ENERGY-CONSTRAINED V_{DD} HOPPING SCHEME WITH RUN-TIME POWER ESTIMATION FOR LOW-POWER REAL-TIME VLSI SYSTEMS

SEONGSOO LEE

School of Electronic Engineering, Soongsil University, Seoul, Korea

SEUNGJUN LEE

Department of Information Electronics Engineering, Ewha Womans University, Seoul, Korea

TAKAYASU SAKURAI

Center for Collaborative Research, University of Tokyo, Tokyo, Japan

Received 10 June 2002
Revised 28 August 2002

In this paper, we propose a novel dynamic voltage scaling algorithm on a variable-voltage processor. It determines the supply voltage on timeslot-by-timeslot basis within the task boundary, and significantly reduces the power consumption by fully exploiting the slack time. Also, we modify this algorithm and propose an energy-constrained dynamic voltage scaling algorithm for low-power multimedia applications. In the multimedia applications, there are usually several alternative algorithms with different performance and power. Considering the trade-off between performance and power, the proposed algorithm adaptively determines the optimal alternative to achieve optimal performance under given energy constraint. Compared with the conventional algorithms, the power consumption is reduced to 1/14.4~ 1/5.6 without performance degradation.

1. Introduction

Recently, 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. For these systems, power consumption is a primary design goal, since battery lifetime is one of the most significant performance measures for commercial portable equipments. However, over the past several decades, improvement of battery capacity cannot catch up with progress of semiconductor industry[1], which demands low-power technologies over device, circuit, logic, and system levels.

Dynamic power of CMOS circuits, which dominates total power consumption in most VLSI systems, is given by $P = \alpha C_L V_{DD}^2 f$, where α is the switching activity, C_L is the load capacitance, V_{DD} is the supply voltage, and f is the clock frequency[2]. Showing quadratic dependency, lowering the supply voltage is the most effective way in power reduction. However, it also increases the circuit delay and decreases the clock frequency, resulting in performance degradation of the system throughput. When the required performance of the target system is lower than the maximum performance, dynamic voltage scaling[3] (DVS) is one of the most promising approaches in power reduction, where the supply voltage can be dynamically reduced to the lowest possible extent that ensures proper operation. In the DVS, significant power reduction is possible due to the quadratic power dependency on supply voltage, while meeting the deadline constraints.

Recently, extensive studies have been carried out on the DVS hardware implementation[4-5] based on the frequency-voltage feedback loop. A built-in ring oscillator, which is a replica of the critical path of a target system, is used to model the CMOS circuit delay for given supply voltage. The output frequency of the ring oscillator is compared with the desired clock frequency, and the supply voltage is adjusted by negative feedback. However, this approach is not suitable for off-the-shelf processors, because critical paths of the target processor are not accessible from outside of the chip. The ring oscillator should be embedded in the target processors, meaning that the processor itself should be redesigned. Moreover, it cannot control the supply voltage efficiently, since the voltage-frequency modeling of critical paths is neither accurate nor flexible. Furthermore, the clock frequency of target system can have arbitrary values, which may cause interface problems to exchange data. Especially, this interface problem becomes serious for peripheral devices or other external systems at fixed clock frequencies, meaning that it requires complicated interface circuits or asynchronous data transfer.

In the DVS, another important issue is how to determine and schedule the supply voltage for effective power reduction while meeting given deadline constraints. Various methods[6-9] have been proposed for real-time systems, based on the following observations.

(i) 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.

(ii) Actual execution time of each task is always smaller than its WCET. Moreover, workload of each task may vary along time, depending on the input data. This leads to another kind of slack time.

The DVS exploits these slack times to lower the supply voltage. In this paper, we denote them as *worst-case slack time* and *workload-variation slack time*,

respectively. Most of the conventional methods[6-8] assumes that all tasks run at their WCETs, exploiting only worst-case slack time to reduce the power consumption. To overcome this problem, the fixed-priority low-power scheduling[9] was proposed where the workload-variation slack time is calculated and exploited during run-time. Nevertheless, this approach cannot fully exploit the workload-variation slack time, because it controls the supply voltage on task-by-task basis.

In this paper, we propose a new DVS method called run-time V_{DD} hopping (VH) scheme. It has the following features[10].

(i) It employs pure software control of the clock frequency and the supply voltage, which can be easily adopted for various target systems including off-the-shelf processors. Furthermore, it needs no modification of the operating system (OS).

(ii) It exploits discrete levels of the clock frequency as f_{CLK}, $f_{CLK}/2$, $f_{CLK}/3$... where f_{CLK} is the master (=highest) system clock frequency. This avoids interface problems with peripherals or other external systems.

(iii) It fully utilizes workload-variation slack time by partitioning a task into several pieces called *timeslots* and dynamically controlling the supply voltage on timeslot-by-timeslot basis. This provides effective power reduction, since the longer slack time we exploit, the lower supply voltage we achieve.

In many practical real-time applications, there is a trade-off between the performance and the computation, since in most cases better performance requires more computation. Usually, an application has several alternative algorithms with different computation and performance, and these alternatives can be selected during run-time to optimize both performance and computation. For example, MPEG-4[11] video encoding, which is a typical real-time application in many portable multimedia terminals, has several motion estimation algorithms such as the three-step search[12] (TSS) and the multi-candidate three-step search[13] (MCTSS). The former reduces computation with poor performance, while the latter shows good performance with heavy computation. During run-time, an MPEG-4 application program can select either TSS or MCTSS based on the current workload. When the current workload is large, it selects TSS to reduce the computation, while it selects MCTSS to improve the performance with small workload.

In the DVS, this trade-off can be also useful to optimize the performance and the power, since more computation usually means more power in real-time applications. In the previous example of MPEG-4 video encoding, it adaptively selects TSS to reduce power when the current workload is large, while it selects MCTSS to improve performance at small workload. Unfortunately, most of the

conventional DVS methods fail to consider this trade-off, i.e. they cannot adaptively determine best alternative for optimal formance and power.

In this paper, we also propose a new DVS method called energy-constrained V_{DD} hopping (ECVH) scheme, which is a modified version of the run-time V_{DD} hopping (VH) scheme. It has the following features.

(i) When the application has several alternative algorithms with different performance and power, it adaptively selects optimum algorithm under given time and energy constraints.

