A Dynamic-Mode DVS Algorithm under Dynamic Workloads*

Yan Wang and Albert M. K. Cheng
Real-Time Systems Laboratory
Department of Computer Science
University of Houston, TX 77204, USA
(rainyday,cheng)@cs.uh.edu

Abstract

Dynamic Voltage Scaling (DVS) is a promising method to achieve energy saving by slowing down the processor into multiple frequency levels especially in battery-operated embedded systems. We introduce a novel Dynamic-Mode EDF scheduling algorithm when workloads change significantly. Single-Mode, Dual-Mode and Three-Mode frequency setting formats can be applied, based on the Real Execution Time (RET) and the slack time at run-time. Only one combination of the time zone length and the frequency scaling factor can lead to the best energy saving. Experimental results show that, given an RET pattern, our Dynamic-Mode DVS algorithm achieves an average 15% energy savings over the traditional two-mode DVS scheme on hard real-time systems.

1. Introduction

Resource requirements change significantly and unpredictably in a dynamic-workload real-time system, where scheduling must be based on the tasks' run-time behavior. Their fluctuating execution times at run-time induce a significant difference between the real and the estimated value of the Worst Case Execution Time (WCET). Pessimistic and optimistic estimates can cause performance degradation and high power consumption in processors.

With the aim of obtaining a correct estimate of the RET, giving more diversification to the scaling frequency adjustment, we present a Dynamic-Mode DVS algorithm for battery-powered real-time systems. A task is split into "three" subtasks. The last time zone is fixed under the maximum frequency. Other two subtasks are assigned to different frequency scaling factors which are proportional to each other by a frequency variant n, which decides the final execution mode of system. At run-time, the frequency scaling factor is adjusted based on S_k, the accumulated slack time and RET. The executing frequency in the dynamic-mode DVS algorithm will be changed more frequently than dual-mode, which provides more opportunity for potential energy saving in the system. Furthermore, we attempt to obtain close-to-optimal energy saving by changing the length of different frequency zones. Only a combination of certain frequency scaling factor and its length time zone assigned to each frequency level can maximize the system's total energy saving.

----

* This work is supported in part by a 2004 grant from the Institute of Space Systems Operations (ISSO).

2. Previous Works

2.1. Dual-mode Voltage Scaling

Power consumption increases proportionally to the processor frequency and to the square of the voltage in the CMOS circuit [7][8], characterized by $P_{cmos} = C_L * V^2_{DD} * f$ (1), where P_{cmos} is the dynamic power dissipation, C_L is the effective switched output capacitance, and f is the clock frequency. DVS [4][5][6] is a promising method for real-time systems to allow multiple voltage levels. Lee et al proposed a flexible dual-mode voltage run-time assignment [3] with the assumption that the WCET of each task is known in advance.

Most of the improvements on accurate prediction of WCET are partially related to feedback control for the real-time scheduler. In Y. Zhu and F. Mueller's recent paper [1], they presented a novel approach combining a DVS scheduler and a feedback controller within EDF scheduling algorithm. The dependency of prediction on previous execution time decides the limitation of feedback technique when dealing with truly random tasks. Our approach uses the idea of feedback control, but the main difference is the frequency setting format for the first two time zones. In our algorithm, even how many modes assigned to the next job is decided at run-time.

2.2. Why Three-Mode is Needed

In recent work, Ishihara and Yasuura [10] proved that for a processor with few multiple discrete voltages, at most two voltages minimizes the energy consumption for a task or schedule with deadline. (Suppose V_{ideal} is the voltage which minimizes the energy consumption for this task. The task runs at a voltage below V_{ideal} up to a point which can be statically determined, and then it runs at a voltage higher than V_{ideal}.) However, this two-voltage theorem is true only for the static case in which the actual execution time of the task is known in advance. For tasks with unknown actual execution times, this theorem is not true since V_{ideal} cannot be determined before the task completes its actual execution. Therefore, our work proposes more than two voltages for running the task to better adapt to its fluctuating execution times and hence reduces energy consumption as much as possible.

