Run-time Voltage Hopping for Low-power Real-time Systems

Seongsoo Lee and Takayasu Sakurai

Center for Collaborative Research and Institute of Industrial Science
University of Tokyo, Japan

{cupid,tsakurai}@iis.u-tokyo.ac.jp

Abstract

This paper presents a novel run-time dynamic voltage scaling scheme for low-power real-time systems. It employs software feedback control of supply voltage, which is applicable to off-the-shelf processors. It avoids interface problems from variable clock frequency. It provides efficient power reduction by fully exploiting slack time arising from workload variation. Using software analysis environment, the proposed scheme is shown to achieve 80~94% power reduction for typical real-time multi-media applications.

1. Introduction

Over the past several years, reduction of power consumption has been emerged as a key technology in VLSI system design, especially for portable and battery-powered systems such as a digital cellular phone. *Dynamic voltage scaling* (DVS) [1] is one of the most promising approaches in power reduction, where supply voltage can be dynamically reduced to the lowest possible extent that ensures proper operation, when the required performance of the target system is lower than the maximum performance. Significant power reduction is possible with the DVS scheme, since dynamic power of CMOS circuits, which dominates total power consumption, is proportional to the square of the supply voltage.

Recently, extensive studies have been carried out on the hardware implementation of the DVS scheme [2][3]. A ring oscillator, which is a replica of the critical path of a system under consideration, is used to model the CMOS circuit delay for given supply voltage. Output frequency of ring oscillator is compared with desired clock frequency, and supply voltage is adjusted by frequency-voltage feedback loop [2]. However, this hard-wired approach does not provide efficient control of supply voltage, because voltage-frequency modeling of critical path as ring oscillator is not accurate, nor flexible. Furthermore, this approach cannot be applied to off-the-shelf processor, because critical path is not accessible outside of the chip.

Moreover, in the DVS scheme, system clock frequency can have arbitrary values, which may cause interface problems to exchange data. Especially, this interface problem becomes serious for peripherals or other systems at different clock frequencies.

Another important issue in the DVS scheme is voltage scheduling, i.e. how to determine and schedule the supply voltage for efficient power reduction. Various voltage scheduling methods [4]-[7] have been proposed for real-time systems, based on the following observations.

In real-time systems, the utilization of the processor is frequently less than 1 even if all tasks run at their worst-case execution time (WCET), meaning that there is always some slack time. Moreover, workload of each task may vary from time to time, which results in another kind of slack time. These slack times can be exploited to lower the supply voltage. In this paper, we denote them as *worst-case slack time* and *workload-variation slack time*, respectively.

However, most of the conventional voltage scheduling methods [4]-[6] exploit only worst-case slack time for power reduction, since they assume that all tasks run at their WCETs. This is overcome by Shin and Choi [7] where both worst-case and workload-variation slack times are exploited. Nevertheless, this approach cannot fully exploit workload-variation slack time, because it controls supply voltage on task-by-task basis.

In this paper, we propose a new DVS scheme called *run-time voltage hopping* (RVH). It has the following features [8]: (1) relationship between clock frequency and supply voltage is measured by experiment and is stored as a lookup table in the device driver, (2) it controls clock frequency and supply voltage by software feedback, which can be easily adopted for various targets, (3) it avoids interface problems by exploiting discrete levels of clock frequency as f_{CLK}, $f_{CLK}/2$, $f_{CLK}/3$… where f_{CLK} is the master (=highest) system clock frequency, and (4) it fully utilizes workload-variation slack time by partitioning a task into several pieces, which we call *timeslots*, then dynamically controlling supply voltage on timeslot-by-timeslot basis.

2. System Architecture

Figure 1 shows the conventional DVS system architecture. Supply voltage is controlled by hard-wired frequency-voltage feedback loop, using ring oscillator as a replica of critical path. All chips operate at the same clock frequency and the same supply voltage, which are generated from ring oscillator and voltage regulator. However, this approach has the following problems.

