

# Moving Average Frequency Reduction for Low Power in Hard Real Time Systems

M.Angels Moncusí, Alex Arenas  
{amoncusi,aarenas}@etse.urv.es

Dpt d'Enginyeria Informàtica i Matemàtiques  
Universitat Rovira i Virgili  
Campus Sescelades, Av dels Països Catalans, 26  
43007 Tarragona, Spain

Jesus Labarta

jesus@ac.upc.es

Dpt d'Arquitectura de Computadors  
Universitat Politècnica de Catalunya  
Jordi Girona, 1-3. D6 Campus Nord  
08034 Barcelona, Spain

## Abstract

*We present a new policy to improve power saving in hard real time systems guarantying all tasks deadlines based on a moving average frequency reduction. Our study focus on the improvement obtained using this policy on the low power dual priority scheduling algorithm [3]. The resulting modified algorithm uses the total workload, the task execution history and the breakdown utilization to estimate the average minimum frequency of the processor to accomplish maximal energy reduction while meeting deadlines. The moving average strategy has been proposed to estimate the empirical execution time beyond the WCET, then updating the processor frequency for every task accordingly. We have performed extensive simulations that show a considerable enhancement in energy saving compared to original low power dual priority scheduling algorithm.*

Keywords: Static priorities, power aware scheduling, Dynamic Voltage Scaling, Worst Case Execution Time.

## 1. Introduction

The energy consumption in portable and hard real time systems is a fundamental problem in the design of modern computational devices [1]. A lot of efforts have been made during the last decade to minimize this drawback, but the high performance of modern microprocessors and micro-controllers jointly with the increasing functionality of them obtained via software still require improvements in the power-efficiency context.

The dynamic power consumption in CMOS circuits is given by the equation  $P \cong p_t C_L V_{DD}^2 f$ , where  $P$  is the power consumption,  $p_t$  is the probability of switching in power transition,  $C_L$  is the load capacitance,  $V_{DD}$  is the voltage supply and  $f$  is the operating clock frequency. Since the power has a quadratic dependency on the supply voltage, it is always energetically favorable scaling the voltage supply down. If the processor uses the voltage scaling technique to scale frequency, the relation between power and frequency is given by  $P(f) \propto C_L V_{DD}^2 f \approx k f^3$  [2-4]. The main techniques that take advantage of this non-

linear dependence are: Clustering Voltage Scaling and Dynamic Voltage Scaling (CVS and DVS) [5-6] Its functioning is based on the reduction of the voltage supply along with the processor frequency, and have been successfully used in many applications.

In hard real time systems these techniques could affect adversely the system performance, because time restrictions are critical. Nevertheless, the DVS technique is used in hard real time systems via power aware scheduling algorithms that determine the operating frequency of the processor that guarantees all real-time constraints while minimizing the energy consumption. Generally speaking, the scheduling algorithms reduce the voltage supply along with the processor operating frequency whenever the full system performance is not necessary and the tasks deadlines are not going to be compromised. Basically, these power aware schedulers use the idle time intervals to slow down the processor, executing tasks at reduced operating frequency.

To calculate the abovementioned reduction of the operating frequency, there are mainly two different approaches: static and dynamic [4,6-10]. In the static approach, the frequency is calculated off-line, before runtime, for each task independently. Once the execution starts the frequency could be readjusted on-line depending on the dynamic slack that has been generated – i.e. part of the worst case execution time not consumed. In the dynamic approach, the operating frequency is calculated on-line, just before running each task, once the scheduler knows exactly what the history about the previous executed tasks have been and when the rest of tasks will arrive [4,8,10].

We will focus our attention on the dynamic approach. In this approach, whenever it exist more than one task in the system, the operating frequency reduction could be performed following at least three general policies:

- Executing ready tasks at the maximum processor operating frequency, and reducing the operating frequency only to execute the last task in the system. This conservative approach cannot use the idle time

that could appear if the last task does not consume all its WCET because there is no task ready to be executed. The Cycle-conservative RT-DVS [8], the dynamic Reclaiming algorithm [10] and the Low Power Fixed Priority algorithm [11] uses a similar policy. See the sketch a) in Figure 1.

