Hybrid Run-time Power Management Technique for Real-time Embedded System with Voltage Scalable Processor

Minyoung Kim
School of Electrical Engineering and Computer Science, Seoul National University
Seoul 151-742 Korea
+82-2-880-7292
mykim@iris.snu.ac.kr

Soonhoi Ha
School of Electrical Engineering and Computer Science, Seoul National University
Seoul 151-742 Korea
+82-2-880-7292
sha@iris.snu.ac.kr

ABSTRACT
This paper presents a new run-time power management technique for real-time embedded systems which consist of a voltage scalable processor and power controllable peripheral devices. We have observed that there exist significant trade-offs in terms of energy consumption between the Dynamic Power Management (DPM) scheme and the Dynamic Voltage Scaling (DVS) scheme over a wide range of system operating conditions. The proposed technique fully exploits workload-variation slack time by partitioning the task into several timeslots and shut down the unneeded peripheral device on timeslot-by-timeslot basis. Through extensive simulations, the novelty and the usefulness of the proposed technique are demonstrated.

Categories and Subject Descriptors
D.4.7 [**Operating Systems**]: Organization and Design – *real-time systems and embedded systems*

General Terms
Design

Keywords
Run-time power management technique, real-time embedded system, DPM(Dynamic Power Management, DVS(Dynamic Voltage Scaling).

1. INTRODUCTION
Over the past several years, two different researches on power minimization have been carried out extensively. One is dynamic power management (DPM) scheme and the other is dynamic voltage scaling (DVS) scheme. In DPM, a device is put into a low-power and low-performance sleeping state to save power when it is not serving any request during a time interval sufficient to compensate for the shutdown and wakeup overhead. A good survey on the DPM techniques can be found in [3].

On the other hand, in a DVS scheme, the voltage scheduler varies the supply voltage so that the processor consumes minimal amount of energy by operating at lower performance level that satisfies the required speed constraints. Recently, hardware implementation [7,8] and many software voltage scheduling methods have been proposed. Especially in [6,11], DVS algorithms for real-time system were proposed considering the worst-case slack time and workload-variation slack time[1] which are inherent in the real-time scheduling. Nevertheless, these approaches focused only on processor energy ignoring energy consumption of peripheral devices.

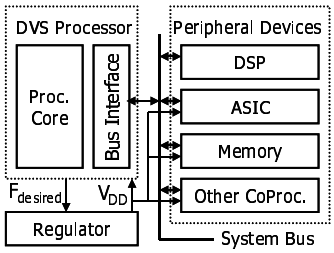

Figure 1. System architecture.

However, a typical embedded system consists of processor, memory, and many other kinds of peripheral devices such as DSP, ASIC, etc., as shown in Figure 1. Under the reasonable assumption that peripheral devices do not offer energy gain by reducing the supply voltage, there is a trade-off relation between DVS and DPM. DVS slows down the processor speed to lower the dynamic power consumption of the processor. But, this results in more static power consumption and more energy consumption of peripheral devices since the active periods of peripheral devices are also lengthened to reduce the time duration for shutdown or idle mode operation. On the other hand, to save more peripheral energy, we may want to shorten the active tasks as much as

[1] In real-time system, there are some slack times since the utilization of the processor is frequently less than 1 even if all tasks run at their worst-case execution times (WCETs). Moreover, when some task instances are completed earlier than their WCETs, there are more idle intervals. We denote them as *worst-case slack time* and *workload-variation slack time*, respectively

possible to gain more energy saving from DPM for peripheral devices.

In this paper, we pay our attention to this tradeoff and propose a new run-time power management technique for real-time embedded systems which consist of a voltage scalable processor and power controllable peripheral devices. Note that peripheral devices of interest in this paper include auxiliary component such as frame memory, fine-grain coprocessors, and flash memory; these devices have no power management module. The voltage scalable processor can issue the power management command to the power controllable peripheral device only at the time slot boundary. The processor and the peripheral device communicate frequently in the given time slot. Neither very slow devices like current hard-disks nor very fast devices which can do DMA operation are not considered adequate for the proposed technique. Our approach is not aimed at devising a new DPM nor DVS algorithm but to combine the existing DPM and DVM algorithms into a new framework. Therefore, it is a complementary work to the previous researches on dynamic power management of systems.

This paper is organized as follows. In the next section, we briefly review the previous works on DPM and DVS separately. In section 3, we introduce motivational examples and state the problem we are addressing. In section 4, the proposed technique is illustrated with simple examples, and discussed extensively. Section 5 presents extensive simulation results. In section 6 we comment on the extension possibilities of this work, and draw conclusions in section 7.

2. PREVIOUS WORKS

2.1 Dynamic Power Management (DPM)

In a DPM scheme, the device changes its power state [3] after staying idle for a certain amount of decision time. So, it is important to predict the next idle period to reduce the wasted energy consumed during the decision time. Several techniques have been proposed to predict the length of the next idle period based on the history of device usage [4,12].

Conventional DPM approaches are based on the observation of the requests at the target device level. This hardware-oriented approach is also known as Device-Level Power Management (DLPM) [13] scheme, in which the target device has an autonomous power manager [14]. DLPM has a major limitation because it does not provide task-specific power control.

In contrast, Task-Based Power Management (TBPM) scheme provides software-oriented power management which uses task information available in the operating system. This observation is reflected in Advanced Configuration and Power Interface (ACPI) [2] and OnNow [1]. Furthermore, process scheduling techniques in the operating system may be used for efficient device power control [13,14]. Especially in [14] this approach is called as the requester aware power management. Each task notifies the scheduler the usage periods of peripheral devices. Then, the scheduler rearranges the execution order of tasks to cluster idle periods of devices to reduce the power-state transition overhead.

