Energy Efficient Scheduling For Hard Real-Time Tasks On Dynamic Pipeline And Voltage Scaled Processor

Hyodo, Akihiko
Department of Computer Science and Communication Engineering Kyushu University

Muroyama, Masanori
Department of Computer Science and Communication Engineering Kyushu University

Tarumi, Kousuke
Department of Computer Science and Communication Engineering Kyushu University

Yasuura, Hiroto
Department of Computer Science and Communication Engineering Kyushu University

ENERGY EFFICIENT SCHEDULING FOR HARD REAL-TIME TASKS ON DYNAMIC PIPELINE AND VOLTAGE SCALED PROCESSOR

Akihiko Hyodo, Masanori Muroyama, Kousuke Tarumi, and Hiroto Yasuura
Department of Computer Science and Communication Engineering
Kyushu University, Japan
akihiko@c.csce.kyushu-u.ac.jp

ABSTRACT

This paper addresses an energy optimization problem of pipeline depth and voltage scheduling for hard real-time tasks on a dynamic pipeline and voltage scaled (DPVS) processor, which can adjust its pipeline depth and operating voltage dynamically depending on the workload characteristics under the timing constraints. For a set of periodic tasks, we formulate the energy optimization problem as a linear programming (LP) problem. Then, we propose the energy efficient version of widely used fixed-priority and dynamic-priority scheduling methods, which produce a feasible task schedule with optimal processor energy consumption. This paper also provides a set of experimental results to show the effectiveness of the proposed techniques in reducing energy over the existing methods.

KEYWORDS: Low Energy, Energy Efficient Scheduling, Variable Pipeline Depth

1. INTRODUCTION

The demands for mobile and pervasive computing devices have made energy efficient computing a critical technology. To reduce system energy consumption, adapting hardware to workload characteristics with supply voltage reduction is one of the strongly effective techniques [5]. As a consequence, the concept of dynamic pipeline and voltage scaling (DPVS), which is the ability to adjust pipeline depth and operating voltage dynamically to workload characteristics under timing constraints, is first proposed in [5] and the pipeline and voltage scheduling problem of a task on a realistic DPVS model is addressed in [6]. In this paper, we present the energy efficient scheduling techniques to integrate DPVS mechanisms into the two most-studied real-time schedulers, Rate Monotonic (RM) and Earliest-Deadline-First (EDF) schedulers. RM is a static- or fixed-priority scheduler, and assigns task priority according to its period [15]. It always selects first the task with the shortest period that is ready to run (released for execution). EDF is a dynamic-priority scheduler that sorts tasks by deadlines and always gives the highest priority to the released task with the most imminent deadline [1]. A fixed-priority scheduling means that a priority is assigned to a computation only once, while a dynamic scheme allows changing priority with time. In the classical treatments of these schedulers, both assume that the task deadline equals the period, that scheduling and preemption overheads are negligible, and that the tasks are independent [4,9]. In this paper, we maintain the same assumptions. The scheduling techniques to be studied in this paper are preemptive and priority-driven ones. This means that whenever there is a request for a task that is of higher priority than the one currently being executed, the running task is immediately interrupted and the newly requested task is started [11].

2. MODELS AND MOTIVATION

2.1 Real-Time Task Model

There are two types of hard real-time tasks, periodic and sporadic tasks [10]. This paper assumes such a real-time system composed of a set of periodic tasks, and the tasks are executed on a single

DPVS processor. Each periodic task T_k (X_{ijk} ,D_k , P_k) is characterized by its worst-case execution times X_{ijk} at each processor mode $mode_{ij}$ = (p_i ,V_j , F_{ij}), relative deadline D_k , and period P_k . The ready times of the tasks occur periodically with period P_k , the period of the task T_k . In most cases, the deadlines are assumed to be equal to the periods. A periodic task set, denoted by T { T_1 ,T_2 ,…, T_k }, is defined to be a set of arbitrary positive number of such periodic tasks. Any periodic task set has its hyperperiod, which is the least common multiple of all the periods of the tasks in it. Each invocation of the task is called a job and the r-th invocation of task T_k is denoted as $J_{k,r}$. Each job $J_{k,r}$ (r P_k , X_{ijk} , D_k) in a job set J is characterized by its release time r P_k , worst case execution times X_{ijk} at each $mode_{ij}$, and relative deadline D_k .

