Power Reduction of Chip Multi-Processors using Shared Resource Control Cooperating with DVFS

Ryo Watanabe, Masaaki Kondo, Hiroshi Nakamura, Takashi Nanya
Research Center for Advanced Science and Technology,
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, Japan
{watanabe,kondo,nakamura,nanya}@hal.rcast.u-tokyo.ac.jp

Abstract

This paper presents a novel power reduction method for chip multi-processors (CMPs) under real-time constraints. While the power consumption of processing units (PUs) on CMPs can be reduced without violating real-time constraints by dynamic voltage and frequency scaling (DVFS), the clock frequency of each PU cannot be determined independently because of the performance impact caused by the conflict for the shared resources. To minimize power consumption in this situation, we first derive an analytical model which provides the optimal priority and clock frequency setting, and then propose a method of controlling the priority of shared resource accesses in cooperation with DVFS. From the analytical model, in dual-core CMPs, we reveal that the total power consumption is minimized when the clock frequency of two PUs becomes the same. An experiment with a synthetic benchmark supports the validity of the analytical model and the evaluation results with real applications show that the proposed method reduces the power consumption by up to 15% and 6.7% on average compared with a conventional DVFS technique.

1. Introduction

To reduce power consumption, and thus reduce energy consumption, dynamic voltage and frequency scaling (DVFS) is widely used, especially in real-time systems. By lowering the clock frequency and supply voltage, the power consumption of a processor is reduced while satisfying the real-time constraint.

Recently, the single chip multiprocessor (CMP) has become an attractive architecture. With multiple processors, we can achieve high throughput without increasing clock frequency; thereby, CMP offers higher power-performance efficiency compared to that of single processor systems.

In CMPs, multiple PUs are often used to improve the performance of multi-program workloads. When PUs execute several independent applications which have different real-time constraints, the suitable clock frequency and voltage for each application should be selected to reduce power consumption, if each PU can choose its own clock frequency and supply voltage independently of other PUs. However, the clock frequency of PUs cannot be determined independently, even if independent programs are executed. This is because of the existence of the shared resources conflict.

In order to utilize limited resources efficiently, PUs on CMPs usually share some hardware components such as the L2 cache, memory bus and main memory. If two or more PUs try to use such shared resources simultaneously, all but one of the PUs have to wait until the ongoing access finishes. The performance penalty (waiting time) due to the conflict heavily depends on the access patterns of running programs. As the performance penalty increases, PUs must use a higher clock frequency and supply voltage to compensate for the performance loss and to keep satisfying the performance constraint. Thus, the power consumption of PUs also heavily depends on the occurrence of the conflict in shared resources. This implies that the power consumption of PUs is altered if we control the effect of the performance penalty caused by the conflict.

In this paper, we propose a method to minimize the power consumption of CMPs by priority control in accessing shared resources. The performance penalty caused by the resource conflict can be altered by the priority control. Suppose that we have two PUs (denoted as PU$_A$ and PU$_B$) and the shared resource accesses from PU$_A$ have a higher priority than those from PU$_B$. Then, the waiting time due to the resource conflict of PU$_A$ decreases, and consequently, PU$_A$ can use a lower clock frequency, which leads to lower power consumption of PU$_A$. On the other hand, the waiting time of PU$_B$ increases, and therefore the clock frequency of PU$_B$ must be increased to satisfy the performance constraint. Clearly, the power consumption of PU$_B$

increases. There should be an optimal priority setting which minimizes the total power consumption, and the proposed method tries to find the optimal priority setting as well as the clock frequency for each PU.

This paper makes the following contributions.

- First, we derive the condition in which the total power consumption becomes minimal by constructing a model which analyzes the performance and power impact caused by the shared resource conflict and priority control.
- Second, we propose a method which cooperatively optimizes the access priority and the clock frequency of each PU based on a proposed model. This method can be easily implemented in conventional CMPs. In this paper, we assumes that the priority control for shared resource accesses is realized by a priority queue.

The rest of this paper is organized as follows. The next section briefly presents the strategy for power reduction using the access priority control. Section 3 clarifies the condition of power minimization with an analytical model. In Section 4, we propose a cooperative priority and clock frequency control method and in Section 5 we show some experimental results. We describe related work in Section 6 and conclude in Section 7.