DPM techniques are usually concerned with peripheral devices even though processors have also several power states eligible for DPM. If a processor is voltage scalable, however, DVS is known to be more efficient than DPM for reducing the processor power. One of the latest DVS approaches such as LPFPS [11] uses both DPM and DVS schemes for processor power reduction for hard real-time systems. This is discussed in the next section.

2.2 Dynamic Voltage Scaling (DVS)

A voltage scalable processor can change the supply voltage along with the operating frequency when the required performance is lower than the maximum performance [7,8]. In recent studies, several DVS algorithms applied to PDA-class devices [10] and to real-time systems [6,11] have shown significant power reduction.

Even though DVS is the most power efficient, as of today, peripheral devices are not ready for DVS. Some peripheral devices do not allow the supply voltage to be changed due to the internal architecture, and others have internal DC-DC converters, thus having no energy gain when the supply voltage is reduced. Therefore, in this paper, we assume that processor is the only voltage scalable component for simplicity.

Among many proposed methods, there are two recent research results: Low Power Fixed Priority Scheduling (LPFPS)[11] and Run-time Voltage Hopping (RVH)[6]. In LPFPS, the scheduler dynamically varies the speed of the processor whenever possible, and brings the processor to a power-down mode when it is predicted to be idle for a sufficiently long time interval. LPFPS assumes that it performs voltage scheduling on task-by-task basis, so it cannot fully use the slack time due to workload variation. In order to overcome the limitation in LPFPS, RVH introduces the timeslot concept. Supply voltage is controlled on a timeslot-by-timeslot basis inside each task. Figure 2 shows an example behavior of LPFPS and RVH.

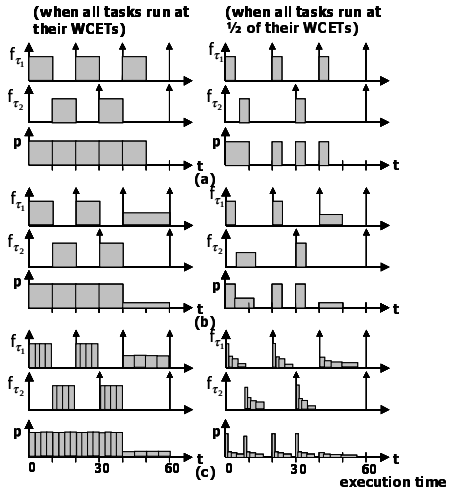

Figure 2. Slack time variation and DVS technique; (a) No voltage scaling, (b) LPFPS, (c) RVH.

Consider two periodic tasks $\tau_1(20,20,10)$ and $\tau_2(30,30,10)$ as shown in Figure 2, where $\tau_i(T_i, D_i, C_i)$ denotes period, deadline, and worst-case execution time (WCET) of task τ_i. f_{τ_i} and P in the vertical axis denote clock frequency and the power consumption of the processor, respectively. Figure 2(b) presents LPFPS and Figure 2(c) presents RVH, respectively. If we assume

all tasks run up to $\frac{1}{2}$ of their WCETs, both LPFPS and RVH result in significant power reduction since these two techniques exploit not only the worst-case slack time but also the workload-variation slack time for lowering the supply voltage. Especially, in the case of RVH, the energy gain is remarkable. As shown in Figure 2(c), the RVH technique fully exploits workload-variation slack time by partitioning the task into a series of timeslots and scaling voltage dynamically inside a task. Since the timeslot-based DVS includes the task-based DVS as an extreme case, the timeslot model of RVH is also used in this paper.

3. MOTIVATIONAL EXAMPLE

We are concerned with a simple system architecture that consists of a DVS processor and a power controllable peripheral device. For explanation of the proposed idea, we have used some simple but motivational examples.

3.1 Task with single timeslot

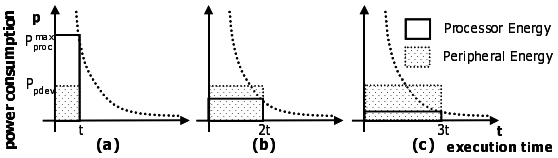

Figure 3. A simple example of single timeslot model.

Figure 3 illustrates the basic idea of our approach. As we discussed earlier, in order to compensate for the state-transition overhead, a device must stay in a sleeping state long enough. We denote this as device break-even time (T_{be}), which is a device dependent parameter. At this moment, we assume that T_{be} is zero and the peripheral device has two power states: active and sleeping for simplicity. Dashed line illustrates the relationship between the processor power and the execution time of the task.

If a processor finishes the task early and goes into power-down mode, there is no need to use peripheral devices. So, the device can be shutdown and energy saved. In the case of Figure 3(a), the processor operates at its maximum frequency and full power level, $P^{\max}_{proc}$. After the processor finishes the task at time t, the peripheral devices can be shutdown. In case of Figure 3(c), on the other hand, the processor operates at $\frac{1}{3}$ of its maximum frequency and $\frac{1}{9}$ of its maximum power level. Consequently, the peripheral device must operate longer than in Figure 3(a), even though the processor can save energy. We can consider Figure 3(a) as a DPM scheme and Figure 3(c) as a DVS scheme. It should be pointed out that neither Figure 3(a) nor Figure 3(c) is the most power-efficient.

