Extended U-Link Scheduling to Increase the Execution Efficiency for SMT Real-Time Systems

Shinpei Kato and Nobuyuki Yamasaki
School of Science for Open and Environmental Systems
Keio University
Yokohama, Japan
{shinpei,yamasaki}@ny.ics.keio.ac.jp

Abstract

This paper extends U-Link Scheduling to increase the average execution efficiency of the system. We first define the execution efficiency. Then we propose a new algorithm that establishes the co-scheduled sets where the execution efficiency can be increased. Also we present the static estimation of the execution time and provide the schedulability analysis for the extended U-Link Scheduling. In the experiments, we evaluate the advancement of the extended U-Link Scheduling from the viewpoint of the execution efficiency and real-time processing.

1 Introduction

Recent embedded systems require not only real-time processing but also high-throughput in the system. However, unlike workstations and personal computers, it is not straightforward to increase the processor frequency in the embedded systems, since power consumption and hardware resources must be concerned. Simultaneous multithreading (SMT) [4] is effective to such a circumstance, which increases throughput based on not the processor frequency but the program parallelism. Although SMT processors provide high-throughput with low power consumption and limited hardware resources, they have a disadvantage to real-time processing that the performance of each thread fluctuates due to resource competition among the multiple threads running simultaneously. As a result, the execution time of periodic tasks also fluctuates, which degrades the predictability of the system.

U-Link Scheduling [3] has been proposed to realize real-time processing for SMT processors. The objective of U-Link Scheduling is to mitigate the fluctuation of the execution time as much as possible. We concern the following two aspects about U-Link Scheduling in this paper. At first, the performance of U-Link Scheduling mainly depends on the algorithm which establishes the co-scheduled sets, since each task always runs with the same task members once the co-scheduled sets are established. RR-DUP was proposed in the previous work [3] as such an algorithm. RR-DUP, however, does not concern the compatibility among the tasks, hence there is a possibility that the execution efficiency of each thread degrades due to hardware resource competition. Secondly, the previous work [3] did not provide the static estimation of the worst-case execution time (WCET) for U-Link Scheduling. Therefore the schedulability analysis has not been done enough.

The contribution of this paper is to increase the average execution efficiency and to provide the schedulability analysis of U-Link Scheduling. We first describe the execution efficiency and static estimation of the execution time. Then we propose a new algorithm that establishes the co-scheduled sets where the average execution efficiency can be increased. Also we present task scheduling and its theorem. In the experiments, we show the advancement of the extended U-Link Scheduling.

2 Framework and Assumption

The SMT processor assumed in this paper has M logical processors (LPs), $LP_1 \sim LP_M$. There are L types of the functional units $\lambda_1 \sim \lambda_L$ such as ALU, FPU and so on. The number of each unit λ_l is expressed by $N(\lambda_l)$. The latency of λ_l is expressed by $D(\lambda_l)$. If a processor has four ALUs, two FPUs, two Branch Units(BU) and one Memory Access Unit(MAU), there are four types of functional units then $L = 4$. Let us assume $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are ALU, FPU, BU, MAU respectively. Then we acquire $N(\lambda_1) = 4$, $N(\lambda_2) = 2$, $N(\lambda_3) = 2$ and $N(\lambda_4) = 1$.

There are N periodic tasks $\tau_1 \sim \tau_N$ in task set τ and their periods are harmonic. Each task τ_i is described by tuple (C_i, T_i, I_i). C_i is WCET of τ_i in single-threading execution.

T_i is a period of τ_i and is equal to the relative deadline. I_i is a function which inputs a function unit and returns the number of instructions using that function unit. That is to say, $I_i(\lambda_l)$ means the number of instructions in τ_i using function unit λ_l. The utilization of τ_i in single-threading execution is defined as $U_i = \frac{C_i}{T_i}$.

The kth co-scheduled set is described by τ'_k and it owns $N(\tau'_k)$ of compounds $(1 \le N(\tau'_k) \le M)$. The ith compound in τ'_k is described by $\tau'_{k,i} = (C'_{k,i}, T'_{k,i})$. $T'_{k,i}$ is a period of $\tau'_{k,i}$ and is $T'_{k,i} = \min\{T_j | \tau_j \in \tau'_{k,i}\}$. $C'_{k,i}$ is the estimated execution time(EET) of $\tau'_{k,i}$ in SMT execution, which is described in Section 3.1. The axis task in τ'_k is defined by α_k. α_k is also described by $\tau'_{k,a}$ where a is the symbol of the index of α_k in τ'_k. The axis set is denoted by $\alpha = \{\alpha_1, \alpha_2, ...\}$.

