Minimizing Response Time Implication in DVS Scheduling for Low Power Embedded Systems

Nasro Min-Allah[1,2,3], Asad-Raza Kazmi[2], Ishtiaq Ali[3], Xing Jian-Sheng[1], Wang Yong-Ji[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
nasar.pathan@gmail.com
[2] Graduate University, Chinese Academy of Sciences, Beijing 100039, China
[3] COMSATS Institute of Information Technology, Islamabad 44000, Pakistan

Abstract

Recently, a lot of work has been done on minimizing energy consumption of real time embedded systems by exploiting hardware characteristics of latest processors. However, these techniques are effective to energy reduction at the expense of delayed responsiveness; a feature highly discouraged in real time embedded systems. As opposed to the previous works, we value response time of higher importance than energy reduction after reliability, when a tradeoff is involved. In this paper, we present a novel technique for scheduling mixed tasks on single dynamic voltage scaling enabled processor. The proposed algorithm, preserves all timings constraints for hard periodic tasks under worst case execution time scenario, improves responsiveness to periodic tasks and, saves as much energy as possible for hybrid workload.

1. Introduction

Due to the efforts of scientists, engineers and mathematicians, the law of Gordon Moore- the number of transistors on a microprocessor would double periodically [1]- is maintained for microprocessors . Unfortunately, on the other front, advances in battery technologies are not in pace with microprocessor improvement, as battery capacity is only tripled since 1990 [2, 3]. This gap is now addressed by applying energy efficient techniques at architecture, operating system, protocol and application levels, since energy reduction has become a major design consideration for computing environments, ranging from wearable devices to gird computing and server farms.

In [4], authors formulated a relation between power consumption, operating frequency and operating voltage for CMOS circuitry, which provided a foundation for Dynamic Voltage Scaling (DVS) in latest processors. It shows that neglecting the leakage and short circuit power, power consumption becomes a linear function of frequency (f) and a quadratic function of the operating voltage (v) i.e. $P_{cmos} = v^2 f$. This voltage/speed adjustment on the fly is called DVS; an effective means for power savings in current systems. Unlike typical processors running at maximum speed throughout, latest processors support discrete speed levels (Table 1).

Today, many real-time embedded systems are capable of equally responding to randomly arriving events called aperiodic tasks, along with periodic task. In such systems a delayed response may result in performance degradation ranging from degraded QoS to user's frustration and even critical system failure. Although the focus of previous studies [5, 6, 7, 8, 9, 10] is minimizing system energy, however energy saving is left behind by system responsiveness on hand held devices running embedded applications in general and interactive application in particular. Applying DVS to mixed tasks require a compromise between two conflicting terms, namely, responsiveness and energy reduction. In this paper, we propose a novel algorithm for hybrid task scheduling which is applicable to power efficient hand held devices running performance intensive applications, where responsiveness is of higher importance than energy reduction after reliability.

The rest of the paper is organized as follows. In Section 2, we discuss related work and formulate our problem in Section 3. A novel technique for scheduling mixed workload from DVS perspective is presented in Section 4. Experimental results are given in Section 5. We conclude our paper in Section 6.

2. Related work

In their pioneer work, Weiser et al. [6] and a year later Chan et al. [7] proposed DVS algorithms for reducing energy consumption of processor by dividing time slots into

978-1-4244-1841-1/08/$25.00 ©2008 IEEE

Table 1. DVS-enabled processors

Processor	Speed (MHz)	Voltage (V)
StrongARM SA[16]	$150-600$	$0.75-1.30$
PXA250[15]	$100-400$	$0.85-1.3$
Cursoe(TM5400)[18, 21]	$200-700$	$1.10-1.65$
ARM7D[19]	$20-33$	$3.3-5.0$
PowerPC860[17]	$25-50$	$2.4-3.3$
Itsy[21]	$59-206$	$1.0-1.5$
Intel XScale [24]	$150-1000$	$0.75-1.80$

fixed-length intervals. Based on the CPU utilization in previous intervals, their algorithms predict the CPU utilization during next interval in advance and adjust the system speed accordingly. Unlike interval based strategies, more promising task based strategies were proposed recently. Pillai and Shin's work [5] provides real-time guarantees for real-time tasks. Yifan and Frank in [8] recently proposed a feedback EDF scheduling for hard real time systems exploiting dynamic workload characteristics, where actual and WCET exhibits a significant variation. Their greedy scheme splits highest priority job into two subtasks. Only highest priority job is scaled exploiting available slack while assuming all other task execute at full speed. Each task is divided into two subtasks i.e. the first subtask exploits available slack and executes at lowest speed to reduce energy consumption while enough time is reserved for second subtask to met deadline. As tasks do not fully utilize WCET, tasks are ideally expected to complete during first subtask. This combination of feedback scheme and task splitting guarantees in deadline requirement of real time tasks while reducing system energy consumption.