Given processor power level P_{proc} and peripheral device power level P_{pdev} , the total energy consumption is calculated by

$$(execution\ time)\times(P_{proc}+P_{pdev})\ .$$

Therefore, the total energy consumption of Figure 3(a) is calculated to

$$t\times(P^{\max}_{proc}+P_{pdev})\ .$$

In the same manner, the total energy consumption of Figure 3(b) is

$$2t\times(\frac{1}{4}\times P^{\max}_{proc}+P_{pdev})$$

and that of Figure 3(c) becomes

$$3t\times(\frac{1}{9}\times P^{\max}_{proc}+P_{pdev})\ .$$

A simple calculation leads to the result that if P_{pdev} is between $\frac{1}{6}$ of $P^{\max}_{proc}$ and $\frac{1}{2}$ of $P^{\max}_{proc}$, the energy consumption of Figure 3(b), neither DPM nor DVS becomes the minimum. We need a new technique such as Figure 3(b) to find the solution.

3.2 Task with multiple timeslots

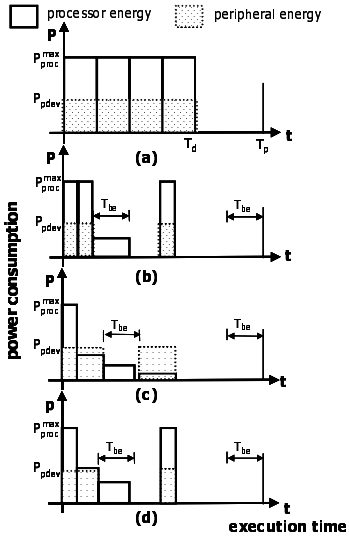

Figure 4. Policy comparison among various power management schemes: (a) no power management, (b) DPM, (c) DVS, (d) the proposed scheme, where T_d and T_p mean the deadline and the next period, respectively.

Now, we extend the single timeslot model to the multiple timeslots model. Figure 4 shows four possible scenarios when the following power management schemes is applied to a real-time task with four timeslots: (a) no power management (no processor DVS, no peripheral shutdown), (b) DPM first (peripheral shutdown first), (c) DVS first (processor energy minimization first) and (d) the proposed scheme. In this example, one task is partitioned into 4 timeslots that may also be considered as sequentially executed tasks. Assume that all timeslots run to $\frac{1}{2}$ of their WCETs. And we assume that timeslot 3 does not use the peripheral device while the other timeslots do.

As shown in Figure 4, two conventional approaches cannot achieve the minimum energy for the entire system. In Figure 4(b), since DPM prefers peripheral shutdown to processor power reduction, the processor runs at full speed to minimize the active period of peripheral devices in timeslots 1,2, and 4. During timeslot 3 where the peripheral device is not used, the DVS scheme is applied to the processor. In the opposite case, like Figure 4(c), DVS sets higher priority to slow down the processor maximally at each timeslot and shutdowns the peripheral whenever possible, at timeslot 3 in this example. As can be seen from the illustration, neither scheme provides the optimal solution. Figure 4(d) illustrates the proposed scheme, which will be discussed in detail in section 4.3.

4. PROPOSED METHOD

To explain the proposed technique, we assume that the task that is characterized as deadline (T_d), period (T_p), and WCET can be partitioned into timeslots and the voltage scheduler has information which timeslot requires the peripheral device. Starting from the single timeslot case, we extend the technique to the general case.

4.1 Technique for single timeslot

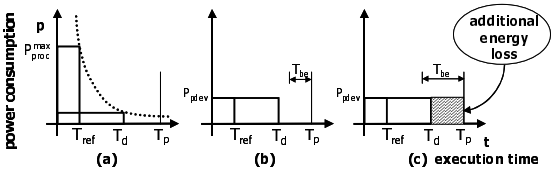

Figure 5. Relationship between power consumption and WCET: (a) for the processor, (b) for the peripheral device with small T_{be}, when peripheral operates at 0.4 of processor full power level, (c) the peripheral device with large T_{be}, where T_d, T_p and T_{ref} mean the deadline, the next period, and WCET at $P_{proc}^{\max}$, respectively.

Processor voltage scaling is determined by the deadline analysis, varying its operating frequency as long as it satisfies the deadline. After the current execution, the processor and the peripheral can go into power-down mode until the next period if the idle time is longer than T_{be}. Since T_{be} of the processor is usually insignificant, we assume that the processor enters the sleeping state at all idle periods. For example, the time overhead associated with voltage switching is on the order of 10 cycles in a processor [10,15]. This overhead is negligible since the computation itself can continue during the voltage and frequency transition. On the other hand, T_{be} for the peripheral should be compared with the expected idle time to make shutdown decision.

Figure 5 illustrates the relation between the WCET and power consumption for the processor (a) and the peripheral device (b, c). The graph should read that the execution time of the peripheral is the same as that of the processor. Note that the power level of the peripheral is assumed to be constant in the active state.

For quantitative analysis, we denote voltage scaled power level, processor maximum frequency, processor scaled frequency as P_{proc}^{scaled}, $f_{\max}$, and f_{scaled}, respectively. If we let t_d be the actual execution time at $P_{proc}^{\max}$, the processor energy gain can be expressed as the following formula (1), where $E\{t_d\}$ is the expected value of the actual execution time:

$$E_{gain} = \left(P_{proc}^{\max} - P_{proc}^{scaled} \times \frac{f_{\max}}{f_{scaled}} \right) \times E\{t_d\} \quad (1)$$

To compute the energy loss associated with the peripheral device, we distinguish two cases shown in Figures 5(b) and 5(c). In Figure 5(b), where the peripheral goes into the idle state after the WCET, the energy loss comes from the extended operating duration of the peripheral. On the other hand, in Figure 5(c), the smaller remaining idle time ($T_p - T_d$) than T_{be} due to processor slow-down results in additional energy loss. This energy loss is illustrated as a dashed box in Figure 5(c).

