Optimal Speed Assignment for Probabilistic Execution Times

Claudio Scordino
University of Pisa, Italy
scordino@di.unipi.it

Enrico Bini
Scuola Superiore Sant'Anna, Italy
e.bini@sssup.it

Abstract

The problem of reducing energy consumption is dominating the design and the implementation of embedded real-time systems. For this reason, a new generation of processors allow to vary the voltage and the operating frequency to balance computational speed versus energy consumption. The policies that can exploit this feature are called Dynamic Voltage Scheduling (DVS).

In real-time systems, the DVS technique must also provide the worst-case computational requirement. However, it is well known that the probability of a task executing for the longest possible time is very low. Hence, DVS policies can exploit probabilistic information about the execution times of tasks to reduce the energy consumed by the processor.

In this paper we provide the foundations to integrate probabilistic timing analysis with energy minimization techniques, starting from the simple case of one task.

1 Introduction

The number of embedded systems operated by batteries is increasing in different application domains, from PDAs (*Personal Digital Assistants*) to autonomous robots, smart phones and sensor networks. Reducing the energy consumed by these systems has become a key design issue, as they can only operate on the limited battery supply. For this reason, a new generation of processors [9, 13, 19] allow to dynamically vary the voltage and the operating frequency to balance computational speed versus energy consumption.

In recent years, as the demand for computing resources has rapidly increased, even normal workstation PCs and servers face energy constraints. Not surprisingly, a significant portion of the consumed energy is due to the cooling devices, which may consume up to the 50% of the total energy [11]. In addition, researchers at IBM showed that average processor use of servers is between 10% and 50% of their peak capacity because the load depends on the time of the day or the day of the week [4]. This suggests that a striking energy reduction can be achieved by enriching DVS policies with a more detailed information on the required workload.

Recently, many DVS algorithms have been proposed in the literature. These algorithms can be divided in two classes: static and dynamic. Static techniques [21, 14, 17, 12, 3] are typically applied to periodic tasks, and make use of off-line parameters, such as periods and worst-case execution cycles (WCECs), to select the appropriate processor speed. Since the worst-case parameters may differ significantly from the actual values, these techniques save less energy than the dynamic ones.

On the other hand, much recent research has focused on dynamic techniques [14, 1, 22, 17, 16, 18], which take advantage of early job completion. Some studies have observed that the actual execution cycles of real-world embedded tasks may vary up to 80% with respect to their measured WCECs [20]. Thus, dynamic methods can exploit information about the run-time behaviour of tasks, which may be very far from the pessimistic assumptions required during the design of static techniques. Dynamic algorithms may take decisions — change the processor speed — at two different instants:

After task completion The algorithm does not make any assumption on task duration, and waits for task completion to know the exact execution time. Then, the processor speed is changed based on this information. The algorithms GRUB-PA [18], DVSST [16] and RTDVS-Cycle Conserving [14] belong to this category.

Before task completion The algorithm tries to foresee the duration of the current task instance, and takes the decision in advance, based on some task's characteristics such as the average execution time. This decision typically depends on the behaviour of previous instances of the task. Obviously, a right prediction allows to reduce considerably the energy consumption. However, when the predicted behaviour is distant from the reality, some undesired side effect, such as a deadline miss in soft real-time systems or an increase of the energy consumed, may occur. The algorithms RTDVS-Look Ahead [14], DVS-EDF [22] and DRA-Aggressive [1] are based on this mechanism.

The success of the second class of methods relies on predicting correctly the task behaviour. For this reason, the use

of a richer task information may enhance the effectiveness of the DVS. This increased information can be provided by the probability density function (p.d.f.) of the task execution time. Recently, the discipline of probabilistic timing analysis has significantly advanced [5, 7], and today there exist some tools which can provide the p.d.f. of task execution times [2].

An attempt to consider stochastic information in energy reduction problems has been done by Gruian [8]. However, it only addresses the case with no transition overheads and with a specific power function.

In this paper we integrate the concept of probabilistic execution time within the framework of energy minimization, providing the basis of a new challenging approach. We preliminary consider the case of only one task, since we believe that it can be extended to the general case of n tasks.

