Energy Efficient Scheduling of Real Time Signal Processing Applications through Combined DVFS and DPM

Erwan Nogues, Maxime Pelcat, Daniel Menard, Alexandre Mercat

Energy Efficient Scheduling of Real Time Signal Processing Applications Through Combined DVFS and DPM

Erwan Nogues, Maxime Pelcat, Daniel Menard and Alexandre Mercat
*IETR, INSA Rennes, UMR 6164, UEB
35000 Rennes, FRANCE
Email: firstname.lastname@mail.com

***Abstract*—This paper proposes a framework to design energy efficient signal processing systems. The energy efficiency is provided by combining Dynamic Frequency and Voltage Scaling (DVFS) and Dynamic Power Management (DPM). The framework is based on Synchronous Dataflow (SDF) modeling of signal processing applications. A transformation to a single rate form is performed to expose the application parallelism. An automated scheduling is then performed, minimizing the constraint of energy efficiency and providing DVFS and DPM decisions. This framework uses an architecture model including the number of available cores, the per-actor processing load and the energy per-cycle, derived from time and power measurements of modelled applications. After introducing the proposed framework, the energy characterization of big.LITTLE SoC systems is described. A generic approach is presented to generate the energy model of a platform from power measurements as customized polynomials. Finally, the experimental results on a Samsung Exynos 5410 big.LITTLE processor show that the energy optimal execution is not obtained by Linux governors that can execute either as-fast-as-possible or as-slow-as-possible. Instead, the most energy efficient scheduling is obtained by adapting both DVFS and DPM to application needs.**

I. Introduction

Signal processing applications (also called stream processing applications) have 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 high data rates. In such systems, respecting energy consumption constraints is a challenge. Thanks to recent advances in microelectronics, modern Systems-on-a-Chip (SoC) based on multi-core processors offer high processing capabilities. The energy efficiency of these SoCs come primarily from parallel processing combined with adaptive frequency. Many functions that were traditionally hardwired can now be implemented in software, bringing flexibility and short time-to-market for applications deployment.

For multi-core processors, two main power management techniques are provided that minimize the energy consumption. By combining clock gating and power gating, Dynamic Power Management (DPM) [2] is used to turn a processing core into a low power state when it is not in use. To reduce the influence of dynamic power, Dynamic Voltage Frequency Scaling (DVFS) [13] reduces both the clock frequency and the supply voltage until the application real-time constraint is slighly exceeded.

The design of energy efficient systems can be done at several levels of abstraction (gate level, register transfer level, core level, operating system level, SoC level...). Dataflow programming provides descriptions of signal processing that exhibit parallelism explicitly. The application is then described with a graph, where the nodes, named actors, represent the processing carried-out on the data. The edges are First-In-First-Out (FIFO) queues with storage capacity that are the only media for actors to communicate. This paper proposes a framework for the energy consumption minimization of signal processing applications that are described with Synchronous Data Flow (SDF) [14]. The targeted system has the following properties : a) the system is constrained by a deadline; b) each actor has a fixed workload; c) the underlying platform supports dynamic frequency scaling and dynamic power management; d) the dataflow model is SDF, and thus static.

There are numerous applications that fulfill these conditions in the signal processing domain [3], [10]. For example, multimedia applications like audio resamplers and video spatial interpolation procedures use chain of filters that are static and deadline-constrained. In digital communications, applications like propagation channel equalization or data error correction also verify these properties and are deadline-constrained.

The power management techniques provided in the Linux operating system do not benefit from the predictability properties offered by static dataflow. State-of-the-Art approaches do not take into account static power [1] or multi-core execution [6], [15]. In this paper, an energy efficient approach combining DVFS and DPM is proposed. An energy model is defined and a convex optimisation framework is used to find the optimal system clock frequency. We show that the combination of DVFS and DPM leads to better energy efficiency than classical approaches executing as-fast-as-possible or as-slow-as-possible.