The peripheral energy before scaling is $P_{pdev} \times (t_d + T_{be})$ where $P_{pdev} \times T_{be}$ represents the energy overhead to change the power-state of the device. After scaling, the peripheral energy for a given instance of execution becomes as follows.

$$E_{pdev}^{scaled} = \begin{cases} P_{pdev}\left(t_d \times \frac{f_{\max}}{f_{scaled}} + T_{be} \right) & if \quad t_d \times \frac{f_{\max}}{f_{scaled}} + T_{be} < T_p \\ P_{pdev} \times T_p & otherwise \end{cases} \quad (2)$$

The bottom formula of equation (2) indicates that if the peripheral is slowed down too much, it loses the chance of power shutdown. Therefore, the expected energy loss associated with the peripheral can be summarized with equation (3).

$$\begin{aligned} E_{loss} &= E\left\{ \min\left[t_d \times \frac{f_{\max}}{f_{scaled}} + T_{be}, T_p \right] \times P_{pdev} - (t_d + T_{be}) \times P_{pdev} \right\} \\ &= P_{pdev} \times E\left\{ \min\left[t_d \times \frac{f_{\max}}{f_{scaled}}, T_p - T_{be} \right] - t_d \right\} \end{aligned} \quad (3)$$

In the case where the peripheral enters the idle state after the deadline like Figure 5(b), equation (3) is reduced to the following equation.

$$E_{loss} = P_{pdev} \times \left(\frac{f_{\max}}{f_{scaled}} - 1 \right) \times E\{t_d\} \quad (4)$$

Suppose the statistical distribution of t_d is uniform between BCET (Best Case Execution Time) and WCET (Worst Case Execution Time). Then, E_{loss} becomes

$$E_{loss} = \begin{cases} P_{pdev} \times \left\{ \left(\frac{f_{\max}}{f_{scaled}} - 1 \right) \times \frac{WCET + BCET}{2} - \frac{1}{2} \frac{f_{scaled}}{f_{\max}} \frac{\left(T_p - T_{be} - WCET \frac{f_{\max}}{f_{scaled}} \right)^2}{WCET - BCET} \right\} & \\ & \text{if } \frac{f_{\max}}{f_{scaled}} WCET + T_{be} > T_p \\ P_{pdev} \times \left\{ \left(\frac{f_{\max}}{f_{scaled}} - 1 \right) \times \frac{WCET + BCET}{2} \right\}, & otherwise \end{cases} \quad (5)$$

In the beginning of the time slot, the voltage scheduler computes

the net energy gain, $\Delta E = E_{gain} - E_{loss}$ for each candidate of scaled voltage and processor frequency, and selects the optimum values to maximize ΔE. Since this decision is made at run time, it should be fast. However, computing the exact energy loss from equation (3) with given statistical information of t_d is likely to take significant amount of time. So, approximate computation is needed. Finding a better approximation is a future research topic.

In this section, we assume a simple architectural model that the peripheral power is not a function of the processor frequency. In reality, reducing the processor frequency also reduces the bus frequency, which may affect the power consumption of the peripherals. Another approximation we made is that the peripheral consumes the same power in the active state regardless of whether it is actually operating or waiting for a request.

4.2 Technique for two timeslots

In the case where a task is partitioned into multiple timeslots, the proposed technique makes the decision examining the previous timeslot and the current timeslot[2]. We also assume that the deadline is equal to the sum of the WCETs of all timeslots, as is the typical working scenario. In this section, we explain our technique with a task with only two timeslots. The extension to the multiple time slots will be discussed in the next section.

At the first timeslot, the processor should be run at full power to guarantee the deadline in the worst case. If we lower the supply voltage from the beginning and both timeslots experience the WCETs, we will miss the deadline since the deadline is set assuming the full speed operation. Hence, the power control decision is effective only at the second timeslot. Since the first timeslot is much likely to complete its execution earlier than the WCET, we can use the slack time from the first timeslot.

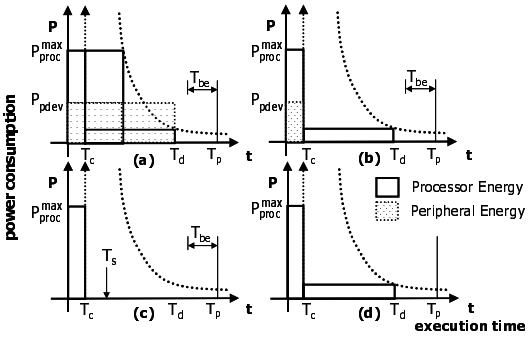

Figure 6. Four device usage patterns and the proposed method: (a) both timeslots require the peripheral device, (b) only first timeslot requires the peripheral device, (c) only second timeslot requires the peripheral device, (d) neither requires the peripheral device, where T_c, T_d, T_p and T_s mean the current time, the deadline, the next period and the initial scheduling time of the second timeslot, respectively.

[2] At this moment, we assume the shutdown decision of the peripheral device at the previous timeslot always follows the request for simplicity. Actually, the proposed technique makes the decision examining the current power state of the peripheral device and the request of the current timeslot.

