in this work as target patform. This SoC embeds two clusters with different power capabilities. Like most modern MPSoCs, the Samsung Exynos 5410 processor is equipped with DVFS and DPM to enhance its energy efficiency. The *cpufreq* and *cpuidle* Linux subsystems are used to set the core clock frequency and the power management states (Advanced Configuration and Power Interface [11]) respectively. The optimization of the joint frequency - parallelism parameters takes on a whole new significance when the criterion to minimize is energy rather than dynamic power. In [4, 6], authors integrate power expression to derive an energy-per-cycle metric E_{cycle} from power values. We propose to upgrade this method to MPSoC and define $E_{cycle}(f,c) = \frac{1}{f} P(f,c)$.

In [10], the energy efficiency is derived from the power measurements $P(f,c)$ carried-out on the Samsung Exynos 5410 processor. The power measurements for each core are performed with embedded power sensors and using the `stress` benchmark [12]. This benchmark is widely used for energy characterization because of its capability to tune the stress parameters [13]. The curve representing the energy per clock cycle is approximated by a convex function with the technique described in [10]. Figure 2 represents the energy per clock cycle as a function of the processing frequency and the number of active cores. It exhibits the efficiency of the different operation points of the design space. From an energy point of view, the performance can change drastically depending on the operation set-up point. Using a small number of active cores with high processing frequency fosters energy efficient systems design. Nonetheless, at a system level point of view, parallel processing can speed-up application execution and respect real-time requirements at a lower energetic cost. This function is injected in the optimization problem (1). The total energy consumed by the system E_{tot} is computed from the energy-per-cycle metric E_{cycle} and the workload $W(\mathbf{c_i})$ for $\mathbf{c_i}$ threads with the following expression

$$E_{tot}(\mathbf{f},\mathbf{c}) = \sum_{i=1}^{N} W(\mathbf{c_i}) E_{cycle}(\mathbf{f_i},\mathbf{c_i}) \tag{3}$$

2.3. Application Model Description

The intrinsic parallelism level of an application can be used to improve its energy efficiency. Parallelism modeling is a key point of the proposed framework. This section details the supported applications as well as the parallelism model.

2.3.1. *Modeling a pipeline of stream processing*

This paper addresses signal processing applications where a pipeline of internally parallel tasks process a stream of data. Typical examples of this type of application are domain transformations such as FFTs, sampling rate conversion, or image processing. Figure 3 is an illustration of such an application representation. All these individual applications have the property of supporting parallel processing. Contrary to previous studies [7, 8], the proposed modeling considers the parallelism as a parameter to adapt from either a fine grain representation (i.e. task per task) or a top level (i.e. at the application level).

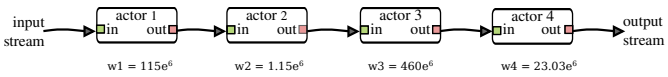

Fig. 3. Example of a streaming application

At the design phase, one can use different strategies to respect application real-time requirements. Using an MPSoC equipped with DVFS, there are two popular options. *As-Fast-As-Possible* scheduling consists of setting the processing frequency to its maximum and using the maximum number of cores. Conversely, *As-Slow-As-Possible* scheduling processes the application at the minimum speed that satisfies the deadline requirement. In this paper, we propose a third option which minimizes the energy of processing while satisfying the deadline. The proposed scheduling can use per-task frequency scaling as well as parallel execution mapping to minimize the energy. Figure 4 illustrates an example of generated schedule.

2.3.2. *Speed-Up models*

Because it is considered as a parameter for optimization, the speed-up after scaling the number of threads has to be characterized. For the targeted applications, two types of behaviors are considered. The first one, denominated *perfect* speed-up, assumes a perfect behavior and scales perfectly when the number of thread grows. Some signal processing applications like FFTs or image filtering can be efficiently sliced with a speed-up equal to the number of cores in the case of a homogeneous architecture. The second one, denominated *limited* speed-up, considers a non-perfect behavior with the following trend: $S(c) = k_0.c^{0.25}$, where S is the achieved speed-up and c is the normalized allocated threads. This model is inspired from the measured parallelism of an HEVC video decoder, discussed later in the experimental result section 3.2. Figure 5 illustrates the two behaviors in the case $k_0 = 2$. The penalty induced by a limited scaling together with the energy efficiency model confirms the challenge for the designer to find the best operating set-point.

