An Integration of Imprecise Computation Model and Real-Time Voltage and Frequency Scaling

Keigo Mizotani, Yusuke Hatori, Yusuke Kumura, Masayoshi Takasu, Hiroyuki Chishiro, and Nobuyuki Yamasaki
Graduate School of Science and Technology, Keio University
3-14-1, Hiyoshi, Kohoku-ku Yokohama, Kanagawa, 223-8522, Japan
Email:{mizotani, hatori, yusuke, takasu, chishiro, yamasaki}@ny.ics.keio.ac.jp

Abstract

As microprocessor performance grows, high throughput and the power management are becoming more important on embedded real-time systems. Real-Time Voltage and Frequency Scaling (RT-VFS) has been proposed to reduce the power consumption and ensure real-time constraints. An imprecise computation model adds an optional part to Liu and Layland’s model to improve the quality of computations. This paper proposes the scheme to integrate an imprecise computation model and RT-VFS that can reduce the power consumption and improve the quality of computations within real-time constraints. Moreover, we implement this scheme on Dependable Responsive Multithreaded Processor (D-RMTP). D-RMTP is a prioritized simultaneous multithreaded processor for embedded real-time systems and D-RMTP system in a package supports RT-VFS. We implement the proposed scheme by use of D-RMTP original features to improve the quality of computations and reduce the energy consumption. Through experimental evaluation, we show that the proposed scheme satisfies both the lower energy consumption and higher performance on real environments. In our evaluation, the proposed scheme achieves a maximum of 135% improvement of the quality of computations per energy consumption.

keywords: Embedded Real-Time System, Power Consumption, RT-VFS, and Imprecise Computation Model

1 Introduction

With the improvement of microprocessor performance, embedded real-time systems require both the low power consumption and high performance. Voltage and Frequency Scaling (VFS) is known as a power management technique [2]. In real-time systems, the VFS scheme must ensure time constraints even if tasks run at a lower frequency. Thus, Real-Time VFS (RT-VFS) is important in embedded real-time systems that require the low power consumption. Moreover, an imprecise computation model is known as a technique to improve the quality of computations [3]. An imprecise computation model adds an optional part to Liu and Layland’s model [4] to improve the quality of computations. An optional part is no real-time task. When a task is ready to execute or executing its optional part and its deadline expires, its optional part is terminated to ensure its real-time constraint.

This paper proposes an integration of RT-VFS and an imprecise computation model as the scheme to satisfy both the low power consumption and high performance. This proposed scheme executes the optional part after scaling the supply voltage and processor frequency by RT-VFS. In an actual system, the configurable frequency is discrete, and hence the slack time remains even if RT-VFS scales the processor frequency. Our proposed scheme reduces the power consumption and utilizes this slack time to improve the quality of computations.

In this paper, we implement the proposed scheme on Dependable Responsive Multithreaded Processor (D-RMTP) [6]. D-RMTP System on a Chip (SoC) integrates many functions for embedded real-time systems. D-RMTP SoC supports the frequency scaling and the original features for embedded real-time systems. Moreover, D-RMTP System in a Package (SiP) supports the voltage scaling. In this way, D-RMTP SoC and SiP support RT-VFS. In experimental evaluation, this paper evaluates the proposed scheme on D-RMTP.

The remainder of this paper is organized as follows. Section 2 describes the background of this paper and Section 3 shows the proposed scheme of this paper. Section 4 describes the detail of D-RMTP and an implementation for D-RMTP. In Section 5, we evaluate the proposed scheme on D-RMTP. Finally, we conclude with a summary in Section 6.
2 Background

2.1 Imprecise Computation Model

The imprecise computation model decomposes each task into a mandatory part and an optional part. A mandatory part is required to complete by the deadline of each task. An optional part is executed after the mandatory part and improves the quality of mandatory part’s results. When a task is ready to execute or executing its optional part and its deadline expires, its optional part is terminated to ensure its real-time constraint. Figure 1 shows how tasks are executed on an imprecise computation model. In Figure 1, the upward arrow means the task release, the downward arrow means the deadline of the task and the instance of the task is called the job. In the second job and the third job, the optional part is terminated because its deadline expires.

