Software Energy Reduction Techniques for Variable-Voltage Processors

Takanori Okuma

Hiroto Yasuura
Kyushu University

Tohru Ishihara
University of Tokyo

A processor consumes far less energy running tasks requiring a low supply voltage than it does executing high-performance tasks. Effective voltage-scheduling techniques take advantage of this situation by using software to dynamically vary supply voltages, thereby minimizing energy consumption and accommodating timing constraints.

ADVANCES IN deep-submicron technologies have enabled system-on-chip (SOC) designs in which a system's entire functionality rests on a single chip. SOCs are embedded in various electric products, such as portable information terminals, digital audio systems, and automobiles. Many of these products are real-time systems with timing constraints. An important consideration in SOC design is minimizing power consumption. Heat due to high power consumption often prevents realization of high-performance SOCs with high transistor density. Moreover, portable systems require a small battery. Thus, design technology for high-performance SOCs with low energy consumption is an important research issue in real-time system design.

The problem is realizing both high-speed computation and low energy consumption. Employing a high-performance processor core may satisfy timing constraints, but will probably not foster low energy consumption. Ishihara et al. have proposed a variable-voltage processor that can vary its supply voltage dynamically to solve this problem.[1] Using the variable-voltage processor, tasks with severe real-time constraints can execute at high supply voltages—and, therefore, high execution speeds—and tasks with loose time constraints can execute at low supply voltages. Reducing the supply voltage leads to drastic energy reduction because energy consumption in CMOS circuits typically increases quadratically with supply voltage. Energy consumption integrates power consumption in the time domain. The energy consumption per clock cycle for a task is

$$E_{\text{cycle}} = \sum_{k=1}^{M} LC_k \cdot SW_k \cdot V_{\text{DD}}^2 \qquad (1)$$

where M is the number of gates in the circuit, LC_k is the load capacitance of gate g_k, SW_k is the switching count of g_k per clock cycle for the task, and V_{DD} is the supply voltage. Now, consider a task with total number of execution cycles CY_{task}. The energy consumption for this task is

$$E_{\text{task}} = CY_{\text{task}} \cdot \sum_{k=1}^{M} LC_k \cdot SW_k \cdot V_{\text{DD}}^2 \qquad (2)$$

We can reduce the energy consumption for the task by lowering V_{DD}. However, this step

0740-7475/01/$10.00 © 2001 IEEE

increases the execution time. The circuit delay τ is

$$\tau \propto \frac{V_{DD}}{(V_G - V_T)^{\alpha}} \sim \frac{1}{V_{DD}} \quad (1 < \alpha \leq 2) \qquad (3)$$

and execution time T_{task} is

$$T_{task} = \tau \cdot CY_{task} \qquad (4)$$

where V_T is the threshold voltage, and V_G ($\sim V_{DD}$) is the input-gate voltage. The α factor depends on the carrier velocity saturation and in advanced MOSFETs is about 1.3.

Using the variable-voltage processor, the application programs or operating systems control the supply voltage. In real-time systems, effective compiler and operating system techniques can manage this function in a way that minimizes energy.[2] If a system includes only one task, finding the optimal supply voltage is easy.[1] However, in a multiple-task system, determining the optimal supply voltage assignment that minimizes total energy consumption without violating real-time constraints is not so simple.

Here, we propose voltage scheduling techniques for real-time applications using a variable-voltage processor. Our proposed techniques always guarantee satisfaction of a system's real-time constraints if the system meets those constraints under the highest supply voltage.

Dynamically controlled variable-voltage processors

A variable-voltage processor employs special instructions for controlling supply voltage. These instructions can change the supply voltage at any clock cycle. The CPU clock is adjusted to the frequency suitable for the present supply voltage. Software can dynamically control the supply voltage and the clock frequency. Therefore, both application programs and operating systems can manage the processor's power consumption.

For variable voltage systems, the time and power overhead to change the supply voltage can be important. However, we will ignore this issue here because Lee and Sakurai have already discussed it,[3] and extending our problem to consider these overheads would not be difficult. Instead, we will concentrate on methods for variable voltage scheduling.