(1) Even in a same chip, critical path may be different along supply voltage, meaning that circuit delay of ring oscillator should have much margin to cover this variation. In this case, however, supply voltage increases, which results in less efficient power reduction.
(2) Since fabrication process technology is different for each chip, circuit delay characteristics may differ a lot, meaning that all chips should be custom-designed to have same voltage-frequency relationship for efficient power reduction.
(3) This approach cannot be applied for off-the-shelf processors, since ring oscillator cannot be inserted into ready-made chips.
(4) In the multi-processor system, it is desirable to control supply voltages separately for each processor, which is impossible in this approach.

Moreover, in the conventional DVS scheme, system clock frequency can have arbitrary values, which may cause serious problems for synchronous data transfer with peripherals or other systems running at different clock frequencies. For example, two

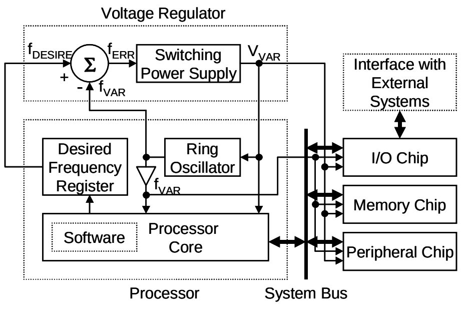


Figure 1: Conventional DVS system architecture.

DVS-oriented systems running at 60 MHz and 49.99 MHz can exchange data only at 10kHz. Similar problems may also occur in the multi-processor system or shared-memory system. Especially, conventional DVS scheme has difficulty in employing synchronous DRAM (SDRAM), which is one of the most popular memory devices.

These problems can be solved if (1) the supply voltage is controlled not by hard-wired feedback but by software feedback, (2) supply voltage is determined based on physical voltage-frequency relationship of each chip, and (3) system clock frequency is restricted to discrete levels. In this paper, we propose a new DVS system architecture, as shown in Figure 2.

Power controller has on-chip DC-DC converter and frequency synthesizer. It generates only f_{CLK}, $f_{CLK}/2$, $f_{CLK}/3$, … to avoid interface problems, where f_{CLK} is the master clock frequency. Device driver has two lookup tables: one for voltage-frequency relationship of the target processor, and the other for transition delay to change clock frequency and supply voltage. These lookup tables are established by measuring the physical characteristics of the chips. For example, voltage-frequency relationship is determined as maximum values among Shmoo plots of all chips in the system. Hardware operation of the proposed system architecture is described as follows.

(1) Desired clock frequency is determined by the proposed voltage scheduling method, which is to be explained in the next section.
(2) Desired supply voltage is looked up from the device driver.
(3) Target processor sets these values into power controller by sending control codes. After that, target processor stops running, and waits while clock frequency and supply voltage are settling down to steady state. Duration of this transition time is looked up from the device driver.
(4) Power controller changes clock frequency and supply voltage. After that, target processor restarts running.

The proposed system architecture is very flexible and can be extended to various applications. For off-the-shelf processors, power controller is implemented as a separate chip, and the power controller is administrated via target processor's I/O port. For custom-designed processors, power controller is embedded in the processor. For multi-processor system or efficient power reduction, power controller has multiple DC-DC converters and common frequency synthesizer, and device driver has individual lookup table for each chip. In this case, simple level converters are employed, because logic thresholds of each chip are different due to different supply voltages.

3. Voltage Scheduling

As explained in Section 1, there are two kinds of slack times inherent in real-time systems, i.e. worst-case slack time and

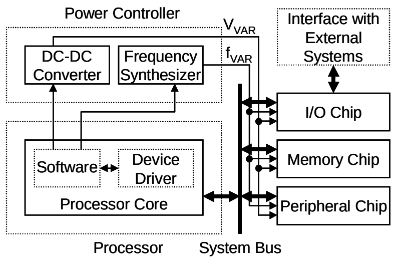


Figure 2: Proposed DVS system architecture.

workload-variation slack time. The former one exists when a real-time system is designed such that utilization, which is computed based on WCETs of tasks, is lower than 1. The latter one exists when the execution time of each task deviates from its WCET.