We can exhibit several scenarios in which we need 3 or more voltages in order to approximate minimum energy consumption for tasks with dynamic and fluctuating actual execution times. One such scenario is given next. If we know a task's actual execution time, then according to [10], at most two voltages V_1 and V_2 are needed to minimize energy consumption, where $V_1 < V_2$. The task runs at V_1 until time instant P, and then runs at V_2 till its completion, where $V_1 < V_{ideal} < V_2$. If we do not know in advance this task's actual execution time, then only a clairyovant scheduler would know the time instant P. However, since such a scheduler does not exist, P has to be guessed or estimated. The chosen P, denoted P', may be < P, = P, or > P. If P' < P, then the task would finish before its deadline. If we notice this after running the task using V_2, we can switch to a lower voltage after an interval to be determined, hence 3 or more voltages may be needed. If P' > P, then the task would miss its deadline, so we would switch to a higher voltage after a while, hence again 3 or voltages may be needed.

3. System Model

This paper shows the current attempt to reduce energy consumption by using a dynamic speed adjustment method on scheduling dynamic-workload real-time system. Assuming that the system workload changes with a specific pattern, without making any prediction for the future workload, we make use of the RET of the previous job as a parameter to further adjust the frequency scaling factor.

3.1. Assumptions

(1) Hard real-time systems; overhead is negligible.
(2) Task set: independent; fully preemptive; periodic; all tasks arrive at time 0; arrival time is fixed.
(3) Voltage can be changed continuously at run-time.
(4) Continuous processor speed within a certain range.

3.2. Dynamic-Mode Task Splitting

First of all, we use $\alpha=f_i/f_m$ (2) to define the frequency scaling factor [10]: where f_i represents the scaled frequency and f_m represents the maximum frequency of the dynamic real-time system. Assuming that the WCET is C_i when the task is executed at f_m, Sk is the slack time accumulated by only the formerly finished tasks. Task splitting, shown in Figure 1, is done based on the estimated WCET plus the total reclaimed slack time. T_l,T_m,T_h represent the subtasks of task T after splitting; C_l,C_m,C_h represent the WCET of three subtasks at maximum frequency. Based these notations, a task is split into a three-mode model, each is executed at a different frequency level.

Under maximum frequency (without scaling):

$$Ci = C_l + C_m + C_h \qquad (3)$$

Under dynamic voltage scaling (with scaling):

$$C_l/\alpha_1 + C_m/\alpha_2 + C_h/1 = Ci + Sk \qquad (4)$$

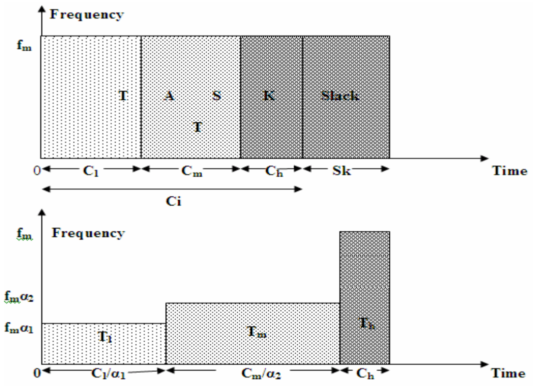


Figure 1. Dynamic-Mode Task Splitting

3.3. Frequency Scaling Factor Setting

The setting of frequency format is based on three thoughts: guarantee time constrains; introduce diversification on frequency scaling; and ensure scaling factor<=1. The last is a crucial requirement that we must take a look when solving α value in equation (4). Based on the definition of α<=1, each scaling factor α_1 and α_2 must also be a value not bigger than 1.

Having known all the constraints and frequency setting thoughts, the value of each assignment format is discussed in Figure 3:

$$\alpha_2 C_l + \alpha_1 C_m = \alpha_1\alpha_2 C_l + \alpha_1\alpha_2 C_m + \alpha_1\alpha_2 Sk \qquad (5)$$

Condition (1): $\alpha_1 = \alpha_2 = \alpha$

$$(C_l + C_m + S_k)\alpha^2 - (C_l + C_m)\alpha = 0$$

$\alpha = (C_l + C_m)/(C_l + C_m + S_k)$ (6)

This condition changes three-mode frequency scaling to two-mode, or even one-mode when S_k=0. This format of frequency setting can be used under the situation that S_k is a relatively small number.