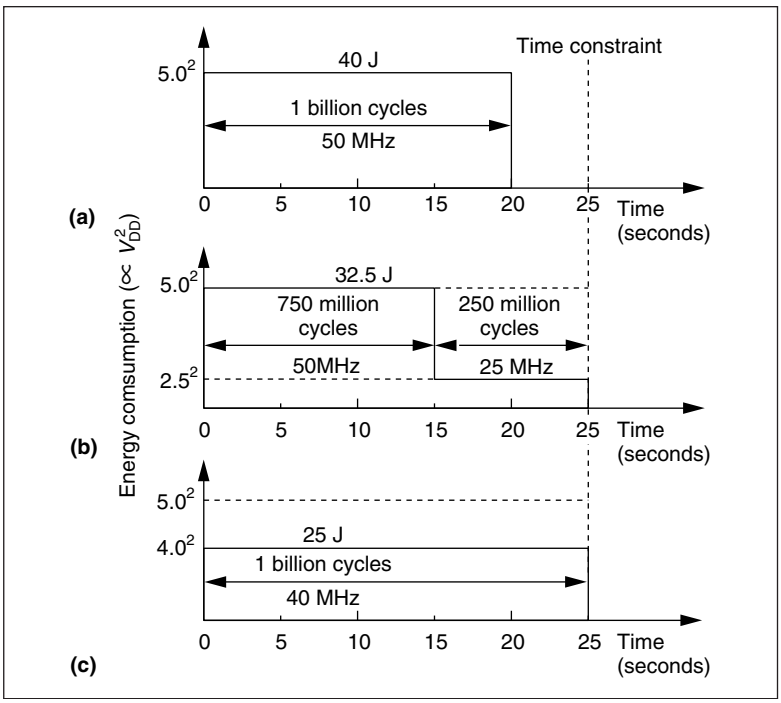

Figure 1. Power-delay optimization for a program with 1 billion execution cycles when (a) the processor uses only 5.0 V. Compare this with (b) voltage scheduling with 2.5 V and 5.0 V, and (c) when the processor uses a single supply voltage that fits the execution time with the given time constraint.

Motivational examples

We illustrate our basic idea with a simple example. Assume a given program's energy consumptions are 10 nJ/cycle at 2.5 V, 25 nJ/cycle at 4.0 V, and 40 nJ/cycle at 5.0 V. The processor's corresponding computational speeds are 25×10^6, 40×10^6, and 50×10^6 clock cycles/second. This assumption roughly accords with Equations 2 and 3. Figure 1 shows three voltage assignments for the given program, which has 1 billion execution cycles. In Figure 1a, the total energy consumption is 40 J, because the processor uses only a 5.0-V supply voltage. Given a time constraint of 25 seconds, voltage scheduling with 2.5 V and 5.0 V reduces the energy consumption from 40 J to 32.5 J, as shown in Figure 1b. Figure 1c shows the lower-bound case of this example. A processor using a single supply voltage that fits the execution time only with the given time constraint minimizes total energy consumption.

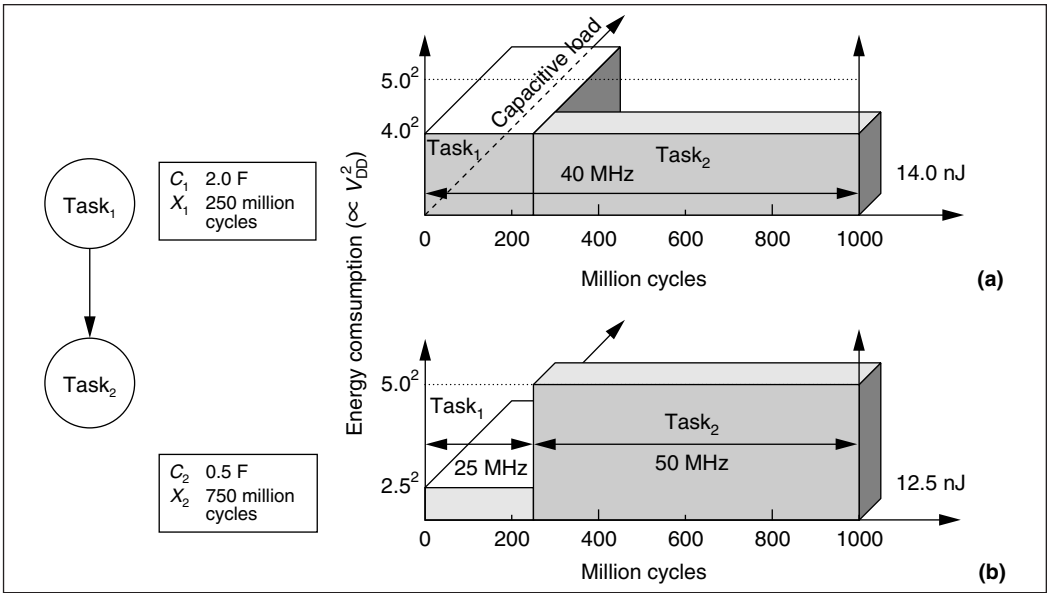

Figure 2. Power-delay optimization, considering the capacitive loads: voltage scheduling (a) with a single voltage of 4.0 V; and (b) with two voltages, 2.5 V and 5.0 V.

Up to this point, we have not discussed load capacitances, which are charged and discharged by the tasks. Capacitances for an addition operation, for example, are quite different from the capacitive load for a multiply operation. Moreover, because of load capacitance, processing the program with a single voltage that adjusts the execution time to the timing deadline does not always minimize energy consumption. The average capacitive load per cycle of task$_j$ is

$$C_j = \frac{\sum_{i=1}^{X_j} \sum_{k=1}^{M} LC_k \cdot SW_{kij}}{X_j} \quad (5)$$

where the jth task (task$_j$) is $\{X_j, C_j\}$, X_j is the number of execution cycles of the jth task ($1 \le j \le N$), C_j is the average capacitive load for the jth task, M is the number of gates in the processor, LC_k is the load capacitance of a gate g_k, and SW_{kij} is the switching count of g_k while the ith cycle of task$_j$ executes.