2. Strategy for Power Reduction

2.1. Problem Settings

We investigate a power minimization problem in a typical dual-core CMP which concurrently executes two programs with real-time constraints. We suppose that the target CMP consists of two identical PUs, PU_0 and PU_1. Each PU has dedicated L1 and L2 caches and can operate with an individual clock frequency and supply voltage denoted as f_i and V_i, respectively. Here, the subscript i indicates the index of the PUs. All the PUs access the main memory through a shared memory bus.

PU_i is supposed to execute an application task $\mathrm{T_i}$. As mentioned in the previous section, application tasks on different PUs are independent and $\mathrm{T_i}$ has its own real-time constraint (or latency constraint) which is denoted as L_i. This means that the execution of one iteration of $\mathrm{T_i}$ must be completed within time L_i. Note that parallel tasks are not considered in this paper.

2.2. Strategy

The key strategies for power minimization are as follows: 1) deriving the optimal priority setting which minimizes the total power consumption by constructing a performance and power model, 2) controlling the priority of the shared resource access in order to alter the power consumption of each PU. We explain the effect of the priority control with the example shown in Figure 1. The figure shows the relationship between the priority setting and power consumption of the PUs. This graph is derived by simulating the formulated model described in Section 3.

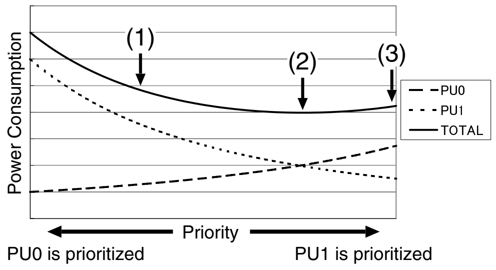


Figure 1. Effect of the priority control.

In Figure 1, the latency constraint of T_1 is supposed to be tighter than that of PU_0. Therefore, PU_1 has to operate with a higher clock frequency than PU_0 to satisfy the constraint. Consequently, the power consumption of PU_1 is larger than that of PU_0 ((1) in Figure 1).

As stated in Section 1, the clock frequency and supply voltage required to meet the constraint can be altered by priority control of the shared resource. When the access from PU_1 is prioritized, PU_1 can decrease its clock frequency because of the shorter waiting time. On the other hand, PU_0 has to increase its clock frequency to compensate for the increasing performance penalty.

The power consumption of the PUs increases quadratically as the clock frequency increases [6]. Therefore, the reduction of power in PU_1 (which uses a higher clock frequency) is larger than the increase of power in PU_0 (which uses a lower clock frequency). For this reason, total power consumption of the CMP decreases with the prioritizing of accesses from PU_1. This trend continues until the power reduction of PU_1 becomes equal to the power increase of PU_0 ((2) in Figure 1). As will be discussed in Section 3, this is the point at which the clock frequency of PU_0 is equal to that of PU_1.

When the access priority of PU_1 is increased beyond this point, the trend of the total power consumption begins to increase ((3) in Figure 1). This is because the increase in power for PU_0 becomes larger than the reduction of power for PU_1 as the clock frequency of PU_0 becomes larger than that of PU_1. Therefore, the total power consumption is a convex function of the access priority; in addition, the total power consumption is minimized when the clock frequency of both PUs becomes the same.

In this paper, we assume the memory bus is the only shared resource of the target CMP. However, the strategy presented in this section is applicable to CMPs which have other shared resources. The essence of the strategy is that the power consumption of the PUs is altered by means of the priority control so that the total power is minimized.

3. Conflict Modeling

This section presents a model for analyzing the performance and power impact of the access priority control on a shared memory bus. From the model, we can obtain the exact priority setting that minimizes the total power consumption of a CMP.

3.1. Parameters and Variables

As stated in Section 2, we use the variables L_i, f_i, and V_i for the latency constraint, clock frequency, and supply voltage for each PU, respectively. In addition to these variables, we introduce a parameter which indicates the performance of the shared memory bus.

- l_B : Bus access latency (time for transferring data from main memory to L2 cache)

We also introduce some parameters for representing the task properties, as follows.

- I_i : Number of instructions for $\mathrm{T_i}$
- m_i : Number of L2 misses for $\mathrm{T_i}$
- s_i : Length of stall time in PU_i due to L2 cache misses when executing $\mathrm{T_i}$

These parameters are assumed to be provided by techniques such as the static analysis of the execution profiles of programs, on-line estimation, and so on. Note that the analytical model discussed later does not depend on how these parameters are extracted.