To assess the feasibility of a schedule, we define the processor utilization factor $U(T)$ to be the fraction of processor time spent in the execution of the periodic task set T. In other words, the utilization factor is equal to one minus the fraction of idle processor time. Since X_{ijk}/P_k is the fraction of processor time spent in executing task T_k, for n periodic tasks, the utilization factor is given by

$$U(T) = \sum_{k=1}^{N} \sum_{j=1}^{M} \sum_{i=1}^{L} \frac{x_{ijk} \cdot X_{ijk}}{P_k} \tag{1}$$

where x_{ijk} denotes the allocation rate of the combination of the i-th pipeline depth and the j-th supply voltage for the k-th task. This paper makes several assumptions for tasks, summarized as follows:

(T1) Deadline for a periodic task's instance is equal to the next request of the tasks.
(T2) Preemption over a task is always possible.
(T3) All overhead for context switching is counted into the corresponding task's computation requirements.
(T4) Tasks are independent with no precedence constraints.
(T5) The worst case execution demand of each task is analyzed and known a priori.

2.2 Energy Model

The dominant source of energy in a digital CMOS microprocessor is the dynamic consumption, which is mainly the charging and discharging of the load capacitances [2]. The dynamic energy consumption is equal to

$$E = \sum_{t=1}^{T} \sum_{g=1}^{G} a_{gt} \cdot C_{L\,gt} \cdot V_{DD}^2 \tag{2}$$

where a_{gt} and $C_{L\,gt}$ denote the switching counts and the load capacitances of the gate g at t-th unit time respectively, T denotes the total execution cycles, G denotes the total number of gates in a circuit, and V_{DD} denotes the supply voltage. When we assume the gates in a circuit form an average collective switched capacitance C, and the total execution cycles $T = IC \cdot CPI$, where CPI is clock cycle per instruction, IC is the total instruction count committed, the energy consumption derived from Eq.(2) can be given by

$$E = IC \cdot CPI \cdot C \cdot V_{DD}^2 \tag{3}$$

Then, we develop this energy model on the following assumption: CPI increases linearly as the number of pipeline depth increases, i.e., $CPI = CPI_o + K_s \cdot p$ where CPI_o denotes the clock

cycles without stall or NOP cycles per instruction, K_s denotes the slope which depends on workload characteristics, p denotes the pipeline depth. This assumption is proved to be reasonable and proper by our extensive simulations [5] and also referred in [3]. Additionally we define the ratio between the average switched capacitance during a NOP or stall cycle C_s and that of a typical instruction execution cycle C_o as $r_s = C_s / C_o$, which depends on the circuit design. The developed energy model derived from Eq.(3) becomes as

$$E = IC \cdot C_o \cdot (CPI_o + r_c \cdot K_s \cdot p) \cdot V_{DD}^2 \tag{4}$$

If there exists a deadline time D, we should satisfy the equation of $IC \cdot CPI / F \le D$ where F is the operating frequency.

3. OPTIMAL PIPELINE AND VOLTAGE SCHEDULING PROBLEM

In this section, we address the problem of the optimal pipeline-depth and voltage scheduling for minimizing energy, which is indispensable for exploiting the benefit of a DPVS processor and formulate it by using Linear Programming (LP) method.

3.1 Assumptions

The assumptions of the processor for formulation are as follows:

(P1) The processor can vary its pipeline depth and operating voltage dynamically without any adaptation overhead in terms of speed and energy.
(P2) The processor has only a small number of available pipeline depths and voltages.
(P3) The processor equips an adaptive clock scheme which tracks the alteration of pipeline depth and voltage.
(P4) The processor exploits instruction level parallelism uniformly over a task.

3.2 Notations

The variables used in the formulation are defined as follows:

- N The number of periodic tasks $N=|\{T_k\}|$
- IC_k The instruction count of the k-th task
- CPI_{ik} The average CPI of the k-th task when the i-th pipeline depth is applied
- C_{ik} The average switched capacitance per cycle of the k-th task when the i-th pipeline depth is applied
- M The number of available voltages $M=|\{V_j\}|$
- V_j The j-th supply voltage $(1 \leqq j \leqq M)$
- L The number of available pipeline depths $L=|\{p_i\}|$
- p_i The i-th pipeline depth $(1 \leqq i \leqq L)$
- F_{ij} The clock frequency when the i-th pipeline depth and the j-th supply voltage are applied
- $mode_{ij}$ The (i,j)-th execution mode for the processor. $mode_{ij} = (p_i, V_j, F_{ij})$
- x_{ijk} The allocation rate of the combination of the i-th pipeline depth and the j-th supply voltage for the k-th task $(0 \leqq x_{ijk} \leqq 1)$