If the C_js differ considerably from one another, voltage scheduling with multiple voltages may minimize the energy consumption. For a given task set {task$_1$, task$_2$}, the voltage scheduling with 2.5 V and 5.0 V, as shown in Figure 2b, reduces more energy than a voltage schedule with a single voltage (4.0 V), as shown in Figure 2a. For both voltage schedules, a processor completes the program's 1 billion cycles only at the timing constraint. The *x*-, *y*-, and *z*-axes of the graphs in Figure 2 indicate the execution cycles, the square of supply voltage, and the capacitive loads. The volume of cubes indicates the energy consumption for processing. Therefore, minimizing the total volume of cubes can help voltage scheduling.

Related work

Lee and Sakurai have proposed a runtime dynamic voltage-scaling scheme for low-power real-time systems.[3,4] This scheme employs a power control chip with an on-chip DC-to-DC converter and a frequency synthesizer, as well as an embedded runtime power control algorithm using the software feedback loop. The scheme 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.

Transmeta has announced the Crusoe processor and proposed LongRun technology: the first commercial variable-voltage processor.[5] This technology can save power by reducing clock speed and voltage when an application doesn't need peak processor performance. Using LongRun power management

technology, software continuously monitors processor demands, dynamically and smoothly adjusting the processor's speed to what is exactly needed to run the application.

Hong et al. describe a design methodology for a real-time system on a chip that uses a dynamically variable-voltage-processor core. This methodology provides an offline scheduling heuristic to handle nonpreemptive, hard real-time tasks and select the processor core. It also determines the configuration and size of the instruction and data caches.[6] Also, Hong and other colleagues have proposed an online preemptive scheduling algorithm for on- and offline tasks on a variable-voltage processor to optimize energy consumption while ensuring that all offline tasks meet their deadlines. They also designed the algorithm to accept the highest possible number of online tasks that can be guaranteed to meet their deadlines.[7]

Shin et al. proposed a power-efficient version of fixed-priority preemptive scheduling, which is widely used in hard real-time system design.[8] Their method reduces energy consumption in the processor by exploiting system-inherent slack times, as well as slack times arising from dynamic variations of execution times for the task.

Pering et al. presented an online scheduling algorithm for soft real-time systems.[9] This algorithm relaxes the deadline constraints and allows application frames to complete after their deadlines. The scheduler can then absorb the effects of high frame-to-frame application variance, which might otherwise increase energy.

Burd et al. have demonstrated dynamic voltage scaling on a complete embedded processor system.[10] This prototype system contains four custom chips in 0.6-µm three-metal CMOS: a battery-powered DC-to-DC voltage converter, a microprocessor (ARM8 core with 16-Kbyte cache), SRAM memory chips, and an interface chip for connecting to commercial I/O devices. The entire system can operate from 1.2 to 3.8 V and 580 MHz, and energy consumption varies from 0.54 to 5.6 mW/MIP.

Software techniques

Voltage scheduling for real-time applications is complex for several reasons:

- A real-time application consists of two or more tasks. In certain applications, precedence relations exist among tasks.
- A variable-voltage processor uses only a few discrete voltages (or frequencies) because preparing a lot of discrete voltages makes the test difficult. Therefore, the scheduler might not be able to assign an ideal voltage to a task.
- The load capacitance is different for each task. It depends on input data and does not remain constant during a task's execution. Because the gated clock scheme is popular, disregarding the change in load capacitance is not possible.
- Tasks typically end earlier than they would in worst-case execution cycles. However, the scheduler can't know the execution cycle of the next executed task before that task executes.
- The scheduler can't execute a task until it's ready for execution—the arrival time. When the scheduler doesn't know a task's arrival time, assigning the lower voltage to tasks for which the arrival time is known is dangerous.

Using a variable-voltage processor, we propose some voltage-scheduling techniques for real-time applications: Static voltage scheduling addresses the first three problems listed, and dynamic voltage scheduling addresses the last two.

Static voltage scheduling

If a processor employs only a few discrete variable voltages, a single voltage might not minimize energy consumption. However, supporting several different supply voltages can be costly, so any feasible voltage-scheduling technique must use only a few different supply voltages.

If the processor cannot supply a single ideal voltage, voltage scheduling with processor-supplied multiple voltages can minimize energy consumption. Consequently, voltage scheduling with a single voltage that adjusts the execution time to the timing deadline does not always minimize energy consumption. In a formulation of static voltage scheduling that supports instruction-level parallelism (ILP), we target a processor that employs just a few discretely variable voltages.