Using the above assumptions, the effective working time of each PU is given as follows.

$$t_i = L_i - s_i \tag{1}$$

The effective working time means the length of time that a PU really performs useful computations. Thus, PU_i has to complete execution of I_i instructions within time t_i. The processing speed of PU_i required to keep the deadline depends on I_i and t_i. Since it has already been shown that theoretically the unique power-optimal clock frequency and voltage level can be determined [8], the power-optimal clock frequency is given as the following formula using a task dependent constant c.

$$f_i = c\frac{I_i}{t_i} \tag{2}$$

The relationship between the supply voltage and clock frequency is approximated by equation (3).

$$V_i = af_i + b \tag{3}$$

Table 1. Relationship between clock frequency and supply voltage.

Processor	Max./Min. Clock Freq. [GHz]	Regression Formula V[V] = a f[GHz]+b	Coefficient of Determination R^2
Pentium M [10]	1.6 / 0.6	V = 0.558 f + 0.609	0.9920
Pentium M [2]	2.1 / 0.6	V = 0.237 f + 0.848	0.9997
Turion 64 [1]	2.0 / 0.8	V = 0.250 f + 0.700	1.0000

This linear relationship is observed in many commercial processors. Table 1 shows some examples of the relationship. Using regression analysis, the relationship between the clock frequency and supply voltage is almost a linear function, as shown in the third column in Table 1. Because the coefficient of determination (the fourth column in the table) is very high, this approximation seems to be fully reasonable.

Then, using system dependent constant k, the energy consumption per instruction for $\mathrm{PU_i}$, denoted as e_i, is derived as follows [6].

$$e_i = kV_i^2 \tag{4}$$

Finally, the average power consumption of PU_i is presented below.

$$P_i = \frac{I_i}{L_i}e_i = \frac{kI_iV_i^2}{L_i} \tag{5}$$

3.2. Impact of Access Conflicts

In this subsection, we formulate the power and performance impact of memory bus conflicts when $\mathrm{T_0}$ and $\mathrm{T_1}$ are executed simultaneously. During the execution of $\mathrm{T_i}$, the length of time that PU_i occupies the bus is given by m_il_B. If L2 misses have uniform distribution, the probability that $\mathrm{PU_1}$ occupies the bus when $\mathrm{PU_0}$ tries to access it is as follows.

$$p_0 = \frac{m_1l_B}{L_1} \tag{6}$$

This also indicates the probability of occurrence of $\mathrm{PU_0}$'s stall due to the bus conflict. Once the conflict occurs, the data transfer of $\mathrm{PU_0}$ on the bus must wait for completion of the ongoing data transfer requested by $\mathrm{PU_1}$. The expectation of the time length that $\mathrm{PU_0}$ waits for the bus to be available is shown in equation (7).

$$E_w = \frac{1}{2}l_B \tag{7}$$

From (6) and (7), the expectation of additional stall time per L2 cache miss for $\mathrm{PU_0}$ is given as P_0E_w. Thus, the effective working time of $\mathrm{PU_0}$ decreases to the value shown in equation (8).

$$t'_0 = t_0 - m_0p_0E_w = t_0 - m_0\frac{m_1l_B^2}{2L_1} \tag{8}$$

The power-optimal clock frequency and supply voltage can be obtained in the same way as stated in the previous section, and is described as follows.

$$f'_0 = c\frac{I_0}{t'_0},\ V'_0 = af'_0 + b,\ e'_0 = kV'^2_0 \tag{9}$$

The same procedure can also be applied for PU_1 to derive t'_1, f'_1, V'_1 and e'_1. Thus, the average power consumption of PU_i, including the impact of the shared bus conflict, is given as follows.

$$P'_i = \frac{I_i}{L_i}e'_i = \frac{kI_iV'^2_i}{L_i} \tag{10}$$

3.3. Power Reduction by Priority Control

In this subsection, we formulate the effect of the access priority control.

3.3.1 Distributing the Performance Penalty

From equation (8), the total extra stall time caused by the conflict per unit of time for all PUs can be derived as follows.

$$l_{total} = \frac{1}{L_0}\frac{m_0m_1l_B^2}{2L_1} + \frac{1}{L_1}\frac{m_0m_1l_B^2}{2L_0} = \frac{m_0m_1l_B^2}{L_0L_1} \tag{11}$$

Here, we suppose the system has an access priority controller. When an access conflict happens, the controller chooses one request among several requests from PU_0 and PU_1 based on the priority setting. Note that we cannot reduce the total amount of the performance penalty l_{total} even with the priority control. The priority control can only alter the allotment of the performance penalties among PUs.