As illustrated in Figure 6, there are four peripheral request patterns; (1) both timeslots require the peripheral device, (2) only the first timeslot requires the peripheral device, (3) only the second timeslot requires the peripheral device, and (4) neither requires the peripheral device. The following algorithm explains each case at the second timeslot scheduling point.

Case 1: Both time slots use the peripheral device (Figure 6(a))

The situation is exactly the same with the single timeslot case discussed in the previous section. The deadline is set to the sum of WCET and the slack time from the first timeslot.

Case 2: Only the first time slot uses the peripheral device (Figure 6(b))

We can shutdown the peripheral if the time left to the next period is greater than T_{be}. Processor supply voltage can be reduced maximally as long as the deadline is not violated in the worst case.

Case 3: Only the second time slot uses the peripheral device (Figure 6(c))

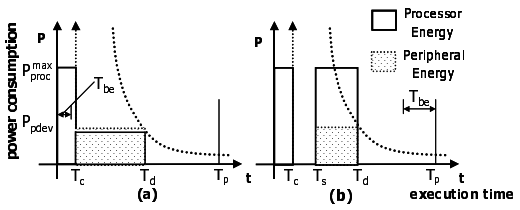

Figure 7. Sub-cases of case 3: (a) the second timeslot scheduled at T_c, (b) the second timeslot scheduled at T_s (without DVS), where T_c, T_d, T_p and T_s mean current time, deadline, next period and initial scheduling time of the second timeslot, respectively.

If the current time (T_c) is greater than T_{be}, there is no harm in starting the second timeslot immediately at T_c. Figure 7(a) describes this situation. However, as shown in Figure 7(b), if T_c is less than T_{be}, the situation becomes more complicated. Suppose that the second timeslot starts execution immediately at T_c. Then, we have to pay the peripheral energy consumption during the first timeslot. If we delay the start time of the second timeslot by a point of time between T_{be} and T_s, we can save the peripheral power from the first timeslot. However, we pay the increased energy cost at the second timeslot since the deadline becomes shorter. Since the power overhead of the second timeslot gets bigger as the second timeslot is delayed more, there is an optimal point such that the net energy becomes minimal. Note that after the delay, minimizing the power cost of the second timeslot is again reduced to the single timeslot case with a shortened deadline parameter. Since the accurate computation is too time consuming, it is not applicable at run-time. Thus, we use a simple heuristic in the experiments of section 5: we compare the energy consumption with a few delay values and choose the minimum.

Case 4: Neither timeslot use peripheral devices (Figure 6(d))

We can apply the DVS technique to minimize the processor power since the peripheral can be shutdown at all times.

4.3 Technique for multiple timeslots

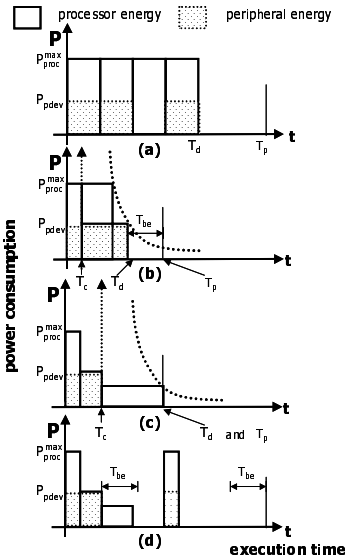

Figure 8. Multiple timeslot example: (a) initial scheduling (WCET analysis), (b) second timeslot scheduling point, (c) third timeslot scheduling point, and (d) the optimal solution, where T_c , T_d , and T_p mean current time, the deadline and the next period, respectively.

In this section, we explain how the proposed technique can be extended to a task with multiple timeslots. Figure 8(a) displays an initial scheduling result of a task with 4 timeslots, which is nothing but the motivational example of Figure 4(a). At the first timeslot, we run the processor at full speed. Then, we arrive at the second timeslot, as shown in Figure 8(b). Since the first and the second timeslot both need the peripheral device, the situation is the same as (Case 1) of the previous section except that we adjust the deadline and the period parameter. The deadline of the second timeslot is fixed as the initial scheduled time. But the next period now moves to the start time of timeslot 4 when the peripheral must turn-on after the second timeslot. Since timeslot 3 does not use the peripheral, we turn off the device after timeslot 2. Hence, the situation becomes exactly the same as with the case described in Figure 6(a), with adjusted deadline and period. Note that the result is different from Figure 4(b) and Figure 4(c) because we consider the peripheral energy and the processor energy simultaneously. The supply voltage during the second timeslot is higher than the voltage proposed in Figure 4(b), but lower than the voltage proposed in Figure 4(c). In the same way, when we schedule the third timeslot and the last timeslot, the problem is reduced to 2 timeslots problems as described in Figure 6(b) and Figure 7(a), respectively. Figure 8(d) reveals the optimal solution. In summary, multiple timeslot problem is reduced to a series of two timeslot problems where the next period is moved to the next peripheral request arrival.

4.4 Technique for multitask environment

Until now, we limit our discussion to the single task. But it can be extended to multi-task environment. In a multitask environment, the assumption that the first timeslot should run at full power level must be discarded since the first timeslot can be scheduled at the end time of the previous task, and earlier than the scheduled time. If there is only one task in the ready queue, the minimum of time remained to next scheduling point and the deadline of the task is added to the slack time.