2 System model

2.1 Processor model

We assume that the processor has a continuous spectrum of operating modes. This means that the speed can continuously vary between zero and some speed upper bound. We know that in real-world architectures this hypothesis does not ever hold. However, many significant contributions in the literature [1, 14, 22] still assume a continuous speed because if the processor speed levels are very close each other then this approximation is very close to reality. Obviously, if the optimal speed is not available, it has to be approximated with the closest discrete level higher than the optimal one. In this case, there is an increase of energy consumption, called *energy quantization error*, that has been studied by Saewong and Rajkumar [17].

The processor model is formalized as follows:

- the speed α can vary within $[0, \alpha_{\max}]$, where $\alpha_{\max}$ is the maximum speed allowed by the processor;
- the power consumption at speed α is modelled by the function $p(\alpha)$. Typically, the power function $p(\alpha)$ is a polynomial [6]. However, due to the advances of semiconductor technology, it is expected that the power/speed relationship may change in the next future [8]. For this reason we model this relationship by a generic function $p(\alpha)$;
- the cost of mode switching is considered in terms of both time and energy. We call o the time overhead needed to switch between any two modes, and e the energy required. Notice that o and e do not depend on the modes before and after the switch.

2.2 Energy management scheme

We focus on the problem of reducing the energy consumed by a task τ on a variable speed processor. Some existing power-aware algorithms have been deployed starting from this simple scheme, since it constitutes a good starting point for more complex analysis [22].

The task τ has period and deadline both equal to T. The number of processor cycles required in the interval $[0, T]$ is modelled by a random variable whose p.d.f. is $f_C(c)$. The maximum possible number of cycles needed by τ is $C_{\max}$. Since the task is hard real-time, $C_{\max}$ cycles must be available in $[0, T]$.

If the number of required cycles in $[0, T]$ was known in advance, the best way to reduce energy consumption would be to keep a constant speed [10, 15]. In fact, the convexity of the power/speed curve implies that maintaining a constant speed α is better than switching between two different speeds. Unfortunately, this number of cycles is unknown in advance, hence we cannot determine the optimal speed α.

A common technique adopted in the literature [1, 22, 14] is based upon the idea of deferring some work, since we expect that the current instance of τ will request much less than its WCEC $C_{\max}$. This technique splits the task execution into two parts, as shown in Figure 1. In the first part, the task runs at a lower processor speed α_L in order to reduce the energy consumed in the average case. In the second part, instead, a higher processor speed α_H is used, so that we can provide up to $C_{\max}$ cycles even in the worst case. The idea is that, if a task tends to use much less than its WCEC, the second part, which consumes more power, may never be needed. When the worst-case condition occurs, instead, the speed increase guarantees the completion of all the deferred work within $[0, T]$.

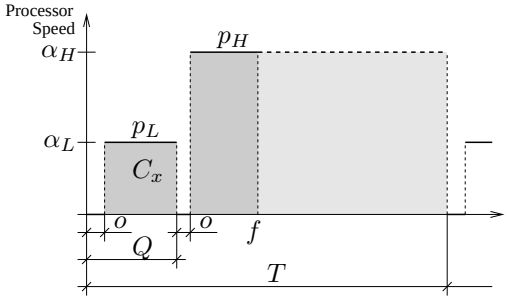

Figure 1. The energy management scheme.

The idea of deferring work has been widely used in the literature to create efficient power-aware algorithms. For instance, Pillai and Shin [14] proposed the RTDVS-Look Ahead algorithm, which tries to defer as much work as possible, and sets the operating frequency to the minimum value to ensure that all future deadlines will be met. This technique has also been used by Aydin et al. in the "aggressive" version of their DRA algorithm [1]. This algorithm speculatively assumes that current and future instances of the task will most probably present a computational demand lower than the worst case. Hence, it tries to reduce the speed of the running task by deferring all the work above a certain threshold, set according to the average workload. A similar

approach has been applied to EDF by Zhu and Mueller [22]. Each task's instance is divided into two portions. The objective is to provide the average number of cycles C_{avg} within the first portion. The second part at speed α_H ensures that the deadline is met even when the task requires C_{max} cycles.