Suppose that the priority controller allocates the penalty so that the ratio of the penalty for each PU is $\mathrm{PU}_0 : \mathrm{PU}_1 = r : (1-r)$ $(0 \le r \le 1)$. In this case, the effective working time of each PU is expressed as the following formulas.

$$t'_0 = t_0 - L_0rl_{total} \tag{12}$$

$$t'_1 = t_1 - L_1(1-r)l_{total} \tag{13}$$

From these formulas, it is shown that the total power consumption of the CMP, expressed as $P_{total}(= P'_0 + P'_1)$, is a function of the variable r.

3.3.2 Power Optimization

Here, we derive a condition in which the total power consumption of the CMP is minimized.

When the supply voltage V is represented as a linear function of the clock frequency, as in equation (3), V^2 is a convex polynomial function of f and is also a convex function of r (from equations (9), (12) and (13)). Hence, P_{total} is also a convex function of r and has a global minimum, as shown in Figure 1.

Using the derivative constants c_2, c_1, c_0 and equation (9), we express e'_i as follows.

$$e'_i = c_2\left(\frac{I_i}{t'_i}\right)^2 + c_1\left(\frac{I_i}{t'_i}\right) + c_0 \tag{14}$$

The convex function becomes the minimum when the derivative is equal to zero. Here, the following formula for the derivative of P_{total} holds.

$$\begin{aligned}\frac{d}{dr}P_{total} &= \frac{I_0}{L_0}\frac{d}{dr}e'_0 + \frac{I_1}{L_1}\frac{d}{dr}e'_1 \\ &= l_{total}\left\{2c_2\left(\frac{I_0}{t'_0}\right)^3 + c_1\left(\frac{I_0}{t'_0}\right)^2\right. \\ &\qquad \left. - 2c_2\left(\frac{I_1}{t'_1}\right)^3 - c_1\left(\frac{I_1}{t'_1}\right)^2\right\}\end{aligned} \tag{15}$$

This immediately leads to the result of the condition for minimizing the total power consumption, which is represented as follows.

$$\frac{I_0}{t'_0} = \frac{I_1}{t'_1} \quad \Rightarrow \quad f'_0 = f'_1 \tag{16}$$

This formula indicates that the total power consumption of the CMP is minimized by controlling the access priority so that the each clock frequency of each PU becomes equal. Therefore, the value of r which minimizes the total power consumption of the CMP is derived as follows.

$$r_{min} = \frac{I_1t_0 - I_0t_1 + I_0L_1l_{total}}{(I_0L_1 + I_1L_0)l_{total}} \tag{17}$$

In some cases, r_{min}, given by equation (17), can be more than 1 or less than 0. Even in this case, the power is minimized by making the clock frequency of the PUs close to each other. Therefore, we use 1 in the case of $r_{min} > 1$, and 0 in the case of $r_{min} < 0$.

3.4. Case for CMPs with More PUs

Although we focus on a dual-core CMP in this paper, the model presented in this section can be easily extended to CMPs with more PUs. We briefly describe how to extend the model.

Suppose N PUs ($\mathrm{PU}_0, \mathrm{PU}_1, \ldots, \mathrm{PU}_{N-1}$) share the memory bus. The probability of p_i (the probability that access from PU_i is delayed due to the bus conflict) is given as the probability that any of the other PUs occupies the bus at that moment. This is shown in equation (18). From this equation, the power and performance impact caused by the access conflict can be derived in the same way as in the case of two PUs.

$$p_i = \sum_{j \ne i}\frac{m_jl_B}{L_j} \tag{18}$$

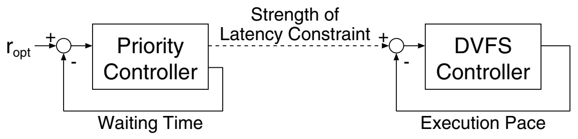


Figure 2. Block diagram of the cooperative priority and DVFS control method.

Here, l_{total} can also be expressed using equation (18,) and it is a constant even with access priority control. Therefore, power minimization is a problem for allocating the performance penalty among N PUs.

Suppose that the ratio of the penalty is $\mathrm{PU_0} : \mathrm{PU_1} : \ldots : \mathrm{PU_{N-2}} : \mathrm{PU_{N-1}} = r_0 : r_1 : \ldots : r_{N-2} : (1 - r_0 - r_1 - \ldots - r_{N-2})$ $(\forall i \ 0 \leq r_i \leq 1)$. In this situation, the partial derivative of P_{total} with respect to r_i is derived similarly to equation (15), because the power consumption of the PUs is constant except for that of PU_i and $\mathrm{PU_{N-1}}$. This leads to the conclusion that the total power consumption of PU_i and $\mathrm{PU_{N-1}}$ becomes a minimum when $f_i = f_{N-1}$.