The rest of this paper is organized as follows. Section II presents existing methods to optimize energy in recent processors. Section III describes how the system energy is modelled. The platform characterization is presented in section IV. Application modeling is described in section V. Finally, section VI concludes the paper.

II. Related work

As defined in Linux, a processor can be in two types of power states named *C-states* and *P-states*. In *C-states* C_1 to C_6, the processor is in idle state. The C_6 state corresponds to the deeper sleep state. Clock gating and power gating techniques consist in pruning the clock tree and switching off the portions of the circuit which are not in use. The management of the different running and idle states to reduce energy consumption is carried-out by the DPM technique [2]. In Linux, the DPM technique is implemented by the *CPUidle* subsystem, providing the ability to the user to switch between different *C-states*.

In the *P-state*, the processor is running and energy consumption can be reduced by adjusting the clock frequency and the supply voltage. The reduction of processor load gives opportunities to the operating system to reduce the clock frequency and hence the supply voltage. In real time systems, the temporal difference between the end of a task and its deadline, named slack-time, provides clock frequency flexibility. In Linux, the DVFS [13] technique is implemented with the *CPUfreq* subsystem. This subsystem scales the core clock frequency and sets the lowest supply voltage compatible with this clock frequency. This technique is widely used in embedded sysmtems [16].

For the energy management of dataflow applications, the work of Nelson et al. [15] can be considered as the closest reference to this paper to the best of our knowledge. Authors propose to model the energy of dataflow applications and use a convex solver to find the best working frequency of the dataflow actors. However the modeling does not consider the energy at a MPSoC level and how the system can be scheduled efficiently from an energy point of view. This paper questions how an efficient implementation can be obtained considering both application requirements and platform characteristics using a fast prototyping framework.

Similarly, De Langen *et al.* [6] introduce the notion of energy modeling for DPM and DVFS systems. They propose heuristics for energy efficient scheduling but only considering mono-core implementations. We propose a more complete formulation for multi-core implementations where solvers can be used to find an optimized scheduling and energy policy. We also experiment our approach on a cutting-edge platform using 28 nm silicon process.

Aupy *et al.* [1] propose several DVFS strategies for real-time systems under bounded execution time with proofs of optimality. However, the modeling part only considers the dynamic power part of the overall power consumption, which limits the scope of the proposed approach especially because the modern SoCs are not only dominated by dynamic power.

The next section explains how energy is modeled in the proposed framework.

III. Energy model and Power management

In this section, the power and energy models basis is given and its application to DVFS and DPM is proposed. Besides, the different strategies using DVFS, DPM and a combination of both is described for embedded operating systems.

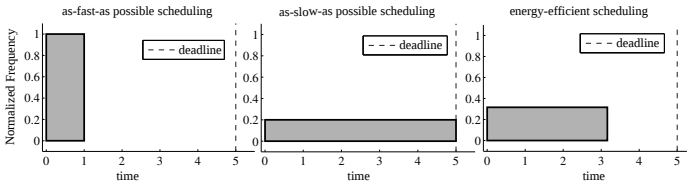

Fig. 1. Comparison of scheduling strategies - left: *as-fast-as-possible* scheduling - center: *as-slow-as-possible* - right: *Energy Efficient* scheduling

A. Power model

The growing offer in terms of high performance embedded SoCs, many of them being based on ARM cores, makes an ever larger number of applications executable in software. The power consumption of the system is a crucial aspect to secure the application viability, especially in the handheld domain where systems are battery-powered.

The power consumption of processors is generally split into two parts: the dynamic part coming from the switching activity of transistors and the static part coming from transistor leakage current. Whereas it has been neglected for a long time, the static part is bound to take a preponderant role in the overall system when considering new technology generations. In this paper, we use the power model from Jejurikar et al. [12] where the total power consumption is given by equation 1.

$$P_{tot} = P_{dyn} + P_{stat} \tag{1}$$

where

$$P_{stat} = V.I_{leak} \quad (2) \qquad P_{dyn} = C_{eff}.f_{proc}.V^2 \quad (3)$$

and where V is the supply voltage matching with the processing frequency f_{proc}, I_{leak} is the leakage current, and $Ceff$ is the circuit effective capacitance. This model has been used in several energy efficiency studies [6], [15].