Target systems. In a simplified static voltage-scheduling problem, we target real-time, processor-based systems where the processor

- can vary its supply voltage dynamically and at any clock cycle,
- uses only one supply voltage at a time,
- employs only a few discrete voltages, and
- has an adaptive clock scheme that closely tracks the supply voltage.

In addition, the target system must be one in which

- time overhead for changing the supply voltage and clock frequency is negligible,
- power loss for the DC-to-DC level converter is negligible, and
- the given program's worst-case execution cycles can be estimated statically.

Notation. We define the variables used in the formulation as follows:

- N is the number of tasks: $N = |\{\text{task}_j\}|$.
- Task_j is the jth task: $\text{task}_j = (X_j, C_j)$.
- X_j is the number of execution cycles of the jth task, such that $1 \le j \le N$.
- C_j is the average capacitive load for the jth task.
- L is the number of variable voltages of the target processor: $L = |\{\text{mode}_i\}|$.
- Mode_i is the processor's ith execution mode: $\text{mode}_i = (V_i, F_i)$.
- V_i is the ith voltage, such that $1 \le i \le L$.
- F_i is the clock frequency when supply voltage is V_i, where Equation 3 calculates F_i.
- T is the time constraint during which all given tasks must be completed.
- x_{ij} is the number of cycles of task j executed with voltage V_i.

ILP formulation. We formulate the voltage-scheduling problem as follows:

Minimize

$$E = \sum_{j=1}^{N} \sum_{i=1}^{L} C_j \cdot x_{ij} \cdot V_i^2 \qquad (6)$$

subject to

$$\sum_{i=1}^{L} x_{ij} = X_j, \qquad \sum_{j=1}^{N} \sum_{i=1}^{L} \frac{x_{ij}}{F_i} \le T \qquad (7)$$
$$0 \le x_{ij} \le X_j$$

We formally define the voltage-scheduling problem as

> for the given $\{\text{task}_j\}$ and $\{\text{mode}_i\}$, find x_{ij} that minimizes E and satisfies time constraint T.

Both the objective function and the constraint are linear functions of variable x_{ij}. The computation time to solve the voltage-scheduling problem strongly depends on $N \times L$. When all C_js are the same value, the problem is far simpler. In this situation, obtaining an optimal solution requires solving only the problem of a single task ($N = 1$).

Dynamic voltage scheduling

These techniques extend voltage scheduling to tasks for which it is difficult to predict start or completion times.

Target systems. A real-time system generally includes both application programs and an operating system to execute those applications. Application programs are divided into several tasks to satisfy real-time constraints. A system designer must estimate each task's worst-case execution time. When external events are detected, the operating system schedules these tasks to satisfy the real-time constraints.

In our dynamic voltage-scheduling techniques, we assume a single-processor system, which uses a variable-voltage processor as a processor core. The variable-voltage processor can discretely change its supply voltage using special instructions for voltage control. Only the operating system—not the applications program—can use these instructions. Thus, switching tasks can vary the supply voltage.

Notation. We assume the voltage is assigned to each real-time task in task set $\{J_1, \ldots, J_n\}$. These parameters characterize a real-time task J_i:

- a_i is the arrival time,

- O_i is the worst-case execution time,
- d_i is the deadline time,
- s_i is the execution start time,
- e_i is the execution completion time, and
- L_i is the remaining time from completion time to deadline, where $L_i = d_i - e_i$

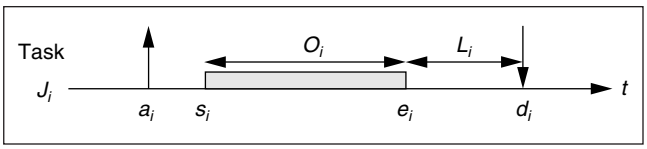

Figure 3. Parameters of a real-time task.

We also define the following parameters regarding the task's energy consumption:

- X_i represents the worst-case execution cycle.
- F_i is the clock frequency when J_i executes.
- O_i, X_i, and F_i have the relation $O_i = X_i/F_i$.
- V_i is the supply voltage when J_i executes.
- C_i is the average capacitive load.
- E_i is the worst-case energy consumption.
- E_i, C_i, X_i, and V_i have the relation $E_i = C_i \cdot X_i \cdot V_i^2$

Figure 3 illustrates some of these parameters.

We assume that F_i and V_i do not change during the execution of J_i. However, the supply voltage and clock frequency can change when another task preempts J_i, which resumes after the preemption.

The following parameters characterize the variable-voltage processor:

- (v_j, f_j) is the processor mode. When the processor's supply voltage is v_j, its clock frequency is f_j.
- $m = |\{(v_j, f_j)\}|$ is the processor's mode number.
- $V_{\max} = \max(v_j)$ indicates the largest supply voltage.

For example, when task J_3 executes with processor mode 2 ($j = 2$), we define V_3 and F_3 as $V_3 = v_2$, and $F_3 = f_2$.