3 Design and Analysis

3.1 Execution Efficiency and Execution Time

We define the execution efficiency in this paper. The execution efficiency of task τ_j which belongs to compound $\tau'_{k,i}$ is denoted by $E(\tau_j \in \tau'_{k,i})$. In order to acquire the execution efficiency, we define the relative instruction ratio(RIR) and the functional unit competition ratio(FCR), first of all.

RIR of $\tau_j \in \tau'_{k,i}$ to functional unit λ_l is denoted by $R_l(\tau_j \in \tau'_{k,i})$, which is the value that the ratio of instructions using λ_l in all the instructions of τ_j is normalized by the utilization of axis task α_k. $R_l(\tau_j \in \tau'_{k,i})$ is acquired by Equation (1). The summation of RIR of the tasks in compound $\tau'_{k,i}$ is the RIR of $\tau'_{k,i}$, which is described by $R_l(\tau'_{k,i})$ and acquired by Equation (2). If the ratio is not normalized by the utilization of the axis task, the value would be unfair among the tasks in a co-scheduled set, since the utilization of the normal task is considered to be that of the axis task.

$$R_l(\tau_j \in \tau'_{k,i}) = \frac{I_j(\lambda_l)}{\sum_{m=1}^{L} I_j(\lambda_m)} \times \frac{U_j}{U_{k,a}} \quad (1)$$

$$R_l(\tau'_{k,i}) = \sum_{\tau_j \in \tau'_{k,i}} R_l(\tau_j) \quad (2)$$

FCR of τ_j is denoted by $F_l(\tau_j \in \tau'_{k,i})$, which describes the inefficiency of using functional unit λ_l. It satisfies $F_l(\tau_j \in \tau'_{k,i}) \ge 1$ and $F_l(\tau_j \in \tau'_{k,i}) = 1$ means τ_j can fully utilize λ_l. FCR is acquired by Equation (3) where τ'_k is composed of M compounds and they are assumed to be sorted so that $R_l(\tau'_{k,1}) \le R_l(\tau'_{k,2}) \le ... \le R_l(\tau'_{k,M})$. Due to the limitation of space, A and B in Equation (3) are replaced as follows.

$$F_l(\tau_j \in \tau'_{k,i}) = \frac{\sum_{m=1}^{M}(A \times B)}{R_l(\tau_j)} \quad (3)$$

$$A = max\left\{1, \frac{M-m+1}{N(\lambda_l)}\right\}$$

$$B = \left[min\{R_l(\tau_j), R_l(\tau'_{k,m})\} - min\{R_l(\tau_j), R_l(\tau'_{k,m-1})\}\right]$$

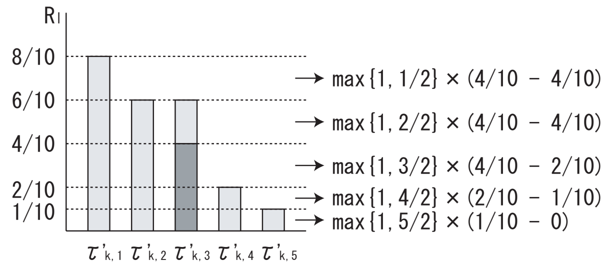

Figure 1. Unit competition ratio

A describes how much the functional unit is competed at a certain point. B describes how often competition of A occurs. For example, let us assume there are five compounds $\tau'_{k,1} \sim \tau'_{k,5}$ and their RIRs are $\frac{8}{10}$, $\frac{6}{10}$, $\frac{6}{10}$, $\frac{2}{10}$, $\frac{1}{10}$, respectively. Also let us assume $N(\lambda_l) = 2$ and RIR of τ_j that belongs to $\tau'_{k,3}$ is $R_l(\tau_j) = \frac{4}{10}$. In this case, Equation (3) calculates recursive processing as depicted in Figure 1. First, the five tasks compete the two of unit λ_l at the rate of $\frac{1}{10}$. Hence $A = \frac{5}{2}$ and $B = \frac{1}{10}$ at first. Then, by the same token, the four tasks compete the two of λ_l at the rate of $\frac{1}{10}$. Hence $A = 2$ and $B = \frac{1}{10}$. By this recursive processing, FCR of $\tau'_{k,3}$ to functional unit λ_l is finally $F_l(\tau_j \in \tau'_{k,3}) = (\frac{5}{2} \times \frac{1}{10} + \frac{4}{2} \times \frac{1}{10} + \frac{3}{2} \times \frac{2}{10} + \frac{2}{2} \times 0 + 1 \times 0) \div \frac{4}{10} = \frac{15}{8}$. Note that FCR assumes the worst case competition.