B. Energy model

The capability of the system to shut down processing cores is a crucial point for energy efficient system design. Low energy and low power are different objectives. Indeed, one can argue that minimizing at each instant the processing power consumption is the most energy efficient strategy. However, this is not systematically true for systems with shut down mechanisms because the processing duration and activation time play a role in energy. For this latter case, the energy needed per cycle count of processing can serve as a comparison basis. It is determined as a function of the processing frequency in equation 4 assuming that the power consumption is close to null when entering to sleep mode.

$$E_{cycle}(f_{proc}) = \frac{1}{f_{proc}.T}.\int_0^T P_{tot}(t)\,\mathrm{d}t \tag{4}$$

Similarly to De Langen *et al.* in [6], a simple model is used to illustrate energy modeling with $P_{dyn}(f) = Cst_A.f^2$ and $P_{stat} = Cst_B$ where Cst_A and Cst_B can be derived from the

power models of equation 1. Figure 2 is an illustration of the energy and power dissipations as functions of the processing frequency. All values are normalized w.r.t to their maximum value for simplicity reasons.

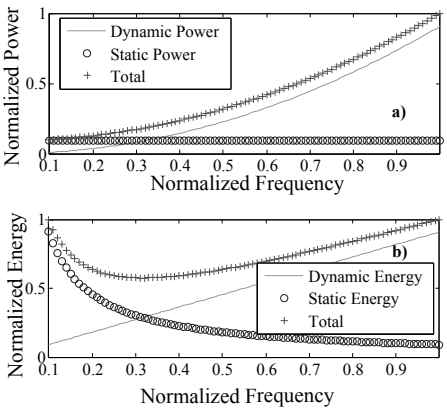

Fig. 2. Power and energy consumption on a mono-processor with DVFS and DPM. Top figure a): Power consumption (normalized) per frequency use (normalized) - Bottom figure b): Energy consumption (normalized) per frequency use (normalized)

In both curves, the convexity of the energy and power characteristics is outlined. However, the minimum value of power and energy is not reached for the same processing frequency. The power model of Figure 2 shows that the minimum power dissipation happens when using the minimum frequency while the energy model exhibits an optimal frequency $f_{efficient}$ at 0.3 that provides a better energy efficient design than the minimum frequency $f_{minimum}$ at 0.1. Therefore, using this processing frequency is the most efficient strategy, providing that the system can enter into a deep sleep mode when the processing ends.

C. Frequency allocation: as-slow-as-possible, as-fast-as-possible and energy efficient schedulings

In standard operating systems such as Linux, the processing frequency can be configured to different modes according to system requirements and constraints. Assuming a simple application which execution shall be terminated before a deadline, the most common strategies consists to run *as-fast-as-possible* at the maximum frequency $f_{maximum}$ or to run *as-slow-as-possible* while respecting the deadline. The first strategy is implemented by the *performance* Linux governor and uses a DPM mechanism when the second strategy is implemented by the *on-demand* Linux governor and uses a DVFS mechanism to scale the frequency correctly [5].
We introduce a new strategy called energy-efficient scheduling that reaches the minimum energy such as in Figure 2, aiming to process at the most energy efficient state.
Figure 1 compares *as-slow-as-possible* scheduling, *as-fast-as-possible* scheduling, and *energy-efficient scheduling*. The application is supposed to finish before a deadline that can be achieved with the minimum frequency.

In the next section, the generation of power and energy models from a real platform are explained. The objective of this section is to verify the suitability of the models in Figure 2 on a State-of-the-Art System-on-Chip (SoC) and find the frequency $f_{efficient}$ corresponding to the energy efficient scheduling.

