Proceedings of the 3rd International Conference on Systems and Control, Algiers, Algeria, October 29-31, 2013

A Real-time Feedback Scheduler for Environmental Energy Harvesting*

A. ABBAS[1, 3], E. GROLLEAU[2], M. LOUDINI[3], and D. MEHDI[4]

Abstract—**The use of environmental energy such as solar energy has recently emerged as an option to increase the operating time of embedded systems (e.g. wireless sensors). It consists in converting ambient energy into electricity to power and lengthen battery life. Dynamic voltage scaling (DVS) is one of the most effective techniques for reducing energy consumption in embedded and real-time systems. However, traditional DVS algorithms have inherent limitations on their capability in energy saving since they rarely take into account the actual application requirements and often exploit fixed timing constraints of the real-time tasks. Some authors used feedback scheduling techniques in order to minimize the consumed energy by observing the actual usage of resources in the system. This paper combines the use of DVS and energy harvesting with the capabilities of a feedback scheduler. Our goal is to minimize the consumed energy, as well as to take the charging model into account.**

I. INTRODUCTION AND RELATED WORK

Wireless and embedded systems are commonly powered using batteries. The battery life is an important quality metric; therefore, the reduction of the energy consumption becomes a crucial optimization criterion in the conception and the realization of these systems. For applications where the system is expected to operate for long durations, energy becomes a severe bottleneck and much effort has been spent on the efficient use of the batteries. More recently, another alternative has been explored to supplement or even replace batteries: harvesting energy from the environment.

Several authors treated the problems of power and scheduling with the objective of minimizing power usage under timing constraints. For example, EDF (Earliest Deadline First) and RM (Rate Monotonic) [1] scheduling algorithms have been extended to variable-voltage processors. The idea is to save power by slowing down the processor just enough to meet the deadlines [2], [3], [4]. For ambient energy harvesting, Moser et al. proposed LSA (Lazy Scheduling Algorithm) [5] to optimally schedule tasks with deadlines. Despite its optimality in terms of schedulability and battery usage, the drawbacks of LSA is that it needs an exponential complexity for periodic task sets, as well as a high complexity for some kinds of complex energy harvesting curves. Several simple deadline-based heuristics have been compared to LSA in [6]. In [7], a DVS algorithm (called EA-DVFS) is proposed to enhance the performance of LSA. This algorithm adjusts the processor's behavior depending on the sum of the stored energy and the harvested energy in a future duration. Particularly, if the system has a sufficient amount of energy, the tasks are executed at full speed; otherwise, the processor slows down to save energy. In [8], [9], the algorithm EDeg is using the slack time (the slack time [10] at a time t is the amount of time the processor can remain idle without missing a future deadline): this notion is extended to the notion of slack energy.

Up to our knowledge, the authors combining DVS and energy harvesting consider a worst-case behavior of the tasks. Nevertheless, the worst-case execution time (WCET) which is taken into account in the task models, considered in the literature concerning energy harvesting, is the upper bound of a highly volatile parameter. As an illustration, Fig. 1 is a histogram, taken from [11], representing the typical effective task duration observed on an important amount of executions of a task. The WCET, taken in the scheduling analysis as an upper bound of the task duration, is a far upper bound of the average execution duration. Since the energy harvesting scheduling techniques are based on the WCET, they overuse the battery in order to insure that task deadlines are met.

To overcome the limitations due to the WCET, in the case of soft real-time systems, the feedback scheduling [12], [13] has recently emerged as a promising technique for resource management. In particular, significant effort has been made on feedback scheduling of control systems [14], [15]. There are also several papers applying feedback control methods to DVS. For instance, the popular PID (Proportional- Integral-Derivative) control has been integrated into several DVS algorithms [16]. Solutions for integrated optimization of sampling periods and CPU speed have been presented in [17], [18]. A feedback fuzzy-DVS scheduling method has been developed in [19]. Marinoni and Buttazzo [20] presented an approach that integrates elastic scheduling with DVS management to fully exploit the available computational resources in processors with limited voltage levels. In [21], a solution is proposed to achieve further reduction in energy consumption over pure DVS while not jeopardizing the quality of control, the sampling period of each control loop is adapted to its actual control performance, thus exploring flexible timing constraints on control tasks. In [26], an approach combining feedback scheduling with DVS is presented for exploiting the slack time of schedule which is generated by actual executions of tasks that complete under their WCET. However, these algorithms have been used without considering energy harvesting capabilities.

* This work was not supported by any organization.

[1]Université Akli Mohand Oulhadj de Bouira, 10000 Bouira, Algeria; (a_abbas@esi.dz).

[2]Laboratoire d'Informatique et d'Automatique pour les Systèmes (LIAS) ISAE-ENSMA, France (grolleau@ensma.fr).