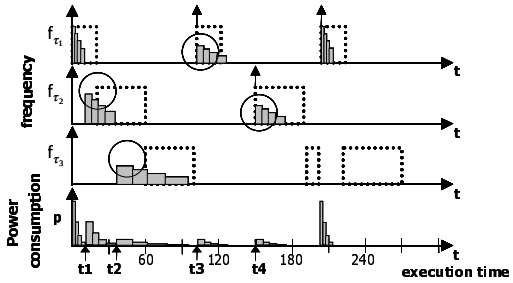

Figure 9. Multitask example.

Consider three periodic tasks $\tau_1(100,100,20)$, $\tau_2(150,150,40)$ and $\tau_3(350,350,100)$ in Figure 9. We know that these three tasks are schedulable using the Rate Monotonic Scheduling (RMS) algorithm and the initial scheduling is depicted with dashed lines. If the task τ_1 finishes its execution at half of its WCET, the remaining time can be used for the next task τ_2. Therefore, the first timeslot of the task τ_2 doesn't have to run at full power level because the remaining time from the previous slot is transferred to the next task τ_2. Then the current task τ_2 can use this time as the initial slack time. The circle at time t1 illustrates this. Note that LPFPS and RHV based on LPFPS intend to minimize the power when only one task is ready to run. Hence, in LPFPS and RHV, there's no supply voltage reduction at time t1 because there are two tasks, τ_2 and τ_3, that are ready to run. When only one task is ready to run the proposed method works in the same manner as LPFPS and RHV. The circles at times t2, t3 and t4 illustrate this situation.

5. EXPERIMENTS

In this section, we evaluate the performance of the proposed technique with simulation since we do not have such an embedded system consisting of a voltage scalable processor and a power controllable peripheral device. Since there are numerous parameters in the problem statement, we take our best efforts to use the realistic values. As for the voltage scalable processor, we obtained the power-latency curve of ARM8 microprocessor core from the author [7,8,9]. The processor can operate from 80MHz with 3.8V down to 8MHz with 1.2V while the energy consumption varies from 5.6 to 0.54mW/MIPS. In our experiments, we change the power level of the peripheral device to 0.5, 1.0 and 1.5 of the processor power in the active state. For example, the active power of a DSP processor falls into this range.

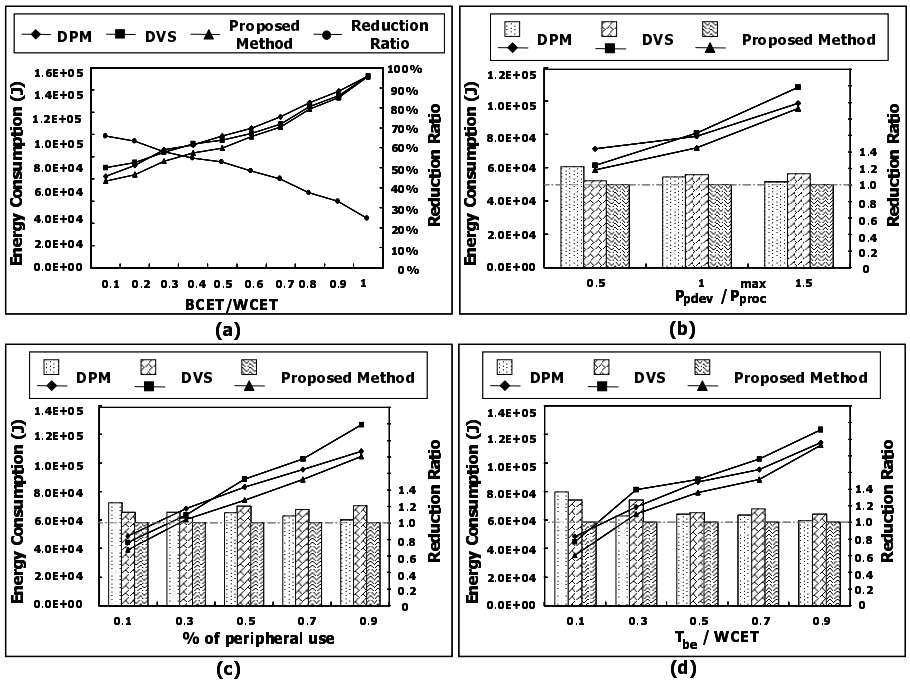

Figure 10. Simulation results of single task environment, where device request probability of each time slot is assumed to be 0.5 except for the third set of experiments: (a) energy consumption and energy reduction ratio on varying $\frac{BCET}{WCET}$, (b) on varying $\frac{P_{pdev}}{P_{proc}^{\max}}$, (c) on varying the device usage probability, (d) on varying the device break-even time.

Through simulations, we compare our proposed technique with the DPM and the DVS schemes in terms of the energy consumption. For conservative comparison, we improve the previous DPM and DVS schemes so that they also operate in a hybrid fashion. In the DPM scheme, the processor is slowed down during the timeslots with no device request to minimize the processor power while the processor runs at full speed to minimize the active period of the device. In the DVS scheme, on the other hand, the processor is maximally slowed down first, and turns off the device during the period without device request.