- Executing the first task at reduced speed and the following tasks at maximum operating frequency, this is a greedy approach. The first task executed uses the maximum possible idle time to reduce the clock operating frequency while the rest of tasks have to be executed necessarily at the maximum operating frequency. The advantage of this policy is that if a task does not consume all its WCET, the following task can use this time and then its operating frequency can be reduced. The Look-Ahead RT-DVS [8], the Aggressive Speed Reduction [10] and the Power Low Modified Scheduling Algorithm [3,12] uses a similar policy. See the sketch b) in Figure 1.
- Executing all ready tasks at some reduced operating frequencies whenever is possible. This scheme is similar to the static calculation of the operating frequency. Within this scheme all tasks execute at reduced operating frequency, trying to avoid any task execution at maximum operating frequency. In this case if the tasks finishes earlier, the slack generated can be used to reduce the operating frequency of the next task. See the sketch c) in Figure 1.

In this paper, we expose how to implement a dynamic approximation to the last policy in a dual priority scheme.

To motivate our work, based on the execution of tasks at a certain average frequency, let us compare the differences in energy saving obtained using the three aforementioned policies in a toy model. Let us assume a task set formed by two tasks,  $task_1$  and  $task_2$ , with a period and deadline of 100 time units, and a WCET of 30 time units each. Their execution is sketched in Figure 1.

Depending on the different operating frequencies at which the tasks represented in Figure 1 are executed, the total energy consumption values are: a)  $E=3.52$ , b)  $E=3.52$  and finally in c)  $E=2.16$ . Note that, in the latest policy, both tasks execute at a certain average operating frequency that shows to be clearly energetically favorable. This efficiency relies on the fact that the maximum operating frequency has been avoided, and because the relation between energy consumption and operating frequency is quadratic.

Note that the value for the average frequency used in Figure 1c corresponds exactly to the processor utilization percentage, in our case 60%, and this indicates a strong relationship between them. A similar approach considering the frequency reduction as a function of the processor utilization has been used in [9,10] for the Earliest Deadline First scheduling (EDF).

The rest of the paper is structured as follows: in section 2 we set the framework of the system, in section 3 we present a simple example of efficient reduction for Rate Monotonic scheduling. In section 4, we expose the modifications in the low power dual priority algorithm. In section 5, we compare the efficiency of the described policy with two energy aware scheduling algorithms. Finally in section 6 we present the conclusions of the current work.

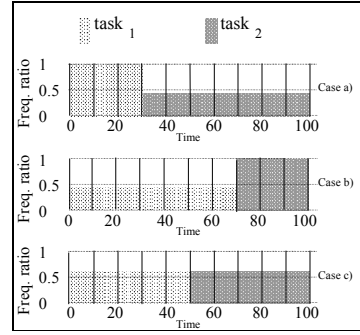


Figure 1: Three possible policies for tasks execution at different operating frequencies: a)  $task_1$  is executed at the maximum operating frequency, and  $task_2$  is executed at 0.42 of the maximum operating frequency; b)  $task_1$  is executed at 0.42 and  $task_2$  executes at the maximum operating frequency; c) the execution of both tasks is at 0.6 of the maximum operating frequency.

## 2. Framework

We consider task sets consisting on  $n$  independent periodic tasks,  $\tau_1 \dots \tau_n$ , each task  $\tau_i$  characterized by a 3-tuple  $(C_i, T_i, D_i)$ , where  $C_i$  is the worst case execution time of  $\tau_i$ . For each task instance the execution time varies from  $0.1 \cdot C_i$  to  $C_i$ .  $T_i$  stands for its period (or minimum inter-arrival time), and  $D_i$  is its relative deadline. The tasks sets are scheduled using a fixed priority pre-emptive algorithm in a multi-operating frequency processor.

The computation time overhead for context switching and for the scheduler are assumed to be negligible. The extent to which these assumptions are realistic is discussed in the analysis of the algorithm given in [13] and it turns out to be practical if the switch is subsumed to the worst-case execution times of the different tasks. We also assume that the voltage scaling overhead is negligible; the safeness of the system under these conditions is proved on theorem 1 of the work by Shin and Choi [11]. When the processor is powered down we consider zero energy consumption. Note that we also are assuming that the energy consumption is minimized whenever the supply voltage is scaled down. A recent work of Miyoshi et al. [14] has pointed out that there exists some practical processors with energy-inefficient operating frequencies for which this hypothesis does not hold, in these cases the current approach should be correctly to avoid entering the range of non-operative frequencies.