The execution efficiency of τ_j is expressed by Equation (4), using FCR and the latency of each functional unit.

$$E(\tau_j \in \tau'_{k,i}) = \frac{\sum_{l=1}^{L} D(\lambda_l)}{\sum_{l=1}^{L}\left\{F_l(\tau_j) \times D(\lambda_l)\right\}} \quad (4)$$

EET of τ_j in SMT execution, $\ddot{C}_j$, is calculated by Equation (5) based on WCET in single-threading execution and the execution efficiency acquired above. Also, the utilization in SMT execution, $\ddot{U}_j$ is expressed by Equation (6).

$$\ddot{C}_j = \frac{C_j}{E(\tau_j \in \tau'_{k,i})} \quad (5)$$

$$\ddot{U}_j = \frac{\ddot{C}_j}{T_j} \quad (6)$$

EET and the utilization of compound $\tau'_{k,i}$, $C'_{k,i}$ and $U'_{k,i}$, are calculated by Equation (7) and (8) respectively.

$$C'_{k,i} = \sum_{\tau_j \in \tau'_{k,i}} \{\ddot{U}_j \times T'_{k,i}\} \quad (7)$$

$$U'_{k,i} = \frac{C'_{k,i}}{T'_{k,i}} \quad (8)$$

Note that the following equation is satisfied.

$$U'_{k,i} = \sum_{\tau_j \in \tau'_{k,i}} \ddot{U}_j \quad (9)$$

```
Algorithm UL-FFDE
Input: Task set $\tau$
Output: Co-scheduled set $\tau'$ and axis set $\alpha$
Begin
 1 : Let $umax(\tau) = (\tau_i : U_i = \max\{U_j | \tau_j \in \tau\})$;
 2 : Let $etest(\tau'_{k,i}, \tau_j) = \bar{E}(\{\tau'_{k,l} | l < i\} \cup \{\tau'_{k,i} \cup \tau_j\})$;
 3 : Let $utest(\tau'_{k,i}, \tau'_{k,j}) = (U'_{k,i} : \tau'_k = \tau'_k \cup \tau'_{k,j})$;
 4 : $k \leftarrow 1$;
 5 : while $\tau \neq \emptyset$
 6 :     $\tau'_{k,1} \leftarrow umax(\tau)$;
 7 :     $\alpha_k \leftarrow \tau'_{k,1}$;
 8 :     $\tau = \tau - umax(\tau)$;
 9 :     for $2 \leq i \leq M$
10 :         sort $\{\tau_j \in \tau\}$ by $etest(\tau'_{k,i}, \tau_j) \geq etest(\tau'_{k,i}, \tau_{j+1})$;
11 :         for each $\tau_j \in \tau$
12 :             if $utest(\tau'_{k,i} \cup \tau_j, \tau'_{k,i} \cup \tau_j) \leq utest(\alpha_k, \tau'_{k,i} \cup \tau_j)$
13 :                 $\tau'_{k,i} = \tau'_{k,i} \cup \tau_j$;
14 :             end if
15 :         end for each
16 :     end for
17 :     $k \leftarrow k + 1$;
18 : end while
End
```

Figure 2. Algorithm UL-FFDE

The average execution efficiency of co-scheduled set τ'_k is denoted by $\bar{E}(\tau_k)$ and acquired by Equation (10).

$$\bar{E}(\tau'_k) = \frac{\sum_{\tau_j \in \tau'_k} C_j}{\sum_{\tau_j \in \tau'_k} \ddot{C}_j} \quad (10)$$

3.2 Co-Scheduled Set Establishment

In order to acquire the co-scheduled sets which maximize the execution efficiency, we need a complicated algorithm and its computation is NP-hard. Therefore we consider an algorithm that increases the execution efficiency sufficiently and does not require complicated calculation.