Even if these techniques have been widely used in the literature, a probabilistic study of this model has not been proposed, yet. For instance, all previous algorithms set the speed α_H equal to the maximum possible value, even if this may not be optimal from the point of view of energy consumption. Even worse, some technique [22] is based on the intuitive idea that the optimal energy reduction is obtained by providing exactly the average execution cycles in the first part. In Section 3.1 we will prove that this intuition is not correct.

We decide to deeply study this model, extending it to the case in which probabilistic information about task execution time is known. Moreover, we use a general model for the processor, accounting for both the time and energy overheads of voltage transition.

3 Optimal speed assignment

Let α_L and α_H be the lower and the higher processor speeds, respectively. The period of the scheme is T. The number of processor cycles required by the task τ in each period is modelled by a random variable whose p.d.f. is $f_C(c)$, and the maximum number of cycles is C_{max}. This amount of cycles must be guaranteed in each period because the task is subject to a hard real-time constraint.

Our goal is to find the two speed levels α_L and α_H and the instant Q when to switch, in order to achieve the minimum energy consumption. Let C_x be the number of cycles provided while running at α_L, as shown in Figure 1. We can express (α_L, α_H) as a function of C_x and Q as follows

$$\alpha_L = \frac{C_x}{Q-o} \qquad \alpha_H = \frac{C_{\text{max}} - C_x}{T-Q-o}, \tag{1}$$

due to the constraint of providing C_{max} cycles within each period T.

Let also be c the number of cycles that actually occur and f the finishing time . We distinguish two different cases:

1. if $c \leq C_x$ then the task terminates before we could actually switch to α_H, and we expect $f \leq Q$;
2. otherwise, if $c > C_x$ then we need to run at speed α_H to provide the required cycles and we expect $f > Q + o$.

We consider the two cases separately. In order to have a more compact notation we set $p_H = p(\alpha_H)$ and $p_L = p(\alpha_L)$.

In the first case ($c \leq C_x$), the finishing time is

$$f = o + \frac{c}{\alpha_L}$$

and the energy consumed in one period T is

$$E = e + p_L\,(f - o) = e + \frac{p_L}{\alpha_L}\,c. \tag{2}$$

On the other hand, when $C_x < c \leq C_{\text{max}}$, we have

$$f = Q + o + \frac{c - C_x}{\alpha_H}$$

and the energy is

$$E = 2\,e + \frac{p_L}{\alpha_L}\,C_x + \frac{p_H}{\alpha_H}\,(c - C_x). \tag{3}$$

Equations (2) and (3) provide the energy E consumed when the number of cycles is c. Since the number of cycles is a random variable with p.d.f. $f_C(c)$, then the energy consumed is a random variable too. Our target then becomes to minimize the expectation E_{avg} of the random variable E. Let us compute this value.

$$\begin{aligned}
E_{\text{avg}} &= \int_0^{C_x} E\,f_C(c)\,\mathrm{d}c + \int_{C_x}^{C_{\text{max}}} E\,f_C(c)\,\mathrm{d}c \\
&= \int_0^{C_x} \left(e + \frac{p_L}{\alpha_L}\,c\right) f_C(c)\,\mathrm{d}c \\
&\quad + \int_{C_x}^{C_{\text{max}}} \left(2\,e + \frac{p_L}{\alpha_L}C_x + \frac{p_H}{\alpha_H}\,(c - C_x)\right) f_C(c)\,\mathrm{d}c \\
&= 2\,e + \frac{p_H}{\alpha_H}\,C_{\text{avg}} - \left(\frac{p_H}{\alpha_H} - \frac{p_L}{\alpha_L}\right) C_x \\
&\quad + \int_0^{C_x} \left(\left(\frac{p_L}{\alpha_L} - \frac{p_H}{\alpha_H}\right)(c - C_x) - e\right) f_C(c)\,\mathrm{d}c \\
&= e\,(2 - F_C(C_x)) + \frac{p_H}{\alpha_H}\,C_{\text{avg}} \\
&\quad - \left(\frac{p_H}{\alpha_H} - \frac{p_L}{\alpha_L}\right)(G_C(C_x) + C_x(1 - F_C(C_x)))
\end{aligned}$$

where

$$F_C(x) = \int_0^x f_C(c)\,\mathrm{d}c \qquad G_C(x) = \int_0^x c\,f_C(c)\,\mathrm{d}c.$$

For compactness we also set

$$\gamma(x) = G_C(x) + x(1 - F_C(x)), \tag{4}$$

so that the average energy consumed in a period is

$$E_{\text{avg}} = e\,(2 - F_C(C_x)) + \frac{p_H}{\alpha_H}(C_{\text{avg}} - \gamma(C_x)) + \frac{p_L}{\alpha_L}\,\gamma(C_x) \tag{5}$$

Notice that $G_C(C_{\text{max}})$ is equal to C_{avg}. For this reason we always have $0 \leq \gamma(x) \leq C_{\text{avg}}$ for all x.