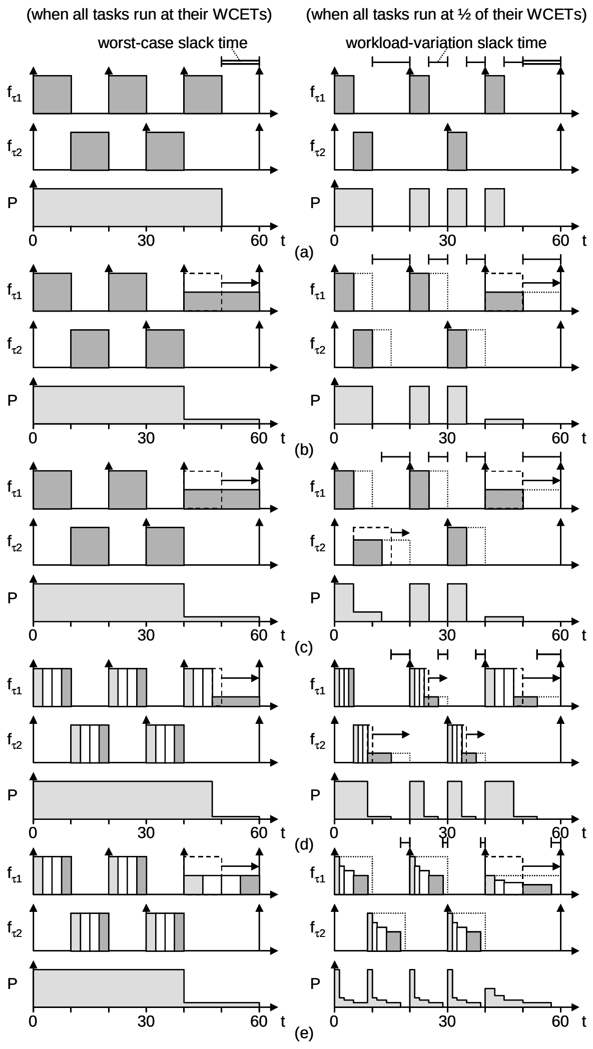
Consider two periodic tasks $\tau_1(20,20,10)$ and $\tau_2(30,30,10)$ as shown in Figure 3, where $\tau_i(T_i,D_i,C_i)$ is characterized by period T_i, deadline D_i, and WCET C_i. As shown in Figure 3(b), most of the conventional voltage scheduling methods [4]-[6] exploit only worst-case slack time, because they assume that tasks *always* run at their WCETs. This problem is partly overcome in [7]. Nevertheless, workload-variation slack time cannot be fully exploited as shown in Figure 3(c), because it performs voltage scheduling on task-by-task basis, i.e. supply voltage cannot be dynamically controlled *inside* a task.

One possible solution is partitioning a task into several pieces, which we call *timeslots*, and considering them as sequentially executed tasks. Seeming to be simple at a glance, however, this approach is impractical from the following problems.

All timeslots have the same period and deadline as original task, meaning that they are released at the same time. Otherwise, these parameters should be updated on every context switching, which is impractical. In [7], supply voltage can be lowered *only if* there's no other task that was already released, meaning that supply voltage can be lowered only for the last timeslot, as shown in Figure 3(d). Therefore, power reduction is not significantly improved, or even degraded.

These problems can be solved if (1) supply voltage is controlled such that each task finishes its execution *within* its WCET, and (2) supply voltage is controlled on timeslot-by-timeslot basis *inside* each task, *independent of other tasks*. One simple solution is embedded voltage scheduling, where supply voltage is controlled in the application program, not in the operating system (OS). As shown in Figure 3(e), this approach significantly improves power reduction. In this paper, we propose a new voltage scheduling method, which is summarized as follows.