By applying the above process for all i $(0 \leq i \leq N-2)$, the relationship $f_0 = f_1 = \ldots = f_{N-1}$ is derived as the condition for power minimization. Thus, even in CMPs with more than two PUs, the settings for priority control that enables power minimization is derived in the same way as in the case of two PUs.

4. Control Method

4.1. Overview

Figure 2 illustrates an overview of the proposed cooperative priority and clock frequency control method. The two cooperating feedback controllers in this method minimize the power consumption of the CMP based on the derived condition presented in Section 3.

The priority controller located on the left side of Figure 2 adjusts the priority for conflicting accesses so that the ratio of the penalty caused by the conflict for the PUs comes close to the power-optimal ratio (r_{min}:1-r_{min}), which is the input of the priority controller. The value of r_{min} is calculated by equation (17). The DVFS controller located on the right side of the figure adjusts the clock frequency and supply voltage of each PU so that the performance constraint of all the PUs is satisfied. The tightness of the latency constraint for each PU, which is the input to the controller, is derived as a result of the priority controller.

In the following sections, we describe the details of the priority control and the DVFS control for a dual-core CMP.

```
Parameter
  $TH_{NQ}$: threshold for changing $NQ$
  $NQ_{MAX}$ : the maximum value of $NQ$
Counters
  $C_i$ : the number of requests from PU$_i$ in the queue
  $W_i$ : waiting time due to conflicts for PU$_i$
Register
  $R$ : the identifier of the PU which currently uses the bus
Monitoring Routine (called every bus cycle)
  for ( $i = 0$ ; $i \leq 1$ ; $i + +$ ) {
    /* Is PU$_i$ waiting due to access from other PUs? */
    if ( $C_i > 0$ and $R \neq i$ ) $W_i$++ ;
  }
Feedback Routine (called at regular intervals)
  if ( $(1 - r_{min})W_0 > r_{min}W_1 \times TH_{NQ}$ ) {
    if ( $NQ < NQ_{MAX}$ ) $NQ$++ ; /* PU$_0$ waits too much */
  } else if (($(1 - r_{min})W_0 \times TH_{NQ} < r_{min}W_1$)
    if ( $|NQ| < NQ_{MAX}$ ) $NQ$– ; /* PU$_1$ waits too much */
  }
```

Figure 3. Algorithm of the priority control.

4.2. Priority Control

The priority control is based on access management with a priority queue. The queues are popular hardware used for managing accesses to shared resources. All of the access requests to the memory (and to the memory bus) due to L2 cache misses in any of the PUs are queued and processed one by one. To use a queue as a priority queue, we introduce an integer parameter NQ. If $NQ > 0$, a request from $\mathrm{PU_0}$ is granted before previously generated NQ requests from $\mathrm{PU_1}$. On the other hand, if $NQ < 0$, a request from $\mathrm{PU_1}$ is granted before previously generated $|NQ|$ requests from $\mathrm{PU_0}$. Otherwise, the requests in the queue are selected based on first-come-first-served (FCFS) manner.

Although we try to allocate the performance penalty to each PU by the priority control with the priority queue, the optimal ratio of the penalty is difficult to achieve because it is not directly handled by the priority control method mentioned above. Therefore, we also introduce a feedback control mechanism which updates the priority setting of the queue according to the observed performance penalty ratio as well as the target penalty ratio. The algorithm of the feedback priority control is shown in Figure 3. In this algorithm, the amount of waiting time due to conflict is monitored for each PU. Based on the actual performance penalty that is corrected for each PU, the new value of priority NQ is decided so that the actual penalty ratio becomes close to the target penalty ratio.