Equation (5) is a new result in the literature because it expresses the average energy consumed as function of the probability density of the task execution cycles $f_C(c)$.

It is very insightful to plot the quantity E_{avg} on a plane (C_x, Q). Figure 2 shows the level curves of the quantity E_{avg} as function of C_x and of Q. In the plot we assumed an exponential p.d.f. with average value $C_{\text{avg}} = 0.2929$, the period T equal to 1 and the power function $p(\alpha) = k\,\alpha^3$.

As you can notice, the minimum at the center of the white

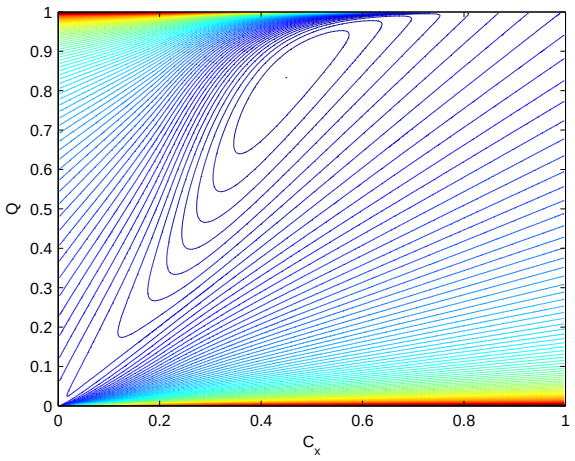

Figure 2. E_{avg} **for uniform execution times.**

region occurs for a value of C_x greater than C_{avg}. In order to find it analytically, we need to compute the partial derivative of E_{avg} with respect to the variables (C_x, Q). After opportune simplifications, we find that:

$$\frac{\partial E_{\text{avg}}}{\partial C_x} = -e\, f_C(C_x) - \left(p'_H - \frac{p_H}{\alpha_H}\right)\frac{C_{\text{avg}} - \gamma(C_x)}{C_{\max} - C_x} + \left(p'_L - \frac{p_L}{\alpha_L}\right)\frac{\gamma(C_x)}{C_x} - \left(\frac{p_H}{\alpha_H} - \frac{p_L}{\alpha_L}\right)\gamma'(C_x) \quad (6)$$

where p'_L and p'_H denote $p'(\alpha_L)$ and $p'(\alpha_H)$, respectively.

Now, we complete the analysis of the function E_{avg} by computing also $\frac{\partial E_{\text{avg}}}{\partial Q}$, which is

$$\frac{\partial E_{\text{avg}}}{\partial Q} = (p'_H \alpha_H - p_H)\frac{C_{\text{avg}} - \gamma(C_x)}{C_{\max} - C_x} - (p'_L \alpha_L - p_L)\frac{\gamma(C_x)}{C_x} \quad (7)$$

Equations (6) and (7) are the components of the gradient ∇E_{avg}. From functional analysis, we know that the minimum satisfies the condition $\nabla E_{\text{avg}} = 0$. Once the optimal (C_x, Q) is found, then the constraint $\alpha_H \leq \alpha_{\max}$ must be checked. In fact, if it is violated, it means that the global minimum would result in a too high value of α_H. It this case we know from the Kuhn-Tucker conditions that the minimum occurs when $\alpha_H = \alpha_{\max}$, which means that

$$\frac{C_{\max} - C_x}{T - Q - o} = \alpha_{\max} \;\Rightarrow\; \alpha_L = \frac{C_x\, \alpha_{\max}}{\alpha_{\max}(T - 2o) - C_{\max} + C_x} \quad (8)$$

From Eq. (5), substituting α_H with $\alpha_{\max}$ and α_L with the expression of Equation (8), we find E_{avg} as function of the unique variable C_x. The minimal energy solution is found by applying classical techniques of functional analysis of one-variable functions.