Dynamic voltage-scheduling algorithm. Dynamic voltage-scheduling time should be short to accommodate online scheduling. In our dynamic voltage-scheduling techniques, we assume the scheduler assigns a supply voltage to only the next executed task just before task execution. Then, the scheduler must assign supply voltage so that all tasks executed later will not violate these real-time constraints.

We define a time slot for each task. A time slot's start time is when the task execution started. The end time is the maximum time that can guarantee that all future tasks will not violate these real-time constraints. Thus, if the next executed task's supply voltage lets it finish within the time slot that the scheduler gives it, satisfaction of the real-time constraints is always guaranteed.

In our techniques, the scheduler's main work is to determine the time slot's length for each task. The scheduler's remaining work is to assign the minimum voltage to the task so that it can finish within its time slot. Two algorithms determine the length of the time slot:

- The SD algorithm assumes every task's arrival time is known.
- The DD algorithm assumes every task's arrival time is unknown.

Figures 4 and 5 (next page) show the SD and DD algorithms.

SD and DD algorithms have three main steps:

1. *CPU time allocation.* Assign the task set CPU time under the condition that all tasks execute on $V_{\max}$ and that the execution cycle for each task is the worst case. Divide the work of the preempted task into other tasks to simplify the problem in the SD algorithm.

2. *End-time prediction.* Determine the time slot's end time for the next executed task, considering real-time constraints of all later-executed tasks.

3. *Start-time assignment.* Determine the time slot's start time. The finished time of the previously executed task dynamically moves the start time; the time slot can be lengthened if the previous task finishes ahead of schedule.

Initialize (static phase)
Assign task set $\{J_1, \ldots, J_n\}$ CPU time under the condition that all tasks execute on V_{max}. Determine s_i, e_i, and L_i for each task. Divide, into several subtasks, a task preempted by another task, and treat those subtasks as different tasks. If task J_i is divided into l tasks, $J_{i.1}, \ldots, J_{i.l}$, then the arrival time, deadline, and remaining time of task $J_{i.j}$ are

$$a_{i.1} = a_i \qquad d_{i.n} = d_i$$
$$a_{i.j} = e_{i.j-1} \qquad di.j = s_{i.j+1}$$
$$L_{i.j} = d_{i.j} - e_{i.j}$$

The following equation gives the worst-case execution cycles $X_{i.j}$:

$$X_{i.j} = \frac{e_{i.j} - s_{i.j}}{\sum_{k=1}^{n} (e_{i.k} - s_{i.k})} \cdot X_i$$

Sorting according to the early order at the deadline, the task set becomes

$$\{J_1, \ldots, J_{n'}\} \quad (n' \geq n,\ d_i \leq d_{i+1})$$

If tse_i is the time slot's end time for task J_i, then the tse_i for each task in $\{J_1, \ldots, J_n\}$ is

$$tse_i = e_i + \min_{k \geq 1}(L_k)$$

Input
Next executed task: J_i
Current time: t

Output
The length of the time slot for J_i: TS_i

Dynamic process
$TS_i = tse_i - t$

Figure 4. SD algorithm for dynamic voltage scheduling.

Initialize (static phase)
$R := \emptyset$
$t_s := 0$
$J_{exe} := J_{idle}$

Input
Current execution task: J_{exe}
Current time: t

Output
Next execution task: J_{ne}
Occupation period for J_{ne}: T_{ne}

Dynamic process
if new task arrived then
  J_{ar} := arrival task
  if priority(J_{ar}) > priority(J_{exe}) then
    $R := \{J_{exe}\} \cup R$
    if $J_{exe} \neq J_{idle}$ then
      X_{exe} := the rest of X_{exe}
    end if
    $t_s := t + O_{ar}|V_{max}$
    $J_{ne} := J_{ar}$
    $T_{ne} := t_s - t$
  else
    $R := \{J_{ar}\} \cup R$
    $J_{ne} := J_{exe}$
    supply voltage is unchanged
  end if
else if J_{exe} finished then
  J_{hi} := the task with maximum priority in R
  $R := R - \{J_{hi}\}$
  $t_s := t_s + O_{hi}|V_{max}$
  $J_{ne} := J_{hi}$
  $T_{ne} := t_s - t$
end if

Figure 5. DD algorithm for dynamic voltage scheduling.

Table 1. Variable supply voltages.

Cases	Variable supply voltages (volts)
$Processor_1$	Only 3.3
$Processor_2$	3.3 and 0.9
$Processor_3$	3.3, 2.5, and 0.9
$Processor_4$	3.3, 2.5, 1.7, and 0.9
$Processor_5$	Any voltage between 3.3 and 0.9

Table 2. Specification of sample programs.

Task sets	No. of cycles for X_1, X_2, X_3 (billions)	Capacitive loads for C_1, C_2, C_3 (pF)
1	50, 50, 50	100, 100, 100
2	50, 50, 50	80, 100, 120
3	50, 50, 50	40, 100, 160
4	50, 50, 50	20, 40, 240

The SD algorithm performs steps 1 and 2 statically, and step 3 dynamically. The DD algorithm performs all three steps dynamically.