4.3. DVFS Control

As mentioned in Section 3.3.2, once the task parameters are given, the clock frequency which minimizes the total power consumption is derived from the model together with r_{min}. However, the assumptions used to derive the

```
Parameter
  $TH_D$: threshold of using lower clock frequency
Counters
  $Ir_i$ : the number of remaining instructions in $T_i$
  $Lr_i$ : remaining time to deadline of $T_i$
  $Ie_i$ : the number of instructions executed on PU$_i$
  $Le_i$ : elapsed execution time for PU$_i$
Feedback Routine (called at regular intervals)
  for ( $i = 0$ ; $i \leq 1$ ; $i++$ ) {
    if ( $Ir_i/Lr_i > Ie_i/Le_i$ ){
      /* current execution pace is too slow */
      Increase clock frequency level of PU$_i$ by 1 step;
      Reset $Ie_i$ and $Le_i$;
    } else if ( $Lr_i - Ir_iLe_i/Ie_i > TH_DLr_i$ ) {
      /* $T_i$ will be completed much before the deadline */
      Decrease clock frequency level of PU$_i$ by 1 step;
      Reset $Ie_i$ and $Le_i$;
    }
  }
```

Figure 4. Algorithm of the DVFS control.

model are slightly inapplicable to typical CMP systems for the following reasons: 1) the number of available clock frequency levels is limited, 2) the target r_{min} is sometimes not achievable because all the access conflicts are not controllable, even with the priority control method. Therefore, we propose the DVFS method, which also includes a feedback control mechanism. The algorithm is shown in Figure 4.

In the algorithm, the current processing speed is checked for each task at regular intervals and the current speed is compared with the target speed required to complete the task just within its deadline. Then the clock frequency for the next interval is decided.

5. Evaluation

We evaluated the proposed method to validate the proposed model and to show the effectiveness of the method. We used SimpleScalar Tool Set [3] as the base simulation environment. We extended SimpleScalar so that the assumed CMP architecture was evaluated. For estimating the power consumption, we used the Wattch [4] extension.

The hardware configuration and parameters for the proposed algorithm are shown in Table 2 and Table 3, respectively. The relationship between clock frequency and supply voltage was assumed to follow the linear approximation of that of Pentium M [10], as shown in Table 1.

5.1. Model Validation

We evaluated a synthetic application to verify the following two results derived by the model analysis: 1) the total power consumption is reduced by making the clock frequency of the PUs closer, 2) the power is minimized by controlling the access priority when the ratio of the performance penalty becomes PU$_0$:PU$_1$ = $r_{min} : (1 - r_{min})$. The

Table 2. Hardware configuration.

PU clock frequency	200 $\sim$ 1600 MHz
PU supply voltage	0.721 $\sim$ 1.502 V
Fetch, Issue, Commit width	4
Branch prediction	Combined bimodal (4K-entry) gshare (1K-entry), selector(1K-entry)
BTB	1,024 sets, 2-way
Mis-prediction penalty	5 cycles
Load / Store Queue size	16
Reorder buffer size	32
Functional units	(INT) 2-ALU, 1-mult/div (FP) 2-FPU, 2-mult/div (MEM) 1-port
L1 I-cache	32 KB, 32 B line, 2-way 1-cycle latency
L1 D-cache	32 KB, 32 B line, 2-way 2-cycle latency
L2 unified cache	256 KB, 64 B line, 4-way 10-cycle latency
Memory latency	100 ns
Bus width	8 B
Bus clock	200 MHz

Table 3. Algorithm parameters.

Priority feedback interval	1.87 us
DVFS feedback interval	187 us
NQ_{MAX}	3
TH_{NQ}	1.01
TH_D	0.05

synthetic application had a constant L2 miss rate throughout the execution, while the interval of consecutive L2 misses follows the Poisson distribution. In this experiment, we assumed DVFS with a very large number of available clock frequency levels. This is for making the condition almost the same as the hardware assumption used in the analytical model. We evaluated several patterns of the combination of L2 miss rates and latency constraints.

The evaluation results are shown in Table 4. For each evaluated case, we present the L2 miss rate, the average clock frequency without priority control (this indicates the strength of the latency constraint), r_{min} derived from the model, the actual waiting time ratio achieved by the control, the average clock frequency when priority control is applied, and the degree of power reduction achieved by the proposed method.

This result shows that the power consumption of the CMP is reduced by controlling the access priority. The degree of power reduction depends on the strength of the latency constraint and the L2 miss rate. As for case 1 and case 2 in the table, r_{min} cannot be achieved even by the priority control. This is due to the limitation of priority controllability of the queue-based method. However, even in such a situation, the proposed method can reduce the power consumption by reducing the difference in the clock frequency of each PU.

Figure 5 shows the changes in the power consumption in

Table 4. Results with a synthetic program for model validation.