We propose U-Link First-Fit in Decreasing Efficiency (UL-FFDE), which builds the co-scheduled sets so that the average execution efficiency can be increased. The algorithm is described in Figure 2. It first chooses the task with the largest utilization in the remaining tasks as the axis task $\alpha_k (= \tau'_{k,1})$(line 6~7). This chosen axis task is removed from τ(line 8). Secondary it acquires normal tasks for the co-scheduled set where the chosen axis task is assigned. For each τ_j, it simulates the case that τ_j is assigned to compound $\tau'_{k,i}$ and sorts tasks in τ to result in high execution efficiency in the case(line 10). Then it conducts the following work in sequence from the head of the sorted task set. If $U'_{k,i}$ which is utilization of compound $\tau'_{k,i}$ including task τ_j in SMT execution does not exceed utilization of the axis task(line 12), τ_j becomes the element of compound $\tau'_{k,i}$(line 13). This process is similar to the First-Fit assignment. By repeating this process, it buids one co-scheduled set(line 9~16). Then, it moves the index to build the next co-scheduled set(line 17).

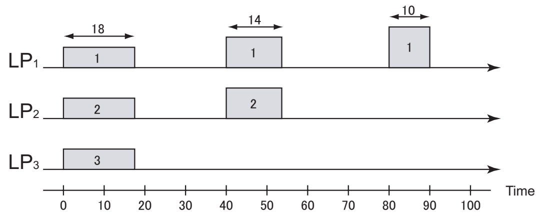

Figure 3. Fluctuation of execution time

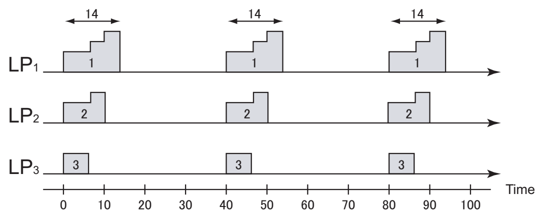

Figure 4. Pseudo task transformation

3.3 Task Scheduling and Its Analysis

In the case that utilizations of compounds in a co-scheduled set are not equal, the execution times of some tasks would fluctuate. Let us consider the case that the three tasks $\tau_1 = (10, 40, I_1)$, $\tau_2 = (20, 120, I_2)$, $\tau_3 = (10, 120, I_3)$ are assigned to the same co-scheduled set. In this case, the execution time of a task might fluctuate as τ_1 in Figure 3. The estimation method presented in Section 3.1 calculates the statistic average value. Therefore, EET becomes $\ddot{C}_1 = 14$ in the above example. However, the execution time is 18 in the first period of τ_1 and it is bigger than EET. It is obvious that it causes missing the deadline if the actual execution time is bigger than the estimation in several periods. In this paper, we introduce the notion of **peudo-task** to overcome this circumstance. The pseudo-task transformation assumes that any task in a co-scheduled set has the same period as the task which has a minimum period in the co-scheduled set. Note that the execution of the pseudo-tasks is equal to that of the original. For the previous example(Figure 3), when the pseudo-task transformation is applied to all the task, τ_2 and τ_3 come to have period 40. The original tasks τ_2 and τ_3 are composed of its three pseudo-tasks respectively. Then the schedule in Figure 3 is changed to that in Figure 4. In this schedule, all the tasks can run without the fluctuation of the execution time. When the pseudo-task transformation is applied to compound $\tau'_{k,i}$, the compound comes to have EET and a period expressed by Equation (11) and (12) respectively.

$$C'_{k,i} = C'_{k,i} \times \frac{\min\{T'_{k,j} | \tau'_{k,j} \in \tau'_k\}}{T'_{k,i}} \quad (11)$$

$$T'_{k,i} = \min\{T'_{k,j} | \tau'_{k,j} \in \tau'_k\} \quad (12)$$

In the previous work [3], U-Link Earliest Deadline First (UL-EDF) was proposed as a scheduling algorithm. This algorithm was renamed to UL-DEDF in the later work. In this paper, we modified this algorithm so as to use Rate Monotonic (RM), so called U-Link Double Rate Monotonic (UL-DRM), since we presume the harmonic task set. In addition, RM is relatively predictable and reasonable compared to EDF. Here we give the following theorem for UL-DRM. Due to the limitation of space, we do not show the algorithm and the proof of the theorem here.

Theorem 1. *The task set satisfying Equation* (13), (14) *and* (15) *is schedulable by UL-DRM.*

$$\sum_{\tau'_{k,i} \in \alpha} U'_{k,i} \leq 1 \quad (13)$$

$$\forall k, U'_{k,a} \geq \max\{U'_{k,i} | i \neq a\} \quad (14)$$

$$\forall k, T'_{k,a} \leq \min\{T'_{k,i} | i \neq a\} \quad (15)$$

The co-scheduled sets built by UL-FFDE meets the Equation (13) and (14). Equation (15) can be satisfied by the pseudo-task transformation.