For multitask experiments, we use the task sets defined in the Computerized Numerical Control (CNC) machine controller example [5], while we arbitrarily define the task characteristic for the single task experiment. Each simulation result is obtained from 30 random experiments, and displayed in Figure 10 and Figure 11. For simulation of single task environment, we assume that the deadline and the period of the application task are equal to the worst-case execution time at full processor speed. The task is partitioned into 10 time slots whose worst-case execution times are assumed to be equal for simplicity. We believe that such assumption does not affect the discussion in a significant way. On the other hand, for simulation of multitask environment with CNC controller application which consists of 8 individual tasks, we partition each task into 7 to 15 timeslots according to its WCET. We also assume that the execution time of the task has the uniform distribution between [BCET, WCET], where we vary the ratio of BCET over WCET from 0.1 to 1.0. Another parameter we are concerned with is the peripheral usage frequency. We change the average number of timeslots that require the peripheral device. Finally, we also vary the device break-even time to examine the energy overhead of the DPM scheme. Note that we ignore the break-even time of the processor since it is usually much smaller than that of the peripheral device.

Figure 10(a) and Figure 11(a) show the energy consumption and the energy reduction ratio when we vary the BCET from 10% to 100% of the WCET for each timeslot. The energy reduction ratio is obtained by the comparison with energy consumption when no power management scheme is used. In this simulation, we use the following parameter values: $\left\{\frac{P_{pdev}}{P_{proc}^{\max}}, \frac{T_{be}}{WCET}\right\}$ = {1,0.1} . Device request probability of each time slot is assumed to be 0.5 except for the third set of experiments. From Figure 10(a), it is seen that the energy reduction ratio increases as the BCET becomes smaller. Even though the BCET equals the WCET and there is only a single task, which is the most degenerate case, the proposed method as well as DVS and DPM scheme obtain about 25% energy gain since peripheral device can be shutdown 50% of duration. As shown in Figure 11(a), multitask environment with no workload-variation slack time results in 62 % of energy reduction. As the execution time varies within a wider range, the proposed technique shows better results than other schemes. With relatively small variation of execution times, the DVS scheme approaches to the proposed technique.

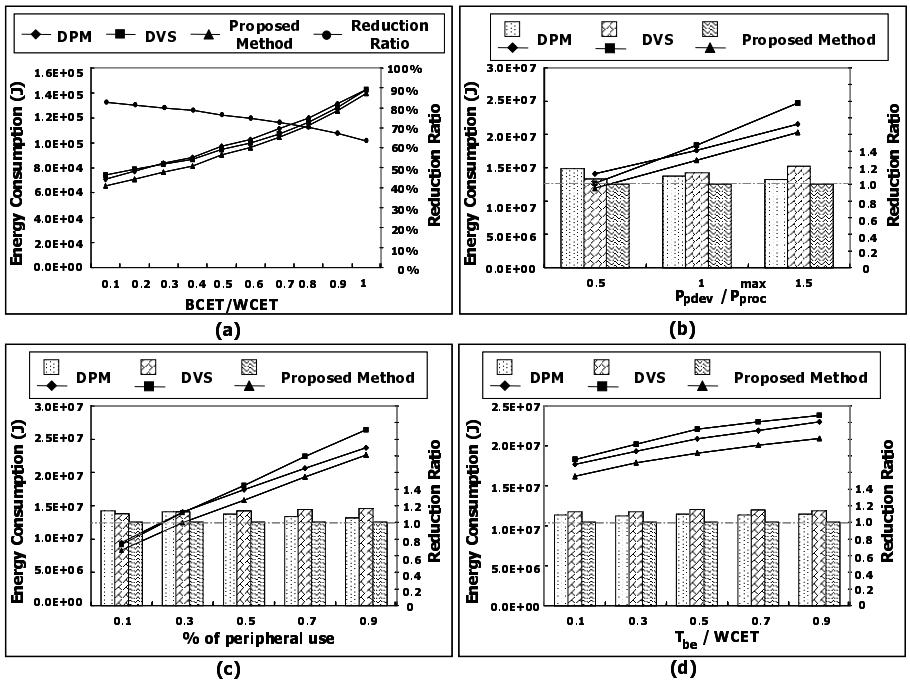

Figure 11. Simulation results of multitask environment with CNC controller application, where device request probability of each time slot is assumed to be 0.5 except for the third set of experiments; (a) energy consumption and energy reduction ratio on varying $\frac{BCET}{WCET}$, (b) on varying $\frac{P_{pdev}}{P_{proc}^{max}}$, (c) on varying the device usage probability, (d) on varying the device break-even time.

Figure 10(b) and Figure 11(b) demonstrate the energy consumption on varying $\frac{P_{pdev}}{P_{proc}^{max}}$ to 0.5, 1.0, and 1.5. Other parameters are set as follows: $\left\{\frac{BCET}{WCET}, \frac{T_{be}}{WCET}\right\} = \{0.1, 0.1\}$. The lines indicate energy consumption and the histogram illustrates the energy reduction ratio compared with proposed method. Both in single task and multitask environment there is about 10% performance improvement. More specifically, in the case of single task, 9.9% energy reduction compared with DPM scheme and 9.1% energy reduction compared with DVS scheme on average. In the case of multitask experiment there is 10.1 % and 11.9% energy reduction, respectively. The performance improvements could grow significantly if our comparison was made with pure DVS or DPM scheme. As expected, the DVS scheme shows the worst result as the ratio increases since it underestimates the device power in nature.