3.1 Polynomial power function

Once the main equations for the general case have been found, we show how they can be applied to find the optimal (C_x, Q) in some significant examples. Due to lack of space, we assume the time and energy overheads equal to zero (i.e. $o = 0$ and $e = 0$).

When considering continuous speed levels, a common assumption is that the relationship between the power consumption p and speed α is

$$p(\alpha) = k\, \alpha^n \quad (10)$$

for some k, n. The typical value of n is 3, however we keep the general form as long as the math is tractable.

In these hypothesis, the gradient can be greatly simplified. In order to find the point of minimal energy we have to set both the gradient components equal to zero. By setting $\nabla E_{\text{avg}} = 0$, we finally find that the pair (C_x, Q) minimizing the average energy E_{avg} must satisfy Equations (9) reported in Table 1. Due to lack of space we don't include all the calculations. For their importance we call the Equations (9) the **minimum stochastic energy equations**. Once we know n and the probability density $f_C(c)$, Equations (9) can be solved and produce the pair (C_x, Q) which minimizes the energy.

Uniform Density Let us now assume a uniform density between $C_{\min}$ and $C_{\max}$. It means that

$$f_C(c) = \begin{cases} \frac{1}{C_{\max} - C_{\min}} & \text{if } C_{\min} \leq c \leq C_{\max} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

and also, when $C_{\min} \leq c \leq C_{\max}$,

$$F_C(c) = \frac{c - C_{\min}}{C_{\max} - C_{\min}} \qquad G_C(c) = \frac{c^2 - {C_{\min}}^2}{2(C_{\max} - C_{\min})}. \quad (12)$$

The function $\gamma(c)$, defined in Eq. (4), is

$$\gamma(c) = \frac{-c^2 + 2c\, C_{\max} - {C_{\min}}^2}{2(C_{\max} - C_{\min})} \quad (13)$$

and its derivative is

$$\gamma'(c) = \frac{C_{\max} - c}{C_{\max} - C_{\min}} \quad (14)$$

In this case the minimum energy can be simply found by properly substituting $\gamma(C_x)$ and $\gamma'(C_x)$ in the minimum stochastic energy equations (9).

To simplify and compact them, it is very convenient to normalize the cycles C_x and $C_{\min}$ with respect to $C_{\max}$. Hence, we set $x = \frac{C_x}{C_{\max}}$ and $a = \frac{C_{\min}}{C_{\max}}$. Due to lack of space here we do not report all the simplifications, which can be accomplished by any symbolic manipulation tool.

When $n = 2$ the optimal number of normalized cycles x, which provides the minimal energy, is

$$x_{\text{opt}} = \frac{1 + \sqrt{1 + 3\, a^2}}{3}. \quad (15)$$

$$\begin{cases} \dfrac{(n-1)\,\gamma(C_x)+C_x\gamma'(C_x)}{(n-1)\,(C_{\text{avg}}-\gamma(C_x))+(C_{\text{max}}-C_x)\gamma'(C_x)}\left(\dfrac{C_{\text{max}}}{C_x}-1\right)\left(\dfrac{C_{\text{avg}}}{\gamma(C_x)}-1\right)=\dfrac{T}{Q}-1 \\ \left(\dfrac{C_{\text{max}}}{C_x}-1\right)^{n-1}\left(\dfrac{C_{\text{avg}}}{\gamma(C_x)}-1\right)=\left(\dfrac{T}{Q}-1\right)^{n} \end{cases} \tag{9}$$

Table 1. Minimum stochastic energy equations.