Experimental results

We demonstrated the effectiveness of our proposed static or dynamic scheduling techniques. We used the virtual variable-voltage processor in the experiments, and applied the techniques to an easy task set.

Static scheduling

We use the following set of tasks: task set = $task_1$, $task_2$, $task_3$. The average capacitive loads and the number of execution cycles of these three tasks are $\{C_1, X_1\} = \{50\text{ pF}, 50 \times 10^9\}$, $\{C_2, X_2\} = \{100\text{ pF}, 50 \times 10^9\}$, and $\{C_3, X_3\} = \{150\text{ pF}, 50 \text{ x } 10^9\}$. These three tasks are sequentially processed under a time constraint. In this

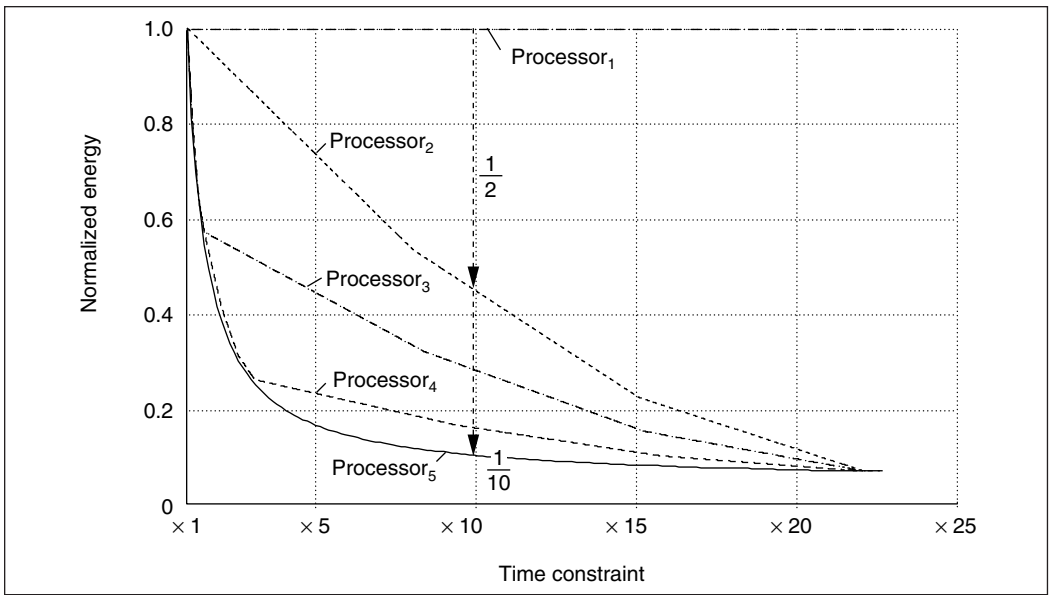

Figure 6. Results for static scheduling, from solving the ILP problem presented earlier. Three tasks are sequentially processed under a time constraint for five kinds of variable-voltage processors, as shown in Table 1. If we relax the time constraint to ×10, we can reduce the energy consumption to 1/10 with an ideal processor (processor$_5$); or we can halve the energy consumption for a processor with only two voltages (processor$_2$).

experiment, we target five kinds of variable-voltage processors, as shown in Table 1. We assume that processors can dynamically vary the supply voltage but can support only one voltage at a time. We obtain the results shown in Figure 6 by solving the ILP problem described in the previous section.

The energy consumption of processor$_1$ is constant even if the time constraint is relaxed, because the total number of tasks remains constant. The more variable voltages there are, the more energy consumption can be reduced. Selecting suitable voltages for the time constraint leads to drastic energy reduction even if the number of variable voltages is small. Therefore, determining the variable voltages that the processor uses is the most important step for optimizing variable voltage scheduling.

Next, we show the experimental results (using processor$_2$) for four kinds of task sets: task set = {task$_1$, task$_2$, task$_3$}), as shown in Table 2. For each set, the processor sequentially processed three tasks. In this experiment, both the total number of execution cycles and the sum of capacitive loads remained the same for each task set. Figure 7 (next page) shows experimental results.

The results demonstrate that the deviation of C_js strongly affects energy reduction. Comparing task set 1 with task set 4, we see a 30% reduction in energy consumption even when the time constraint remains the same. If the tasks' C_js differ, assigning the lower voltage to the tasks with the larger C_js, and the higher voltage to the tasks with the smaller C_js, drastically reduces energy consumption. Of course, energy consumption also decreases according to how much the time constraint is relaxed.