[3]Ecole Nationale Supérieure d'Informatique (ESI), Laboratoire de Communication dans les Systèmes Informatiques (LCSI), B.P 68M, 16270 Oued Smar, El Harrach, Algiers, Algeria; (m_loudini@esi.dz).

[4]Laboratoire d'Informatique et d'Automatique pour les Systèmes (LIAS), Université de Poitiers, France; (driss.mehdi@univ-poitiers.fr)

In this paper, we consider that the system is powered by rechargeable battery and it evolves in a dynamic environment in which the task parameters (mainly their execution times) are not fixed.

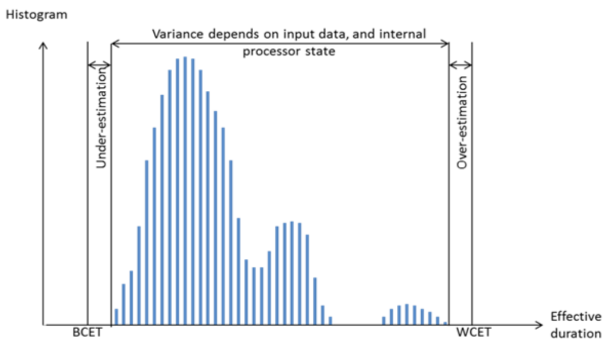

Figure 1. Histogram of effective task duration vs. WCET

The outline of the paper is as follows. In section 2, we describe the system model: tasks, processor ambient source model. Our main contribution concerning the feedback scheduling under energy harvesting is presented in section 3. Experimental results are included and discussed in Section 4. Finally, the main conclusions and discusses topics for future work are highlighted in section 5.

II. Models and Assumptions

This section describes the system model and energy source- consumption model, thus formulating the problem to be addressed in this paper.

A. *Task system model*

A real-time task is classically defined as a set of temporal parameters [1]. In the sequel, we consider independent tasks: they do not communicate or share resources, and cannot suspend themselves. This assumption can be removed in some subsequent work. We consider sets of *n* periodic real-time tasks, which are defined by:

- r_i the first release time of the task τ_i; every release of a task is named a job $\tau_{i,k}$, $k \geq 0$;
- $T_{i,nom}$ the nominal release period of the task τ_i: the job $\tau_{i,0}$ is released at the date $r_{i,0}=r_i$, and every subsequent job $\tau_{i,k}$ is released at the date $r_{i,k}=r_{i,k\text{-}1}+T_i$, with T_i derived from $T_{i,nom}$;
- $WCET_i$ is the Worst-Case Execution Time of every job of τ_i at full processor speed;
- D_i is the relative deadline of the task τ_i: we assume the absolute deadline of a job $\tau_{i,k}$ to be given by $r_{i,k}+D_i$.

Note that in our model, the release time (resp. absolute deadline) is computed given the previous release time (resp. the job release time). This definition comes from the fact that in a feedback scheduler, the period T_i may vary.

The underlying idea of feedback scheduling is that for tasks integrating a control loop (a task controlling an independent physical process), meeting the deadlines is not giving the best quality of the process response. Indeed, several factors, like the release jitter, and the input-output jitter, have a higher impact on the quality of the process response than meeting the deadlines or not. The jitter is an important metric for control tasks: a jitter may be defined in different ways, but it expresses the variation of the execution times between subsequent jobs. In some cases, it is better to increase the period of a control task in order to lower the jitters [14], than to increase the sampling period (which side effect may be to increase the jitters).

Since we consider DVS systems, the WCET of a task is given for a full processor speed, which is called unitary speed. We suppose that if the processor is running at a speed $0 \leq \alpha \leq 1$, the WCET of the tasks are scaled to match the speed. Using a DVS system, we can change the WCET and the actual execution times of every task at once. Therefore, at any time, the WCET of the tasks depends on α.

- $WCET_{i,\alpha}$ is the Worst-Case Execution Time of every job of τ_i at an α -processor speed;

Using a classic feedback scheduler (FS), we can also change independently the release periods of some chosen tasks in order to allow the DVS system to be scaled even more than when considering only the WCET of the tasks, consuming less power than without the FS. We will assume the FS to be executed periodically. For the FS part, we use, like in [15], the following notations:

- $C_{i,1}$ estimated execution time at full CPU speed;
- $C_{i,\alpha}$ actual estimated execution time of the task τ_i when the CPU speed is scaled by α. $C_{i,\alpha} = C_{i,1} / \alpha$;
- c_i measured execution time of the job $\tau_{i,k}$;
- T_i is the actual period of the task τ_i chosen by the FS;

To include energy consumption per task τ_i we consider the following notations:

- E_i is the Worst Case Energy Consumption (WCEC) by the task τ_i;
- $E_{i,\alpha}$ actual energy consumption by the task τ_i when the CPU speed is scaled by α.