Mandatory-First with Earliest Deadline (MFED) [1] is an imprecise computation real-time scheduling algorithm based on Earliest Deadline First (EDF) [4]. Each task is scheduled by EDF. However, the priority of each task during executing an optional part is lower than that during executing a mandatory part. Therefore, MFED is an optimal real-time scheduling algorithm on a single processor as with EDF. In this paper, we employ MFED as the imprecise computation scheduling algorithm.

2.2 Real-Time Voltage and Frequency Scaling

In recent years, an integrated circuit such as a processor is composed of CMOS. This power consumption $P_{\text{TOTAL}}$ is expressed in the following Equation (1):

$$P_{\text{TOTAL}} = P_{\text{SW}} + P_{\text{LEAK}},$$

where $P_{\text{SW}}$ is the switching power and $P_{\text{LEAK}}$ is the leakage power. The power consumption of processor is composed of these power. The switching power occurs by the switching of transistors on CMOS. This power is expressed in the following Equation (2):

$$P_{\text{SW}} = \alpha \times C \times V_{DD}^2 \times f,$$

where $\alpha$ is the activity ratio, $C$ is the load capacity, $V_{DD}$ is the supply voltage and $f$ is the processor frequency. The switching power is proportional to the square of the supply voltage. Moreover, it is proportional to the processor frequency. On the other hand, the leakage power occurs by the power supply. The leakage power is expressed in the following Equation (3):

$$P_{\text{LEAK}} = I_{\text{LEAK}} \times V_{DD},$$

where $I_{\text{LEAK}}$ is the leakage current. The leakage power is proportional to the supply voltage. Therefore, the power consumption of processor can be reduced by scaling down the processor frequency and supply voltage on the RT-VFS scheme.

RT-VFS is classified into RT-Static VFS (RT-SVFS) and RT-Dynamic VFS (RT-DVFS) (Figure 2). RT-SVFS sets the schedulable lowest processor frequency and supply voltage at a system initialization off-line. RT-DVFS scales the schedulable processor frequency and supply voltage while a system run on-line. RT-DVFS can reduce the power consumption more than RT-SVFS but RT-DVFS raises execution overhead because the voltage and frequency scaling is executed at each schedule event by utilizing the dynamic slack on RT-DVFS.

Cycle-Conserving RT-DVFS and Look-Ahead RT-DVFS [5] are the representative RT-DVFS schemes. These algorithms utilize the dynamic slack to scale the processor frequency. Cycle-Conserving RT-DVFS calculates dynamic slack by recalculating the utilization using the actual computing time consumed by tasks at each scheduling event. On the other hand, Look-Ahead RT-DVFS minimizes the current processor frequency by deferring task execution within its real-time constraint. In this paper, an optional part is executed by use of remaining slack. Therefore, Look-Ahead RT-DVFS which minimizes current processor frequency is unsuited for the integration with an imprecise computation model. In addition, Cycle-Conserving RT-DVFS is the more simple and practical algorithm. Thus, we employ Cycle-Conserving RT-DVFS as the RT-DVFS scheme.

3 Integration of Imprecise Computation Model and RT-VFS

3.1 System Model

This paper considers a preemptive real-time system on multiprocessors or simultaneous multithreaded (SMT) processors. The processor frequency $f =$
\{f_1, f_2, \ldots, f_l \mid f_1 < \cdots < f_l \}$ is discretely scaled. The supply voltage $V_k$ is scaled to the required value by $f_k$. In this paper, we implement the proposed scheme on the SMT processor. Each executing thread on a SMT processor is defined as a logical processor. \( \text{logical processors } P = \{ p_1, p_2, \ldots, p_m \} \) are supplied with the common voltage and frequency because all executing threads on a SMT processor share hardware resources. However, our proposed scheme can be extended easily to the independent voltage and frequency model. Task set $\mathcal{T} = \{ t_1, t_2, \ldots, t_n \}$ is composed of $n$ independent periodic tasks. Each task $t_i$ has the period $T_i$, the worst-case execution time (WCET) of the mandatory part $M_i$ and the WCET of the optional part $O_i$. These WCET are defined as a worst-case execution time at the highest frequency $f_l$. The relative deadline $D_i$ is equal to $T_i$. In addition, this paper considers partitioned scheduling that assigns tasks to processors off-line. At a system initialization, tasks are assigned to each processor by a partitioned algorithm. Each processor utilization $U_j$ is defined as $U_j = \sum_{t_i \in p_j} M_i / T_i$. The system utilization $U$ is defined as $U = \sum_j U_j$. Since an optional part is not a real-time task and can be terminated, the processor utilization does not include $O_i$.