Instead, when $n = 3$, the solution is

$$x_{\text{opt}} = \frac{5-\sqrt{5}+\sqrt{2}\sqrt{5(3-\sqrt{5})-8(1-\sqrt{5})\,a^2}}{8} \tag{16}$$

Interestingly, this result proves that the approach proposed by Zhu and Mueller [22] is sub-optimal, as stated by themselves. In fact, they suggested to set C_x equal to C_{avg}. From both Equations (15) and (16) we see that **the optimal value is always greater than C_{avg}** (see also Figure 3). Providing C_{avg} cycles at speed α_L would lead to increase

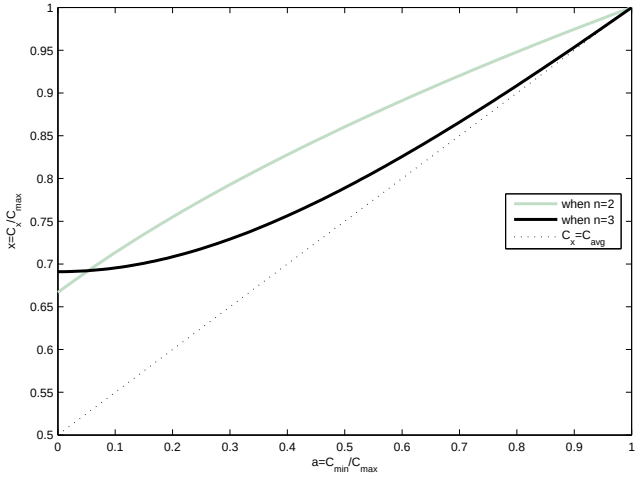

Figure 3. The optimal number of cycles.

the average energy consumed in a period.

Exponential Density The probability density considered previously is very simple and it allows to exactly find the pair (C_x, Q) which minimizes the average energy consumed. We consider now a more complex density $f_C(c)$ which better captures the characteristics of real execution times. Without loss of generality, we normalize the number of cycles toward C_{max} so that the possible values of cycles are in $[0, 1]$. As done before we set $a = \frac{C_{\text{min}}}{C_{\text{max}}}$.

We consider the following exponential p.d.f.:

$$f_C(c) = \begin{cases} \frac{1}{K}e^{\beta c}(1-c)(c-a) & \text{if } c \in [a,1] \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where K is a proper constant such that $\int_a^1 f_C(c)\mathrm{d}c = 1$.

The presence of β allows to alter the symmetry of the density. In fact, for negative β the density shifts to the left, meaning that values close to C_{min} are more likely to happen. On the other hand, positive values of β means that execution cycles close to C_{max} occur more frequently. Figure 4 shows some possible functions.

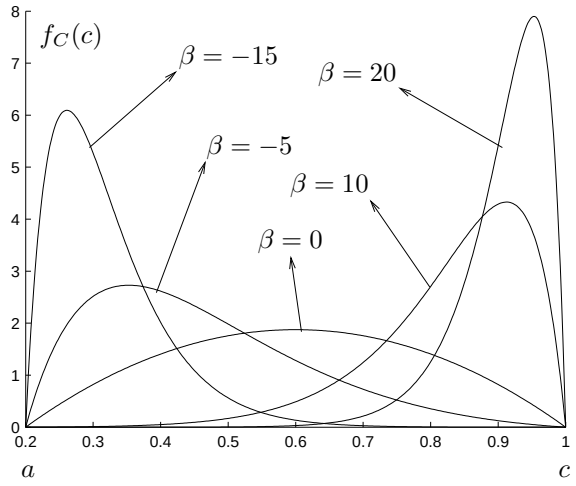

Figure 4. Exponential probability density functions.

For exponential densities, the minimal energy (C_x, Q) pair can only be found by numerical approximation. We investigated the effect of the p.d.f. asymmetry onto the solution. The result is quite interesting. In Figure 5 we plot the ratios $\frac{C_x}{C_{\text{avg}}}$ and $\frac{Q}{T}$, assuming $a = \frac{C_{\text{min}}}{C_{\text{max}}} = 0.2$. A first result, also noticed for uniform density, is that **the optimal C_x is always greater than C_{avg}**, differently than what suggested in a previous paper [22]. This fact is evidenced by the black curve which is always above 1. We also highlight that for big positive values of β (meaning that values close to C_{max} are more likely to occur), C_x tends to C_{avg}.

4 Conclusions and future work

Deferring the work is an effective technique already proposed in the literature to reduce the energy consumed by the processor. However, often this technique has been blindly applied, without a a systemic search of the minimal.