Condition (2): $\alpha_1 < \alpha_2$ assume $\alpha_2 = n\alpha_1$ (n>1)

$\alpha_1 = (nC_l + C_m) / (nC_l + nC_m + nS_k)$ (7)

$\alpha_2 = (nC_l + C_m) / (C_l + C_m + S_k)$ (8)

The scaling factor definition constraints: $n\alpha_1 <= 1$ decides S_k must have a value not less than $(n-1)C_l$. A lower warm-up frequency is followed by a higher frequency that will reserve sufficient time for the task to finish within a certain time constraint.

Condition (3): $\alpha_1 > \alpha_2$ assume $\alpha_1 = n\alpha_2$ (n>1)

$\alpha_1 = (nC_m + C_l) / (C_l + C_m + S_k)$ (9)

$\alpha_2 = (nC_m + C_l) / (nC_l + nC_m + nS_k)$ (10)

in which S_k must have a value not less than $(n-1)C_m$.

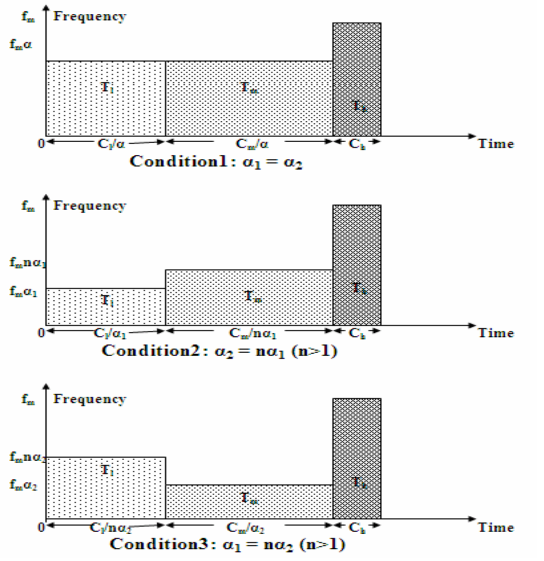


Figure 2. Frequency setting format

After analyzing the three conditions, we declare that α depends on the length of first two time zones: C_l and C_m as well as the accumulated slack time. What kind of workload does each condition suitable for? Firstly, when S_k=0, there is a high rate of missing deadline if frequency scaling is immediately follows. So, for the next job, instead of lowering frequency, we have the aim of accumulating S_k as much as possible. Condition1 should be chosen under such situation that S_k is a relatively small number. Secondly, condition 2 offers 3 different frequency levels. As long as constraint $S_k >= (n-1)C_l$ is satisfied, frequency is gradually increased based on factor n, and deadline is guaranteed to be met. Similarly, condition 3 with the constraints of $S_k >= (n-1)C_m$, frequency is dropped and then increased based on n.

3.4. Frequency Setting Idea

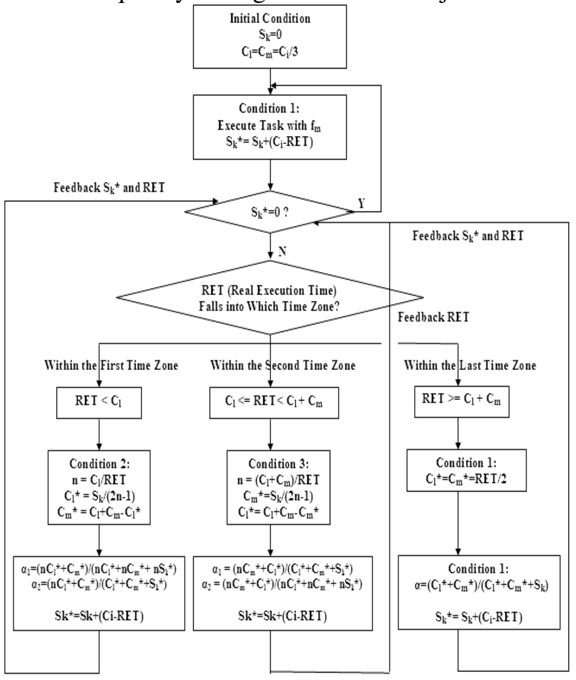
Dynamic-Mode DVS algorithm (Figure 3):