3.2 Integration of MFED and RT-SVFS

First, we propose an integration of MFED and RT-SVFS (S-MFED). On S-MFED, the processor frequency is scaled by RT-SVFS at a system initialization and tasks are scheduled by MFED. At a system initialization, all tasks are assigned to each processor and the supply voltage and processor frequency is scaled to the lowest schedulable value by EDF on the processor with highest utilization. This processor frequency $f_{\text{min}}$ is expressed in the following Equation (4).

$$f_{\text{min}} = \min \{ f_k \in f \mid \frac{f_i}{f_k} U_{\text{max}} \leq 1 \} \quad (4)$$

$$U_{\text{max}} = \max_{1 \leq j \leq m} U_j$$

Figure 3 shows an example of S-MFED dispatch on two processors. The example uses the task set in Table 1 which indicates each task’s mandatory part and optional part WCET and period. $t_i$ is defined as task’s mandatory part and $t'_i$ is defined as task’s optional part. The example assumes that three normalized discrete frequencies are available from (0.25, 0.50, and 1.00). It uses Worst-Fit algorithm for partitioning the example task set. In the example task set, $t_1$ is assigned to $p_1$ and $t_2, t_3$ are assigned to $p_2$ by Worst-Fit. Thus, $U_1$ and $U_2$ are 0.375, and hence the selected frequency by Equation (4) is set to 0.50. As a result, mandatory parts of all tasks are schedulable on selected frequency. Moreover, the optional parts of $t_1$ and $t_2$ are executed by use of remaining idle time in Figure 3.

Figure 4 shows the task execution example scheduled by S-MFED on the same condition as shown in Figure 3 but using the actual execution time from Table 2. Comparing Figure 4 with Figure 3, the execution time of each optional part is increased because actual execution time is shorter than WCET. In this way, there is an opportunity to execute optional parts even if processor frequency is scaled down by RT-SVFS.

3.3 Integration of MFED and Cycle-Conserving RT-DVFS

Secondly, we propose an integration of MFED and Cycle-Conserving RT-DVFS (CC-MFED). On
CC-MFED, tasks are scheduled by MFED and the supply voltage and processor frequency is scaled by Cycle-Conserving RT-DVFS for EDF scheduling [5]. Whenever each scheduler event occurs, it recalculates the utilization of each processor and scales the lowest schedulable frequency based on the highest utilization of all processors.

Figure 5 shows the task execution example scheduled by CC-MFED on the same condition as shown in Figure 4. The example uses the task set in Table 1 and the actual execution time in Table 2. The remaining execution time of each task is set to WCET at each task release and measured at each task completion. The utilization of each processor is recalculated by use of this execution time. CC-MFED scales the processor frequency based on the highest processor utilization at each task release and completion. In this example, \( U_1 \) and \( U_2 \) are 0.375 and hence the selected frequency by Equation (4) is 0.50 at the system initialization. At 2 ms, \( U_1 \) is 0.125 and \( U_2 \) is 0.208. The highest processor utilization is \( U_2 \), and hence CC-MFED scales down the processor frequency to 0.25. At 8 ms, \( U_1 \) is 0.375 and \( U_2 \) is 0.146. The highest processor utilization is \( U_1 \), and hence CC-MFED scales up the processor frequency to 0.50.

CC-MFED achieves the lower power consumption than S-MFED but S-MFED executes the optional part of each task more than CC-MFED. This relation is the trade-off between energy consumption and the quality of execution result.

4 Dependable Responsive Multithreaded Processor

In this paper, we implement the proposed schemes on D-RMTP. D-RMTP is one version of Responsive Multithreaded Processor [7]. This section introduces the detail of D-RMTP and presents how to use the D-RMTP original features in our implementation.

4.1 Overview of D-RMTP

D-RMTP is a prioritized SMT processor for embedded real-time systems. It can execute up to 8 threads simultaneously with prioritizing each thread. In this paper, each executing thread on D-RMTP is defined as a logical processor. Table 3 shows the outline of D-RMTP. It has 32 KBytes of instruction and data cache and 64 KBytes of SRAM. In this paper, all programs are executed on SRAM.