The whole system will be characterized by the processor utilization (U) and the breakdown utilization (BU) [15,16]. The Processor Utilization (U) is defined as

$$U = \sum_{i=1}^n \frac{C_i}{D_i} \quad (1)$$

and the BU is defined as the fraction of the utilization factor that marks the border for a system to be schedulable. To calculate BU, each execution time  $C_i$  in a given task set is multiplied by a constant scaling factor  $\alpha$ , while the periods remain fixed. The task set is scaled to the point at which it is just schedulable, such that any increment of  $\alpha$  would cause at least one task to miss its deadline. The utilization of the task set at that point,  $\Sigma_i (\alpha C_i / T_i)$  is the BU [16]. This scaling factor affects schedulability the same way the operating frequency reduction affects, then the BU should be considered as a lower bound to the static operating frequency reduction.

### 3. Average frequency reduction policy

Suppose that we have a system with only one task whose period and deadline is set to 100 time units and whose WCET is set to 60 time units. The processor utilization is 60%, the hyper-period is 100 and the Breakdown Utilization is 100 %, that is, we can scale the task set up to a real utilization of 100% (i.e. no idle time). In this simple situation, the execution frequency should be set exactly to 0.6 of the maximum processor frequency corresponding to 60(workload)/100(time units). Note that in this case the frequency reduction is optimal, i.e. a reduction below 0.6 makes the system not to meet the deadlines and a reduction over 0.6 will imply larger energy consumption.

Let us now consider an heterogeneous task set (see Table 1). The system is characterized by  $U=80\%$  and  $BU=88\%$ . If the scheduler extrapolates the average frequency policy presented before, the estimated frequency turns out to be 0.8. Using this frequency, the WCRTs (Worst Case Response Time) of tasks are 3.75, 26.25 and 63.75 respectively, and therefore  $Task_3$  misses its deadline. This simple example shows that this frequency reduction is not feasible due to the real time constraints. A more accurate estimation of the average frequency in this case should take into account that the spanning time of tasks in a Fixed Priority system is constrained by the BU to 88%, then the appropriate frequency should be represented by the ratio  $U/BU$ . Using this ratio, the frequency reduction is 0.91 and the WCRTs of tasks are 3.34, 23.36 and 59.4 respectively, consequently all deadlines are meet.

	Period	Deadline	WCET	WCRT
Task <sub>1</sub>	10	10	3	3
Task <sub>2</sub>	40	40	12	18
Task <sub>3</sub>	60	60	12	33

Table 1: Characteristics of the task set.

The determination of the average frequency reduction in a more general situation where there are N tasks in the system competing for real time execution is far more complicated. In this scenario, the tasks priorities as well as the constraints imposed by the deadlines increase the complexity of the optimization problem. An off-line optimization algorithm will not provide the correct values due to the dynamic interferences that take place on-line, and on the other hand an optimization algorithm on-line will require as much computational resources as the real time system itself.

Our idea in this general case is to use the ratio  $U/BU$  as an estimation of the average frequency. To show the reliability of this estimation, we analyze the behavior of Rate Monotonic scheduling using an average frequency reduction policy.

In Figure 2, we present the results of the average operating frequency for different Us and BUs. Symbols represent the empirical frequency we found via simulation (we simulate the execution of all tasks sets reducing progressively the processor operating frequency from the maximum to the minimum, to all tasks in each task set. We stop when one deadline is missed), while the lines represent the theoretical estimation of this frequency using the ratio  $U/BU$ .

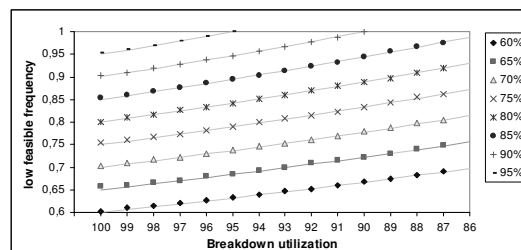


Figure 2: Lowest feasible frequency vs breakdown utilization for 4000 random task-sets with 8 independent task characterized by non-harmonics periods. Different symbols correspond to processor utilizations ranging from 60% to 95%. Lines represent the theoretical estimation based on the ration  $U/BU$ .