(1) A task is divided into N timeslots. Following parameters are obtained through static analysis [9] or direct measurement.
·T_{WC}, T_{WCi}: WCET of whole task and ith timeslot
·T_{Ri}: WCET from (i+1)th to Nth timeslots
(2) For each timeslot, target execution time T_{TAR} is calculated as $T_{TAR} = T_{WC} - T_{WCi} - T_{ACC} - T_{TD}$, where T_{ACC} is accumulated execution time from 1st to (i-1)th timeslots, and T_{TD} is transition delay to change clock frequency and supply voltage.
(3) For each candidate clock frequency f_j, = f_{CLK}/j (j=1,2,3…), estimated maximum execution time T_j is calculated as $T_j = T_{wi}\times j$., where f_{CLK} is the master clock frequency. If f_j is not equal to clock frequency of (i-1)th timeslot, $T_j = T_j + T_{TD}$.
(4) Clock frequency f_{VAR} is determined as minimum clock frequency f_j whose estimated maximum execution time T_j does not exceed target time T_{TAR}, as shown in Figure 4.



(when all tasks run at their WCETs)
(when all tasks run at ½ of their WCETs)
worst-case slack time
workload-variation slack time
(a)
(b)
(c)
(d)
(e)

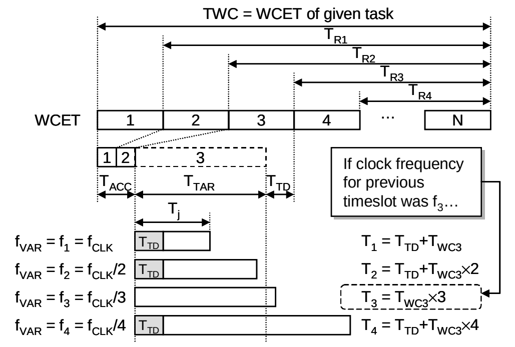
Figure 3: Slack times and voltage scheduling methods. (a) No voltage scheduling, (b) voltage scheduling in [4]-[6], (c) voltage scheduling in [7], (d) voltage scheduling in [7] (4 timeslots), and (e) proposed voltage scheduling (4 timeslots), where $f_{\tau i}$ and P denote clock frequency of task τ_i and power consumption of processor, respectively.

(5) Supply voltage V_{VAR} is determined from lookup table.

Steps (1)-(2) are performed at compile-time, while steps (3)-(5) are performed at run-time. Voltage scheduling is performed independent of other tasks, meaning that it is not affected by preemption or interrupt. Note that given task always finishes its execution within its WCET from following observation. If worst-case occurs, ith timeslot runs at f_{VAR}, and all after (i+1)th timeslot run at f_{CLK}. In this case, total execution time is T_{ACC} (from 1 to (i-1)th timeslot) + $T_{WCi} \times f_{CLK}/f_{VAR}$(ith timeslot) + T_{TD} (transition delay from f_{VAR} to f_{CLK}) + T_{Ri} (from (i+1)th to Nth timeslot), which does not exceed T_{WC} (WCET of given task).

4. Performance Evaluation

In this paper, three typical real-time applications were tested, i.e. MPEG-4 SP@L1 video encoding, MPEG-2 MP@ML video decoding, and RCR-STD27 vector-sum-excited linear prediction (VSELP) speech encoding, which are going to be the killer applications for portable and battery-powered systems.



TWC = WCET of given task
T_{R1}
T_{R2}
T_{R3}
T_{R4}
WCET
1
2
3
4
···
N
T_{ACC}
T_{TAR}
T_{TD}
T_j
If clock frequency for previous timeslot was f_3…
$f_{VAR} = f_1 = f_{CLK}$
$f_{VAR} = f_2 = f_{CLK}/2$
$f_{VAR} = f_3 = f_{CLK}/3$
$f_{VAR} = f_4 = f_{CLK}/4$
$T_1 = T_{TD}+T_{WC3}$
$T_2 = T_{TD}+T_{WC3}\times 2$
$T_3 = T_{WC3}\times 3$
$T_4 = T_{TD}+T_{WC3}\times 4$

Figure 4: Determination of clock frequency.