(ii) To cover various applications and trade-off models, it provides three user-selectable modes, i.e. the energy-scalable mode, the minimum power mode, and the maximum performance mode. In the energy-scalable mode, it provides the energy-performance scalability, i.e. the performance and the power are roughly proportional to the constrained energy budget. In the minimum power mode, it achieves largest power reduction under given real-time constraint. Similarly, in the maximum performance mode, it achieves highest performance improvement.

The rest of the paper is organized as follows. In Section 2, the problems of the conventional DVS methods are discussed and the run-time V_{DD} hopping (VH) scheme is proposed. The experimental results of the proposed VH scheme are shown in Section 3. The proposed energy-constrained V_{DD} hopping (ECVH) scheme and its experimental results are shown in Section 4. Section 5 concludes the paper.

2. Run-time V_{DD} Hopping Scheme

2.1. Hardware architecture

The system architecture of the conventional DVS scheme is shown in Figure 1. The ring oscillator is embedded in the target processor as a replica of the critical path. After the desired clock frequency is calculated to meet given deadline constraint, it is compared with the output frequency of the ring oscillator. Difference of these two frequencies controls a switching power supply, using frequency-voltage feedback loop. All chips operate at the same clock frequency from the ring oscillator and the same supply voltage from the switching power supply. However, this approach has the following problems.

(i) This voltage-frequency feedback approach cannot be applied for off-the-shelf processors, since the ring oscillator cannot be inserted into ready-made chips. This is a severe disadvantage in the commercial applications where time-to-market is a primary goal.

(ii) The system clock frequency can have completely arbitrary values, which may cause serious problems for synchronous data transfer. For example, two 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 the shared-memory system.

The system architecture of the proposed run-time V_{DD} hopping scheme is shown in Figure 2. It exploits pure software control of the supply voltage and the clock frequency. The OS or the application program determines the desired clock frequency, and then looks up the corresponding supply voltage from the device driver. These supply voltage and clock frequency are applied to the target system via a simple power controller. To avoid the interface problems, the system clock frequency is restricted to discrete levels of f_{CLK}, $f_{CLK}/2$, $f_{CLK}/3$,... where f_{CLK} is the master (= maximum) clock frequency. The power controller can be implemented

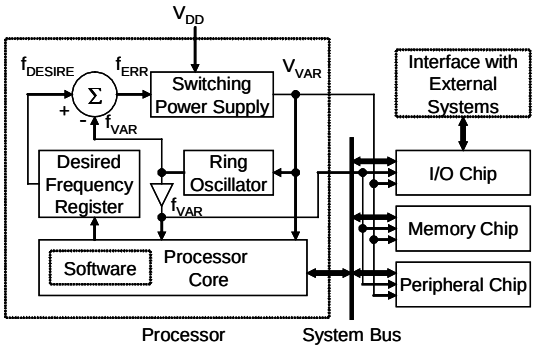

Fig. 1. Conventional DVS system architecture.

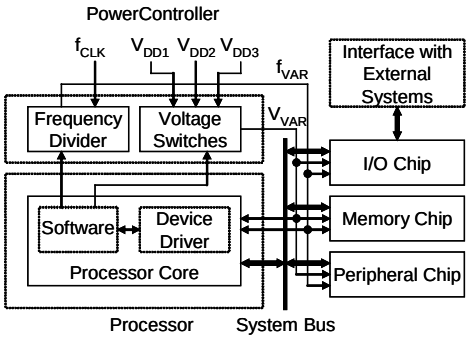

Fig. 2. System architecture of the proposed run-time V_{DD} hopping scheme.

as off-chip or on-chip. It consists of simple voltage switches and frequency dividers, which are much smaller than the switching power supply in the conventional DVS scheme. Note that the power controller needs neither external inductors nor capacitors in the output filter of switching power supply. The device driver has two lookup tables: one for the voltage-frequency relationship of the target processor, and the other for the transition delay to change the clock frequency and the supply voltage. These lookup tables are established by measuring the physical characteristics of the chips. In the proposed system architecture, the supply voltage and the clock frequency may be unstable during the transition delay T_{TD}, which may cause invalid operations. To avoid this problem, the target system stops running in this interval. Note that the transition of the supply voltage and the clock frequency is very fast (~ several tens of microseconds), because only analog voltage switches and frequency dividers are used. Therefore, T_{TD} is negligible for most cases.

The hardware operation of the proposed system architecture is described as follows.

(i) A task is portioned into several segments called timeslots. For each timeslot, the desired clock frequency is determined based on the proposed voltage scheduling method. This step is to be explained in the next subsection.

(ii) The desired supply voltage and its transition delay are looked up from the device driver.

(iii) The target processor sets these values into the power controller by sending control codes. After that, the target processor stops running, and waits until the clock frequency and the supply voltage settle down to steady state.

(iv) After the transition of the clock frequency and the supply voltage, the target processor restarts running.

2.2. Voltage scheduling

In the DVS, the supply voltage is reduced to the lowest possible extent while meeting the deadline constraints. This is achieved by exploiting slack times in real-time systems. These slack times can be classified into two categories, i.e. *worst-case slack time* and *workload-variation slack time* as explained in the introduction.

Consider two periodic tasks $\tau_1(20,20,10)$ and $\tau_2(30,30,10)$ 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. Feasible schedule of a task set can be found only when sum of each task's WCET is equal or less than total time assigned to the given task set. The worst-case slack time comes from the fact that there may be some idle time to wait for the arrival of next task even if all tasks run at their WCETs. In Figure 3(a1), the target processor