We also checked the effect of variability in number of periodic tasks, the harmonicity of the periods and the processor utilizations. In particular we have simulated a number of independent periodic tasks varying from 8 to 16 for both harmonic and non-harmonic periods. The results do not differ from those presented in Figure 2.

### 4. The Enhanced Power Low Dual Priority

The benefits in energy saving obtained in the Rate Monotonic Algorithm by applying the average operating frequency policy can be generalized to other fixed priority pre-emptive scheduling algorithms.

In particular, we are interested into extend these results to the Power Low Modified Dual Priority Scheduling algorithm (based on the Dual Priority scheduling [13]) because its adequacy to manage power saving in more

complex scenarios that could include aperiodic requests and because its performance has been contrasted against other fixed priority algorithms [11]. The original Power Low Dual Priority Scheduling algorithm (PLMDP) [12] guarantees to meet the temporal constraints and a significant energy consumption reduction.

Based on the results obtained for the RMA we propose the use of a balanced operating frequency reduction policy that gives to all ready tasks the opportunity to reduce its execution operating frequency. The balance is intended to provide an average operating frequency according to U/BU and it is controlled dynamically. Qualitatively the idea work as follows: If a task should execute at a certain operating frequency higher than the estimated average to meet its deadline, then the following tasks try to execute at an operating frequency lower than the estimated average to compensate the global effect in the system. Then our algorithm is designed to achieve an average operating frequency according to the ratio U/BU while meeting deadlines. BU is statically calculated off-line.

The PLMDP defines three levels of priorities that are organized as follows, the highest level, or upper run queue (URQ) is for tasks that can no longer be delayed by less priority tasks otherwise they could miss their deadlines. The lowest level, or lower run queue (LRQ) occupied by those periodic tasks whose execution time can still be delayed without compromising their deadlines. At the beginning of each hyper-period the remaining processor utilization ( $W_{rem}$ ) is set to the total workload of the task set = U.

The scheduling algorithm is driven by the following events:

1. Promotion time instant ( $Tp_i$ ) [12] The moment at which the task is promoted from the LRQ to the URQ. At this moment the task can pre-empt a lower priority task currently in execution. At this time instant,  $W_{rem}$  is updated according to its real use, it is decremented by the consumed time of the pre-empted task.
2. Activation time ( $Ta_i$ ). The task is queued in the LRQ sorted by its promotion time instant. At this moment this task can pre-empt a lower priority periodic task currently in execution, and  $W_{rem}$  is updated.
3. Task finalization time. At this time instant,  $W_{rem}$  is decremented by the consumed time plus the spare time of this task. After that, the highest priority task from the highest non-empty priority level (i.e. URQ or LRQ, in this order) is selected for execution.

In the new algorithm Enhanced Power Low Dual-Priority EPLDP, the processor operating frequency is individually calculated for each task, once the scheduler decides which task must be executed (Figure 3). The algorithm reduces the operating frequency at the maximum value between the frequency estimated by the original PLMDP, and the ratio between the processor utilization and the breakdown utilization:  $U_{rem}/BU$ . This operating frequency is the lowest frequency that assures that no

deadline will be missed<sup>1</sup>. Before calculating the operating frequency  $U_{rem}$  is updated to the ratio between the workload that remains to be executed and the remaining time to arrive to the end of the hyperperiod.

$$U_{rem} = \left( \frac{W_{rem}}{\text{hyperperiod} - tc} \right) \quad (2)$$

Summarizing, the resulting algorithm (EPLDP) works as follows:

```

// Tp is the promoted time; Td is the deadline time; tc is the current time
// τi is the active task; τk is the next promoted task
// Urem is the remaining utilization = U - executed workload
L1 if not empty (URQ) then
L2   Active Task (τi)= URQ.head;
L3   if URQ.head.next = NIL
L4     Frequency = max( ( min(Tpk - tc, remaining_i) / ( min(Tpk, Tdi) - tc ), Urem / BU )
L5   else
L6     Frequency = MAX_FREQ;
L7   endif
L8 else
L9   if not empty (LRQ) then
L10    Active task (τi)= LRQ.head;
L11    if Tpk < Tp_i then
L12      Frequency = ( Urem / BU )
L13    else
L14      Frequency = max( ( min(Tpk - Tp_i, remaining_i) / ( min(Tpk, Tdi) - tc ), Urem / BU )
L15    endif
L16  else
L17    Set timer to (next Ta_i - wake up delay);
L18    Enter power-down mode;
L19  endif
L20  endif
L21 endif
L22 execute Active Task (τi) at calculated operating frequency;