To estimate the execution time $C_{i,1}$, we use a low-pass filter given below with a forgetting factor λ in [0, 1] proposed by Cervin in [15].We initialize the estimator of a task as its worst execution time (WCET). The index k represents the execution number of the task τ_i.

$$\begin{aligned} C_{i,1}(0) &= WCET_i \\ C_{i,1}(k) &= \lambda * C_{i,1}(k-1) + (1-\lambda) * c_i \end{aligned} \quad (1)$$

where, at each termination of the job $\tau_{i,k}$, $C_{i,1}(k)$ is updated.

The instantaneous state of the task system is given by α, the scaling speed of the CPU, and for every subsequent job $\tau_{i,k}$<$r_{i,k}$, $C_{i,1}$,T_i>.

A task system is schedulable by a scheduling policy if every job $\tau_{i,k}$ can be completely executed in the time window [$r_{i,k}$, $r_{i,k}+D_i$].

EDF is optimal regarding schedulability in the context of independent tasks. The density test is an easy way to test (linear time) a sufficient condition of schedulability for periodic tasks. When the tasks have their nominal period, and are executed by a unitary speed processor this schedulability test is [22]:

$$\Delta = \sum_{i=1}^{n} \frac{WCET_i}{\min\{D_i, T_{i,nom}\}} \leq 1 \qquad (2)$$

If the processor runs at a scale factor of α, this sufficient schedulability condition can be expressed as: $\Delta \leq \alpha$.

Given a system under a feedback scheduler, we can adapt the definition of the density test, during the j^{th} execution of the feedback scheduler, as the instantaneous processor utilization:

$$\Delta_{inst} = \sum_{i=1}^{n} \frac{C_{i,1}(k)}{\min\{D_i, T_i\}} \leq 1 \qquad (3)$$

where the term $C_{i,1}(k)$ is equal to the latest filtered value of the task τ_i according to (1).

$\Delta_{inst} \leq \alpha$ can be used as an indicative sufficient schedulability test on the average observed execution time during the execution of the feedback scheduler.

B. *Energy source model*

We assume that ambient energy is harvested and converted into electrical power. To model the solar energy source behavior, we use the following model [5].

$$Ps(t) = \left| 10 \times unif(t) \times \cos\left(\frac{t}{0.7 \times \pi}\right) \times \cos\left(\frac{t}{0.1 \times \pi}\right) \right| \qquad (4)$$

where *unif(t)* denotes a uniform distributed random variable between 0 and 1. The values of *Ps* have been cut off at the value *Ps, max = 10*. As illustrated in Fig. 2, the obtained power trace *Ps (t)* is simulating day and night periods similar to those experienced by solar cells in an outdoor environment.

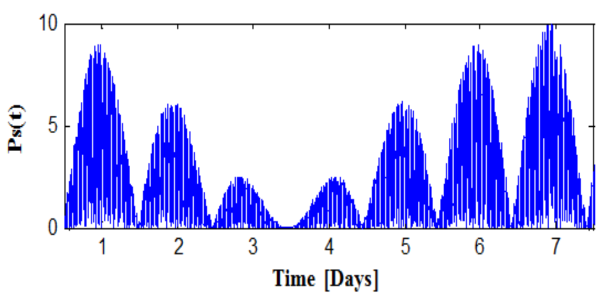

Figure 2. Power trace Ps(t)

In this energy source model, the input power *Ps (t)* has excluded the loss incurred by the auxiliary circuitry. In another word, *Ps (t)* is the net power to feed the storage. The harvested energy *Es (t1, t2)* by *PS (t)* at time interval *[t1, t2]* is given as:

$$Es(t1,\ t2) = \int_{t1}^{t2} Ps(t)dt \qquad (5)$$

The system uses an energy storage unit that has a nominal capacity (noted E), corresponding to a maximum energy (expressed in Joules or Watts-hour). The energy level has to remain between two boundaries E_{min} and E_{max} therefore the maximal available energy for the CPU is $E = E_{max} - E_{min}$.