Majority of available literature [5, 12, 14] focuses mainly on reducing energy consumption and preserves timing constraints for hard periodic tasks. However, unlike energy efficient hard periodic tasks scheduling, applying DVS to hybrid/mixed task scheduling is very recently addressed in [11, 12], where focus is primarily on energy reduction. In contrast, we propose a novel algorithm for hybrid task scheduling, applicable to power efficient hand held devices running performance intensive applications, where responsiveness is of higher importance than energy reduction after reliability, whenever a tradeoff is involved between these conflicting factors.

3. System Model

Throughout the paper, we assume a DVS-enabled processor running RTOS having negligible task switching overheads. We schedule hybrid/mixed tasks (both periodic and aperiodic) according to EDF scheduling policy, where the task which has the earliest deadline among all ready tasks has highest priority. Our target processor offers discrete speed levels in voltage-frequency (v, f) pairs and is controlled by operating system instructions, where $v_1 < v_2 < ... < v_n$ and $f_1 < f_2 < ... < f_{max}$. Furthermore the overhead of scheduling algorithm and voltage transition is negligible when compared to the WCET of tasks. The power consumption of a processor under the speed f is given by $g(f)$ and energy consumption during the interval $[t_0, t_1]$ is $\int_{t_0}^{t_1} g(f(t))dt$ [12].

Hybrid schedule consists of

3.0.1. Periodic Tasks.

- A task set $T = \{T_1, T_2, ..., T_n\}$ represents hard periodic tasks simultaneously ready at $t = 0$, where every T_i is characterized by a pair of parameters (p_i, c_i), where p_i is time period and c_i is WCET of the task .
- All tasks are independent and preemptable.
- The relative deadline D_i of a task T_i is equal to p_i.
- The periodically released instance $R_{i,j}$ of T_i is called $j-th$ job of periodic task T_i. The release time of $R_{i,j}$ instance is $p_i \times (j-1)$.
- The WCET of T_i is known in advance.

3.0.2. Aperiodic Tasks. Aperiodic jobs $\{\sigma_k | k = 1, 2, ...\}$ are modeled by a pair (r, e) of parameters, where r is release time of job and not known in advance, e is average WCET of σ_k, and is known only when job arrives at $t = r_k$. Considering r and e as mean inter-arrival and execution times respectively, aperiodic load is represented by $\omega = e/r$. To evaluate our algorithm, we vary this load in our experiments up to the maximum limit.

We deploy a dedicated server to handle aperiodic load called Total Bandwidth Server (TBS) [22]. TBS is represented in terms of capacity $u_s = c_s/p_s$, where c_s is called execution budget and p_s is period of the server. In this algorithm, the $k-th$ aperiodic job σ_k, with execution time e_k arrives at $t = r_k$, is assigned deadline

$$d_k = max(r_k, d_{k-1}) + \frac{e_k}{u_s} \tag{1}$$

where e_k is WCET of aperiodic task σ_k. The higher is u_s, the earlier is d_k. Initial deadline d_0 is always 0.

4. Scheduling Mixed workload

4.1. TBS at Full Speed (No-DVS)

According to EDF scheduling test, a task set can be feasibly scheduled iff

$$u_{tot} = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1 \qquad (2)$$

Where u_{tot} is called total system utilization. As we have to accommodate aperiodic jobs along with period tasks, total CPU utilization is portioned between u_p and u_s such that, the timing constraints of periodic tasks remain intact in presence of TBS iff

$$u_p + u_s \leq 1 \qquad (3)$$

subject to $0 \leq u_p, u_s < 1$ and $0 < u_p + u_s \leq 1$, at frequency $f = f_m$ ($f = 1$ in this case), where u_p is worst case utilization of periodic tasks. We represent this approach by No-DVS in rest of the paper.