```

**Figure 3: Enhanced Power Low Dual-Priority (EPLDP) Scheduling.**

1. If there is not any ready task in the system we set the timer to the next arriving task minus the wake up delay, and power down the processor.
2. If there is more than one task in the URQ then it must be executed at the maximum operating frequency.
3. If there is only one task in the URQ then it can be executed at low frequency:

$$Frequency = \max \left( \frac{\min(Tp_k - tc, remaining_i)}{\min(Tp_k, Td_i) - tc}, \frac{U_{rem}}{BU} \right) \quad (3)$$

where  $Tp_k$  is the promotion time instant of any task in the system excluding the current executing task,  $remaining_i$  is the non-executed worst execution time of the current task,  $Td_i$  is the deadline of the current task,

<sup>1</sup> The feasibility of our algorithm is achieved whenever the Dual Priority Algorithm is feasible [13] because our algorithm always uses the Dual Priority schedule as a lower bound for feasibility.

and finally  $tc$  is the current time. The desired operating frequency is based on the remaining workload, the remaining time to the hyper-period, and the BU.

4. If there is not any task in the URQ but there are some tasks in the LRQ then we can execute at the average operating frequency  $U_{rem}/BU$ .

At practice only certain discrete values of the frequency of the clock, and then speed, are attainable depending on the accuracy of the tuning, in this case the frequency selected should be a frequency equal or larger than the frequency obtained by the calculations to ensure time constraints.

The algorithm is designed to achieve an average operating frequency equal to the ratio  $U_{rem}/BU$ . This average is achieved whenever all tasks consume the 100% of its WCET. When the WCET is not totally consumed then this average is overestimated (in the next section we will discuss how to take advantage of this fact). The performance of the PLMDP is flexible to different WCET consumptions adapting its behavior when needed. At a certain critical value of the WCET consumption we expect PLMDP to overcome the performance of the EPLDP because this overestimation. In figure 4 we show the experimental critical curve for the WCET consumption delimiting the area of efficiency of both algorithms.

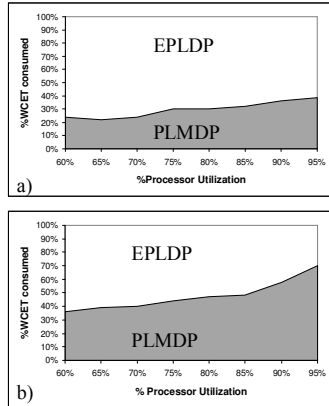


Figure 4: Critical line showing the transition performance between EPLDP and PLMDP. Above the line EPLDP is energetically favorable, below the line the PLMDP is energetically favorable. The line is obtained by simulation of 100 tasks sets (formed by 8 tasks each one) for each value of the processor utilization, varying the processor utilization in 5% each step. Harmonic periods from 1024 to 65536 are considered. The maximum task workload is set to 20%.

## 5. Moving average estimation of the empirical utilization of the processor

The WCET of a task depends on both the program flow and architectural factors like pipelines and caches. It must guarantee and not underestimate the real execution time, but often provides an overestimation of it. To reach maximum effectiveness of the use of the processor, the overestimation of the empirical execution time should be

as small as possible. But note that as the processors have more complex features like for example out-of-order execution, the overestimation becomes usually large.

We want to point out that a dynamic estimation of the real utilization (EU) is possible by using the history of past executions ( $U_m$ ) where  $m$  refers to different hyper-periods.

Here we present a moving average process that takes advantage of this information to determine the correct average frequency reduction that adapts to the real calculation consumption of tasks ( $U_i$ ).