IV. Platform characterization

A. Platform selection: focus on 28 nm Exynos 5410

In our experiments, an Exynos 5410 SoC is used. This SoC is based on a big.LITTLE ARM configuration with four ARM Cortex-A15 cores and four ARM Cortex-A7 cores providing a set of 17 DVFS configurations from 250 MHz to 1.6 GHz. Only four cores can run at a time and all cores share the same working frequency. From 250 MHz to 600 MHz, the SoC uses the Cortex-A7 cluster. From 800 MHz to 1.6 GHz, the Cortex-A15 cluster is used. This platform provides DVFS and DPM.

B. Building a power model from measurements

For conciseness reasons, applications are considered to necessitate all the processing cores in the system. It means that all the available cores are used. The SoC is considered as a whole and the power consumption is measured for an *intensive* processing. The power model is derived from effective measurements using the `stress` benchmark. This benchmark is widely used for energy characterization because of its capability to tune the stress parameters [9]. It is run under Linux on four threads with all the available DVFS configurations provided by the SoC. In Figure 3, the power consumption is depicted as a function of the processing frequency and is normalized versus the maximum frequency $f_{max} = 1600MHz$. As expected, the minimum power state is attained when the frequency is minimum.

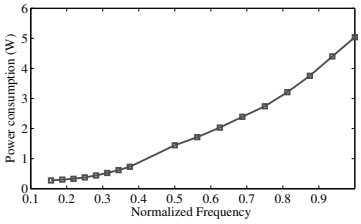

Fig. 3. Power consumption vs. operating frequency with 4 fully loaded cores on an Exynos 5410

Section III-B establishes the relation between the power and the energy per cycle of processing. At the system level, the energy per cycle can be computed as a function of the processing frequency from power and time measurements.
Figure 4 depicts the energy behaviour of the system. Contrary to the power, the most energy efficient state is not reached at the minimum frequency $f_{minimun}$. The optimal frequency $f_{efficient}$ is 15 % more energy efficient than the minimum frequency for the Cortex-A7 cluster. It is also the case for the complete system that includes both a Cortex-A7 cluster and a Cortex-A15 cluster. These measurements confirm the initial model depicted in Figure 2.

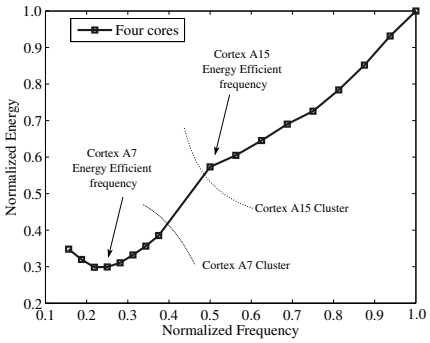

Fig. 4. Energy per cycle vs. operating frequency with 4 fully loaded cores on an Exynos 5410

Approximation of the platform power characteristics: In order to optimize the frequency, we build a regression model that computes the power as a function of the processing frequency. The model aims at fitting and interpolating the power measurement. Since these variables are used in normalized ranges, they can be considered unit-less and in the range of [0 - 1]. We used Levenberg-Marquardt's algorithm [11] to minimizes the error between the model and the real data. The goal is to define a curve as close as possible to the data values in Figure 4. The obtained equation is:

$$E_{cycle}(f_{norm}) = p_0 \frac{1}{f_{norm}} + p_1 f_{norm} + p_2 f_{norm}^3 \quad (5)$$

with $p_0 = 0.1730$, $p_1 = 0.1564$, $p_2 = 0.3367$.

Figure 6 shows the energy function used in the optimization problem together with the actual measurements. The modeling errors are close to the measurement accuracy.