4.2. Static Speed

Equation 3 gives the lowest possible system utilization running at maximum speed. Assuming all task instances run for their WCET, the processor utilization is often far lower than 1.0 and results in idle intervals. Such intervals can be exploited to reduce energy consumption by statically slowing down the processor and operating at a lower voltage. System utilization can be increased and energy consumption is reduced by lowering operating frequency. However, lowering frequency also means performance degradation of the system as the execution time of a task always takes longer at lowered speed than running at maximum speed. Based on [5], the frequency component can be added to Equation 3 as

$$u_p + u_s \leq \frac{f_i}{f_m} \qquad (4)$$

where f_i is the suitable speed for task set, so that no task misses the deadline and f_m gives the maximum speed ($0 < f_i/f_m \leq 1$). This arrangement is done statically and we denote this initial speed f_i by f_{static}, which is the minimum speed to successfully execute hybrid task set. For the sake of brevity, we denote f_i/f_m by α_i , hereafter throughout in this paper.

4.3. Deadline-based Frequency Scaling Algorithm (DFSA)

Although, Equation 4 provides the lowest possible frequency to successfully schedule mixed tasks, execution times are scaled by a factor of $1/\alpha_i$. Lowering frequency means lowering voltage and it is clear from $P_{cmos} = v^2 f$ that lowering speed offers gain in power reduction at the expense of performance (response time) degradation. Since energy acquired by an application during time t is $E = P.t$. Consequently, energy consumption becomes $E \propto v^2$.

Since execution times of jobs are scaled when running at lower frequency α_i. The deadline assignment policy of TBS becomes [12]

$$d_k(\alpha_i) = max(r_k, d_{k-1}) + \frac{e_k}{u_s.\alpha_i} \qquad (5)$$

Hence, the deadline for TBS is delayed as

$$\begin{aligned} d_k(\alpha_i) - d_k &= max(r_k, d_{k-1}) + \frac{e_k}{u_s.\alpha_i} - max(r_k, d_{k-1}) + \frac{e_k}{u_s} \\ &= \frac{e_k}{u_s}\left(\frac{1}{\alpha_i} - 1\right) \qquad (6) \end{aligned}$$

As deadline is prolonged, aperiodic task's response time also increases. In a large number of real-time applications, aperiodic jobs contribute very little to total workload such as Java based videophone, which needs to run garbage collector(aperiodic job) almost every 600ms for 3.732ms [12]. For systems, where aperiodic jobs rarely arrive as compared to periodic tasks and need quick responsiveness, one possible solution to Equation 6 is running aperiodic jobs at maximum available speed, however this scheme is impractical as energy-voltage curve is convex in nature [25], a small increase in voltage brings quadratic increment in power consumption.

Equation 6 clearly means lowering scheduling priority and delayed response time for response sensitive aperiodic jobs. Like periodic tasks, initially, we assume, as long as response time of aperiodic task is less than p_s (worst case), frequency scaling is unnecessary. In order to avoid performance degradation of aperiodic jobs, we restrict this deadline delay. As mentioned earlier, it is safely assumed that TBS has to execute aperiodic jobs for c_i intervals during any interval of length p_s. Similarly, when applying DVS to our model, we provide a constraint that this delay must be less than or equal to p_s i.e. $d_k(\alpha_i) - d_k \leq p_s$. As we have a range of speed levels ($f_1 < f_2 < ... < f_m$), suitable frequency f_k for aperiodic job $\sigma_k$can be obtained by

$$\alpha_i = \left(1 + \frac{p_s.u_s}{e_k}\right)^{-1} \qquad (7)$$

In case $d_k(\alpha_i) - d_k \leq p_s$, our algorithm runs aperiodic load with $\alpha_i = f_{static}$.

5. Experimental Results and Analysis

For our experiments we use the power energy model of Transemmta's Cursoe processor as given in Table 2 [24].

Table 2. Characteristics of Cursoe processor

Frequency	Voltage	Power
300	1.20	1.30
400	1.23	1.80
500	1.35	2.73
600	1.53	4.21
700	1.75	6.43
800	2.00	9.60
900	2.35	14.91
1000	2.80	23.52

To apply Crusoe processor to our task model, we extrapolate the last three rows of Table 3. To study energy consumption and response times of aperiodic jobs are shown in the following. We varied aperiodic load from 0.1 to 0.8 (maximum possible load).