Evaluated Case	L2 miss rate [%]		Avg. Freq. of PUs [MHz] only DVFS is applied		r_{min} derived from the model	ratio of W_0&W_1 achieved by the control	Avg. Freq. of PUs [MHz] priority control is applied		Power reduction [%]
	PU_0	PU_1	PU_0	PU_1			PU_0	PU_1	
1	1.76	0.91	810	1322	1.000	0.734	884	1226	3.5
2	1.76	1.76	800	1312	0.837	0.736	930	1038	8.4
3	1.76	2.71	802	1308	0.645	0.655	928	980	10.4
4	2.71	2.71	910	1258	0.594	0.598	1080	1070	6.6
5	1.76	3.36	802	1268	0.626	0.623	928	934	11.1

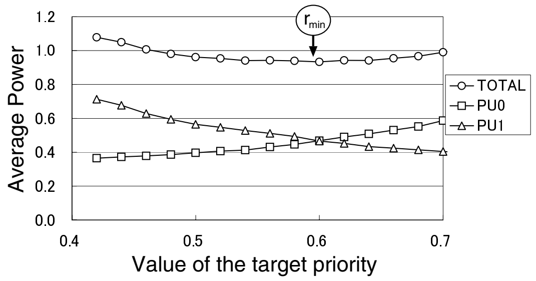
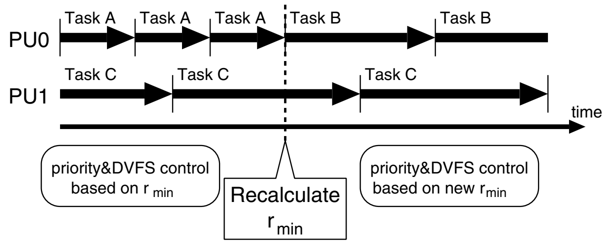


Figure 5. Relationship between power consumption and priority setting.



PU0
PU1
Task A
Task A
Task A
Task B
Task B
Task C
Task C
Task C
time
priority&DVFS control based on r min
Recalculate r min
priority&DVFS control based on new r min

Figure 6. Priority control when combination of executing tasks changes.

case 4 in Table 4 when the target value of the priority ratio is varied. The average power shown in the figure is normalized to the case that the proposed method is not used. The figure shows that the power consumption of the CMP is really a convex function of the variable r, and the value which minimizes the power is almost the same as r_{min}, given by the proposed analytical model.

5.2. Evaluation with Real Applications

We applied the proposed method to real applications. We used the H.264/AVC decoder reference software [7] along with several programs from SPEC CPU2000 benchmark [12]. In this experiment, we assumed DVFS with eight levels of clock frequency and supply voltage. The clock frequency ranged from 200 MHz to 1600 MHz with 200 MHz steps. We fast-forwarded the initialization part of each application and performed the simulation until 1.2 billion instructions of H.264 were completed. The constraint for PU_0 (H.264) was assumed to be tighter than that of PU_1.

In real applications, the L2 cache miss rate is not uniform throughout the execution. Therefore, we partitioned the target applications into small tasks so that the L2 cache miss rate could be assumed to be uniform within the tasks. In this case, the combinations of tasks which are simultaneously executed on the PUs change as the execution progresses. Therefore, r_{min} should be re-calculated whenever the combination changes, as shown in Figure 6).

The evaluation results are shown in Figure 7. This figure presents the power consumption of the CMP with the proposed priority and DVFS control methods relative to the CMP, to which only the DVFS algorithm was applied. We evaluated two cases of latency constraints for PU_1, which are expressed by the required instruction throughput (instructions per second: IPS) in the figure.

As seen from the figure, the proposed method reduces the power consumption of the CMP by up to 15% and 6.7% on average. The effect on the power reduction varies when the latency constraint is varied. For example, in H.264+art, the reduction of power in the case of IPS=147M, which indicates a tight constraint, is larger than the case of IPS=132M. This is because more bus conflicts occur when the latency constraint is tight. On the other hand, in H.264+mcf, the case of the looser latency constraint (IPS=89M) can save more power consumption compared with the tighter latency constraint (IPS=93M). Because the gap between the used clock frequency is large for a looser constraint, we have more potential to reduce power.

Figure 8 shows the utilized clock frequency for both PUs when H.264 and mcf (IPS=89M) are executed simultaneously. Figure 8-(1) shows the case when the priority control is not applied, and Figure 8-(2) shows the case when it is applied. Each point in the figure shows the average clock frequency for 200 ms time intervals. As seen from the figures, we see that the clock frequency of both PUs becomes closer by applying priority control. This is the expected behavior and this indicates that the proposed control method works very well.

