Energy-Efficient Scheduling Fixed-Priority Tasks with Preemption Thresholds on Variable Voltage Processors

XiaoChuan He and Yan Jia

Institute of Network Technology and Information Security
School of Computer Science
National University of Defense Technology
Changsha, China 410073

Abstract. Slowdown factors determine the extent of slowdown a computing system can experience based on functional and performance requirements. Dynamic Voltage Scaling (DVS), which adjusts the clock speed and supply voltage dynamically, is an effective technique in reducing the energy consumption of embedded real-time systems. We address the problem of computing static and dynamic slowdown factors in the FPPT algorithm. In this paper, Sufficient constraints have been identified for the feasibility of the task set using slowdown factors. We formulate this problem of computing the static slowdown factors for tasks as an nonlinear optimization problem to minimize the total energy consumption of the system. Our simulation experiments show on an average 17%∼53% energy gains over FPPT scheduling policy.

1 Introduction

Dynamic Voltage Scaling (DVS), which adjusts the supply voltage and its corresponding clock frequency dynamically, is an effective low-power design technique for embedded real-time systems. Since the energy consumption of CMOS circuits has a quadratic dependency on the supply voltage, lowering the supply voltage is one of the most effective techniques for reducing the energy consumption. There exist two primary ways of reducing the power consumption in embedded computing systems: *processor shutdown* and *processor slowdown*. Slowdown using frequency and voltage scaling has been shown to be effective in reducing the processor energy consumption [1], [2], [3].

It is known that preemptability is a necessary requirement to achieve higher processor utilization and optimal slowdown [2],[4]. However, preemptive scheduling has its additional costs as compared to non-preemptive scheduling. Though preemptive scheduling achieves higher utilization, it is not always required to preempt a lower priority task.

Fixed-Priority scheduling with preemption threshold (FPPT) [5],[6] allows a task to disable preemptions from tasks up to a specified preemption threshold priority. Tasks with a priority greater than the preemption threshold priority are

K. Li et al. (Eds.): NPC 2007, LNCS 4672, pp. 133–142, 2007.

still allowed to preempt. The preemption threshold scheduling model has been shown to reduce the run-time costs by eliminating unnecessary task preemptions. Furthermore, FPPT allows tasks to be partitioned into non-preemptive groups to minimize the number of threads [6] and the stack memory requirement [7], thereby leading to scalable real-time systems.

Fixed-Priority scheduling with preemption threshold (FPPT) eliminates unnecessary context switches, thereby saving energy. Slowdown using frequency and voltage scaling is more effective in reducing the energy consumption [4], [2], [8], [1]. In this paper, we integrate processor slowdown techniques with FPPT to enable scalable and energy efficiency real-time systems. Given task with predefined priorities and corresponding preemption thresholds, we propose an algorithm to compute the *static slowdown factors* by formulating the problem as a linear optimization problem. Furthermore, we consider the energy consumption of task set under different preemption threshold assignments. We gain as much as 17%∼53% energy savings over the usual FPPT scheduling algorithm.

The remainder of the paper is organized as follows. The next section presents the assumed computation model. The related work is introduced in Section 2. Section 3 we formulate the computation of slowdown factors as an nonlinear optimization problem. The experimental results are given in Section 4. Finally, the conclusion is made in Section 5.

2 Preliminaries

In this section, we introduce the necessary notation and formulate the problem. We first describe the system model followed by the related works.

2.1 System Model

This study deals with the fixed priority preemptive scheduling of tasks in a real-time systems with hard constraints, i.e., systems in which the respect of time constraints is mandatory. The activities of the system are modeled by periodic tasks. The model of the system is defined by a task set Γ of cardinality n, $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$. A periodic task τ_i is characterized by a 3-tuple (C_i, T_i, D_i) where each request of τ_i, called instance, has an execution time of C_i, a relative deadline D_i. T_i time units separate two consecutive instances of τ_i (hence T_i is the period of the task). Note that C_i is the worst case execution time (WCET) of the task at maximum speed, given that it is the only task running in the system. The system is said schedulable if each instance finishes before its deadline.

Each task is associated with a deadline D_i, that D_i may be arbitrarily large. This means that many instances of the same task can be active (in the ready queue) at the same time. We assume that the scheduler handles tasks with the same priorities using a FIFO rule. Hence an early instance of a task has priority over a later, and must be completed before the later instance is allowed to start.