We implement the partitioned MFED scheduler by use of the D-RMTP original feature. Threads which have the same priority are executed simultaneously on D-RMTP, and hence a resource contention among these threads delays the execution of each thread. Since a mandatory part is a real-time task and an optional part is no real-time task, the thread executing an optional part should not delay the thread executing a mandatory part simultaneously by a resource contention. Thus, on our implementation, a thread executing the mandatory part is set to a high priority and a thread executing the optional part is set to a low priority. In this way, a thread executing the mandatory part can use hardware resources preferentially.

The D-RMTP SoC integrates D-RMTP, SRAM and various I/Os. These modules are supplied with clock by the clock tree. Figure 6 shows the clock tree of D-RMTP SoC. This clock tree supports the frequency scaling by use of a variable frequency divider on each hardware module. Moreover, this clock tree supports clock gating by use of a gating logic on each hardware module.
### Table 3: Outline of D-RMTP

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Threads</td>
<td>8</td>
</tr>
<tr>
<td>Fetch Width</td>
<td>8</td>
</tr>
<tr>
<td>Issue Width</td>
<td>4</td>
</tr>
<tr>
<td>Integer Register 32bit</td>
<td></td>
</tr>
<tr>
<td>Floating Point Register 64bit</td>
<td></td>
</tr>
<tr>
<td>ALU 4 + 1 (Divider)</td>
<td></td>
</tr>
<tr>
<td>FPU 2 + 1 (Divider)</td>
<td></td>
</tr>
<tr>
<td>Branch Unit</td>
<td>2</td>
</tr>
<tr>
<td>Memory Access Unit</td>
<td>1</td>
</tr>
</tbody>
</table>

### Figure 6: Clock Tree of D-RMTP SoC

4.2 **Context Cache**

D-RMTP has the context cache which is an on-chip memory for thread contexts [7]. This context cache can save a maximum of 32 thread contexts. In general, a context switch requires saving and restoring thread contexts to main memory by software instructions. That consumes a large amount of clock cycles. Figure 7 illustrates the context cache implemented on D-RMTP. Bus connection between hardware contexts and the context cache is exclusive. It can transfer context data in 4 clock cycles.

When the execution of mandatory part is completed, MFED scheduler must save the thread context for the termination of executing the optional part at any point. We use this context cache for the general scheduling context switch and saving the context of mandatory part on MFED scheduler, and hence we can reduce the scheduler overhead.

4.3 **D-RMTP SiP**

The D-RMTP SiP integrates the D-RMTP SoC, DDR-SDRAM modules, a power supply module and a voltage sensor, etc. on a 30mm square package as shown in Figure 8. The D-RMTP SiP supports the voltage scaling by a DC-DC converter and a potentiometer. These devices can scale the processor supply voltage within the range of [0.8, 1.1] V by a serial peripheral interface. We implement RT-VFS by use of the frequency scaling in Section 4.1 and this voltage scaling.

5 **Experimental Evaluation**

5.1 **Evaluation Environment**

In experimental evaluation, we use a D-RMTP evaluation kit which has the D-RMTP SiP, an oscillator, I/O pins and FPGA, etc. as shown in Figure 9. Also, Figure 10 shows measurement environment. The energy consumption is precisely measured by synchronization of oscilloscope and logic analyzer. The sampling rate is 100 KSamples per second and the measurement time is 1,024 ms.

We evaluate S-MFED, CC-MFED and only-MFED on a single logical processor and two logical processors. Moreover, we measure the energy consumption of EDF on RT-SVFS (S-EDF) and Cycle-Conserving RT-DVFS (CC-EDF). By comparing MFED with EDF on the same RT-VFS scheme, we analyze the influence of executing optional parts in the energy consumption.