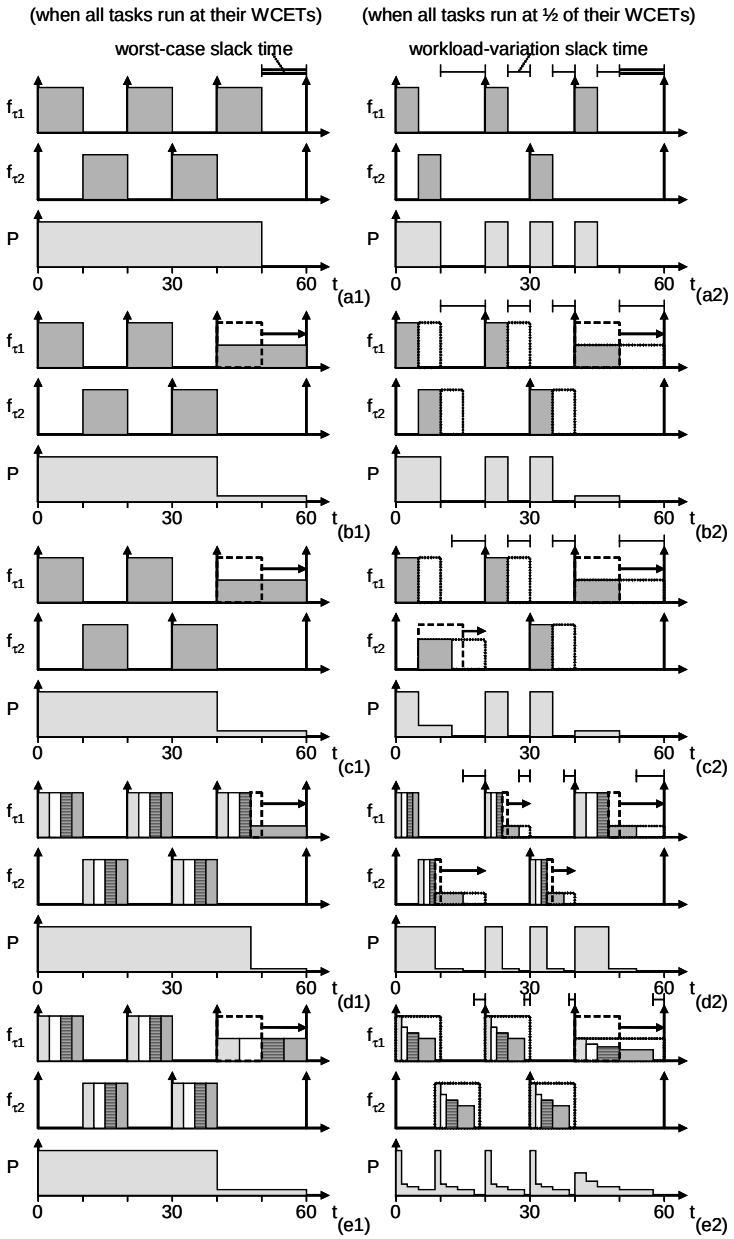

Fig. 3. Slack times and voltage scheduling methods. (a) Power-down only, (b) conventional voltage scheduling[6-8], (c) LPFPS[9], (d) LPFPS[9] where a task is partitioned into 4 timeslots, and (e) voltage scheduling of the proposed run-time V_{DD} hopping scheme, where $f_{\tau 1}$, $f_{\tau 2}$, and P denote clock frequency of τ_1 and τ_2, and power consumption of the target processor, respectively.

finishes both τ_1 and τ_2 at t=50 and waits next task to arrive until t=60. The workload-variation slack time comes from the fact that some tasks finish their execution before WCETs. In Figure 3(a2), the target processor finishes both τ_1 and τ_2 at t=10 and waits next task to arrive until t=20. Note that there is no idle time between t=10 and t=20 in Figure 3(a1). Main difference between the worst-case and the workload-variation slack times is that the former is known at compile time, while the latter is not. Therefore, the former can be easily exploited using simple algorithms, while it is not so easy for the latter.

As shown in Figure 3(b), most of the conventional voltage scheduling methods[6-8] exploits only worst-case slack time, because they assume that all tasks always run at their WCETs. This problem is partly overcome by the low-power fixed-priority scheduling[9] (LPFPS), as shown in Figure 3(c). At the beginning of each task's execution, the accumulated slack time from previous tasks is utilized by current task to lower the clock frequency and its corresponding supply voltage. In Figure 3(c2), there is a workload-variation slack time of τ_1 between t=5 and t=10. At the beginning of τ_2, it exploits this slack time and lowers the clock frequency to 2/3. Nevertheless, LPFPS cannot fully utilize slack time, because it performs the voltage scheduling on task-by-task basis, i.e. the supply voltage cannot be dynamically controlled inside a task. In Figure 3(c2), τ_2 has a slack time between t=12.5 to t=20. This slack time cannot be utilized since next task arrives at t=20.

When the clock frequency is restricted to discrete levels as explained in the previous subsection, there are some cases where conventional voltage scheduling schemes cannot exploit even worst-case slack time. Consider two periodic tasks τ_3(60,60,20) and τ_4(60,60,25) in Figure 4. If continuous clock frequency levels are used, the conventional voltage scheduling schemes can utilize the worst-case slack time, as shown in Figure 4(a1). However, they fail to utilize it when the clock frequency is restricted to only f and f/2, because the desired clock frequency is f/1.6, which is larger than f/2.

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

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

even degraded in some cases.

(ii) Partitioning tasks into timeslots results in tremendous increases of task scheduling overhead. When the number of tasks increases *N* times, the number of context switching increases *N* times. Most schedulers employ

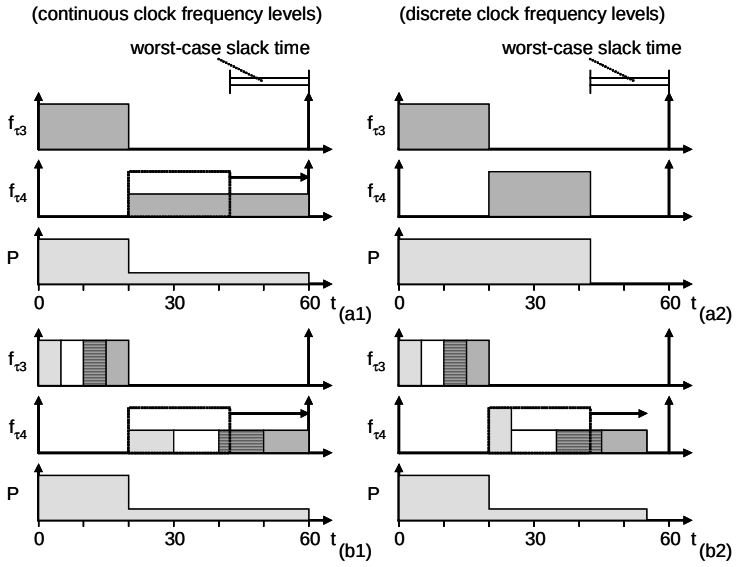

Fig. 4. Clock frequency levels and voltage scheduling methods. (a) Conventional voltage scheduling[6-9] and (b) voltage scheduling of the proposed run-time V_{DD} hopping scheme, where $f_{\tau 3}$, $f_{\tau 4}$, and P denote clock frequency of τ_3 and τ_4, and power consumption of the target processor, respectively.