C. *Energy consumption model*

A wide range of processors supports variable voltage and frequency levels which are tightly coupled. A processor is an integrated circuit of the CMOS family; such a circuit answers the following generic equations [23]:

Instantaneous power and energy consumed: Power consumption (with joules/second or watt for example) can be divided into two types, static and dynamic:

$$P_{CMOS} = P_{stat} + P_{dyn} \qquad (6)$$

where, P_{dyn} is power consumption in switching (dynamic) and P_{stat} is power consumed by a CMOS gate at rest (static).In CMOS circuits, the dynamic power represents about 80-85% of the power-dissipated. Conventionally, we neglect the static power. The total power dissipated can be expressed by:

$$P_{CMOS} \approx P_{dyn} \qquad (7)$$

The total energy consumed (in joules), at t operating time, is given by the following formula:

$$E = \left(P_{stat} + P_{dyn}\right) \cdot t = E_{stat} + E_{dyn} \approx E_{dyn} \qquad (8)$$

Relationship between frequency and supply voltage: The operating frequency of the CMOS circuits is given by.

$$f \approx \frac{(v - v_t)^{\gamma}}{v} \qquad (9)$$

with γ a constant, v_t is the threshold voltage. In the model of metal oxide semiconductor field effect transistor (noted MOSFET) a classical value for γ is approximated by two [24]. For a threshold voltage sufficiently small relatively to the supply voltage, the ency and supply voltage becomes.

$$f \approx v \qquad (10)$$

where f is the operating frequency and v is

Relationship between power, energy, frequency and supply voltage: The dynamic power is calculated by.

$$P_{dyn} = C \cdot f \cdot v^2 = C \cdot f^3 = C \cdot v^3 \qquad (11)$$

where C is a constant related to the type of processor.

Under the operating frequency f, a task requires n clock cycle's, which will give a n/f seconds. Thus, we obtain a consumed energy equal to:

$$E_{dyn} = \frac{n}{f} * P_{dyna} = n * C * f^2 = n * C * v^2 \qquad (12)$$

D. Dynamic Voltage Scaling

The Dynamic Voltage Scaling aims at the dynamic adaptation of processor voltage and thus, its frequency, to the current needs of the application in terms of performance. The scheduler, in this case, is not only defining the order of the tasks to be executed by the processor. It also has to define the processor speed. We assume that the speed can be varying continuously from S_{min} to the maximum supported speed S_{max}. We normalize the speed to the maximum speed to have a continuous operating range of [S_{min}, S_{max}], where $S_{min}=f_{min}/f_{max}$ and $S_{max}=f_{max}/f_{max}=1$, with the minimum and maximum frequencies represented by f_{min} and f_{max} respectively. The important point to note is that, when we perform a slowdown factor α, we change both the frequency and supply voltage of the processor with a slowdown factor $\alpha \in [S_{min}, 1]$.

The key idea is that, if we consider periodic tasks and if the instantaneous densityΔ_{inst} given in (3) is less than 1 and the processor operates with a maximum speed, it is possible to reduce the processor speed (so increasing the execution times of tasks) up to a speed α which gives an instantaneous density factor equal to 1, simply take $\alpha=\Delta_{inst}$.

The current processor frequency noted f expressed in terms of slowdown factor

$$f = \alpha \times f_{\max} \tag{13}$$

By referring to the above equations, the current energy consumption relative to the current slowdown factor α can be written:

$$E_{dyn} = C \times f^{2} = E_{dyn\_\max} \times \alpha^{2} \tag{14}$$

with E_{dyn_max} the maximum energy consumption under the maximum frequency f_{max}.

III. Feedback Scheduling under Energy Harvesting

We present here a new scheduler algorithm noted EDfbs-eg (Earliest Deadline Feedback Scheduling with Energy Guarantee) in order to enhance the performances of classical EDF under energy constraint. Before presenting our algorithm, we consider the example given in [25] to compare EDeg (which consists in putting the processor in the idle state when the available energy E=0) and EDfbs-eg. For this, we consider the set of tree tasks given in [25] $\Gamma = \{\tau i \mid 1 \le i \le 3\}$ and $\tau_i = (C_i, D_i, T_i, E_i)$ with E_i is the worst case energy consumption (WCEC) of the task τ_i. Let $\tau_1 = (1, 5, 6, 12)$, $\tau_2 = (2, 8, 10, 15)$ and $\tau_3 = (4, 11, 15, 22)$. We assume that the energy storage capacity is $C = 25$ energy units at $t = 0$. For simplicity, we assume that the rechargeable power is constant along time with ($Ps = 5$). We choose a forgetting factor λ equal to 0 in order to have $C_{i,1} = c_i$.

A. Case 1: rechargeable power is constant along time but without EDeg and Edfbs-eg

We show below the scheduling of the given configuration Γ for a classical EDF policy at full processor speed. In this case, the system stops at time t = 22. This is due to the depletion of the battery (Fig. 3). The average energy available is E = 7.14.