RET falls into 3 kinds:

(1). If RET is within the first time zone, the RET and C_i usually have a large estimation difference. Condition 2 is used as the frequency setting format for the next job. At the start, a lower frequency is used with a higher chance of possibly shorter execution time occurring. If this is not the case, a higher frequency is used to speed-up the execution.

(2). If RET is within the second time zone, the difference between RET and estimation is small. Condition 3 is used as the frequency setting format for next job. At the beginning, a larger frequency is used with the higher chance of possibly longer execution time occurring. If this is not the case, the frequency is lowered to achieve energy saving.

(3). If RET is within the last time zone, the value of RET and estimation is very much the same. This job will leave little slack time for others. Condition 1 is used as frequency setting format for next job.



Initial Condition
S_k=0
$C_l=C_m=C_i/3$
Condition 1:
Execute Task with f_m
$S_k^* = S_k + (C_i - RET)$
Feedback S_k^* and RET
$S_k^*=0$?
Y
N
Feedback S_k^* and RET
RET (Real Execution Time) Falls into Which Time Zone?
Feedback RET
Within the First Time Zone
Within the Second Time Zone
Within the Last Time Zone
RET < C_l
C_l <= RET < $C_l + C_m$
RET >= $C_l + C_m$
Condition 2:
$n = C_l/RET$
$C_l^* = S_k/(2n-1)$
$C_m^* = C_l + C_m - C_l^*$
Condition 3:
$n = (C_l+C_m)/RET$
$C_m^* = S_k/(2n-1)$
$C_l^* = C_l + C_m - C_m^*$
Condition 1:
$C_l^* = C_m^* = RET/2$
$\alpha_1=(nC_l^*+C_m^*)/(nC_l^*+nC_m^*+nS_k^*)$
$\alpha_2=(nC_l^*+C_m^*)/(C_l^*+C_m^*+S_k^*)$
Sk*=Sk+(Ci-RET)
$\alpha_1 = (nC_m^*+C_l^*)/(C_l^*+C_m^*+S_k^*)$
$\alpha_2 = (nC_m^*+C_l^*)/(nC_l^*+nC_m^*+nS_k^*)$
Sk*=Sk+(Ci-RET)
Condition 1:
$\alpha=(C_l^*+C_m^*)/(C_l^*+C_m^*+S_k)$
$S_k^* = S_k + (C_i - RET)$

Figure 3. Algorithm Specification

S_k is updated and checked upon every completion of jobs. The frequency setting format is changed to f_m as soon as it drops to zero. In the full speed running period, we don't care the length for each time zone. The only purpose under this condition is to accumulate as much slack time as possible. The frequency variant factor n's definition is closely related to the proportion of time zone length and the real execution time of the previous job. The time zone length definition should not disobey the constraint under each condition: C_l/C_m should not be bigger than $S_k/(n-1)$. The larger the value of n is, the smaller the time zone length, and the larger the slack time should be compared to C_l or C_m. In order to satisfy the constraints, as well as make the required value not so difficult to satisfy, we choose $S_k/(2n-1)$ as the length of each required time zone. An inter-task level slack reclaiming policy is used in the algorithm with time complexity of O(N), where N is the number of task instances to be scheduled. S_k is updated after the completion of each job, and the value is defined as the previous slack time plus the difference between the WCET and the RET. If the current job uses up all its reserved time as well as the previously accumulated slack time, S_k will drop to zero. Under this situation, a full speed execution will be used on the next job instance. If the job uses only part of its reserved time, S_k should be the previous slack time plus the unused time. The value of (C_i-RET) can be positive, negative, or zero depending on the RET at run-time.