To evaluate performance improvement of the proposed scheme, these applications were programmed as a single task whose period and deadline are equal to its WCET, where conventional DVS approaches [4]-[7] cannot lower supply voltage at all. In this case, the only way for power reduction is processor shutdown, which is quite less efficient than lowering supply voltage.

Above target applications can be easily regarded as a single task. For example, MPEG-4 application performs video encoding at 15 frame/s, which can be programmed as a single task with period, deadline, and WCET of 1/15 second. We measured the execution time and WCET of every timeslot by running the target application programs on Pentium II processor platform. Based on this timing information, we assumed a virtual processor, which completes these applications just at their deadlines. Voltage scheduling was emulated based on this virtual processor.

Voltage-frequency relationship was obtained from $f^{-1} \propto V/(V-V_{TH})^{\alpha}$ [10] where master supply voltage V_{DD}, threshold voltage V_{TH}, and velocity saturation index α are assumed to be 2.5V, 0.5V, and 1.3. Overhead of embedded voltage scheduling was measured to be less than 0.01% of total workload, which is negligible. Number of timeslots N is 33, 30, and 40 for MPEG-4, MPEG-2, and VSELP applications, respectively. Power efficiency is turned out to be quite insensitive for a wide range of V_{DD}, V_{TH}, α, and N.

Figure 5 shows power consumption of the proposed scheme. This power consumption is normalized by P_{FIX}, which is the power consumption of non-DVS, non-shutdown processor. Power consumption of ideal voltage scheduling [6] is also calculated using post-simulation analysis, which is used as a lower bound. It is unfeasible in real-time systems, because it determines supply voltage of a given task from its actual execution time. In this simulation, all conventional DVS approaches in [4]-[7] have same power consumption, because they can use only processor shutdown for power reduction.

From Figure 5, it is seen that power consumption of the proposed voltage scheduling method is about 6~20% of non-DVS, non-shutdown processors, while those of conventional approaches and lower bound are 20~49% and 3~10%, respectively. Only two (= f, f/2) discrete levels of clock frequency are sufficient, meaning that the proposed scheme is very simple, in both hardware and software. Power efficiency degrades as transition delay of power controller increases, because the target processor stops its execution during transition delay. However, this degradation is not serious when transition delay lies within the practical range.

Figure 6 shows the power consumption for various V_{TH} and V_{DD}, when V_{TH} = 0.2~0.5V, V_{DD} = 0.5~2.5V, and the transition delay T_{TD} = 500μs. Power consumption is independent of