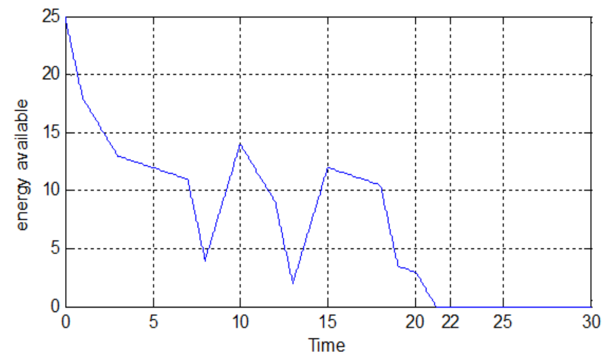

Figure 3. Energy available without EDeg and EDfbs-eg

B. Case 2: rechargeable power is constant along time with EDeg

When scheduling the configuration Γ under EDeg the system never stops. A time $t = 19$, an idle time of four time units is inserted: its duration is computed using the slack time. The variation of the energy consumption is shown in Fig. 4. In this case, the average available energy is equal to E = 13.03.

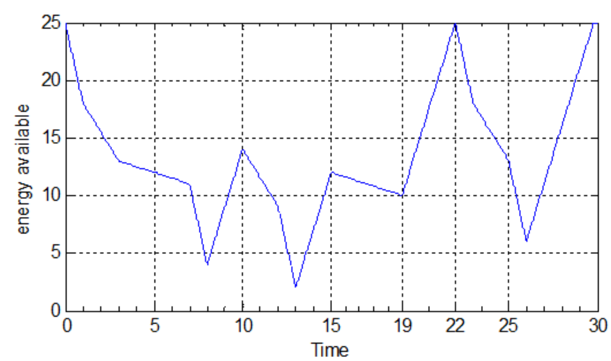

Figure 4. Energy available with EDeg

C. Case 3: rechargeable power is constant along time with EDfbs-eg

The scheduling configuration in EDfbs-eg causes some deadline misses. However, in terms of consumed energy, the algorithm allows EDfbs-eg to consume way less than EDeg (see Fig. 5). In this case, the average available energy is E = 19.43

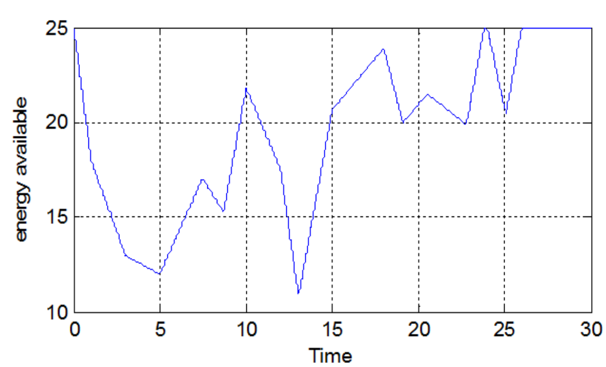

Figure 5. Energy available with EDfbs-eg

D. The EDfbs-eg algorithm

When a task (other than FS) ends its execution, its actual estimated execution time $C_{i,\alpha}$ is evaluated according to (1) For each execution of FS the instantaneous density Δ_{inst} for all tasks is evaluated according to (3) and is taken as the new scaling factor α.

IV. SIMULATION RESULTS

In order to simulate the algorithm EDfbs-eg, we used the tool TrueTime [14] in which we implemented a random task execution time generator according to a weibull low given below in (15), with $k = 3$, $\beta = 1.5$, and U is an uniformly distributed random variable between 0 and 1. We choose these parameters so that the *max* of c_i is less than the $WCET_i$,

$$c_i = (-\log(1-U)^{1/k} \times \beta) \quad (15)$$

The goal of this generator is to simulate a typical distribution (Fig. 6) of actual execution times.

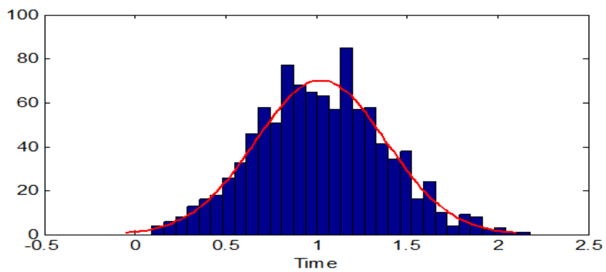

Figure 6. Distribution of actual execution times

We have implemented a power generator source which supplies a battery according to the model given in (4). We choose the forgetting factor of Eq. (1) λ=0.9.We consider the tasks system Γ but for these experiments, we consider that the actual execution times of each instance of the tasks are given by the random generator (15).

In order to compute the rate of missed deadlines, the average and minimum energy available, we perform 100 simulations according to the specifications listed above.

A. Rate of missed deadlines

In this section, we present results of simulations performed to compute the rate of missed deadlines for each simulation where the results are shown in Fig. 7. This rate is equal to the number of instances of tasks that have missed their deadlines divided by the number of instances of all tasks. The maximum rate of missed deadlines is equal to 7.69%. The average rate for 100 simulations is equal to 0.84%.