We further assume that the total utilization of all tasks, U, is strictly less than 1. This will later be shown to be a necessary condition for the analysis.

Each task is assigned with a priority π_i and preemption threshold γ_i, and $\gamma_i \geq \pi_i$. The smallest number is given to the task with highest priority.

2.2 Variable Speed Processors

A wide range of processors like the Intel XScale [9] and Transmeta Crusoe [10] support variable voltage and frequency levels. Voltage and frequency levels are tightly coupled. The important point to note is when we perform a slowdown we change both the frequency and voltage of the processor. Given the minimum frequency f_{min} and the maximum supported frequency f_{max}, we normalize the speed to the maximum frequency to have discrete points in the operating range $[\eta_{min}, 1]$, where $\eta_{min} = f_{min}/f_{max}$.

The *slowdown factor* can be viewed as the normalized frequency. At a given instance, it is the ratio of the scheduled frequency to the maximum frequency of the processor.

2.3 Related Works

Previous investigations on the voltage scheduling problem have focused mainly on real-time jobs running under dynamic-priority scheduling algorithms such as the EDF (earliest-deadline-first) algorithm [4], [11], [12], [13]. For example, the problem of energy-optimal EDF scheduling has been well understood. For EDF job sets, the algorithm by Yao et al. [14] computes the energy-optimal voltage schedules in polynomial time. Although the EDF scheduling policy makes the voltage scheduling problem easier to solve, fixed-priority scheduling algorithms such as the RM (rate monotonic) algorithm are more commonly used in practical real-time systems due to their low overhead and predictability [15].

Most of the earlier work dealt with general fixed-priority task sets. Shin et al. [1] have computed uniform slowdown factors for an independent periodic task set. Jejurika and Gupta [3] proposed an algorithm to compute a near optimal constant slowdown factor based on the bisection method, and furthermore for the case of tasks with varying power characteristics, [3] develop the computation of near optimal slowdown factors as a solution to convex optimization problem using the ellipsoid method. Yao, Demers and Shanker [14] presented an optimal off-line speed schedule for a set of N jobs. An optimal schedule for tasks with different power consumption characteristics is considered by Aydin, Melhem and Mosse [4]. The same authors [2] prove that the processor utilization (at maximum speed) is the optimal slowdown factor when the deadline is equal to the period. Quan and Hu [16] [17] discuss off-line algorithms for the case of fixed priority scheduling.

Since the worst case execution time (WCET) of a task is not usually reached, there is dynamic slack in the system. Pillai and Shin [14] recalculate the slowdown to use the dynamic slack while meeting the deadlines. Low-power scheduling using slack reclamation heuristic is studied by Aydin et al. [2] and Kim et al. [12]. Scheduling of task graphs on multiple processors has also been considered. Luo and Jha [18] have considered scheduling of periodic and aperiodic task

graphs in a distributed system. Non preemptive scheduling of a task graph on a multi processor system is considered by Gruian and Kuchcinski [19]. Zhang et al. [8] have given a framework for task scheduling and voltage scaling of dependent tasks on a multi-processor system. They have formulated the voltage scaling problem as an integer programming problem.

3 Static Slowdown Factors

We compute static slowdown factor for a system with an underlying fixed-priority with preemption threshold (FPPT) scheduler. For a task τ_i in FPPT, the k^{th} instance of τ_i is denoted as J_i^k. Before J_i^k begin execution, the interference caused by other tasks with regular priority higher than π_i is possible; Once J_i^k starts actually, only the tasks with regular priorities higher than γ_i can preempt J_i^k. Thus the starting time of J_i^k play a necessary role in the analysis of FPPT, so a notation S_i^k is introduce to denote it, S_i^k is used to represent the starting point of J_i^k, F_i^k denote the finish time of J_i^k.

3.1 FPPT Scheduler

The existing schedulability analysis for FPPT [5], [20], [21]is based on the FPPT critical instant [21] which says that if a task meets its deadline whenever the task is requested simultaneously with requests for all higher priority tasks and the lower priority task that contribute the largest blocking time to it, furthermore all of the task instance in the level-i busy period must meets its deadline too, then the deadline will always be met for all task phrasings. The WCRT of a task τ_i can be derived through the following equations [20]:

$$B_i = \max_{\tau_j \in \Gamma} \{C_j | \pi_i > \pi_j \wedge \pi_i \leq \gamma_j\} \tag{1}$$

$$L_i = B_i + \sum_{\forall j, \pi_j \geq \pi_i} \lceil \frac{L_i}{T_j} \rceil \cdot C_j \tag{2}$$