In this paper we have provided the foundations to integrate the probabilistic timing analysis with energy minimization techniques, starting from the simple case of one task. This problem has been studied using a general model

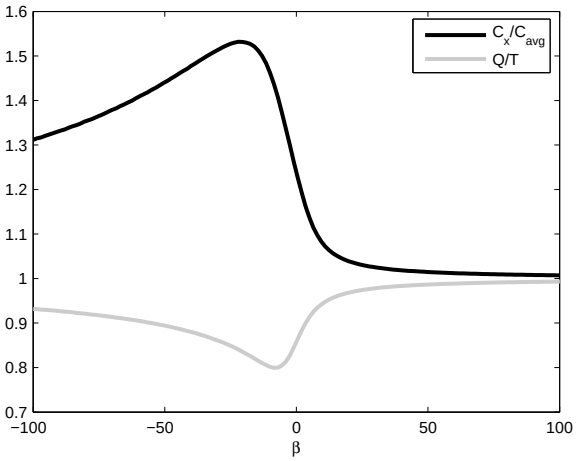

Figure 5. The optimal (C_x, Q) pair as function of the symmetry.

for the processor, taking into account both time and energy overheads. Thanks to this research, the design of effective energy minimization schemes using information about probabilistic execution times is now possible.

Finally, we refuted the idea that providing the average number of cycles at the lower speed is the best possible strategy.

References

[1] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, May 2004.

[2] G. Bernat, A. Colin, and S. M. Petters. pWCET: A tool for probabilistic worst-case execution time analysis of real-time systems. Ycs-2003-353, Computer Science dept., University of York, Feb. 2003.

[3] E. Bini, G. C. Buttazzo, and G. Lipari. Speed modulation in energy-aware real-time systems. In *Proc. of ECRTS*, Palma de Mallorca, Spain, July 2005.

[4] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. *The case for power management in web servers*. Kluwer Academic Publishers, 2002.

[5] A. Burns, G. Bernat, and I. Broster. A probabilistic framework for schedulability analysis. In *Proc. of EMSOFT*, Philadelphia, PA, Oct. 2003.

[6] A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[7] A. Ermedahl, F. Stappert, and J. Engblom. Clustered calculation of worst-case execution times. In *Proc. of CASES*, San José, CA, Oct. 2003.

[8] F. Gruian. *Energy-Centric Scheduling for Real-Time Systems*. PhD thesis, Computer Science dept., Lund Institute of Technology, Lund, Sweden, Nov. 2002.

[9] Intel, `http://www.intel.com/design/intelxscale/`. *Intel XScale Technology*.

[10] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proc. of the International Symposium on Low Power Electronics and Design*, Monterey, CA, Aug. 1998.

[11] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller. Energy management for commercial servers. *IEEE Computer*, Dec. 2003.

[12] Y. Liu and A. K. Mok. An integrated approach for applying dynamic voltage scaling to hard real-time systems. In *Proc. of RTAS*, Washington DC, May 2003.

[13] Motorola, `http://e-www.motorola.com/webapp/sps/library/prod_lib.jsp`. *MPC5200: 32 bit Embedded Processor*.

[14] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proc. of SOSP*, Banff, Canada, Oct. 2001.

[15] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proc. of MOBICOM*, Rome, Italy, July 2001.

[16] A. Qadi, S. Goddard, and S. Farritor. A dynamic voltage scaling algorithm for sporadic tasks. In *Proc. of RTSS*, Cancun, Mexico, 2003.

[17] S. Saewong and R. Rajkumar. Pratical voltage-scaling for fixed-priority RT-systems. In *Proc. of RTAS*, Washington, DC, May 2003.

[18] C. Scordino and G. Lipari. Using resource reservation techniques for power-aware scheduling. In *Proc. of EMSOFT*, Pisa, Italy, Sept. 2004.

[19] Transmeta, `http://www.transmeta.com/crusoe/`. *The Crusoe Processor*.

[20] J. Wegener and F. Mueller. A comparison of static analysis and evolutionary testing for the verification of timing constraints. In *Real-Time Systems*, Nov. 2001.

[21] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. of the* 36^{th} *Annual Symposium on Foundations of Computer Science*, Milwaukee, WI, Oct. 1995.

[22] Y. Zhu and F. Mueller. Feedback EDF scheduling exploiting dynamic voltage scaling. In *Proc. of RTAS*, Toronto, Canada, May 2004.