Table 4 shows the combination of the scaled processor frequency and supply voltage by RT-VFS on D-RMTP. In addition, this experimental evaluation applies clock gating to unused modules in Figure 6, specifically, DMA Controller, PCI, IEEE1394, Vector Unit, PWM...
Table 4: Combination of Frequency and Voltage

<table>
<thead>
<tr>
<th>Frequency [MHz]</th>
<th>Voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.10</td>
</tr>
<tr>
<td>25</td>
<td>0.88</td>
</tr>
<tr>
<td>12.5</td>
<td>0.80</td>
</tr>
</tbody>
</table>

This shows that D-RMTP is superior in the energy efficiency. S-MFED and CC-MFED achieve maximums of 57% and 58% reduction in the energy consumption against only-MFED, respectively. CC-MFED achieves the lower energy consumption than S-MFED within the system utilization 30% to 70% on a single logical processor and 60% to 120% on two logical processors. This reason is that CC-MFED gains more opportunity to scale down the processor frequency by measuring the actual execution time of each task. However, CC-MFED scales down the processor frequency and supply voltage on-line but the result of this scaling is same as that of S-MFED in the other range of system utilization. Therefore overhead of Cycle-Conserving algorithm causes the more energy consumption than S-MFED. On the other hand, the difference between S-MFED and S-EDF is a maximum of 19 mJ. Also, the difference between CC-MFED and CC-EDF is a maximum of 27 mJ. Thus, the execution of optional parts has a small influence on the energy consumption.

5.3 Quality of Execution Result

Figure 12 shows the quality of execution results on a single logical processor and two logical processors. This quality is defined as the ratio of the executed optional part. Each evaluation result is the arithmetic mean quality of each job in a task set. The difference with the trends on a single logical processor and on two logical processors is so small. This shows that the execution on two logical processor does not often cause the resource contention. S-MFED and CC-MFED execute less optional parts than only-MFED because of scaling down the processor frequency. Especially, the difference between CC-MFED and only-MFED is a maximum of 95%. CC-MFED prioritizes scaling down the processor frequency, and hence the optional part execution ratio is reduced to a greater degree. On the other hand,
the difference between S-MFED and only-MFED is a maximum of 25%. The optional part execution ratio on S-MFED is not reduced so much. This reason is that the actual execution time of each task was shorter than its WCET, and hence the optional part of each task could be executed for more time. In addition, S-MFED and CC-MFED are not monotonically decreased when the system utilization become high because of scaling the processor frequency discretely.

5.4 Trade-off between Energy Consumption and Quality of Execution Result

As discussed in Section 3, there is the trade-off between the energy consumption and the quality of execution result. We calculate the optional part execution ratio per energy consumption by use of evaluation results in Section 5.2 and Section 5.3 for analyzing this trade-off. Figure 13 shows this calculation result on a single logical processor and two logical processors. S-MFED and CC-MFED achieve maximums of 135% and 127% improvement in the optional part execution ratio per energy consumption against only-MFED, respectively. In particular, the proposed schemes achieve the higher improvement against only-MFED on the low system utilization. This reason is that there is the sufficient slack to reduce the energy consumption by RT-VFS and improve the quality of execution results by executing an optional part. S-MFED always achieves the higher optional part execution ratio per energy consumption than only-MFED. Therefore it can achieve both the lower energy consumption and higher performance. On the other hand, CC-MFED has lower the optional part execution ratio per energy consumption than only-MFED when the system utilization is higher than or equal to 30% on a single logical processor and 60% on two logical processors. However, as discussed in Section 5.2, CC-MFED achieves the lower energy consumption within the system utilization 30% to 70% on a single logical processor and 60% to 120% on two logical processors. When the energy consumption is regarded as important, CC-MFED can satisfy the requirement.

6 Conclusion

This paper proposed S-MFED and CC-MFED as an integration of imprecise computation model and RT-VFS. These proposed schemes satisfy both requirements, reduction of the energy consumption and improvement of the quality of computations within a real-time constraint. Moreover, we implemented these schemes on D-RMTP. We implemented the proposed schemes by use of D-RMTP original features to improve the quality of computations and reduce the energy consumption. Through experimental evaluation, we showed the proposed schemes satisfying both the lower energy consumption and higher performance on real environments. In our evaluation, the proposed schemes achieved a maximum of 135% improvement of the optional part execution ratio per energy consumption. For future work, we plan to evaluate the proposed schemes over two logical processors. Moreover, we plan to extend CC-MFED to be aware of the optional part execution ratio of each task.

Acknowledgements

This research was supported by JST, CREST for fundamental technologies for dependable VLSI systems.
Figure 12: Quality of Execution Result when $m = 1$ and $m = 2$

Figure 13: Relation between Energy Consumption and Quality of Execution Result when $m = 1$ and $m = 2$

References