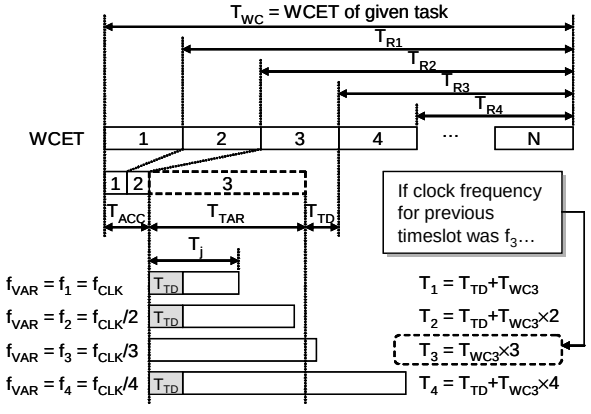

Fig. 5. Determination of clock frequency in the proposed run-time V_{DD} hopping scheme.

sorted queues whose computational complexities are O(Nlog$N$). Therefore, total scheduler overhead increases with N^2log$N$ times. When each task is partitioned into 100 timeslots, the scheduler overhead increases approximately 20000 times.

These problems can be solved if supply voltage is controlled on timeslot-by-timeslot basis such that each task finishes its execution within its WCET. One simple solution is the embedded voltage scheduling, where the supply voltage control is embedded in the application program of each task. In this case, a task scheduler in the OS has no voltage scheduling functionality, meaning that this approach can be applied to most of conventional non-DVS OS.

Voltage scheduling of the proposed run-time V_{DD} hopping scheme is summarized as follows. Step (i) is performed at compile-time, while (ii)-(iv) are performed at run-time.

(i) A task is divided into N timeslots at compile time. Following parameters are obtained and stored in the application program codes, through static analysis[14] or direct measurement.
- T_{WC}, T_{WCi}: WCET of a given task and that of ith timeslot, respectively
- T_{Ri}: WCET from (i+1)th to N^{th} timeslots

(ii) For each timeslot, the target execution time T_{TARk} is calculated as $T_{TAR} = T_{WC} - T_{Rk} - T_{ACC} - T_{TD}$, where T_{ACC} is the accumulated execution time from 1st to (i-1)th timeslots, and T_{TD} is the transition delay to change the clock frequency and the supply voltage.

(iii) For each candidate clock frequency f_j, = f_{CLK}/j (j=1,2,3...), the estimated maximum execution time T_j is calculated as $T_j = T_{wi}$×j., where f_{CLK} is the master clock frequency. If f_j is not equal to the clock frequency of (i-1)th timeslot, $T_j = T_j + T_{TD}$.

(iv) The clock frequency f_{VAR} is determined as the minimum clock frequency f_j whose estimated maximum execution time T_j does not exceed the target time T_{TAR}, as shown in Figure 5. The supply voltage V_{VAR} is determined from lookup table.

As shown in Figure 3(e) and Figure 4(b), this approach achieves a significant improvement of the power reduction over the conventional voltage scheduling scheme. 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}×f_{CLK}/f_{VAR} (ith timeslot) + T_{TD} (transition delay from f_{VAR} to f_{CLK}) + T_{Ri} (from (i+1)th to N^{th} timeslot), which does not exceed T_{WC} (WCET of given task).

3. Experimental Results of Run-time V_{DD} Hopping Scheme

In this paper, four real-time applications were tested, i.e. MPEG-4[11] (motion picture expert group) SP@L1 video encoding, MPEG-4 SP@L1 video decoding, RCR-STD27[15] VSELP (vector-sum-excited linear prediction) speech encoding, and RCR-STD27 VSELP speech decoding. These applications are briefly described in Table 1.

Table 1. Description of the target applications in the experiment.

Parameters		MPEG-4 video encoding	MPEG-4 video decoding	VSELP speech encoding	VSELP speech decoding
General	Frame rate (frames/s)	15	15	25	25
	Frame size (pels, samples)	176×144	176×144	320	320
Single-task	Worst-case execution time (ms)	66.667	66.667	40.000	40.000
	Average execution time (ms)	17.332	9.905	19.672	19.672
Multi-task	Task priority	4(lowest)	3	2	1(highest)
	Period (ms)	66.667	66.667	40.000	40.000
	Deadline (ms)	66.667	66.667	40.000	40.000
	Worst-case execution time (ms)	50.386	9.826	1.844	1.383
	Average execution time (ms)	13.099	1.460	0.907	0.680
	Number of timeslots	33	33	40	40

* Deadline, period, and execution times were normalized based on a virtual processor.

We measured the execution time and WCET of each frame and each timeslot by running the target application programs on a Pentium II processor platform. Based on this timing information, we assumed a virtual processor that completes the worst-case executions of these applications just at their deadlines. Conceptually, it is the slowest (or lowest throughput) processor that can perform the worst-case execution of the target applications. Based on this virtual processor, all deadline, period, and execution times were normalized and voltage scheduling was emulated. The reason we assumed a virtual processor is that the power efficiency is strongly related with the speed of target processor. Increasing the target processor's speed is equivalent to decreasing WCET, which results in better power efficiency. This virtual processor quite makes sense, because it is practically the cheapest processor meeting the computational requirement.

To evaluate the proposed run-time V_{DD} hopping (VH) scheme, two different simulations were performed. In the single-task simulation, each application was programmed as a single-task whose period and deadline are equal to its WCET. In the multi-task simulation, four target applications were merged into a single mobile videophone application with four tasks. The priority-based scheduling[16] was used for real-time task scheduling, where a higher priority is assigned to a

task with shorter period.

For both simulations, the voltage-frequency relationship was obtained from the alpha-power delay model[17], i.e. $f^{-1} \propto V/(V-V_{TH})^{\alpha}$. The master supply voltage V_{DD}, the threshold voltage V_{TH}, and the velocity saturation index α are assumed to be 2.5V, 0.5V, and 1.3, respectively. The overheads of embedded voltage scheduling were measured to be less than 0.01% of the total workload, which is negligible. The number of timeslots N is determined as 33 by the simulation. According to the simulation results, the power efficiency is turned out to be quite insensitive for a wide range of V_{DD}, V_{TH}, α, and N.