We modified our algorithm introducing a moving average of the utilization that reduces the overestimation of the WCET of tasks in the following manner (Figure 5):

1. Initially the estimated utilization (EU) is set to the total workload with the WCET provided by the application designers ( $U_0$ ).
2. The hyper-period is executed using the current EU. At the same time the real workload of each task is updated after its real execution.
3. At the end of the hyper-period the EU is updated according to the available history ( $\bar{U}$ ) and the recent hyper-period execution ( $U_i$ ), using a moving average.

```

L1   $U_0 = \sum_{\forall \tau} C_i \quad EU = U_0$ 
L2   $i=1;$ 
L3  while real_time_application_not_finished do
L3      execute hyper-period  $i$  with  $EU$  and update  $U_i$ ;
L4       $\bar{U} = \frac{(\bar{U} * i) + U_i}{i + 1} \quad EU = U_i - \bar{U}$ 
L5  enddo

```

Figure 5: Moving average estimation of the empirical utilization of the processor.

In Figure 6 we present the evolution of the EPLDP using this estimation over different hyper-periods (EPLDP-m). The maximum utilization of the system is set to 80 % with a maximum workload for tasks of 20%. All tasks consumption is obtained from a Gaussian distribution with an average of 50% of its WCET and with a standard deviation of 10%. The results are obtained averaging over 100 different task sets. Note that a lower bound for the frequency reduction is provided by this estimation while the low power algorithm fixes the upper bound (Figure 3). The energy consumption obtained by EPLDP-m tends to be the same as the energy consumption obtained by the theoretical EPLDP-f, which is the EPLDP behavior assuming that the real utilization is known and fixed to 50% of the WCET (note that this information is unknown in real applications). The small divergence between EPLDP-m and EPLDP-f are consequence of the variations of the real use of the WCET that has been set to 10%.

It is important to note that this moving average strategy does not interfere with the hard real time because the determination of the operating frequency is conservative with respect to deadlines i.e. we use the highest frequency

between the calculated EU and the operating frequency calculated by the PLMDP algorithm. (see equation 3), to do not compromise any deadline.

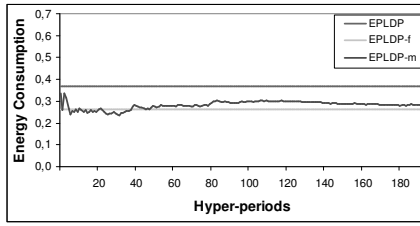


Figure 6: Evolution of EPLDP-m compared with EPLDP and with EPLDP-f.

The improvement in energy consumption provided by EPLDP-m is contrastable, and it should be more evident as the overestimation of the WCET is larger. In figure 7, we show this improvement as a function of the percentage of the WCET really consumed for a harmonic task sets (Figure 7a), and for a non-harmonic task sets (Figure 7b).

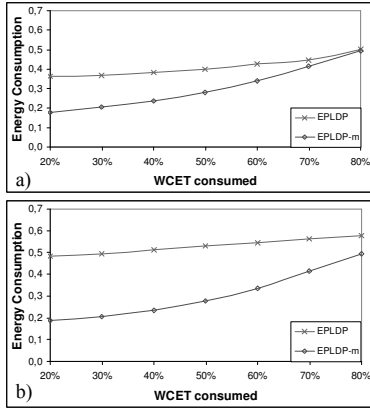


Figure 7: Performance of EPLDP and EPLDP-m with different percentage of WCET consumption. The total utilization of the system is  $U=80\%$  and the maximum task workload is set to 20. The results are obtained averaging over 100 task sets. In a) the task sets are harmonic, and in b) there are non-harmonic task sets.

## 6. Results and discussion

In this section we test the energy saving efficiency of the proposed EPLDP-m algorithm versus the original Power Low Modified Dual Priority (PLMDP) [3,12] for real and synthetic task sets.

The first test (Figure 8) corresponds to the energy consumption for different fixed workloads of the system ( $U=60\%$ ,  $75\%$  and  $95\%$ ) (Figure 8). We observe that the average energy consumption is energetically favorable to EPLDP. The average energy consumption is 59%, 61% and 68% for PLMDP and 17%, 28% and 49% for EPLDP-m ( $U=60\%$ ,  $75\%$  and  $95\%$  respectively) compared to the execution at maximum operating frequency.