From the results, we observe the following:

- Increasing the number of variable voltages may greatly reduce energy consumption. However, having more than three variable voltages will saturate energy reduction.
- Selecting suitable voltages for the time constraint significantly reduces energy consumption even if the number of variable voltages is very small.
- Even if the time constraint is constant, assigning lower voltage to the tasks with the larger C_js, and higher voltage to the tasks with the smaller C_js, reduces total energy consumption by 30%. Voltage scheduling is

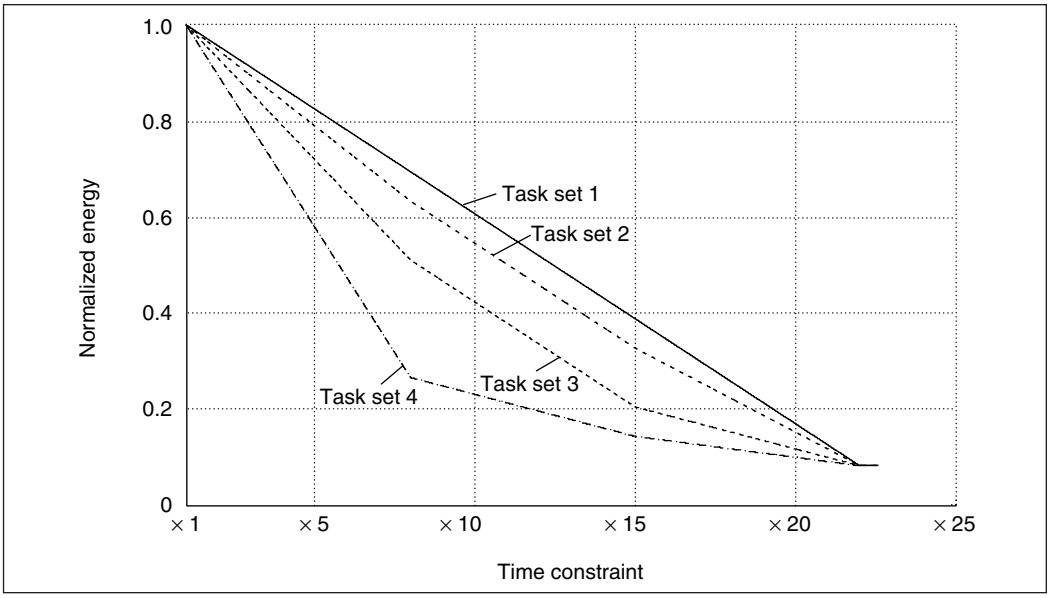

Figure 7. Results for static scheduling that considers the capacitive load. Three tasks are sequentially processed for the four task sets shown in Table 2.

Table 3. Processor mode.

Supply voltage (volts)	Clock frequency (MHz)
5.0	50
4.0	44
2.5	32

Table 4. Task set.

Task	X_i (cycles × 10^6)	Capacitive load C_i (pF)
J_1	10	1.000
J_2	8	1.875
J_3	15	1.333
J_4	5	1.000
J_5	4	7.500

Table 5. Arrival time and deadline time for two scenarios of task execution.

	Scenario 1		Scenario 2		Actual
Task	Arrival time a_i(s)	Deadline time d_i(s)	Arrival time a_i(s)	Deadline time d_i(s)	execution time (million cycles)
J_1	0.0	0.2	0.0	0.5	9.3
J_2	0.0	0.4	0.0	0.7	7.0
J_3	0.0	0.8	0.0	0.9	14.0
J_4	0.4	0.5	0.4	0.7	3.0
J_5	0.5	1.2	0.5	1.4	3.0

very effective for the application program whose capacitive load is widely biased.

Dynamic scheduling

We assume a preemptive real-time system where the five tasks of Table 4 execute on a variable-voltage processor. Table 3 shows the processor's modes. We observe the behavior of the energy consumption for two scenarios of task execution, as shown in Table 5. Each task has an earlier deadline in scenario 1 than in scenario 2.

We scheduled the order and voltage of tasks

using the following methods:

- *Normal*. Assign the maximum supply voltage to all tasks.
- *SD*. Use the SD algorithm to dynamically assign each task a supply voltage after statically scheduling the execution order.
- *DD*. Use the DD algorithm to dynamically assign each task a CPU time and supply voltage.

We assume that a_i and d_i are known in advance for the SD algorithm, and unknown for the DD algorithm. The SD algorithm has an advantage in this assumption because the scheduler in the SD algorithm knows more information than the DD algorithm. For each scheduling method, the scheduler scheduled tasks in the order $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_3 \rightarrow J_5$, where J_3 was preempted by J_4.

Table 6 shows the energy estimation results for scenarios 1 and 2. In scenario 1, the energy reduction rate was 38% for SD and 32% for DD, compared with the normal case. In scenario 2, the energy reduction rate was 62% for SD and 32% for DD, compared with normal.

From the experimental results, we observe the following:

- SD and DD always give better results than the normal case.
- In SD, looser deadline constraints lead to better energy reduction rates because the scheduler has more time to lower the supply voltage.
- In contrast to SD, power consumption using the DD scheduler is independent of deadline constraints.