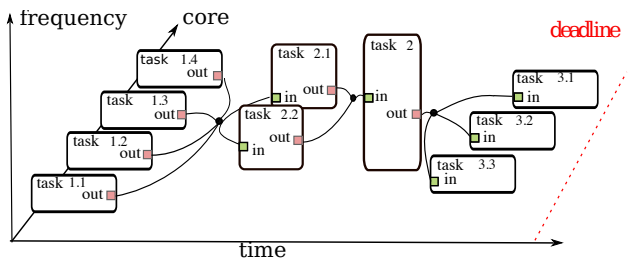

Fig. 4. Example of realization of the proposed scheduling. The number of instantiated tasks can vary as well as the frequency scaling

2.4. Final Optimization Formulation

Given the energy model proposed in [10] and the streaming application representation proposed in Section 2.3 with a *limited* speed-up, the optimization problem given in Eq. 1 can be reformulated as :

$$\begin{aligned} \min_{\mathbf{f_i},\mathbf{c_i}} \quad & \sum_{i=1}^{N} \frac{\mathbf{w_i}}{k_0\,\mathbf{c_i}^{0.25}}\left(a_1\frac{1}{\mathbf{f_i}} + a_2\sqrt{\mathbf{f_i}}\mathbf{c_i}^{1.5} + a_3 + a_4\mathbf{f_i}^2\mathbf{c_i} + a_5\mathbf{f_i}^6\mathbf{c_i}\right) \\ \text{s. t.} \quad & \sum_{i=1}^{N} \frac{\mathbf{w_i}}{k_0\,\mathbf{c_i}^{0.25}} \cdot \frac{1}{\mathbf{f_i} f_{max}} \leq D \\ & \mathbf{f_i} \geq \frac{f_{min}}{f_{max}},\ \mathbf{f_i} \leq 1,\ \mathbf{c_i} \geq \frac{c_{min}}{c_{max}},\ \mathbf{c_i} \leq 1 \end{aligned} \tag{4}$$

where N is the number of tasks composing the application (e.g $N = 4$ in Figure 3, $\mathbf{f_i} \in \mathbb{R}^N$ the normalized processing frequency for task i, $\mathbf{c_i} \in \mathbb{R}^N$ the normalized number of allocated threads for task i, $a_{0..4}$ the coefficients of the power model, and D the deadline of the application completion in seconds.

Adding the parallelism through the speed-up model as a parameter in the optimization process moves the problem to a multidimensional problem. As such, the *Disciplined Convex Programming* technique [14] cannot be exploited directly. Indeed, the product of convex functions is not convex if no particular care is taken. It is proposed to reformulate the problem to remove the unwanted products. If the problem is transformed with logarithm function, then it becomes an affine function problem where the products are replaced by sums. Boyd *et al.* [15] propose a solver, available in `CVX` tool [16] for this class of problem called Geometric Programming (GP) [17].

3. PERFORMANCE EVALUATION

Two types of experimental results are presented. First, the *As-Fast-As-Possible* and *As-Slow-As-Possible* scheduling strategies are compared with the scheduling output by the proposed framework. In a second set of experiments, the scope is narrowed to address HEVC decoding at a coarse grain and leveraging on its parallelism characteristics. Significant energy gains are obtained by the proposed convex modeling technique.

3.1. Comparison of Scheduling Strategies

Choosing the best operating point on an MPSoC is a challenging task. An *As-Fast-As-Possible* strategy leads to a straightforward approach that consists in racing to idle. Conversely, *As-Slow-As-Possible* consists in never idling and meeting the deadline as closely as possible. None of these approaches directly considers the energy

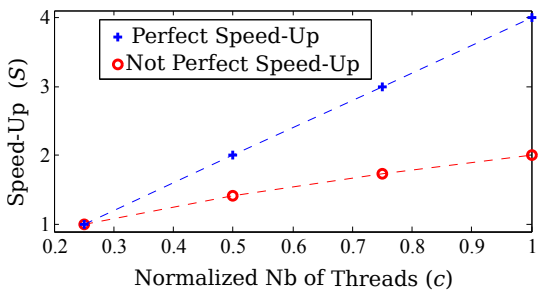

Fig. 5. Speed-up models: perfect and limited

Table 1. Comparison of the Energy consumption of proposed method, ASAP and AFAP for 3 deadlines with 2 speed-up types. The evaluated metrics are the execution time in seconds and the energy (mJ and normalized) and **f** and **c** are the normalized vectors for the frequencies and parallelism parameter.