We have also checked the dependence of energy consumption on the workload of the system ( $U$ ). We calculate the average energy consumption of schedulable

tasks sets composed by 100 synthetic task sets. The workloads range from 60% to 95%, in steps of 5%. The maximum task workload was fixed to 20%. There are 8 tasks in each task set. The periods range from 1024 to 65536 (harmonic task sets). (Figure 9). The experiment represents the results of the normalized average energy obtained with respect to the execution at maximum operating frequency. We run the simulation over 200 hyper-periods.

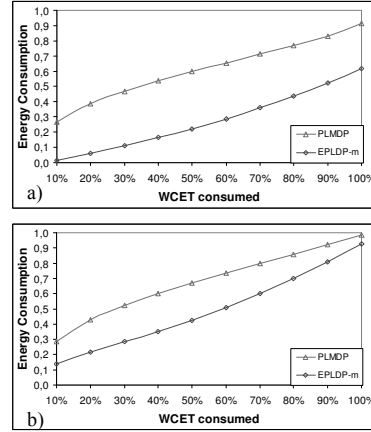


Figure 8: Normalized average energy consumption for different values of WCET consumption. We simulate 100 task sets, of 8 tasks each one, with a maximum task workload of 20%, in a)  $U=75\%$ , and in b)  $U=95\%$ .

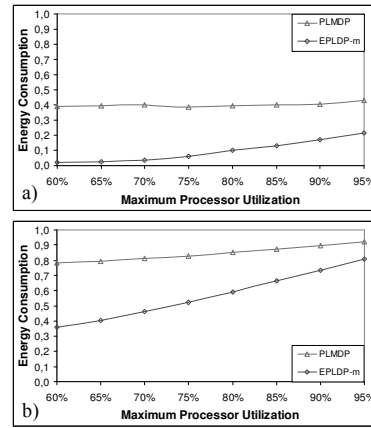


Figure 9 Normalized energy consumption versus processor utilization ( $U$ ). Each dot corresponds to the average energy consumption of 100 different harmonic task sets. The consumed WCET is in a) 20% and in b) 90% .

To conclude the present analysis, we have also collected some real time applications: the avionics task set reported in [17], an Inertial Navigation System (INS) [18] and a Computerized Numerical Control Machine (CNC) [19]. The two first sets represent critical mission applications and the last one is an automatic control for specific machinery.

The results of energy consumption for each application varying the percentage of WCET consumed are drawn in

Figures 10 to 12. The average energy consumption referred to the execution at maximum frequency are 76% for PLMDP and 35% for the EPLDP-m in the avionics task set; 36% for PLMDP and 9% for the EPLDP-m in the INS task set; and 56% for PLMDP and 26% for the EPLDP-m in the CNC task set.

We have also performed simulations considering variations of the task sets specifications: doubling the number of tasks to 16 instead of 8, varying the maximum workload per task from 20% to 10%, and considering non-harmonic periods ranging from 1000 to 70000. The results obtained in these different experiments do not differ qualitatively from the results presented, although the precise values vary. In particular, non-harmonic periods introduce a shift on the energy consumption performance of all the algorithms we have studied. The main reason is that schedulability becomes more complex, and the breakdown utilization decreases.

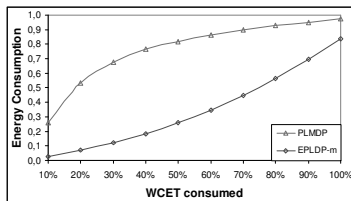


Figure 10: Comparison of the algorithms for the Avionics set [17]

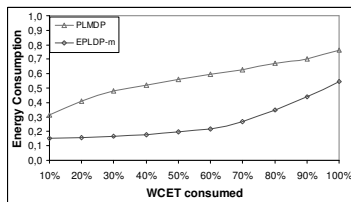


Figure 11: Comparison of the algorithms for the INS set [18]

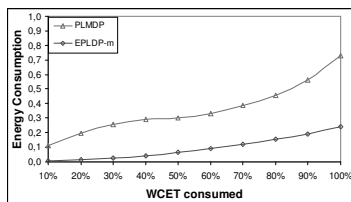


Figure 12: Comparison of the algorithms for the CNC set [19]

## 7. Conclusions

We have proposed a new version of the PLMDP algorithm that enhances energy saving based on the dynamic calculation of an average operating frequency EPLDP-m. The advantage of this energy reduction policy, consisting on giving the opportunity to the processor to reduce the operating frequency of every task, has been demonstrated to improve energy saving substantially without missing any deadline. The current performance could be extended to other dynamic power aware scheduling algorithms.