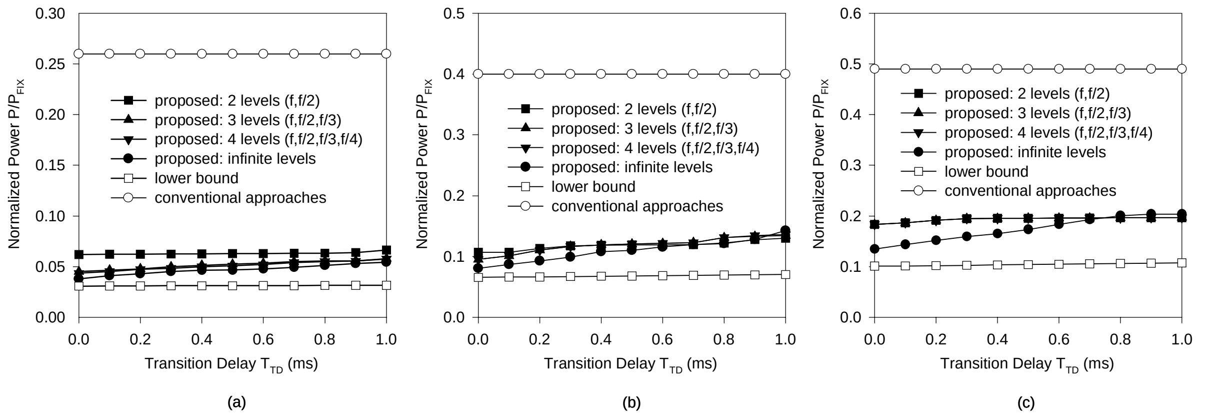


Figure 5: Normalized power consumption for (a) MPEG4, (b) MPEG-2, and (c) VSELP.

individual V_{TH} or V_{DD}, but it is approximately linear function of V_{TH}/V_{DD}. It is seen that the proposed scheme works correctly for a wide range of V_{TH} and V_{DD}.

5. Conclusion

A novel run-time dynamic voltage scaling scheme is proposed for low-power real-time systems. It can be easily applied to various targets including off-the-shelf processors, by employing software feedback control of supply voltage and device driver from physical measurement of voltage-frequency relationship. It avoids interface problems between DVS-conscious systems by exploiting discrete clock frequency of f_{CLK}, $f_{CLK}/2$, $f_{CLK}/3$, … It fully exploits slack time arising from workload variation by partitioning a task into several timeslots and performing run-time supply voltage control on timeslot-by-timeslot basis. When applied to three real-time multi-media applications, the proposed scheme is shown to achieve 80~94% power reduction compared to non-DVS, non-shutdown processors. Currently, extensive studies are in progress on the hardware implementation.

Acknowledgements

The authors thank to Dr. Y. Shin for his valuable comments. This research was partly supported by Mirai-Kaitaku Project.

References

[1] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.

[2] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *Proceedings of IEEE International Solid-State Circuits Conference*, pp. 294-295, 2000.

[3] V. Gutnik and A. Chandrakasan, "An efficient controller for variable supply-voltage low power processing," *Proceedings of IEEE Symposium on VLSI Circuits*, pp. 158-159, 1996.

[4] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," *Proceedings of IEEE Annual Foundations of Computer Science*, pp. 374-382, 1995.

[5] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. Srivastava, "Power optimization of variable voltage-core based systems," *Proceedings of Design Automation Conference*, pp. 176-181, 1998.

[6] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," *Proceedings of IEEE International Symposium on Low Power Electronics and Design*, pp. 197-202, 1998.

[7] Y. Shin and K, Choi, "Power conscious fixed priority scheduling for hard real-time systems," *Proceedings of Design Automation Conference*, pp. 134-139, 1999.

[8] S. Lee and T. Sakurai, "Run-time power control scheme using software feedback loop for low-power real-time applications," *Proceedings of Asia and South Pacific Design Automation Conferences*, pp. 381-386, 2000.

[9] S. Lim, Y. Bae, G. Jang, B. Rhee, S. Min, C. Park, H. Shin, K. Park, and C. Kim, "An accurate worst case timing analysis for RISC proecssors," *Proceedings of IEEE Real-time Systems Symposium*, pp. 97-108, 1994.

[10] T. Sakurai and A. Newton, "Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas," *IEEE Journal of Solid State Circuits*, vol. 25, no. 2, pp. 584-594, Apr. 1990.

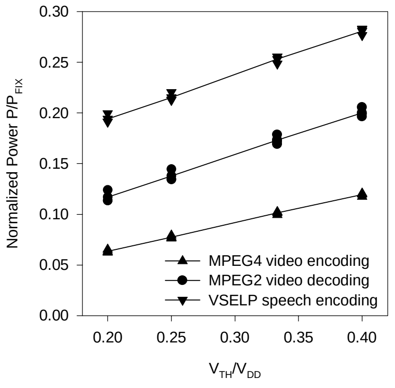


Figure 6: Power consumption for various V_{TH} and V_{DD}.