$$S_i^k = B_i + k \cdot C_i + \sum_{\forall j, \pi_j > \pi_i} \left(1 + \lfloor \frac{S_i^k}{T_j} \rfloor\right) \cdot C_j \tag{3}$$

$$F_i^k = S_i^k + C_i + \sum_{\forall j, \pi_j > \gamma_i} \left(\lceil \frac{F_i^k}{T_j} \rceil - \left(1 + \lfloor \frac{S_i^k}{T_j} \rfloor\right)\right) \cdot C_j \tag{4}$$

$$R_i = \max_{k=0,1,2,\ldots,\lfloor \frac{L_i}{T_j} \rfloor} (F_i^k - k \cdot T_i) \tag{5}$$

In the course of schedulability analysis of FPPT, as for task τ_i, WCRT of τ_i, R_i can be derived. At the same time, the specific instance of τ_i which contribute this WCRT can be derived too. This instance is denoted as $J_i^{k_r}$. Assuming the task are independent, let η_i be the slowdown factors for task τ_i. As we know,

the allowably finish time of $J_i^{k_r}$ is $D_i + k_r T_i$, thus equation 4,3 is rewritten as following:

$$S_i^{k_r} = \frac{B_i}{\eta_b} + k_r \cdot \frac{C_i}{\eta_i} + \sum_{\forall j, \pi_j > \pi_i} \left(1 + \lfloor \frac{S_i^{k_r}}{T_j} \rfloor\right) \cdot \frac{C_j}{\eta_j} \tag{6}$$

$$D_i + k_r T_i = S_i^{k_r} + \frac{C_i}{\eta_i} + \sum_{\forall j, \pi_j > \gamma_i} \left(\lceil \frac{D_i + k_r T_i}{T_j} \rceil - \left(1 + \lfloor \frac{S_i^k}{T_j} \rfloor\right)\right) \cdot \frac{C_j}{\eta_j} \tag{7}$$

In the equation 6,7, $S_i^{k_r}$ is calculated in the course of schedulability analysis, all of other terms are known already except for the *slowdown factors*, denoted as $\eta_i, i = 1, 2, ..., n$. From the combination of equation 6 and 7, the following theorem can be derived.

Theorem 1. *A task set of n independent fixed-priority periodic tasks, scheduled with preemption thresholds, is feasible at a slowdown factor of η_i for task τ_i if the following relations satisfy:*

$$D_i + k_r T_i \geq \frac{B_i}{\eta_b} + \frac{C_i}{\eta_i} + \sum_{\substack{\forall j, \pi_j > \pi_i, \\ \pi_j \leq \gamma_i}} \left(1 + \left\lfloor \frac{S_i^{k_r}}{T_j} \right\rfloor\right) \frac{C_j}{\eta_j} + \sum_{\forall j, \pi_j > \gamma_i} \left(\left\lceil \frac{D_i + k_r T_i}{T_j} \right\rceil \right) \frac{C_j}{\eta_j} \tag{8}$$

where τ_b is the blocking task which contribute to the longest blocking time to τ_i.

In the equation 8, term $\sum_{\forall j, \pi_j > \gamma_i} \left(\left\lceil \frac{D_i + k_r T_i}{T_j} \right\rceil \right) \frac{C_j}{\eta_j}$ represent the workload of the tasks with the regular priority higher than γ_i in the interval of $[0, D_i + k_r T_i]$, term $\sum_{\forall j, \pi_i < \pi_j \leq \gamma_i} \left(1 + \left\lfloor \frac{S_i^{k_r}}{T_j} \right\rfloor\right) \frac{C_j}{\eta_j}$ represent the workload of the tasks with the regular priority higher than π_i before $J_i^{k_r}$ start the execution. once $J_i^{k_r}$ started actually, these tasks can not preempt $J_i^{k_r}$ again. The term $\frac{B_i}{\eta_b}$ represent the blocking time that the task with the priority lower than π_i contributes. The term $\frac{C_i}{\eta_i}$ represent the workload of $J_i^{k_r}$ itself.