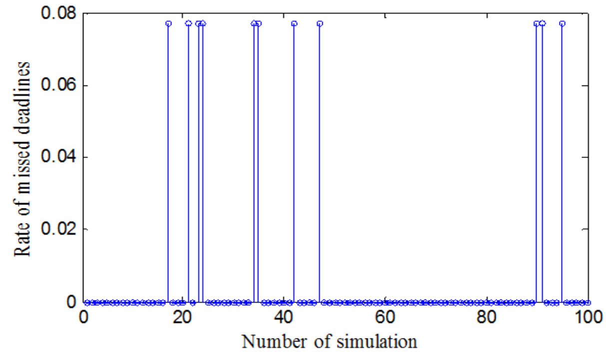

Figure 7. Rate of missed deadlines under EDfbs-eg

From Fig. 7, we can conclude that the EDfbs-eg algorithm causes in average a small amount of deadline misses. Nevertheless we know that in the control systems the number of missed deadlines is less important than their execution frequencies. An example is the control of an inverted

in the position of stable equilibrium (an angle of 180°). Miss some samples (deadlines) of

m position, but with more

pendulum will fall.

B. Average and minimum energy available

We also measured for each simulation, the average and minimum energy available. The results are illustrated in Fig. 8.

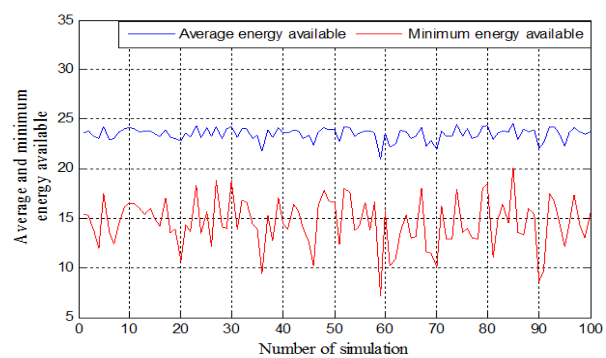

Figure 8. Energy available with EDfbs-eg for 100 simulation

As shown in Fig. 8, the EDfbs-eg algorithm saves a significant amount of energy where the average energy available is near to the maximum capacity of the battery (here it equals to 25). This algorithm protects in average against a total discharge of the battery, since the minimum energy available for each simulations is superior to 5.

C. Quality of control

In this section, we consider an embedded control system that consists of three independent control loops. Each plant is controlled using a PID algorithm where parameters are well designed and remain the same as those in [14]. The transfer function of each plant is $G(s) = 1000/(s^2 + s)$. In our experiments, the nominal sampling periods of three loops are set to be $T_{1,nom}$ =0.006 s, $T_{2,nom}$ =0.01 s, and $T_{3,nom}$ =0.015 s, respectively. To measure the QoC (Quality of Control), the Integral of Absolute Error (IAE) is used for each loop, i.e $J_i(t) = \int_0^t e_i(t)dt$, where the absolute control error e_i is defined as the absolute difference between the reference input x_i (blue line) and the system output y_i (red line), i.e., $e_i = |x_i - y_i|$. The total control cost of the system is calculated as $J_{sys}(t) = \sum_{1}^{3} J_i(t)$. We assume that the energy storage capacity is C = 6 energy units at t = 0. In the first experiment (case 1), we consider an EDF policy at full processor speed where the results of control performance are shown in Fig. 9.

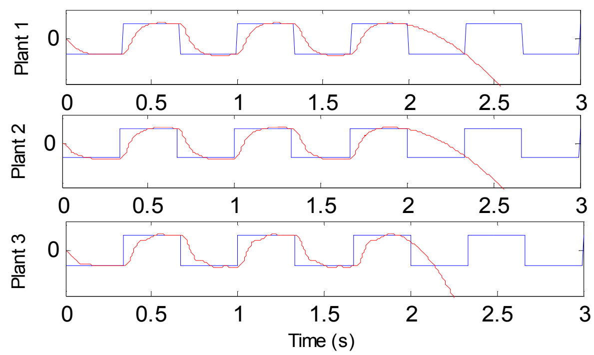

Figure 9. Control performance of three plants at full speed (case 1)

From Fig 9, we can see that all plants remains stable and exhibit satisfactory performances until t= 1.897 s. After this time the execution tasks stops and the plants become unstable and will fall. This is due to the depletion of the battery. In this case, the total control cost of the system is equal to J_{sys}= 25704.16. The average energy available is equal to 2.23 with considering the cycle of charge and discharge of battery and the minimum energy available is equal to 0.

The EDF schedule produced before the system stops and will fall is depicted in Fig. 10.