Figures 6 and 7 show the power consumption in the single-task and the multi-

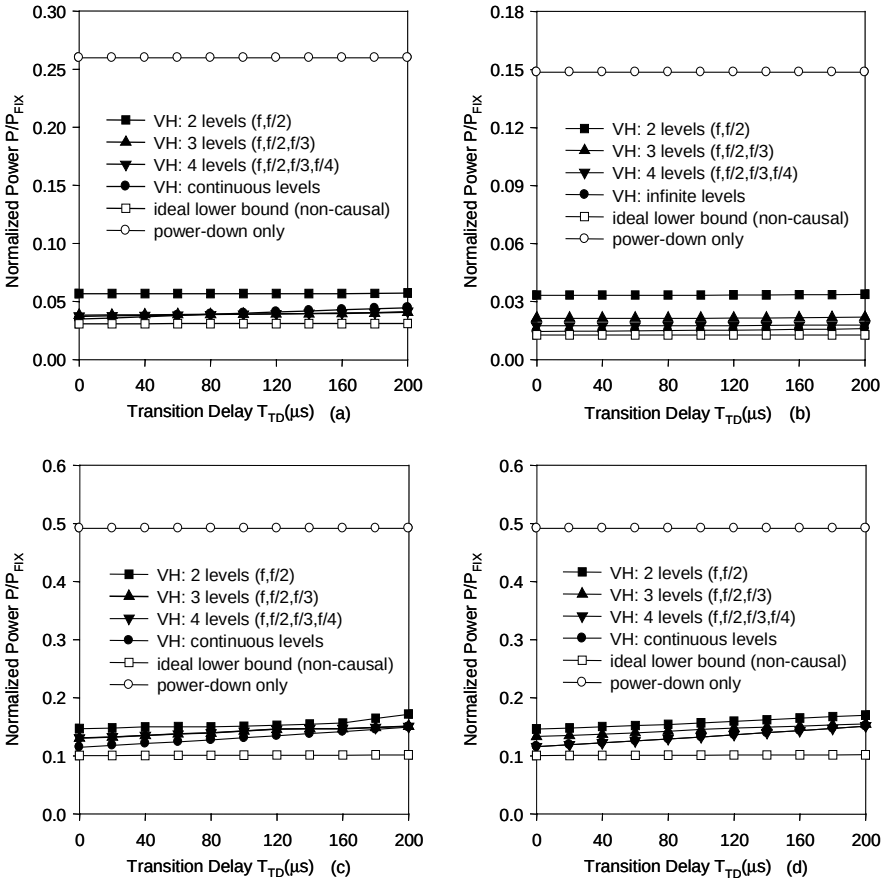

Fig. 6. Normalized power consumption in single-task simulation.
(a) MPEG-4 encoding (b) MPEG-4 decoding (c) VSELP encoding and (d) VSELP decoding

task simulations, respectively. The power consumption is normalized by P_{FIX}, which is the power consumption of a fixed voltage processor without power-down modes. Lower bound of the power consumption[8] is also calculated using post-simulation analysis. This lower bound cannot be achieved in real-time systems, because it determines supply voltage of a given task from its actual execution time, i.e. it is non-causal process.

From Figures 6 and 7, it is seen that the power consumption of the proposed voltage scheduling method is about 2~18% of non-DVS processors without the power-down modes, while those of the power-down approaches and the lower bound are 15~49% and 1.5~10%, respectively. In the multi-task simulation, both

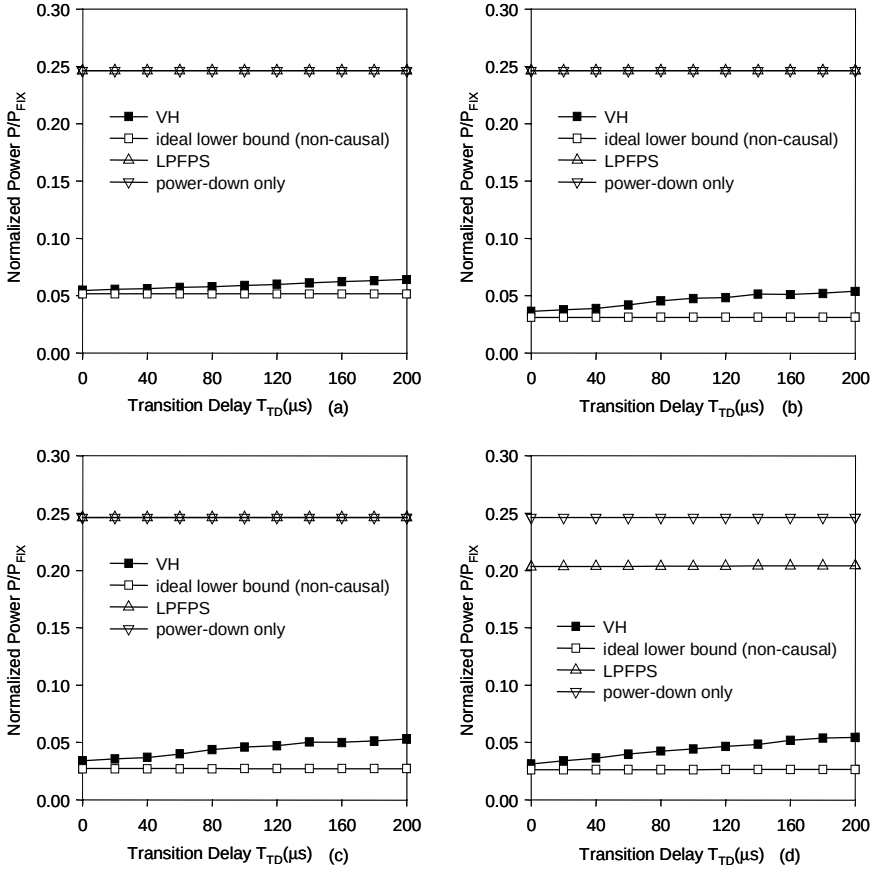

Fig. 7. Normalized power consumption in multi-task simulation.
(a) 2 clock levels (b) 3 clock levels (c) 4 clock levels and (d) continuous clock levels

LPFPS[9] and power-down method achieves same power efficiency of 25% for discrete clock frequency levels, meaning that there is no supply voltage control in LPFPS. This is due to the fact that LPFPS sometimes fails to exploit even the worst-case slack time, as explained in Figure 6. On the contrary, LPFPS shows better performance for continuous clock frequency levels.