3.2 Computing Slowdown Factors

Power Characteristics. The number of cycles, C_i that a task τ_i needs to complete is a constant during voltage scaling. The processor cycle time, the task delay and the dynamic power consumption of a task vary with the supply voltage V_{DD}. The power delay characteristics of the CMOS technology [21] are as given below.

$$P_d = C_{eff} V_{DD}^2 f \tag{9}$$

$$f = \alpha k' \frac{(V_{DD} - V_{TH})^\alpha}{V_{DD}} \tag{10}$$

where k' is a device related parameter, V_{TH} is the threshold voltage, C_{eff} is the effective switching capacitance per cycle and α ranges from 2 to 1.2 depending on the device technology. Since power varies linearly with the clock speed and the square of the voltage, adjusting both can produce cubic power reductions, at least in theory. Since the time needed to execute task τ_i is $t_i = C_i/f_i$, the energy consumption of the task executing for an interval of time I, is $E = P_d I$.

Linear Optimization Problem. We formulate the energy minimization problem as an optimization problem. The voltage and slowdown factors are normalized to the maximum values. We compute normalized voltage levels for the tasks such that the conditions in Theorem 1 are satisfied. Let $\bar{f} \in R^n$ be a vector representing the normalized frequencies f_i of task τ_i. The optimization problem is to compute the optimal vector $\bar{f}^* \in R^n$ such that the system is feasible and the total energy consumption of the system is minimized. Let P_d be the normalized energy consumption of task τ_i as a function of the normalized frequencies f_i (for the case of identical power characteristics $P_d = k' f^3$. The total energy consumption of the system E, a function of the voltage vector $\bar{f} \in R^n$, is given by Equation 11. Thus, we have the following optimization problem:

minimize:

$$E(v) = \sum_{i=1}^{n} P_d \frac{C_i}{\eta_i} \frac{I}{T_i} \tag{11}$$

under the constraints:

$$\overset{\forall i}{i = 1, 2, ..., n} \quad D_i + k_r T_i \geq \frac{B_i}{\eta_i} + \frac{C_i}{\eta_i} + \sum_{\substack{\forall j, \pi_j > \pi_i, \\ \pi_j \leq \gamma_i}} \left(1 + \left\lfloor \frac{S_i^{k_r}}{T_j} \right\rfloor\right) \frac{C_j}{\eta_j} + \sum_{\forall j, \pi_j > \gamma_i} \left(\left\lceil \frac{D_i + k_r T_i}{T_j} \right\rceil\right) \frac{C_j}{\eta_j} \tag{12}$$

$$\forall i \quad \eta_{\min} \leq \eta_i \leq 1 \tag{13}$$

Equation 12 ensures the feasibility when the tasks are independent. Equation 13 constraints the slowdown factor to be greater than or equal to the slowdown in the independent mode. The normalized slowdown factors are between the normalized minimum frequency η_{min} and 1. The optimization function depends on the power characteristics P_d of the task. If P_d is convex, the optimization function is convex. Thus we have a convex Quadratic minimization problem.

4 Experiments

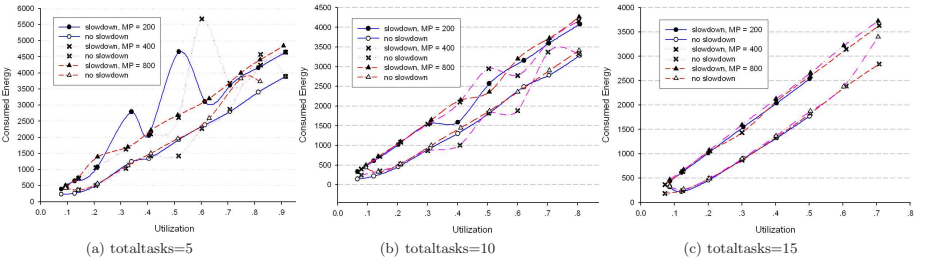
We use randomly generated periodic task sets for our simulations. Each task is characterized by its computation time C_i, its period T_i, its deadline D_i. We vary three parameters in our simulations: (1) number of tasks *totaltasks*, from 5 to 15; and (2) maximum period for tasks *maxperiod*, from 200 to 800; and (3) utilization for task set, from 0.1 to 0.9. For any given pair of *totaltasks*, *maxperiod*,

and utilization, we randomly generate 1000 task sets, and the expreiment result is the average value over these 1000 task sets.

A task set is generated by randomly selecting a period, a deadline, and computation time for each of the *totaltasks*. First, a period T_i is assigned randomly in the range [1, *maxperiod*] with a uniform probability distribution function. Then,we assign a utilization U_i in the range $[0.1/totaltasks, 2.0/totaltasks]$, again with uniform probability distribution function. The computation time of the task is then assigned as $T_i * U_i$. Deadlines are assigned just as periods, and deadline of each task is unique. Considering an interval of time $I = 5,000$, the experiment calculated the energy consumption of task set in this time interval.