3.3 Formulation

The energy optimal pipeline depth and voltage scheduling problem for a set of periodic tasks is formulated as follows:

$$\textit{Minimize} \quad E = \sum_{k=1}^{N}\sum_{j=1}^{M}\sum_{i=1}^{L} x_{ijk} \cdot \frac{lcm(T)}{P_k} \cdot IC_k \cdot CPI_{ik} \cdot C_{ik} \cdot V_j^2 \tag{5}$$

$$\textit{Subject to} \quad \forall i \in [1, L], \forall j \in [1, M], \forall k \in [1, N],\ 0 \le x_{ijk} \le 1,$$

$$\forall k \in [1, N], \quad \sum_{j=1}^{M}\sum_{i=1}^{L} x_{ijk} = 1, \quad \sum_{j=1}^{M}\sum_{i=1}^{L} x_{ijk} \cdot X_{ijk} \le D_k \quad ,$$

$$U(T) \le U_{\text{lub}}(T) = \text{The lest upper bound of processor utilization}$$

$$\text{where} \quad U(T) = \sum_{k=1}^{N}\sum_{j=1}^{M}\sum_{i=1}^{L} \frac{x_{ijk} \cdot X_{ijk}}{P_k},\ X_{ijk} = \frac{IC_k \cdot CPI_{ik}}{F_{ij}}$$

$lcm(T)$ denotes the least common multiple of T.

Find the optimal processor mode assignment x_{ijk} to each task, which minimizes the processor energy consumption. Note that $U_{\text{lub}}(T)$ differs among the scheduling algorithms.

4. PRIORITY-DRIVEN OPTIMAL SCHEDULING

In the following sections, we present both fixed-priority and dynamic-priority scheduling algorithms under fixed processor mode assignment policy. Fixed-priority means the priorities are assigned to tasks once and for all, while dynamic-priority means the priorities of tasks might change from request to request. In this paper, we also use the word of fixed of dynamic or in the same manner to classify the methods of processor mode assignment.

4.1 Fixed-Priority Fixed-Mode Scheduling

In a typical real-time system, there are many periodic tasks that share hardware resources [8]. To ensure that each task satisfies its timing constraint, the execution of tasks should be coordinated in a controlled manner. This is often done via fixed-priority scheduling. Fixed-priority scheduling has several advantages over other scheduling schemes. It is quite simple to implement in most kernels and is adopted in most real-time scheduling algorithms of practical interest due to its low overhead and predictability. Also, many analytical methods are available to determine whether the system is schedulable. Rate Monotonic (RM) scheduling is the first scheduling scheme that falls into this category. It assigns a higher priority to a task with a shorter period or with a higher execution rate. It is proved to be optimal in the sense that if a given task set fails to be scheduled by RM, it cannot be scheduled by any fixed priority scheduling. Computing the priorities of a set of n tasks for the RM priority rule amounts to ordering the task set according to their periods. Hence the time complexity of the RM priority assignment is the time complexity of a sorting algorithm, i.e., $O(n \log n)$ [10]. As the RM priority assignment is optimum, the utilization factor achieved by RM for a given task set is greater than or equal to the utilization factor for any other priority assignment for that task set. Then let us consider the worst case utilization bound of RM.

THEOREM 1:[12] *For a set of n periodic tasks with fixed-priority assignment, the least upper bound to the processor utilization factor is $U = n(2^{1/n} - 1)$.*

Since $U = n(2^{1/n} - 1)$ decreases monotonically from 0.83 when $n = 2$ to $\log_e 2 = 0.693$ as $n \to \infty$, it follows that any periodic task set of any size will be able to meet all deadlines all of

the time if the RM algorithm is used and the total utilization is not greater than 0.693. This utilization bound is a sufficient. It is quite possible that a given task set T { T_1 ,T_2 ,…, T_k } with its utilization exceeding the bound above be fixed priority feasible. It is known that any periodic task set in which their periods are harmonic, i.e., for every pair of periods P_{k1} and P_{k2} in the task set, it is either the case that P_{k1} is an integer multiple of P_{k2} or P_{k2} is an integer multiple of P_{k1}, is fixed-priority feasible if and only if its utilization is at most one. Nevertheless, this is the best possible test using the utilization of a given task set, and the number of tasks in it, as the sole determinants of feasibility.