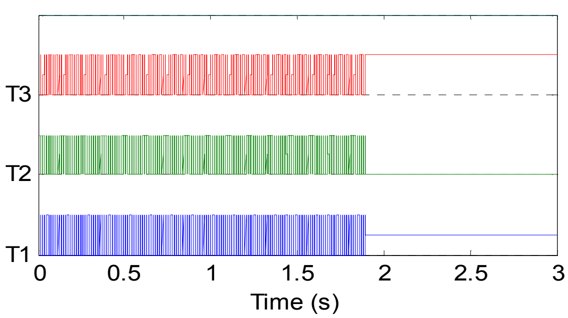

Figure 10. EDF schedule at full speed

In the second experiment (case 2), we use the EDfbs-eg with a FS period equal to 0.007 s. The results are shown in Fig.11.

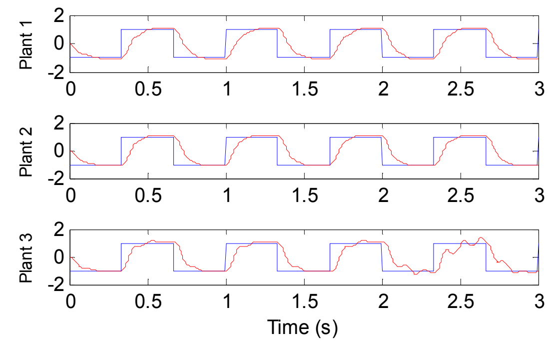

Figure 11. Control performance of three plant with EDfbs_eg (case 2)

From Fig. 11, we can see also that the three plants remains stable and never stops with the total control cost of the system is J_{sys} = 13156.54. We note that in this case, the rate of missed deadlines is equal to 5.68 %. Nevertheless, the average energy available is equal to 3.46 and the minimum energy available is equal to 1.49 which is better than for EDF.

To show the impact of the choice of the FS period, we performed a third experiment (case 3) with a FS period equal to 0.02 s. The results of control performance are shown in Fig. 12.

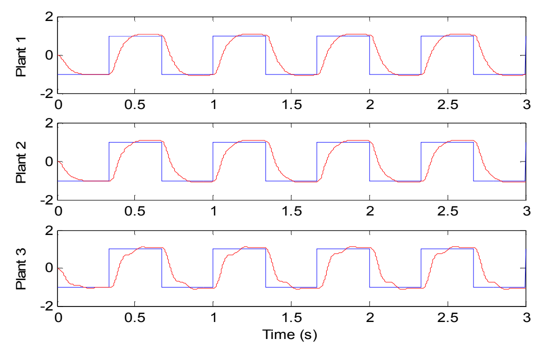

Figure 12. Control performance of three plant with EDfbs_eg (case 3)

From Fig. 12, we can see also that the three plants remain stable with the total control cost of the system J_{sys} = 10479.67. We note that in this case the rate of missed deadlines is equal to 0 %. The average energy available is equal to 2.92 where the minimum energy available is equal to 0.26.

V. CONCLUSION AND FUTURE WORKS

We have shown through a case study that the energy consumption can be drastically reduced in the case of soft real-time system. Moreover, the number of missed deadlines is limited giving a weibull distribution of the execution times. For some types of tasks in embedded systems, like process control tasks, meeting the deadlines is less important than a relative regularity of their execution. In the future, we plan to combine this approach with a delay control taking a maximum lateness of the control task into account, insuring the convergence of the state of the system towards the command. In the other work, we plan to consider some recent works with adaptive control withe considering energy harvesting capabilities

References

[1] C. L. Liu, and J. W. Layland, “Scheduling algorithms for multiprogramming in real-time environment”. Journal of the ACM 20(1): p. 46–61. 1973

[2] F. Yao, A. J. Demers, and S. A. Shenker, “Scheduling Model for Reduced CPU Energy”, in IEEE Symposium on Foundations of Computer Science, pp. 374-382. 1995

[3] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez, “Determining optimal processor speeds for periodic real-time tasks with different power characteristics”. Euromicro Conference on Real-Time Systems, pp. 225-232. 2001

[4] D. Qian, Z. Zhang, X. Tian and C. Hu “Low power scheduling for periodic real-time systems with Dynamic Voltage Scaling processor”. International Conference on Computer Application and System Modeling (ICCASM) 11:244-248. 2010

[5] C. Moser, D. Brunelli, L. Thiele and L. Benini. “Real-time scheduling for energy harvesting sensor nodes”, Real-Time Systems, Volume 37, Issue 3, Pages: 233 - 260, 2007.

[6] M. Chetto and H. Zhang, “Performance Evaluation of Real-Time Scheduling Heuristics for Energy Harvesting Systems”, Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, GREENCOM-CPSCOM '10, pp. 398-403,2010.