From the simulation results, only two (= *f*, *f*/2) or three (=*f*, *f*/2, *f*/3) discrete levels of clock frequency are sufficient, meaning that the proposed scheme is extremely simple in both hardware and software. Note that the power efficiency degrades as the transition delay of the power controller increases, because the target processor stops its execution during the transition delay. However, this degradation is not serious when the transition delay lies within the practical range. Figure 8 shows the transient curves of the normalized power and the clock frequency in the 194th frame in MPEG-4 SP@L1 video encoding when the transition delay is 100 μs. The proposed scheme keeps track of the ideal supply voltage quite well, with limited number of clock frequency levels.

4. Energy-Constrained V_{DD} Hopping Scheme

4.1. Motivation

As explained in the introduction, trade-off between the performance and the computation is useful for efficient power reduction when the application has several alternative algorithms with different performance and computation. For example, MPEG-4[11] video encoding has several motion estimation algorithms

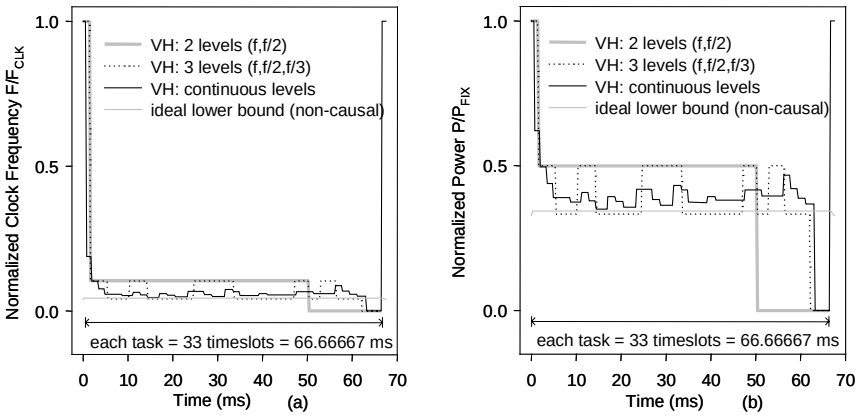

Fig. 8. Transient characteristics of the 194th frame in MPEG-4 SP@L1 video encoding.
(a) Power and (b) clock frequency

such as three-step search[12] (TSS), multi-candidate three-step search[13] with two candidates (MCTSS2) and three candidates (MCTSS3). The characteristics of these algorithms are summarized in Table 2. When TSS is used, the peak signal-to-noise ratio (PSNR) performance is degraded by 0.51dB with respect to MCTSS3, while the power consumption is reduced to 1/5. On the contrary, MCTSS3 improves PSNR performance while sacrificing the power consumption.

In general, how to select proper algorithm is closely related to the application purpose of the system. In the telemedicine for doctorless islands, for example, high performance should be achieved no matter how much power is consumed, and MCTSS3 is always selected. On the contrary, long battery lifetime is indispensable in the military communication, and TSS is always selected. Then, how about private videophone calls in the daily life? In most cases, both the performance and the power are considered. The user determines a certain threshold of performance (or power), and tries to adjust power (or performance) to roughly keep performance above (or power below) the threshold. This is the basic idea of the proposed energy-constrained V_{DD} hopping (ECVH) scheme.

In this paper, the energy constraint is exploited instead of the performance from the following reasons. (1) The power consumption is, in general, much easier to calculate and estimate than the performance measure. (2) It is advantageous to control total power consumption and its corresponding battery operation time to meet various user request, e.g. 60min, 120min, or 240min operation modes. The proposed scheme performs simple run-time power estimation of the application program, and selects proper algorithm to achieve optimum performance under given energy constraint.

Table 2. PSNR performance, normalized worst-case execution time, and normalized power consumption of various motion estimation algorithms in MPEG-4 video encoding.

Motion estimation algorithm	TSS	MCTSS2	MCTSS3
PSNR performance (dB)	31.51	31.79	31.87
Normalized worst-case execution time	0.59	0.79	1.00
Normalized power consumption	0.20	0.46	1.00

* Worst-case execution time and power consumption is normalized with MCTSS3.

4.2. Algorithm

Suppose the following assumptions. (1) A given task has M alternative algorithms denoting ALG_i (i=1...M) whose computation and performance increases in ascending order of i. (2) The worst-case execution time (WCET) and the worst-case power consumption (WCPC) occur when the processor runs at master clock frequency f_{CLK} and master supply voltage V_{DD}. (3) A given task is divided into N

timeslots. (4) WCET of a given task and that of kth timeslot with ALG_M are T_{WC} and T_{WCk}, respectively. (5) WCPC of a given task and that of kth timeslot with ALG_M are P_{WC} and P_{WCk}, respectively. (6) WCET of kth timeslot with ALG_i is $T_{ALGi} = T_{WCk} \times \lambda_{ALGi}$, where λ_{ALGi} ($0<\lambda_{ALGi} \leq 1$) is the normalized computational complexity of ALG_i. (7) The period T_P and the deadline T_D of a given task equals its WCET T_{WC}. Assumptions (1)-(6) are valid for most real-time applications. Assumption (7) comes from the fact that all task scheduling with non-equal T_P, T_D, and T_{WC} can be converted into task scheduling with equal T_P, T_D, and T_{WC}[16].

Our aim is to select most suitable algorithm ALG_{OPT} adaptively for each kth timeslot to achieve optimal performance under real-time constraint $T_{CON} = T_{WC}$ and energy constraint $P_{CON} = \rho \times P_{WC}$, where ρ ($0<\rho \leq 1$) is the normalized energy budget. Note that only energy constraint P_{CON} is adjustable by the user, since the real-time constraint T_{CON} ($=T_{WC}$) equals the deadline T_D and it should be strictly guaranteed for real-time operation. Therefore, the time budget T_{SLk} and power budget P_{SLk} for kth timeslot are $T_{SLK} = T_{WCk}$ and $P_{SLk} = \rho \times P_{WCk}$, respectively.

As illustrated in Figure 9, how to select proper algorithm ALG_{OPT} is very similar with the voltage scaling procedure of the run-time V_{DD} hopping scheme, except for power constraint. It consists of three steps as follows. Step (i) is performed at compile-time, while (ii)-(iii) are performed at run-time.