We propose the fixed-priority fixed-mode (FPFM) scheduling as is shown in Figure 1. When the input parameters which include a task set T and available processor modes { $mode_{ij}$ } are given, it produces a table of feasible scheduling result during T's highperperiod which minimizes the total energy consumption under RM.

FPFM Scheduling Procedure:

Input: A periodic task set $T\{T_1,T_2,\ldots,T_N\} \neq \{\phi\}$, where $T_k(X_{ijk}, D_k, P_k)$.

Available processor modes $mode_{ij} = (p_i, V_j, F_{ij})$.

Output: A table of feasible energy-efficient schedule during a hyperperiod of T, i.e., a job set assigned its start and completion times, processor modes is produced

0. Initially assign CPU time X_k^{ref} on reference mode to each task T_k.
1. Compute the processor utilization on reference mode $U^{ref}(T) = \sum_{k=1}^{N} \frac{X_k^{ref}}{P_k}$
2. If $U^{ref}(T) \le U_{\text{lub}}(T) = N(2^{1/N} - 1)$ then

   Find the optimal processor mode assignment $x_{ijk} = x_k^{opt}$ to each task T_k by using proposed LP problem (Eq.(4)). Then, assign CPU time X_k^{opt} to each T_k.

   Else Return no feasible schedule exists
3. Sort the task set T according to the order of higher execution rate first.

   $T\{T_1,T_2,\ldots,T_N\}$ $(P_k \le P_{k+1})$
4. Generate the invoked job set J from the sorted task set T.

   $J\{J_{k,r} \mid k = 1,2,\ldots,N,\ r = 0,1,\ldots,lcm(T)/P_k - 1\}$
5. Compute start time $s_{k,r}$ and completion time $e_{k,r}$ of each $J_{k,r}(r \cdot P_k, X_k^{opt}, D_k)$.
6. Reclaim x_k^{opt} by exploiting any slack time, i.e., find $x_k'^{opt}$ such $U'^{opt}(T) = 1$
7. Return J with $x_k'^{opt}$, $s_{k,r}$ and $e_{k,r}$

Figure 1. A pseudo code of FPFM Scheduling

4.2 Dynamic-Priority Fixed-Mode Scheduling

The most widely studied and adopted dynamic-priority scheduling algorithm is Earliest Deadline First (EDF) algorithm. EDF is said to be optimal in the sense that no other dynamic, as well as fixed, priority-driven scheduling algorithm can lead to a feasible schedule which cannot be obtained by EDF [11]. The EDF scheduling algorithm is defined as follows. At each time instant *t*, schedule the job active at time instant *t* whose deadline parameter is the smallest [10]. EDF has an

apparent dominance over RM because it can schedule a task set if and only if the processor utilization is lower than or equal to 1 [10], meaning that a schedule with zero slack time is possible.

THEOREM 2:[11] *For a given set of n periodic tasks with dynamic priority assignment, a necessary and sufficient condition to yield the feasible EDF schedule is* $U \le 1$.

This theorem gives us a simple O(n) procedure based on the processor utilization to check the feasibility. The proposed dynamic-priority fixed-mode (DPFM) scheduling is shown in Figure 2.

DPFM Scheduling Procedure:

Input: A periodic task set $T\{T_1, T_2, \ldots, T_N\} \neq \{\phi\}$, where $T_k(X_{ijk}, D_k, P_k)$.

Available processor modes $mode_{ij} = (p_i, V_j, F_{ij})$.

Output: A table of feasible energy-efficient schedule during a hyperperiod of T,
i.e., a job set assigned its start and completion times, processor modes is produced

Since Step 0 to 2 are the same as FPFM Algorithm by substituting $U_{\text{lub}}(T) = 1$,
these steps are omitted to avoid duplications. See Figure 1 about these steps.

0.- 2. The optimal processor mode assignment x_k^{opt} of each T_k is found under EDF.
Then, the corresponding CPU time X_k^{opt} is assigned to each T_k.