[7] S. Liu, Q. Qiu and Q. Wu, “Energy aware dynamic voltage and frequenc selection for real-time systems with energy harvesting”. Proceedings of thconference on Design, automation and test in Europe 236-241. 2008.

[8] H. El Ghor, M. Chetto and R.H. Chehade “A real-time scheduling framewor for embedded systems with environmental energy harvesting”. Computers an Electrical Engineering 37: 498-510. (2011)

[9] M. Chetto, D. Masson, and S. Midonnet, “Fixed priority scheduling strategies for ambient energy-harvesting embedded systems,” Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference, 2011.

[10] J. P. Lehoczky and S. Ramos-Thuel, “An optimal algorithm for scheduling soft-aperiodic tasks fixed priority preemptive systems,” in proceedings of the 13th IEEE Real-Time Systems Symposium, Phoenix, Arizona, , pp. 110–123, Dec. 1992.

[11] P. Axer, R. Ernst, H. Falk, A. Girault, D. Grund, N. Guan, B. Jonsson, P. Marwedel, J. Reineke, C. Rochange, M. Sebastian, R. von Hanxleden, R. Wilhelm and W. Yi, “Building Timing Predictable Embedded Systems”, ACM Transactions on Embedded Computing Systems, 2012.

[12] F. Xia, “Feedback Scheduling of Real-Time Control Systems with Resource Constraints”, PhD thesis, Zhejiang University, 2006.

[13] K.-E. Årzén, A. Robertsson, D. Henriksson, M. Johansson, H. Hjalmarsson and K. H. Johansson, “Conclusions of the ARTIST2 Roadmap on Control of Computing Systems”, ACM SIGBED Review 3(3) 11-20, (2006)

[14] A. Cervin, D. Henriksson, B. Lincoln, J. Eker and K.-E. Årzén “How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime”. IEEE Control Systems Magazine, 23:3, pp. 16–30, 2003.

[15] A. Cervin. “Integrated Control and Real-Time Scheduling”. Thèse PHd. s.l., Department of Automatic Control - Lund Institute of Technology, Sweden :Lund Tryckeri AB, 2003.

[16] A. Soria-Lopez, P. Mejia-Alvarez and J. Cornejo,“Feedback Scheduling of Power-Aware Soft Real-Time Tasks”, in: Proc. of the Sixth Mexican Int. Conf. on Computer Science (ENC'05), , pp. 266-273 , 2005

[17] H. S. Lee and B. K. Kim, “Dynamic Voltage Scaling for Digital Control System Implementation”, Real-Time Systems 29 263-280, 2005

[18] H. A. Wang, H. Jin, H. Wang and G. Z. Dai, “EnergyAware Optimization Design of Digital Control Systems with Evolution Strategy”, Dynamics of Continuous, Discrete and Impulsive Systems - Series B, 13(S1) 1893-1898, 2006

[19] H. Jin, D. L. Wang, H. A. Wang, H. Wang, “Feedback fuzzy-DVS scheduling of control tasks”, Journal of Supercomputing 41(2) 147-162, 2007

[20] M. Marinoni and G. Buttazzo, “Elastic DVS Management in Processors With Discrete Voltage/Frequency Modes”, IEEE Trans. Industrial Informatics 3(1) 51-62, 2007

[21] F. Xia, L. Ma, W. Zhao, Y. Sun, and J. Dong, “Enhanced energy-aware feedback scheduling of embedded control systems”, Journal of Computers, , 4(2): 103-111, 2008.

[22] J. Liu, “Real-Time Systems”, Prentice Hall, 2000

[23] W. C. Kwon, and T. Kim, “Optimal Voltage Allocation Techniques for Dynamically Variable Voltage Processors”. ACM Transactions on Embedded Computing Systems, Vol. 4, No.l, pp.211 -230, 2005,

[24] H. Aydin, R. Melhem, D. Mosé and P.Mejia-Alvarez. “Determining optimal processor speeds for periodic real-time tasks with different power characteristics”. In Euromicro Conference on Real-Time Systems, pages 225–232, 2001.

[25] M. Chetto, H. El Ghor and R. H. Chehade, “real-time scheduling for energy harvesting sensors”, The 6th International Conference for Internet Technology and Secured Transactions, Abu Dhabi : United Arab Emirates Page(s): 396 - 402 , 2011.

[26] A. Dudani, F. Mueller, and Y. Zhu. Energy-conserving feedback edf scheduling for embedded systems with realtime constraints. In ACM SIGPLAN Joint Conference Languages, Compilers, and Tools for Embedded Systems (LCTES’02) and Software and Compilers for Embedded Systems (SCOPES’02), pages 213.222, June 2002.