(i) *Initialization*: The normalized energy budget ρ is set to appropriate value based on the application purpose. A task is divided into N timeslots at compile time. For kth timeslot and ith algorithm, T_{WC}, T_{WCk}, P_{WC}, P_{WCk} and λ_{ALGi} are obtained and stored in the application program codes, through static analysis[14] or direct measurement. Note that, from assumption (2), the ratio of WCET and WCPC is same for each timeslot, i.e. $T_{WC}/P_{WC} = T_{WCk}/P_{WCk}$ for kth timeslot. It means that P_{WC} and P_{WCk} are directly calculated from T_{WC} and T_{WCk}.

(ii) *Determination of candidate algorithms using real-time constraint*: The execution time to process kth timeslot should not exceed the target execution time $T_{TAR} = \sum_{j=1}^{k} T_{SLj} - T_{ACC} - T_{TD}$, where T_{ACC} is the accumulated execution time from 1st to (k-1)th timeslots, and T_{TD} is the transition delay to change the clock frequency and the supply voltage. It means that ALG_i can be applied to k^{th} timeslot only if T_{ALGi} does not exceed T_{TAR}. For example, in Figure 9, only ALG_1, ALG_2, and ALG_3 can be applied to kth timeslot considering the real-time constraint, and they are determined as candidate algorithms for ALG_{OPT}. When these candidate algorithms ALG_i are determined, their clock frequencies f_{VARi} and supply voltages V_{VARi} are determined as ordinary run-time V_{DD} hopping scheme in Figure 5. After V_{VARi} is determined, WCPC of kth timeslot with ALG_i at V_{VARi} is calculated as P_{ALGi} =

$P_{WCk} \times \lambda_{ALGi} \times (V_{VARi}/V_{DD})^2$, since the ratio of the power consumption to process k^{th} timeslot at (f_{VARi}, V_{VARi}) and (f_{CLK}, V_{DD}) is $P_{ALGi}/P_{WCk} = (V_{VARi}^2 \times T_{ALGi})/(V_{DD}^2 \times T_{WCk}) = \lambda_{ALGi} \times (V_{VARi}/V_{DD})^2$.

(iii) *Determination of optimum algorithm using energy constraint*: In a similar manner, the power consumption to process k^{th} timeslot should not exceed the target power consumption $P_{TAR} = \sum_{j=1}^{k} P_{SLj} - P_{ACC}$, where P_{ACC} is the accumulated power consumption from 1^{st} to $(k-1)^{th}$ timeslots. It means that ALG_i can be applied to k^{th} timeslot only if P_{ALGi} does not exceed P_{TAR}. Therefore, ALG_{OPT} is determined as ALG_i with the maximum performance while meeting above energy constraint. For example, in Figure 9, only ALG_1 and ALG_2 can be applied to k^{th} timeslot considering both real-time constraint and energy constraint, and ALG_2 is determined as ALG_{OPT}.

The proposed scheme provides three user-selectable modes. In the minimum power mode, ALG_{OPT} is selected as the least-energy algorithm under real-time

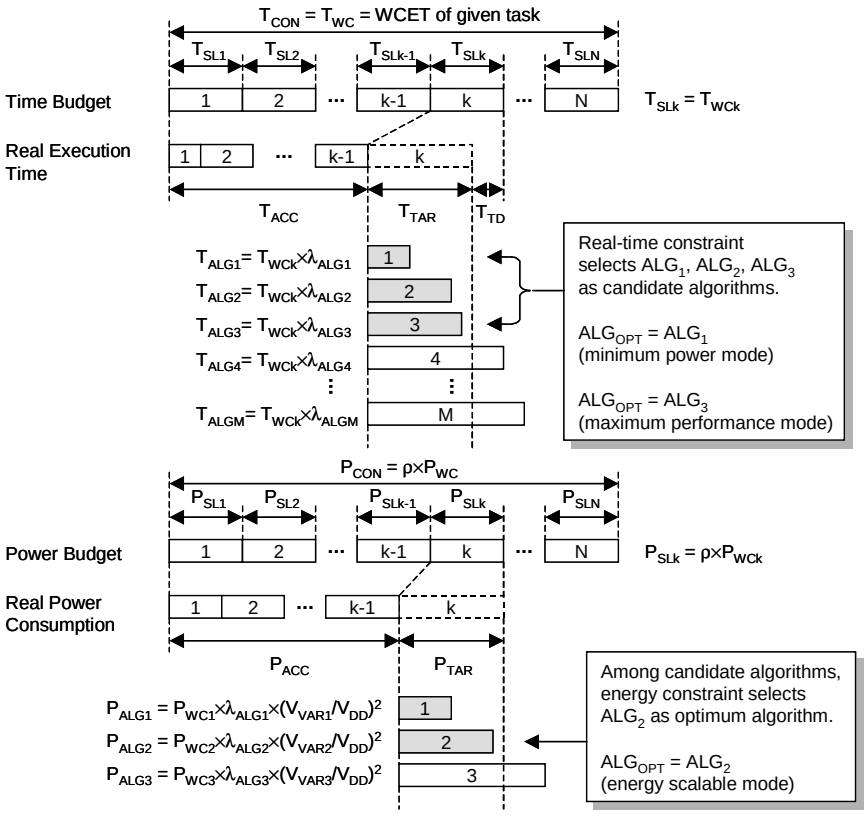

Fig. 9. Algorithm selection in the proposed energy-constrained V_{DD} hopping scheme.

constraint only, i.e. ALG_i with lowest i whose $T_{ALGi} \le T_{TAR}$. Similarly, in the maximum performance mode, ALG_{OPT} is selected as the best-performance algorithm under real-time constraint only, i.e. ALG_i with highest i whose $T_{ALGi} \le T_{TAR}$. On the contrary, in the energy scalable mode, ALG_{OPT} is selected as the best-performance algorithm under both real-time and energy constraints, i.e. ALG_i with highest i whose $T_{ALGi} \le T_{TAR}$ and $P_{ALGi} \le P_{TAR}$. Note that the minimum power mode and the maximum performance mode correspond to the energy scalable mode with $\rho = 0$ and ∞, respectively.