3. Generate the invoked job set J from a given periodic task set T.
$J\{J_{k,r} \mid k = 1,2,\ldots,N,\ r = 0,1,\ldots,lcm(T)/P_k - 1\}$
where $lcm(T)$ denotes the least common multiple of T.
Identify each job $J_{k,r}$ as a independent job $J_{k'}$ in J.
$J\{J_{k'} \mid k' = 1,2,\ldots,N'\},\ N' = N \cdot (lcm(T)/P_k - 1)$

4. Sort J according to the order of earlier deadline first.
$J\{J_1, J_2, \ldots, J_{N'}\}\quad (D_{k'} \le D_{k'+1})$

5. Compute start time $s_{k'}$ and completion time $e_{k'}$ of each job $J_{k'}$

6. Return J with x_k^{opt}, $s_{k'}$ and $e_{k'}$

Figure 2. A pseudo code of DPFM Scheduling

5. EXPERIMENTAL RESULTS

We performed several simulations to assess the benefits of the proposed DPVS scheduling techniques over the simply system shutdown approach and dynamic voltage scaling (DVS) approach [13,14]. In experiments, the system shutdown approach assumes the processor has fixed deepest pipeline depth and fixed highest voltage, while DVS assumes the processor can adjust its operating voltage with fixed deepest pipeline depth. Then, the assumed DPVS processor is such that the architecture applied is the same as proposed in [5], which has the power-of-two pipeline depths available, and the available voltage and its voltage-frequency correspondence is the same as Intel's XScale processor [16], while its pipeline depth-frequency correspondence assumed to be linear for simplicity. The available pipeline depths and voltages are {1, 2, 4, 8, 16} stage and {0.75, 1.0, 1.3, 1.6, 1.8} volt respectively. Besides, the corresponding frequency to voltage is {150, 400, 600, 800, 1000} MHz. For the experiments, 10 periodic tasks are randomly generated.

The total number of jobs generated from each task T_k are P_k /*lcm*(T). The period and the worst-case execution time of each task are randomly generated from uniform distribution with the ranges of [10,100] ms and [1, period] ms in steps of 1 ms, respectively. In order to estimate the processor energy consumption, we use the energy model of Eq.(3), assuming the average energy consumed during a NOP or a stall cycle is 50% of that consumed by a typical instruction, i.e., r_s=0.5. The pipeline depth elasticity slope K_s is randomly generated from uniform distribution with the range of [0.1, 0.8] in steps of 0.1 for each task.

For randomly generated periodic tasks, we produce an energy optimal schedule by using FPFM and DPFM, then, estimate energy reduction results of each energy optimization approach, i.e., shutdown, DVS, and DPVS. We repeat this procedure varying the average processor utilization from 0.1 to 1.0 in steps of 0.1 for DPFM, from 0.1 to 0.7 for FPDM, by stretching the worst-case execution time of each task at equal rate. The least upper bound utilization of FPFM for 10 periodic tasks becomes 0.72 while that of DPFM is always 1.0. In order to solve the LP problem, we used Optimization Toolbox 2.0 in MATLAB environment [7], which contains routines that implement the most widely used methods for performing minimization or maximization. The experimental results are shown in Figure 3.

Our comparative study shows there exits the clear advantage of DPVS in energy reduction. When tasks are scheduled by FPFM, DPVS can reduce -49% energy from shutdown and -39% from DVS in average, as is shown in the left figure. When tasks are scheduled by DPFM, DPVS can reduce -72% energy from shutdown and -45% from DVS in average, as is shown in the right figure.