For the simplicity of description, the FPPT scheduling algorithm using slowdown factors is denoted as *ES_FPPT*. We note that our experiments are executed in the environment of Pentium 1.8G, 786M RAM, Redhat9 (linux-2.4.20-8, gcc-3.4.2, X11, SPAK-0.3).

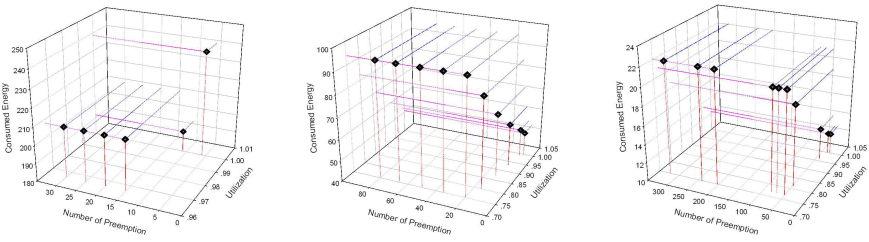
Although different task might have varying power characteristics [4], we don't compute slowdown factors based on the task characteristics in this paper. k' is assumed to be 1. In other words, all tasks have the same power coefficient.



(a) totaltasks=5 (b) totaltasks=10 (c) totaltasks=15

Fig. 1. Average energy savings of ES_FPPT algorithm over the FPPT algorithm

In the first set of experiments, we guarantee the utilization of task set is unchanged when the respective slowdown factors of tasks are put in execution. Figure 1 shows the consumed energy of ES_FPPT scheduling algorithm in comparison with the FPPT scheduling algorithm. The energy consumption of both algorithms are collected in the time length of 5000ms, where figure 1(a), 1(b) and 1(c) describe respectively the experiment results of *totaltasks* is 5, 10 and 15. From the figure, it is seen obvioulsy that the ES_FPPT algorithm performs more 13%∼42% energy-saving than the FPPT algorithm. ES_FPPT assigns different slowdown factors for the tasks to have better energy gains. ES_FPPT performs up to low utilization of task set better than FPPT at high utilization. Since we compute continuous slowdown factors and assign them to the closest discrete voltage levels, there is run time slack even at worst case execution time. The complication workload demand analysis is required to implement the dynamically reclaim the run-time slack, which is the future working problem of our research.



(a) totaltasks = 5, maxperiod = 400, utilization = 0.4 (b) totaltasks = 10, maxperiod = 200, utilization = 0.3 (c) totaltasks = 15, maxperiod = 800, utilization = 0.2

Fig. 2. Energy savings of ES_FPPT algorithm under different PTAs

In the second set of experiments, we allow the utilization can change after using slowdown factors. Figure 2 shows the energy consumption of the ES_FPPT algorithm under different preemption threshold assignments(PTAs) [21]. With the partial relations between the PTAs, which is defined in [21], It can be inferred that the smaller PTA, the more number of preemptions, and the more additional energy dissipation of context switches. It is seen that the energy gains increase with the different PTAs. Furthermore, Computing task slowdown factors considering the varied PTAs, results in more energy gains. From the graph of energy gains, it is seen that the another gains under maximal PTA are as high as 11%∼46% over the ES_FPPT algorithm under minimal PTA, which adds up to 2%∼11% of total energy savings.

To our surprise, the consumed energy doesn't decreased steadily with some drops in number of preemptions due to rising PTA. While most of experiment samples have illustrated that the energy saving maximized by the slowdown factors coming from maximal PTA, there still exist some exceptions, such as figure 2(a) presents.

5 Conclusion

In this paper, we present algorithms to compute static slowdown factors under FPPT scheduling policy. The sufficient constraints which slowdown factors must satisfy to maintain the feasibility of FPPT task set are derived. Furthermore, We formulate the computation of slowdown factors for tasks under different preemption threshold assignments as an optimization problem. Experimental results show that the computed slowdown factors save on an average 17%∼53% energy over the known techniques. The techniques are energy efficient and can be easily implemented. This will have a great impact on the energy consumption of embedded real-time systems.

We plan to further exploit the static and dynamic slack in the system to make the FPPT algorithm more energy efficient. As a future work, we plan to compute discrete slowdown factors for the tasks.