4.3. Experimental results

In this paper, MPEG-4 SP@L2 video encoding was tested with three alternative algorithms, i.e. three-step search[12] (TSS) as ALG_1, multi-candidate three-step search[13] with two candidates (MCTSS2) and three candidates (MCTSS3) as ALG_2 and ALG_3, respectively. The characteristics of these alternative algorithms are shown in Table 2. "Foreman" and "Carphone" were used as test image sequences. As explained in Section 3, we measured the timing information on a Pentium II processor platform and assumed a virtual processor with the alpha-power delay model[17]. The master supply voltage V_{DD}, the threshold voltage V_{TH}, the velocity saturation index α, and the number of timeslots N are 2.5V, 0.5V, 1.3, and 99, respectively. Overheads of OS, embedded voltage scheduling, and optimal algorithm selection were measured to be less than 0.01% of total workload, which is negligible.

Figure 10 shows the power consumption and the PSNR performance of the proposed energy-constrained V_{DD} hopping (ECVH) scheme. The power

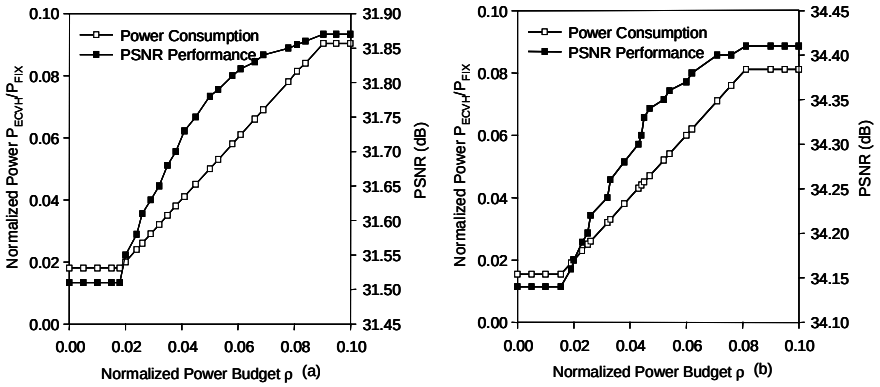

Fig. 10. Normalized power consumption of the proposed energy-constrained V_{DD} hopping scheme. (a) "Foreman" sequence and (b) "Carphone" sequence

consumption is normalized by P_{FIX}, which is the power consumption of a fixed voltage processor without power-down modes. In the "Foreman" sequence, the normalized power consumption on the variable-voltage processor is 0.018~0.091, while those of conventional TSS, MCTSS2, and MCTSS3 algorithms on the fixed-voltage processor are 0.259~0.506. As a result, the proposed scheme reduces the power consumption to 1/14.4~1/5.6 while maintaining same PSNR performance. Note that it runs at the minimum power mode when $\rho \leq 0.018$ and the maximum performance mode when $\rho \geq 0.091$. It is because ALG_1 (=TSS) is always selected when ρ is too small, and ALG_3 (=MCTSS3) is always selected when ρ is too large. We got similar results in the "Carphone" sequence, where the power consumption was reduced to 1/15.7~1/5.9 while maintaining same PSNR performance.

5. Conclusion

In this paper, we propose a novel DVS scheme called run-time V_{DD} hopping. It can be easily applied to various targets including off-the-shelf processors, by employing software control of the supply voltage and the device driver from physical measurement of the voltage-frequency relationship. It employs voltage switches and frequency divider instead of switching power supplies in the conventional schemes, which results in smaller and simpler hardware. It restricts the clock frequency to discrete levels of f_{CLK}, $f_{CLK}/2$, $f_{CLK}/3$, ... to avoid the interface problems due to different clock frequencies of target system and external systems. It fully utilizes slack time arising from workload variation by partitioning a task into several timeslots and performing supply voltage control on timeslot-by-timeslot basis. It is applicable to conventional non-DVS OS since voltage control routine is embedded in the target applications. When applied to real-time multi-media applications, the proposed scheme is shown to achieve 82~98% power reduction compared to non-DVS processors without power-down modes, and 70~84% power reduction compared to conventional DVS scheme.

Based on the run-time V_{DD} hopping scheme, we also propose an energy-constrained V_{DD} hopping scheme, exploiting trade-off between the performance and the power. It adaptively selects optimum algorithm under given time and energy constraints when the application has several alternative algorithms with different performance and power. To cover various applications and trade-off models, it provides three user-selectable modes, i.e. the energy-scalable mode, the minimum power mode, and the maximum performance mode. In the energy-scalable mode, it provides energy-performance scalability, i.e. performance and

power is roughly proportional to the energy constraint. In the minimum power mode and the maximum performance modes, it achieves largest power reduction and highest performance improvement under given real-time constraint. When applied to the real-time multimedia applications, it achieves 82~97% power reduction while maintaining same performance, compared to non-DVS processors without power-down modes.

Acknowledgements

The authors would like to thank IC Design Education Center for supporting simulation environments.

References

1. J. Rabaey, "Low-Power Silicon Architectures for Wireless," *Proceedings of Asia and South Pacific Design Automation Conference* (2000) 379-380.
2. A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid State Circuits*, **27**, 4 (1992) 473-484.
3. A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.
4. T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *Proceedings of IEEE International Solid-State Circuits Conference* (2000) 294-295.
5. V. Gutnik and A. Chandrakasan, "An efficient controller for variable supply-voltage low power processing," *Proceedings of IEEE Symposium on VLSI Circuits* (1996) 158-159.
6. F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," *Proceedings of IEEE Annual Foundations of Computer Science* (1995) 374-382.
7. I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **18**, 12 (1999) 1702-1714.
8. T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," *Proceedings of IEEE International Symposium on Low Power Electronics and Design* (1998) 197-202.
9. Y. Shin and K, Choi, "Power conscious fixed priority scheduling for hard real-time systems," *Proceedings of Design Automation Conference* (1999) 134-139.
10. S. Lee and T. Sakurai, "Run-time voltage hopping for low-power real-time systems," *Proceedings of Design Automation Conference* (2000) 806-809.
11. ISO/IEC JTC1/SC29/WG11 13818-1, "Coding of moving pictures and associated audio," 1994.

12. T. Koga, K. Ilnuma, K. Hirano, Y. Iijima, and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” *Proceedings on National Telecommunication Conference* (1981) G.5.3.1-5.3.5.
13. H. Jong, L. Chen, and T. Chiueh, “Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, **4**, 1 (1994) 88-90.
14. 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 procecssors,” *Proceedings of IEEE Real-time Systems Symposium* (1994) 97-108.
15. Research and Development Center for Radio System, RCR-STD27, 1991.
16. C. Liu and J. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *Journal of ACM*, **20**, 1 (1973) 46-61.
17. 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*, **25**, 2 (1990) 584-594.