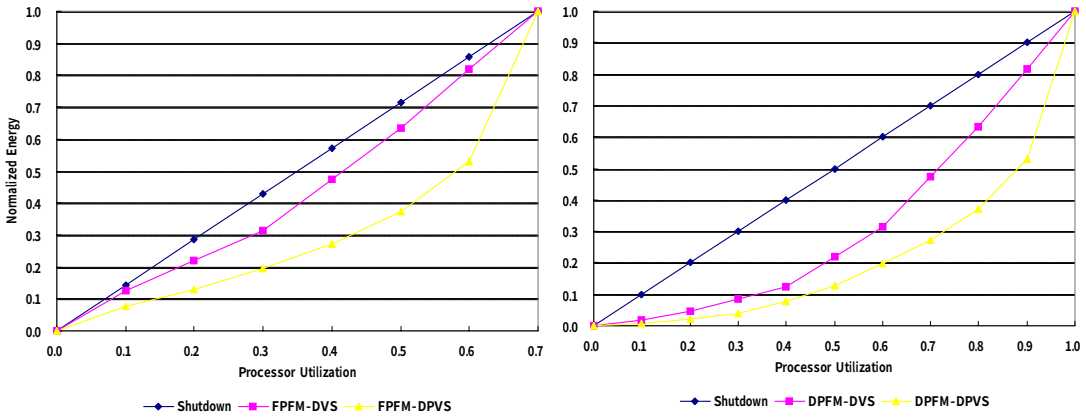

Figure 3. Normalized Energy of Shutdown, DVS, and DPVS

6. CONCLUSION

In this paper, we presented an energy optimization problem for real-time tasks on a DPVS processor. We formulated the optimal scheduling problem of pipeline depth and voltage for minimizing energy consumption by using LP method. We also proposed the energy optimal priority-driven scheduling techniques based on existing task scheduling algorithms such as RM and EDF. The experimental results showed the DPVS processor could reduce energy consumption effectively over well-known existing energy minimization techniques such as system shutdown approach and DVS, when tasks are scheduled by FPFM or DPFM. We observed DPVS can reduce -49% and -39% of energy consumption in average by FPFM, compared to system shutdown and DVS approach respectively, and also -72% and -45% of energy reduction is observed by DPFM.

7. ACKNOWLEDGEMENTS

This work has been supported by the Grant-in-Aid for Creative Scientific Research No. 14GS0218 and the Silicon Sea-Belt Project "Establishing Project of a Cluster for System-LSI Design and Development". We are grateful for their support.

8. REFERENCES

[1] C. L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", Journal of the AACM, 20(2), pp46-61, 1973.

[2] A. Chandrakasan, S. Sheng, R. Brodersen, "Low power CMOS digital design", IEEE Journal of Solid-State Circuits, April 1992.

[3] A. Hartstein, T. R. Puzak, "The optimum pipeline depth for a microprocessor", In International Symposium on Computer Architecture, pp. 7-13, May 2002.

[4] Hong, I., Potkonjak, M., and Srivastava, M. B.,"On-line scheduling of hard real-time tasks on a variable voltage processor", Proc. International Conference on Computer Aided Design (ICCAD), pp. 653-656, 1998.

[5] A. Hyodo, M. Muroyama, and H.Yasuura, "Variable Pipeline Depth Processor for Energy Efficient Systems", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E86-A, No.12, pp.2983-2990, 2003.

[6] Akihiko Hyodo, Masanori Muroyama, Kousuke Tarumi, Kouji Makiyama, Hiroto Yasuura, "Dynamic Pipeline and Voltage Scaling on a Low-Power Microprocessor," Proc. of International Symposium on Information Science and Electrical Engineering, pp444-447, 2003.

[7] TheMathWorks, Inc. MATLAB Optimization Toolbox. http://www.mathworks.com/

[8] G. Quan and X. (Sharon) Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors", Proc. of IEEE/ACM Design Automation Conference, pp 828-833, 2001.

[9] Woonseok Kim, Dongkun Shin, Han-Saem Yun, Jihong Kim, Sang Lyul Min, "Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems", IEEE Real Time Technology and Applications Symposium 2002.

[10] Jean-François Hermant, Laurent Leboucher, and Nicolas Rivierre, "Real-Time Fixed and Dynamic Priority Driven Scheduling Algorithms: Theory and Experience", INRIA Research Report 3081, 1996.

[11] Johngwon Lee, Sungyoung Lee, Hyungill Kim, "Scheduling of Hard-Aperiodic Tasks in Hybrid Static/Dynamic Priority Systems", ACM(American Computing Machinery) SIGPLAN Notices, Vol.30, No.11, pp7-19, ACM, U.S.A., November 1995.

[12] Han-Saem Yun, Jihong Kim, "On energy-optimal voltage scheduling for fixed-priority hard real-time systems", ACM Transactions on Embedded Computing Systems (TECS) August 2003.

[13] T. Pering, T. Burd, R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", In International Symposium on Low Power Electronics and Design, pages 76-81, August 1998.

[14] T. Ishihara, H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors", In International Symposium on Low Power Electronics and Design, pages 197-202, August 1998.

[15] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior", In IEEE Real-Time Systems Symposium, 1989.

[16] Intel Corporation, "Intel Xscale core Developer's Manual", January 2004.