4. Experimental Results

The normalized processor energy consumption is used as a characterized variable to evaluate the performance of the dynamic Multi-Mode DVS algorithm under different system utilizations. A fixed energy consumption is used for a given frequency. Frequency can take value from 0 to f_m; voltages are {0V,1V,…,10V}. Relation between percentage of maximum frequency and voltage is defined as: Voltage = percentage of f_m/10. RET follows a mathematical function pattern: RET=Cmax*Sin(t) and RET=-Cmax*Sin(t), which has both gradually increasing and decreasing parts. Cmax is generated by a random selection function, and has a variation range of 50% to 100% WCET. Figure 6 shows the power consumption for fixed dual-mode, fixed three-mode and our dynamic multi-mode DVS scheme.

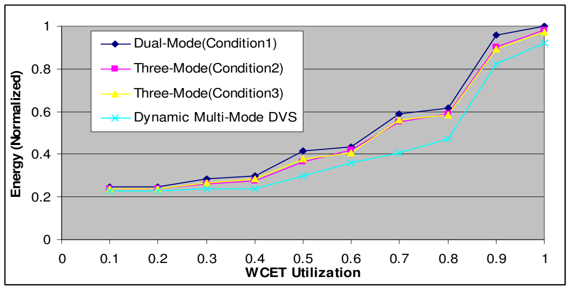


Figure 4. Comparison of Different Mode DVS

The analysis of the experimental results shows that as long as the time constraint is met, fixed mode DVS cannot get the optimal energy saving, whereas our dynamic-mode DVS provides a novel idea to further energy saving. Comparing the result, three-mode DVS saves 5.35% energy consumption compared to dual-mode. Multi-Mode DVS achieves 17% more energy saving over Dual-Mode as well as 12.5% more energy saving over Three-Mode DVS.

5. Ongoing Works

The current idea we hold is still on the preliminary level for the promising dynamic-mode DVS. Further optimization discussion still has potential benefit on more energy saving. Our future goal is to find and apply the improved slack estimation method and optimized parameter tuning to each time zone length.

References

[1] Yifan Zhu, Frank Mueller, "*Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling*", IEEE RTAS, 2004.

[2] M. Weiser, B. Welch, A. Demers, S. Shenker, "*scheduling for reduced CPU energy*," Processings of st USENIX Symposium on Operating Systems Design and Implementation (OSDI'94), pp.13-23.

[3] Yann-Hang Lee, Yoonmee Doh, C. M. Krishna, "EDF Scheduling Using Two-Mode Voltage-Clock-Scaling for Hard Real-Time Systems", ACM CASES'01, Atlanta, Georgia, USA, November 2001.

[4] N. AbouGhazaleh, D. Mosse, B. Childers and R. Melhem. Toward The Placement of Power Management Points in Real Time Applications. *Workshop on Compilers and Operating Systems for Low Power (COLP'01)*, September 2001.

[5] D. Shin, J. Kim and S. Lee. Intra-task voltage scheduling for low-energy hard real-time applications. *IEEE Design and Test of Computers, 18:(2), March-April 2001.*

[6] Rami Melhem, Nevine AbouGhazaleh, Hakan Aydin, Daniel Mosse, *Chapter 7 Power management points in power-aware real-time systems,* University of Pittsburgh.

[7] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *In proceedings of the 1998 international symposium on Low power electronics and design*, pages 197-202. ACM Press, 1998.

[8] T. D. Burd, R. W. Brodersen, Energy efficient CMOS microprocessor design. In *proceedings of the 28th Annual Hawaii International Conference on System Sciences. Volume1: Architecture*, T. N. Mudge and B.D Shriver, Eds., IEEE Computer Society Press, pp.288-297, January 1995.

[9] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. of the Association for Computing Machinery*, 20(1): 46-61, January 1973.

[10] Tohru Ishihara and Hiroto Yasuura, *Voltage Scheduling Problem for Dynamically Variable Voltage Processors*, ACM ISLPED 98, August 1998.

[11] C.-C. Chu and A. M. K. Cheng, ``Static and Dynamic Methods to Improve Total Reward of Tasks in Battery-Powered Devices," Proc. WIP Session, IEEE-CS Real-Time and Embedded Technology and Applications Symposium, 2004.