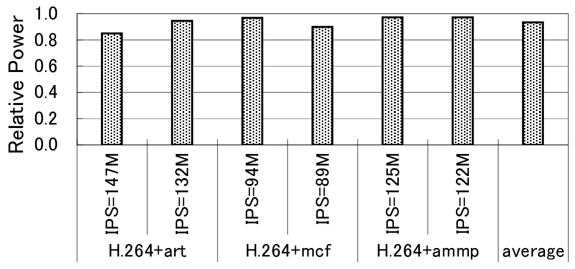


Figure 7. Results with real applications.

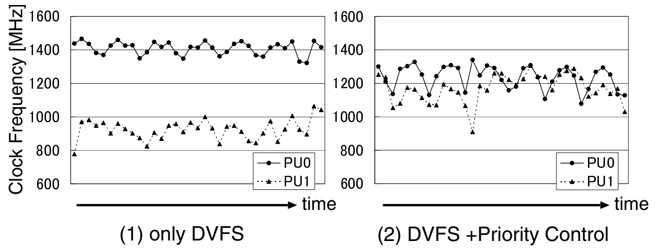


Figure 8. Used clock frequency throughout execution in H.264+mcf (IPS=89M).

6. Related Work

There has been a tremendous effort directed toward the DVFS technique to reduce power consumption. Many of these efforts have attempted to find the optimal choice of clock frequency and voltage level, as in [8], under a real-time constraint. Our work differs from these efforts since we use priority control in cooperation with the DVFS method.

There have been proposed many techniques to alleviate the performance impact caused by the resource conflicts. L2 cache partitioning algorithms have been presented to reduce the overall miss rate on a CMP [13] [9]. A performance model that predicts the impact of cache sharing is presented in [5]. In addition to share cache management, the problems with shared main memory resources is addressed in [11].

Resource conflicts waste power as well as degrade performance in CMPs. The proposed method in this paper provides power reduction by priority control and DVFS for real-time systems when resource conflicts occur. The method should be very useful for future CMP based real-time systems.

7. Conclusion

We have proposed a method of access priority control of shared resources in cooperation with the DVFS technique to reduce power consumption for CMPs. We derived an analytical model that provided the optimal priority and clock frequency, and described a feedback-based priority and clock frequency control method. The evaluation results reveal that the analytical model accurately provides the optimal priority, and the proposed cooperative priority and DVFS method can reduce power consumption compared with the conventional power reduction technique that uses only DVFS. Future work includes evaluating the proposed method in a CMP with three or more PUs and investigating the effectiveness of the method with a wide variety of real-time applications.

Acknowledgment

This work was supported in part by STARC (Semiconductor Technology Academic Research Center) and by Japan Science and Technology Agency as a CREST research program entitled "Innovative Power Control for Ultra Low-Power and High-Performance System LSIs."

References

[1] *AMD Athlon 64 Processor Power and Thermal Data Sheet*. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/30430.pdf.

[2] *Intel Pentium M Processor on 90nm Process with 2-MB L2 Cache Datasheet*. http://developer.intel.com/design/mobile/datashts/302189.htm.

[3] T. M. Austin, E. Larson, and D. Ernst. Simplescalar: An infrastructure for computer system modeling. *IEEE Computer*, 35(2):59–67, Feb. 2002.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *27th ISCA*, pages 83–94, June 2000.

[5] D. Chandra, F. Guo, S. Kim, and Y. Solihin. Predicting inter-thread cache contention on a chip multi-processor architecture. In *11th HPCA*, pages 340–351, 2005.

[6] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power CMOS digital design. *IEEE J. of Solid-State Circuits*, 27(4):473–484, Apr. 1992.

[7] H.264/AVC Software Coordination. *H.264/AVC reference software*. http://iphome.hhi.de/suehring/tml.

[8] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED 1998*, pages 197–202, Aug. 1998.

[9] S. Kim, D. Chandra, and Y. Solihin. Fair cache sharing and partitioning in a chip multiprocessor architecture. In *13th PACT*, pages 111–122, Oct. 2004.

[10] K. Krewell. Pentium M hits the street. *Microprocessor Report*, Mar. 2003.

[11] K. J. Nesbit, N. Aggarwal, J. Laudon, and J. E. Smith. Fair queuing memory systems. In *39th MICRO*, pages 208–222, Dec. 2006.

[12] Standard Performance Evaluation Corporation (SPEC). *SPEC CPU2000*. http://www.specbench.org.

[13] G. E. Suh, S. Devadas, and L. Rudolph. A new memory monitoring scheme for memory-aware scheduling and partitioning. In *8th HPCA*, pages 117–128, Feb. 2002.