Table 6. Energy-consumption estimation results in scenarios 1 and 2 for normal (always maximum), SD-, and DD-assigned supply voltages.

		Scenario 1		Scenario 2	
Task	Normal (volts)	SD (volts)	DD (volts)	SD (volts)	DD (volts)
J_1	5.0	5.0	5.0	4.0	5.0
J_2	5.0	5.0	5.0	4.0	5.0
J_3	5.0	2.5	5.0	2.5	5.0
J_4	5.0	5.0	5.0	2.5	5.0
J_3	5.0	4.0	4.0	2.5	4.0
J_5	5.0	2.5	2.5	2.5	2.5
Energy (Joules)	1,665	1,036	1,130	634	1,130

EXPERIMENTAL RESULTS demonstrate that using software to control processor supply voltage can significantly reduce energy consumption. Voltage scaling is far more effective than the shutdown approach, which simply stops the power supply when the system is inactive. Moreover, because of the tasks' capacitive loads, the reduction in energy consumption is even better. Dynamic approaches can drastically reduce energy consumption, depending on the input data. In our approach, time overhead to compute the optimal supply voltage is negligible because the SD and DD algorithms optimize voltages within a few clock cycles.

Our approach is important for complex and low-power SOC design. Current semiconductor technology enables integrating a large system on a single chip. Several application programs require high clock frequency (more than 1 GHz), yet low-performance applications still play an important role in many of today's systems. As this trend continues, some application programs will require extremely high performance, while others will require only low performance. As system size increases, this gap will widen. Sophisticated energy management will therefore be crucial in future SOC designs.

Acknowledgment

This work is partly supported by Grant-in-Aid for Scientific Research 12558029, 11003357, and STARC 987.

References

1. T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable-Voltage Processors," *Proc.1998 Int'l Symp. Low Power Electronics and Design* (ISPLED 98), ACM Press, New York, 1998, pp. 197-202.
`2. T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for a Variable-Voltage Processor," *Proc. 12th Int'l Symp. System Synthesis* (ISSS 98), IEEE CS Press, Los Alamitos, Calif., 1999, pp. 25-29.

3. S. Lee and T. Sakurai, "Run-Time Power Control Scheme Using Software Feedback Loop for Low-Power Real-Time Application," *Proc. 2000 Asia and South Pacific Design Automation Conf.* (ASP-DAC 2000), ACM Press, New York, 2000, pp. 381-386.
4. S. Lee and T. Sakurai, "Run-Time Voltage Hopping for Low-Power Real-Time Systems," *Proc. 37th Design Automation Conf.* (DAC 00), ACM Press, New York, 2000, pp. 806-809.
5. "Crusoe Processor," Transmeta Corp., Santa Clara, Calif., http://www.transmeta.com/crusoe/.
6. I. Hong et al., "Power Optimization of Variable Voltage Core-Based Systems," *Proc. 35th Design Automation Conf.* (DAC 98), ACM Press, New York, 1998, pp. 176-181.
7. I. Hong, M. Potkonjak, and M.B. Srivastava, "On-Line Scheduling of Hard Real-Time Tasks on Variable-Voltage Processor," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design* (ICCAD 98), ACM Press, New York, 1998, pp. 653-656.
8. Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems," *Proc. 36th Design Automation Conf.* (DAC 99), ACM Press, New York, 1999, pp. 134-139.
9. T. Pering, T. Burd, and R. Brodersen, "Voltage Scheduling in the lpARM Microprocessor System," *Proc. Int'l Symp. Low Power Electronics and Design* (ISPLED 00), ACM Press, New York, 2000, pp. 96-101.
10. T. Burd et al., "A Dynamic Voltage Scaled Microprocessor System," *Proc. IEEE Int'l Solid-State Circuits Conf.* (2000 ISSCC), IEEE Press, Piscataway, N.J., 2000, pp. 294-295.

Takanori Okuma is a PhD candidate in the Department of Computer Science and Communication Engineering at Kyushu University, Fukuoka, Japan. His research interests include real-time operating systems, system-level design, and low power design. Okuma has a ME in computer science and communication engineering, from Kyushu University, Fukuoka, Japan. Contact him at okuma@c.csce.kyushu-u.ac.jp.

Hiroto Yasuura is a professor in the Department of Computer Science and Communication Engineering at Kyushu University, Fukuoka, Japan, and a research director of the Institute of Systems and Information Technologies, Kyushu, Fukuoka, Japan. His research interests include parallel computer architectures, VLSI CAD, hardware algorithms for VLSI, and system design methodology. Yasuura has a PhD in computer science from Kyoto University, Kyoto, Japan. Contact him at yasuura@c.csce.kyushu-u.ac.jp.

Tohru Ishihara is a research associate at the VLSI Design and Education Center, University of Tokyo. His interests include low-power VLSI system design. He has a PhD in computer science and communication engineering from Kyushu University, Fukuoka, Japan. He is a member of the IEEE. Contact him at ishihara@silicon.u-tokyo.ac.jp.