The third set of experiments (Figure 10(c) and Figure 11(c)) shows the relation between the energy consumption and the device usage frequency. Other parameters are set as $\left\{\frac{P_{pdev}}{P_{proc}^{max}}, \frac{BCET}{WCET}, \frac{T_{be}}{WCET}\right\} = \{1, 0.1, 0.1\}$. As the device usage probability increases, the DVS scheme degrades most while the DPM scheme converges to the proposed technique. The proposed method outperforms 7.0% and 14.7% compared with the DPM scheme and the DVS scheme, respectively. The last set of experiments (Figure 10(d) and Figure 11(d)) shows the relation between the energy consumption and the ratio $\frac{T_{be}}{WCET}$ with $\left\{\frac{P_{pdev}}{P_{proc}^{max}}, \frac{BCET}{WCET}\right\} = \{1, 0.1\}$. In the case of multitask environment, we take the ratio $\frac{T_{be}}{average \text{ of } WCETs}$ because WCETs of each tasks are different. As the ratio increases, the DVS scheme degrades most and the DPM scheme converges to the proposed approach. As shown in Figure 10 and Figure 11, the experimental results show similar trends in all cases.

In all the experiments, it is shown that the proposed technique outperforms the existent DPM and DVS schemes even though we made conservative comparison. The simulation results will be also dependent on the statistics of the execution time of the task. More extensive simulations with other statistical models is reserved for future work.

6. EXTENSIONS AND FUTURE WORK

In this paper, our discussion is limited to real-time fixed priority preemptive scheduling. But it can be extended to non real-time adaptive scheduling environment. In that case, we use the predicted value for the next device use time.

Trade-offs between the DPM scheme and the DVS schemes open a wide design space for future research subjects, including the following:

(1) Consideration of the multiple peripheral devices; the optimal technique is unlikely to be expected in real-time. Then, heuristics should be searched for.

(2) Consideration of the multiple power modes for peripherals as well as for processors.

(3) Consideration of voltage scalable peripheral devices.

(4) Accurate peripheral power models; Since the peripheral power is also dependent on the operating frequency, the simple model needs refinements for more accurate analysis.

(5) Consideration of the actual operating duration of the peripheral device in case that power consumption in active mode depends on whether it is actually operating or waiting for request. This is similar to the problem of multiple power modes.

(6) Consideration of the autonomous power manager for peripheral devices which is general power management scheme for peripheral devices.

7. CONCLUSION

In this paper, we proposed a new dynamic power management technique to minimize the system energy assuming that the system consists of a voltage scalable processor and power controllable peripheral devices. We observed that there is a significant tradeoff between the DPM scheme and the DVS scheme over the wide range of system operating conditions. The proposed hybrid scheme fully exploits workload-variation slack time by partitioning the task into several timeslots and shut down the unneeded peripheral device on a timeslot-by-timeslot basis. Our technique integrates both DPM and DVS schemes to make an optimal supply voltage at each time slot. Extensive simulation shows the novelty and usefulness of the proposed technique. We believe that there are many research topics to be pursued in this line of research: the system energy should be considered when choosing the optimal power management policy.

8. ACKNOWLEDGEMENTS

This work is supported by Brain Korea 21 program.

9. REFERENCES

[1] http://www.microsoft.com/hwdev/onnow/

[2] http://www.teleport.com/~acpi

[3] L. Benini, A. Bogliolo, and G. De Micheli, “A Survey of Design Techniques for System-level Dynamic Power Management”, *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 8, No. 3, pp. 299-316, June, 2000.

[4] Mani B. Srivastava, Anantha P. Chandrakasan, and Robert W. Brodersen, “Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation”, *IEEE Transactions on VLSI Systems*, Vol. 4, NO.1, Mar. 1996.

[5] N. Kim, M.Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin., “Visual assessment of a real-time system design: a case study on a CNC controller”, *Proc. of IEEE Real-Time Systems Symposium*, Dec. 1996.

[6] Seongsoo Lee, and Takayasu Sakurai, “Run-time voltage Hopping for Low-power Real-time Systems”, *Proc. of Design Automation Conference*, 2000.

[7] Thomas D. Burd and Robert W. Brodersen, “Design Issues for Dynamic Voltage Scaling”, *Proc. of International Symposium on Low Power Electronics and Design*, Jul. 2000.

[8] Thomas D. Burd, Trevor A. Pering, Anthony J. Stratakos, and Robert W. Brodersen, “A Dynamic Voltage Scaled Microprocessor System”, *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 294-295, Feb. 2000

[9] Thomas D. Burd and Robert W. Brodersen, “Processor Design for Portable Systems”, *IEEE Transactions on VLSI Systems* 1996.

[10] Trevor Pering, Tom Burd, and Robert Brodersen, “The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms”, *Proc. of I International Symposium on Low Power Electronics and Design*, Aug. 1998.

[11] Youngsoo Shin, and Kiyoung Choi, “Power-Conscious Fixed Priority Scheduling for Hard Real-Time Systems”, *Proc. of Design Automation Conference*, 1999.

[12] Yung-Hsiang Lu, Eui-Young Chung, Tajana Simunic, Luca Benini, and Giovanni De Micheli, ”Quantitative Comparison of Power Management Algorithms”, *Proc. of Design Automation and Test In Europe*, pp. 20-26, 2000.

[13] Yung-Hsiang Lu, Luca Benini, and Giovanni De Micheli, “Operating-System Directed Power Reduction”, *Proc. of International Symposium on Low Power Electronics and Design*, 2000.

[14] Yung-Hsiang Lu, Luca Benini, and Giovanni De Micheli, “Request-Aware Power Reduction”, *Proc. of International Symposium on System Synthesis*, Sept. 2000.

[15] Won Namgoong, Mengchen Yu, Teresa Meng., “A high-efficiency variable-voltage CMOS dynamic dc-dc switching regulator Low-Power CMOS Digital Design”, *Proceedings of IEEE International Solid-State Circuits Conference*, 1997.