5.1. Energy Savings

Figure 1 gives the energy consumption of aperiodic tasks, when we apply No-DVS, Static-Speed and DFSA. The highest energy consumption is attributed to No-DVS because all tasks are executed at maximum speed. Both Static-Speed and DFSA are normalized to No-DVS. The static adjustment is directly linked to system utilization, the higher is utilization the more is energy consumption. The reason for this behavior is that it executes all tasks with same frequency that is decided statically $f_{static} \leq f_m$. Both, No-DVS and Static-Speed consume a fixed amount of energy, which remains constant throughout the application, as their speed is not influenced by variation in aperiodic load. However, for DFSA, aperiodic load plays a crucial role here; influences energy consumption greatly. When aperiodic utilization is increased, larger deadlines are assigned by TBS. This is the point where tradeoff has to be made: either execute aperiodic job at lower speed with minimum energy consumption and accept maximum response time or, execute aperiodic job at maximum speed with maximum energy consumption so that no performance lost are observed. In such situation, we opt for system responsiveness, while keeping energy consumption lowered, where possible. It can be seen that when aperiodic load is increased, the slope of DFSA becomes steeper because aperiodic tasks are executed with higher speed.

5.2. Performance Degradation

Figure 2 provides a comparison among average response times of aperiodic jobs, running at maximum (f_m), Static (f_{static}) and restricted speed (α_i). Maximum speed (f_m) gives shortest response time, while Static speed (f_{static}) results in highest values; tasks are running at lowest possible speed such that timing constraints are preserved. DFSA has restricted the response time delay ($d_k(\alpha_i) - d_k \leq p_s$). We varied aperiodic load from to 0.1 to 0.8. Initially, response times are the same for all techniques, as aperiodic utilization is low and the difference becomes clear when higher aperiodic load is applied. DFSA never exceeds static graph, whatever is the utilization. At higher aperiodic load, the response time of jobs with static technique experience quick raise; can not comply with higher aperiodic demands. In contrast DFSA closely follow No-DVS approach since $d_k(\alpha_i) - d_k \leq p_s$. When aperiodic load is high, higher speed is assigned to aperiodic jobs by DFSA and thus its average response times become similar to No-DVS, since $\alpha_i = f_m$. Although No-DVS has better results, the energy consumption is very high, as shown in Figure 1. The performance loss with associated DVS schemes such as static scheme is effectively overcome by DFSA, which is the main contribution of this paper.

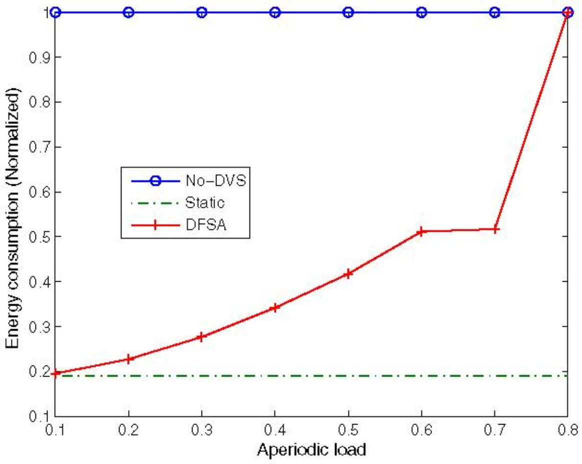

Figure 1. Energy consumption as a function of aperiodic load for DFSA

6. Conclusions and Future Work

Dynamic Voltage Scaling has been projected as a promising technique for minimizing power consumption of low powered devices. An inherit drawback associated with DVS is performance degradation.

We have proposed a novel technique that minimizes power consumption of latest real-time systems by avoiding performance lost. The performance lost is bounded by restricting aperiodic tasks deadlines. The scheme is evaluated

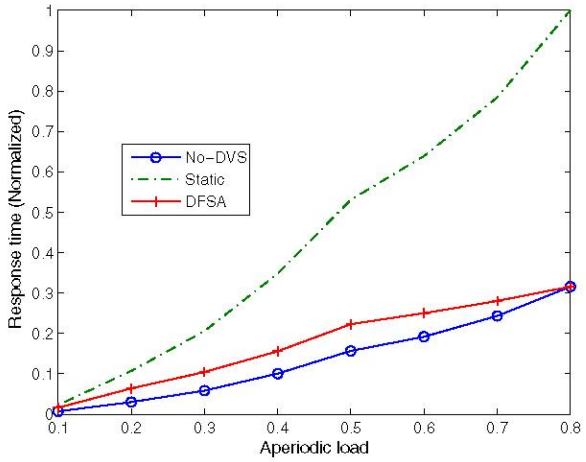

Figure 2. Effectiveness of DFSA over Static approach

in light of maintaining pre-defined performance criteria, assuming task's WCET and , varying aperiodic load. As a future work, we are indented to further reduce performance penalty through slack stealing mechanism by considering the early completion of jobs.