Acknowledgment

The authors wish to express their gratitude to ChuangGuo Guo, Yan Jia for useful discussions, and to the reviewers for their helpful comments on the paper.

References

1. Shin, Y., Choi, K., Sakurai, T.: Power optimization of real-time embedded systems on variable speed processors. In: 2000 International Conference on Computer-Aided Design (ICCAD '00), pp. 365–368 (2000)
2. Aydi, H., Mejia-Alvarez, P., Mosse, D., Melhem, R.: Dynamic and aggressive scheduling techniques for power-aware real-time systems. In: 22nd IEEE Real-Time Systems Symposium (RTSS'01), p. 95. IEEE Computer Society Press, Los Alamitos (2001)
3. Jejurikar, R., Gupta, R.K.: Optimized slowdown in real-time task systems. In: 16th Euromicro Conference on Real-Time Systems (ECRTS '04) (2004)
4. Aydin, H., Melhem, R., Mosse, D., Mejia-Alvarez, P.: Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In: 13th Euromicro Conference on Real-Time Systems (ECRTS'01), p. 225 (2001)
5. Wang, Y., Saksena, M.: Scheduling fixed-priority tasks with preemption threshold. In: The Sixth International Conference on Real-Time Computing Systems and Applications, pp. 328–335. IEEE Computer Society Press, Los Alamitos (1999)
6. Saksena, M., Wang, Y.: Scalable real-time system design using preemption thresholds. In: 21st IEEE Real-Time Systems Symposium(RTSS'2000), pp. 25–34. IEEE Computer Society Press, Los Alamitos (2000)
7. Gai, P., Lipari, G., Di Natale, M.: Minimizing memory utilization of real-time task sets in single and multi-processor system-on-a-chip. In: 22nd Real-Time Systems Symposium, London, England (2001)
8. Yumin, Z., Xiaobo, H., Chen Danny, Z.: Task scheduling and voltage selection for energy minimization. In: 39th design automation conference (DMC), New Orleans LA, pp. 183–188 (2002)
9. Intel: Intel xscale processor (2000)
10. Transmeta: Transmeta crusoe processor (2000)
11. Inki, H., Qu Gang, M.M., Potkonjak, M., Srivastavas, M.B.: Synthesis techniques for low-power hard real-time systems on variable voltage processors. In: 19th IEEE Real-Time Systems Symposium, Madrid, Spain, pp. 178–187. IEEE Computer Society Press, Los Alamitos (1998)
12. Kim, S., Hong, S., Kim, T.: A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis. In: Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. IEEE Computer Society Press, Los Alamitos (2002)
13. Pillai, P., Shin, K.G.: Real-time dynamic voltage scaling for low-power embedded operating systems. ACM SIGOPS Operating Systems Review 35, 89–102 (2001)
14. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced cpu energy. In: 36th Annual Symposium on Foundations of Computer Science (FOCS'95), pp. 374–382. IEEE Computer Society Press, Los Alamitos (1995)
15. Liu, J.: Real-Time Systems. Prentice Hall, Upper Saddle River (2000)

16. Quan, G., Hu, X.: Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In: Annual ACM IEEE Design Automation Conference (38th conference on Design automation), Las Vegas, Nevada, United States, pp. 828–833 (2001)
17. Quan, G., Hu, X.: Minimum energy fixed-priority scheduling for variable voltage processors. In: Design, Automation and Test in Europe Conference and Exhibition (DATE'02), pp. 782–787 (2002)
18. Luo, J., Jha, N.K.: Power-conscious joint scheduling of periodic task graphs and a periodic tasks in distributed real-time embedded systems. In: 2000 International Conference on Computer-Aided Design (ICCAD '00), p. 357 (2000)
19. Kuchcinski, Gruian, F., Krzysztof.: Lenes: task scheduling for low-energy systems using variable supply voltage processors. In: The 2001 conference on Asia South Pacific design automation with EDA Technofair Design Automation Conference Asia and South Pacific (ASP-DAC), Yokohama, Japan, pp. 449–455. ACM Press, New York (2001)
20. Regehr, J.: Scheduling tasks with mixed preemtion relations for robustness to timing faults. In: 23rd IEEE Real-Time System Symposium, pp. 315–326. IEEE Computer Society Press, Los Alamitos (2002)
21. Chen, J.: Extensions to Fixed Priority with PreemptionThreshold and Reservation-Based Schedulin. PhD thesis, University of Waterloo (2005)