D=0.5	Time (s)	**f** (%)	**c** (%)	Energy
		Perfect Speed-up		
Proposed	0.44	[56 56 56 56]	[100 100 100 100]	**198**—1.00
ASAP	0.50	[41 41 53 41]	[100 100 100 100]	208—1.05
AFAP	0.25	[100 100 100 100]	[100 100 100 100]	219—1.10
		Limited Speed-up		
Proposed	0.50	[100 100 100 100]	[100 100 100 100]	**438**—1.0
ASAP	0.50	[100 100 100 100]	[100 100 100 100]	**438**—1.0
AFAP	0.50	[100 100 100 100]	[100 100 100 100]	**438**—1.0
D=1.0				
		Perfect Speed-up		
Proposed	0.44	[56 56 56 56]	[100 100 100 100]	**198**—1.00
ASAP	0.60	[41 41 41 41]	[100 100 100 100]	227—1.19
AFAP	0.25	[100 100 100 100]	[100 100 100 100]	219—1.10
		Limited Speed-up		
Proposed	0.82	**[75 75 75 75]**	**[41 41 41 41]**	**356**—1.00
ASAP	1.00	[41 41 53 41]	[100 100 100 100]	415—1.16
AFAP	0.50	[100 100 100 100]	[100 100 100 100]	438—1.23
D=1.5				
		Perfect Speed-up		
Proposed	0.44	[56 56 56 56]	[100 100 100 100]	**198**—1.00
ASAP	0.60	[41 41 41 41]	[100 100 100 100]	227—1.19
AFAP	0.25	[100 100 100 100]	[100 100 100 100]	219—1.10
		Limited Speed-up		
Proposed	0.82	**[75 75 75 75]**	**[41 41 41 41]**	**356**—1.00
ASAP	1,20	[41 41 41 41]	[100 100 100 100]	454—1.27
AFAP	0,50	[100 100 100 100]	[100 100 100 100]	438—1.23

efficiency as the optimization objective. Table 1 compares our proposed optimization method to the two ASAP and AFAP with different deadlines ranging from a tight deadline, *D=0.5*, to a loose one *D=1.5*. The deadline for the execution of one iteration of the streaming application has been defined to cope with the frequency of the current sensor used for power measurement on the Odroid XU3 platform. These experiments are performed with the two types of speed-up illustrated in Figure 5: *perfect* or *limited* speed-up. In all cases, the proposed method achieves the most energy efficient scheduling. Reducing the processing frequency is known to improve energy efficiency, but ASAP is less energy efficient than AFAP on half of the tested configurations. This is due to the high impact of static power on this platform at low clock frequency. The gains are shown to depend on the deadline requirement and the speed-up model. An interesting result is that for *limited* speed-up model, the optimal solution does not lead necessary to the maximal parallelism parameter.

3.2. HEVC Decoding Case study

In this section, another example of system design is presented. An MPEG HEVC decoder (*openHEVC* [18–20]) is considered at a top level where the decoding is performed by single parallel task. This use case is an example of a pipelined streaming application as depicted in Figure 3 but with a unique task. The HEVC decoder has a degree of parallelism [8]. The number of threads used for the decoding can be adjusted through an input parameter of the *openHEVC* software. We measure the decoder speed-up over different numbers of threads on the considered platform. An HEVC decoder is first benchmarked on different sequences (Kimono, Park Scene...) and a speed-up model is extracted as depicted in Figure 6. The level of parallelism is observed by affecting an increasing number of execution threads, from 1 to 12, to the decoder.

From this information, our proposed framework computes the couple frequency - parallelism parameter that minimizes the energy consumption. The result of the rapid prototyping process, shows

Table 2. Measured Energy consumption (J) of the proposed set-up ($frequency = 351MHz, number\ of\ cores = 4$) on 100 s of MPEG HEVC decoding. Gains (%) are compared to the remarkable points of the design space (the corners in Figure 2).

Seq.	Energy	Gains (%) wrt. c_{min} or c_{max}, f_{min} or f_{max}			
		min, min	min, max	max, min	max, max
BD	5.19	52.3	76.8	12.7	71.5
BQ	3.94	51.15	76.7	15.8	72.35
KS	5.82	56.5	78.9	20.1	73.8
Km	23.70	59	79.5	11.8	77.4
PS	7.36	60.4	80.9	13.2	75.5

that the optimal configuration consists of processing the video at $f_{proc} = 351MHz$ (neither the minimum frequency of 250 MHz nor the maximum frequency of 600 MHz) onto all the available four cores.

The frequency - parallelism parameter design space has many available configurations and we choose to compare the output of the proposed framework to the other remarkable set points that are (f_{min}, c_{min}), (f_{max}, c_{min}), (f_{min}, c_{max}) and (f_{max}, c_{max}) (the corners in Figure 2). Table 2 discloses the gains of the proposed set-point compared to the best of the other points for HEVC 100-second bitstreams available at [21]. The energy is computed from instant power measurements done at a sampling rate of 10 Hz. It can be noted that the proposed set-point achieves the lowest energy. For example, our proposed set-point outperforms the *As-Fast-As-Possible* set-point, (f_{max}, c_{max}), by more than 70 % and the (f_{min}, c_{max}) setup by more than 12 %.