## 8. References

- [1] A.P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-power CMOS digital design", *IEEE Journal of Solid-State circuits*, vol. 27, pp. 473-484, April 1992.
- [2] C. M. Krishna, Y.H. Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems", in *Real-Time Technology and Applications Symposium*, pp. 156-165, 2000.
- [3] M.A. Moncusi, A. Arenas and J. Labarta, "A modified dual priority scheduling in hard real time systems to improve energy saving." in *Compilers and Operating systems for Low Power* pp 17-36. *Kluwer academic/Plenum publishers* 2003.
- [4] S. Saewong and R. Rajkumar, "Practical voltage-scaling for fixed-priority RT-systems." *The 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp106-115, May 2003
- [5] U. Kimiyoshi, M. Horowitz, "Clustered voltage scaling technique for low-power design", in *International Symposium on Low Power Electronics and Design*, pp. 3-8, 1995.
- [6] J. Rabaey and M Pedram (Editors). "Low power design methodologies". *Kluwer Academic Publishers*, May 1996.
- [7] S.T. Cheng, S.M. Chen and J.W. Hwang, "Low-power design for real-time systems", *Real-Time Systems*, 15, pp 131-148, 1998.
- [8] P. Pillai and K.G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems" *18th ACM Symposium on Operating Systems Principles*, October 2001.
- [9] H. Aydin, R. Melhem, D. Mosse and P. Mejia-Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics" *13th Euromicro Conference on Real-Time Systems*, June 2001.
- [10] H. Aydin, R. Melham, D. Mosse, P. Mejia-Alvarez. "Dynamic and Aggressive scheduling techniques for power-aware real-time systems." *Proc. Real-Time Systems Symposium*, pp. 95-105, 2001.
- [11] Y. Shin and K. Choi, "Power conscious fixed priority scheduling in hard real-time systems" *DAC 99, ACM 1-58113-7/99/06*, 1999.
- [12] M.A.Moncusi, A.Arenas, J.Labarta, "Improving energy saving in hard real time systems via a modified Dual Priority scheduling", *ACM SigArch Computer Architecture Newsletter*, Vol 29, 19-24 (2001)
- [13] R. Davis and A.J. Wellings, "Dual Priority scheduling", *Proceeding IEEE Real Time Systems Symposium*, pp. 100-109, December 1995.
- [14] A. Miyoshi, C.Lefurgy, E.V. Hensbergen, R.Rajamony, and R. Rajkumar. "Critical power slope: Understanding the runtime effects of frequency scaling." In *Proceedings of the 16th Annual ACM International Conference on supercomputing*, June 2002.
- [15] J. Lehoczky, L. Sha, and Y. Ding. "The rate monotonic scheduling algorithm: Exact characterization and average case behavior". In *Proceedings of IEEE Real-Time Systems Symposium*, pages 166-171. IEEE Computer Society Press, December 1989.
- [16] Katcher, D.I.; Arakawa, H.; Strosnider, J.K.; "Engineering and analysis of fixed priority schedulers" *Software Engineering, IEEE Transactions on*, Volume: 19, Issue: 9, Sept. 1993 Pages: 920-934
- [17] C. Locke, D. Vogel and T. Mesler, "Building a predictable avionics platform in Ada: a case study", *Proceedings IEEE Real-Time Systems symposium*, December 1991.
- [18] A. Burns, K. Tindell and A. Wellings, "Effective analysis for engineering real-time fixed priority schedulers", *IEEE Transactions on Software Engineering*, 21, pp. 475-480, 1995.
- [19] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi and H. Shin, "Visual assessment of a real-time system design: a case study on a CNC controller", *Proceedings IEEE Real-Time Systems symposium*, December 1996.