V. Application modeling

A. From static Dataflow to real-time constraints estimation

Dataflow modeling is used in this paper to assist the operating system in finding the adequate processing frequency. Modeling a real-time application with an SDF enables an off-line analysis by the designer. For simplification reasons, this study is limited to SDF graphs with only data parallelism and a succession of actors such as in Figure 5. Each actor is annotated by a weight representing the number of processor cycles necessary to execute it. This number can be either an exact number in case of a completely deterministic execution or a Worst Case Execution Time (WCET) in case of varying loads. For signal processing applications like filtering and interpolation, the processing is independent from the data content and static dataflow can be used to model the system at a high level of abstraction [3]. Thus, the overall complexity can be estimated and used to compute the energy efficient frequency of the system. Such a description is a precious tool to make early-design trade-offs and verification of the throughput requirements of the application.

For a given number of processing cores, the SDF graph can provide the number of instructions that need to be executed by each core before reaching the deadline. The appropriate

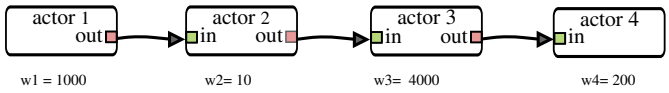

Fig. 5. Example of a dataflow application.

processing frequency of operation of the processor f_k can then be derived. $f_{i,k}$ is a vector where each entry matches the instance of actor k on a given processor i. Because of hardware and technology constraints, the operating frequency is bound by a minimum $f_{minimum}$ and a maximum $f_{maximum}$. The vector w_k corresponds to the work to be executed for each actor k (in cycles).

An optimisation process is conducted, based on the application SDF graph, the platform model and the energy model.

B. Frequency optimization process

The goal of the optimization is to find the most energy efficient frequency for each actor in the SDF graph while satisfying the global application real-time constraints. The problem can be formulated as follows:

$$\begin{aligned} \underset{f_i}{\text{minimize}} \quad & \sum_{i=1}^{N} w_i E_{cycle}(f_{norm}) \\ \text{subject to} \quad & \sum_{i=1}^{N} \frac{w_i}{f_i} \leq Deadline \\ & f_i \geq f_{min} \\ & f_i \leq f_{max} \end{aligned} \quad (6)$$

where f_{norm} is the normalized frequency. The trend of $E_{cycle}(f)$ function is convex as shown in Figure 5. The $E_{cycle}(f)$ modeling function is computed with the convex polynomial approximation from equation 5.

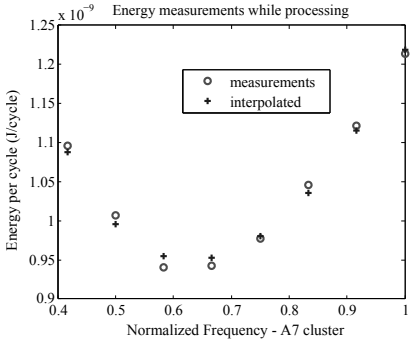

Fig. 6. Energy interpolated function used in the optimization problem

The Disciplined Convex Programming technique [4] is used to solve the problem. Convex problems are proven to be solved in polynomial time. The output values are obtained with the `CVX` tool [7], a convex optimization solver. The energy model and constraints are given as formulated in problem 6.

TABLE I. ENERGY MEASUREMENTS OF THE COMPUTE SYSTEM INCLUDING THE APPLICATION, THE RUNNING OPERATING SYSTEM AND THE POWER MEASUREMENTS

Processing Frequency	250	300	350	400	450	500	550	600
Percentage of processing (%)	100	85	73	64	57	51	47	43
Energy (Joules)	1.15	1.10	1.08	1.19	1.34	1.52	1.71	1.89
Gain vs *as-fast-as-possible*(%)	39.2	41.6	42.6	37.25	28.90	19.58	9.53	0

C. Experimental Verification on the Exynos SoC

The dataflow example of Figure 5 is implemented on the Exynos 5410 where the power and energy characteristics are derived from Figure 6. To illustrate the energy properties of the scheduling strategies, we focus on the Cortex-A7 cluster. The Cortex-A7 cluster shows clearly a energy efficient processing frequency at $f = 350MHz$.
DVFS is supported from 250 MHz to 600 MHz with steps of 50 MHz. The DPM - or deep sleep mode - is implemented through cluster migration, *i.e.* all running processes are migrated to the A15 cluster so the A7 cores can enter into a deep sleep mode. The time needed to switch frequencies or to migrate task is small enough on this platform and can be neglected [8]. The sensor logs the power consumed by the Cortex-A7 cluster. The application is run on top of of a Linux operating system.