4. CONCLUSION

In this paper, a model of energy efficiency for stream processing applications on an MPSoC has been introduced and demonstrated. The model jointly considers the DVFS, DPM and parallelism capabilities of an MPSoC and an application to minimize the energy consumption of the implementation. Energy savings of up to 27% wrt. ASAP and AFAP approaches have been demonstrated. In future work, a finer tuning will be proposed by dynamically modifying the frequency and number of cores during the execution of the application.

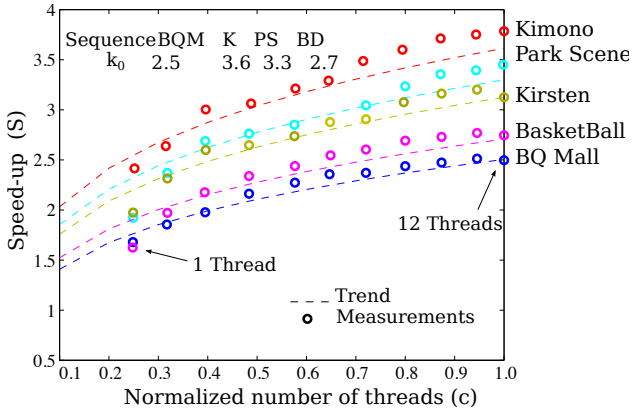

Fig. 6. Speed-up model of an MPEG HEVC decoder from measurements on various reference sequences, on the Exynos 5410 processor

5. REFERENCES

[1] Luca Benini and Giovanni DeMicheli, *Dynamic power management: design techniques and CAD tools*, Springer Science & Business Media, 2012.

[2] Woonseok Kim, Dongkun Shin, Han-Saem Yun, Jihong Kim, and Sang-Lyul Min, "Performance comparison of dynamic voltage scaling algorithms for hard real-time systems," in *Real-Time and Embedded Technology and Applications Symposium*, 2002, pp. 219–228.

[3] Guillaume Aupy, Anne Benoit, Fanny Dufossé, and Yves Robert, "Reclaiming the energy of a schedule: models and algorithms," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 11, pp. 1505–1523, 2013.

[4] Pepijn De Langen and Ben Juurlink, "Leakage-aware multiprocessor scheduling for low power," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. IEEE, 2006, pp. 8–pp.

[5] Marco E. T. Gerards and Jan Kuper, "Optimal dpm and dvfs for frame-based real-time systems," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 4, pp. 41:1–41:23, Jan. 2013.

[6] Andrew Nelson, Orlando Moreira, Anca Molnos, Sander Stuijk, Ba Thang Nguyen, and Kees Goossens, "Power minimisation for real-time dataflow applications," in *Digital System Design (DSD), 2011 14th Euromicro Conference on*. IEEE, 2011, pp. 117–124.

[7] Gang Chen, Kai Huang, and Alois Knoll, "Energy optimization for realtime multiprocessor soc with optimal dvfs and dpm combination," *ACM Trans. Embed. Comput. Syst.*, vol. 13, 2014.

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

[9] Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 275–280.

[10] E. Nogues, D. Menard, A. Mercat, and M. Pelcat, "On learning the energy model of an mpsoc for convex optimization," in *14th ACM International Conference on Computing Frontiers, CF 2017*, Ischia, Italy, May 2017.

[11] Len Brown, "Acpi in linux," in *Proceedings of the Linux Symposium, 51*, 2005.

[12] Amos Waterland, "The STRESS benchmark," Available at http://people.seas.harvard.edu/ apw/stress/.

[13] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sébastien Lafond, and Johan Lilius, "Energy efficiency and performance management of parallel dataflow applications," in *The 2014 Conference on Design & Architectures for Signal & Image Processing*, 2014.

[14] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[15] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi, "A tutorial on geometric programming," *Optimization and engineering*, vol. 8, no. 1, pp. 67–127, 2007.

[16] Michael Grant, Stephen Boyd, and Yinyu Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[17] David G Luenberger, *Optimization by vector space methods*, John Wiley & Sons, 1997.

[18] Erwan Raffin, Erwan Nogues, Wassim Hamidouche, Seppo Tomperi, Maxime Pelcat, and Daniel Menard, "Low power hevc software decoder for mobile devices," *Journal of Real-Time Image Processing*, vol. 12, no. 2, pp. 495–507, 2016.

[19] E. Nogues, S. Holmbacka, M. Pelcat, D. Menard, and J. Lilius, "Power-aware hevc decoding with tunable image quality," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct 2014, pp. 1–6.

[20] E. Nogues, R. Berrada, M. Pelcat, D. Menard, and E. Raffin, "A dvfs based hevc decoder for energy-efficient software implementation on embedded processors," in *2015 IEEE International Conference on Multimedia and Expo (ICME)*, June 2015, pp. 1–6.

[21] "BBC HEVC bistreams: ," in *ftp://ftp.kw.bbc.co.uk/hevc/hm-15.0-anchors/*.