References

[1] G.E. Moore, Cramming More Components onto Integrated Circuits, Electronics, vol. 38, No. 8, pp. 114117, 1965.

[2] N. A. Ghazaleh, B. Childers, D. Mosse, R. Melhem, and M. Craven, Energy Management for Real-Time Embedded Applications with Compiler Support, In Proceedings of ACM SIGPLAN Conference on Languages, Compilers, and Tools for Embedded Systems, 2003, pp. 284-293.

[3] Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, MA: Addison-Wesley Reading, 1998.

[4] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low Power CMOS Digital Design, IEEE Journal of Solid State Circuits, 1992, pp. 472-484.

[5] P. Pillai, and K. G. Shin, Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, In Proc. of ACM Symp. On Operating Systems Principles, pages 89-102, 2001.

[6] Weiser, B. Welch, A. Demers, and S. Shenker, Scheduling for reduced CPU energy, In Proceedings of the 1st Symposium on Operating Systems Design and Implementation, pages 13-23, November 1994.

[7] E. Chan, K. Govil, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power CPU, In Proceedings of the 1st ACM International Conf. on Mobile Computing and Networking (MOBICOM 95), pages 13-25, November 1995.

[8] Yifan Zhu, and Frank Mueller, Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling, In Proceedings of 10th IEEE Real-Time and Embedded Technology and Applications Symposium, 2004.

[9] Y. Shin, and K. Choi, Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems, In Proceedings of Design Automation Conference, 1999, pp. 134-139.

[10] V. Raghunthan and C. L. Pereia, Energy Aware Wireless Systems with Adaptive Power-Fidelity Tradeoffs, IEEE Transactions on Very Large Scale Integration Systems, Vol. 13, No. 2, 2005.

[11] H. Aydin and Q. Yang. Energy-Responsiveness Tradeoffs for Real-Time Systems with Mixed Workload, In Proceedingsof of 10th IEEE Real-time and Embedded Technology and Applications Symposium, pages 74-83, 2004.

[12] D. Shin and J. Kim:, Dynamic voltage scaling of mixed task sets in priority-driven systems. IEEE Transaction on CAD of Integrated Circuits and Systems 25(3): 438-453 (2006)

[13] Jay Heeb, The next generation of StrongArm, Embedded Processor Forum, MDR (1999).

[14] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez, Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems. In Proceedings of IEEE Real-Time Systems Symp., pages 95-106, 2001.

[15] "Introduction to Thumb, ARM Documentation, Advanced RISC Machines Ltd.

[16] MPC860 PowerPC Hardware Specification, MPC860EC/D, Motorola (1998).

[17] Transmeta Corporation, TN5400Processor Specification, (2000).

[18] AMD, Inc. AMD PowerNow Technology, 2000.

[19] R. Hamburgen, D. Wallach, M. Viredaz, L. Brakmo, C. Waldspurger, J. Bartlett, T. Mann, and K. Farkas. Itsy: Stretching the Bounds of Mobile Computing. IEEE Computer, 34(4):28-36, 2001.

[20] http://www.transmeta.com, 2000

[21] D. Shin, J. Kim, and S. Lee, "Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications",In Proceedings of IEEE Design & Test of Computers, 2001.

[22] M. Spuri and G. Buttazzo,Scheduling Aperiodic Tasks in Dynamic Priority Systems, Journal of Real-Time Systems, 10(2):179-210, 1996.

[23] J. Pouwelse, K. Langendoen, and H. Sips, Dynamic voltage scaling on a low-power microprocessor, In International Conference on Mobile Computing and Networking, pages 251-259. ACM Press, 2001.

[24] Intel, Inc. The Intel(R) XScale(TM) Microarchitecture Technical Summary, 2000.

[25] T. Ishihara and H. Yasuura, Voltage Scheduling Problem for Dynamically Variable Voltage Processors, In Proceedings of International Symposium On Low Power Electronics and Design, 1998, pp. 197-202.

[26] T. D. Burd., T. A. Pering, A. J. Stratakos, and R. W. Rodersen, A dynamic voltage scaled microprocessor system. IEEE Journal of Solid-State Circuits, Vol. 35, No. 11, 2000.