In Table I, the performance of each available running frequency of the SoC is given. If the operating system uses the *as-fast-as-possible* policy, the energy consumption is 1.89J. The *as-slow-as-possible* policy reduces the energy consumption by 39%. The most energy efficient normalized frequency, closest to $0.58 * 600MHz$, achieves 6% of further energy savings. The experimental results thus confirm the theoretical ones.

Compared to the gains obtained by simulation in Figure 4, the energy gains are slightly under the predicted ones for several reasons. First, the SoC cannot enter into a zero energy mode and needs some energy to idle whereas the model assumes null energy consumption in sleep mode. Moreover, the energy probes themselves need energy to perform the measurements. In future work, an external energy probe will be tested to enhance the accuracy of the measurements.

VI. CONCLUSIONS

This paper proposes a energy efficient programming policy that exploits the properties static dataflow graphs namely fixed processing load and bounded by a deadline constraint. We compare the typical policies *as-slow-as possible* and *as-fast-as possible* of Linux *on-demand* and *performance* governors to a more efficient one using both DVFS and DPM. Thanks to an energy model of the platform, the operating system can determine an energy efficient processing frequency that is different from the minimum system frequency. The model is tested on a 28 nm big.LITTLE SoC and shows up to 42.6 % of energy saving compared to the *as-fast-as possible* scheduling and 6% when compared to the *as-slow-as possible*. In future work, dynamic applications will be considered and on-line convex optimizer will be included into the operating system.

VII. ACKNOWLEDGMENTS

This work is partially supported by BPI France, Region Ile-de-France, Region Bretagne and Rennes Metropole through the GreenVideo Project.

REFERENCES

[1] G. Aupy, A. Benoit, F. Dufossé, and Y. Robert. Reclaiming the energy of a schedule: models and algorithms. *Concurrency and Computation: Practice and Experience*, 25(11):1505–1523, 2013.

[2] L. Benini and G. DeMicheli. *Dynamic power management: design techniques and CAD tools*. Springer Science & Business Media, 2012.

[3] S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala. *Handbook of signal processing systems*. Springer Science & Business Media, 2013.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] D. Brodowski. Cpu frequency and voltage scaling code in the linux(tm) kernel. https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, 2013.

[6] P. De Langen and B. Juurlink. Leakage-aware multiprocessor scheduling for low power. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8–pp. IEEE, 2006.

[7] M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[8] S. Holmbacka, J. Keller, P. Eitschberger, and J. Lilius. Accurate energy modelling for many-core static schedules. In *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, 2015.

[9] S. Holmbacka, E. Nogues, M. Pelcat, S. Lafond, and J. Lilius. Energy efficiency and performance management of parallel dataflow applications. In *The 2014 Conference on Design & Architectures for Signal & Image Processing*, 2014.

[10] S. Holmbacka, E. Nogues, M. Pelcat, S. Lafond, D. Menard, and J. Lilius. Energy-awareness and performance management with parallel dataflow applications. *Journal of Signal Processing Systems*, pages 1–16, 2015.

[11] K. Iondry. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999.

[12] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st annual Design Automation Conference*, pages 275–280. ACM, 2004.

[13] W. Kim, D. Shin, H.-S. Yun, J. Kim, and S.-L. Min. Performance comparison of dynamic voltage scaling algorithms for hard real-time systems. In *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, pages 219–228, 2002.

[14] E. Lee, D. G. Messerschmitt, et al. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

[15] A. Nelson, O. Moreira, A. Molnos, S. Stuijk, B. T. Nguyen, and K. Goossens. Power minimisation for real-time dataflow applications. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 117–124. IEEE, 2011.

[16] 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 *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.