4 Experiments

In this section, we evaluate the advancement of UL-DRM with UL-FFDE (UL-DRM-FFDE) by comparing to UL-DRM with RR-DUP (UL-DRM-DUP), RM-FF [2] and RM-US [1] in the point of the execution efficiency, deadline miss ratio and task rejection ratio. Since there is no method to estimate the execution time for RM-FF and RM-US in SMT execution, we apply EET acquired by UL-DRM-FFDE for them as well.

We made use of the same SMT processor and sample tasks (PID-INT, PID-FP and MEM) as the previous work [3]. We prepared two types of MEM, MEM1 and MEM2, where the computation amount of MEM2 is larger than that of MEM1. The periods of PID-INT and PID-FP are chosen from $\{100\mu s, 200\mu s, 400\mu s\}$. Meanwhile the periods of MEM1 and MEM2 are chosen from $\{2ms, 4ms, 8ms\}$. The execution time in single-threading execution were measured by the pre-experiments; PID-INT was $7.87\mu s$, PID-FP was $8.46\mu s$, MEM1 was $89.77\mu s$ and MEM2 was $180.12\mu s$. Therefore we set $10\mu s$ to PID-INT and PID-FP, $100\mu s$ to MEM1 and $200\mu s$ to MEM2 as EET.

Table 1. The number of instructions

	ALU	ALU D	FPU	FPU D	MAU	BU
PID-INT	39	1	0	0	12	1
PID-FP	15	0	24	1	15	1
MEM1	27	0	0	0	31	5
MEM2	49	0	0	0	62	10

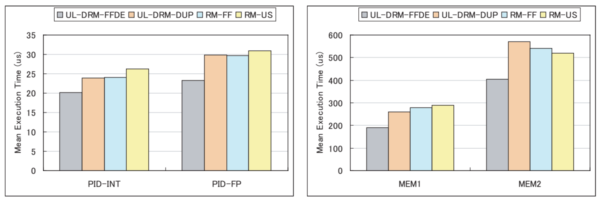

Figure 5. Execution time of the PID/MEM tasks

The number of instructions using each functional unit was counted in advance by disassembling as shown in Table 1. Here, ALU D and FPU D indicate the dividers. The latency of each functional unit is: {ALU, ALU D, FPU, FPU D, MAU, BU} = {1, 9, 3, 12, 30, 1}.

4.1 Execution Time Evaluation

Figure 5 shows the average execution times of PID-INT, PID-FP, MEM1 and MEM2 when the system load is 400%. The average execution time of PID-INT was $20.11\mu s$ in UL-DRM-FFDE, $23.89\mu s$ in UL-DRM-DUP, $23.99\mu s$ in RM-FF and $26.21\mu s$ in RM-US. From this result, there is little difference among these algorithms, though UL-DRM-FFDE slightly reduced the execution time compared to the others. We consider that is because the ALUs which PID-INT mainly utilizes are also utilized by other tasks such as PID-FP, MEM1 and MEM2. Namely there is little advancement to combine the compatible co-scheduled tasks to increase the execution efficiency. In addition, since the processor has four ALUs, the threads hardly compete for the ALUs. The average execution time of PID-FP was $23.31\mu s$ in UL-DRM-FFDE while that was $29.80\mu s$ in UL-DRM-DUP, $29.61\mu s$ in RM-FF and $30.93\mu s$ in RM-US. It means that UL-DRM-FFDE made use of FPUs more efficiently than the other algorithms. Since the processor has only two FPUs, resource competition for the FPUs occurs more often than the case of the ALUs. As for MEM1 and MEM2, UL-DRM-FFDE dramatically reduced the execution time up to $70.1\mu s \sim 166.10\mu s$ over the other algorithms, which means UL-DRM-FFDE increased the execution efficiency about 20% ~ 30%. The effectiveness of UL-DRM-FFDE was quite larger than the case of PID-INT and PID-FP. We consider that is because the latency of memory access is larger than other functions, hence the execution time that

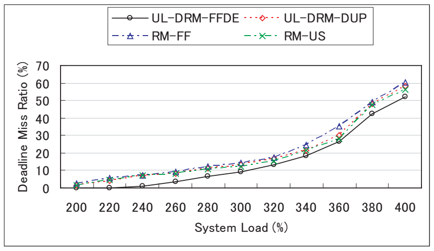

Figure 6. Deadline miss ratio (without schedulability test)

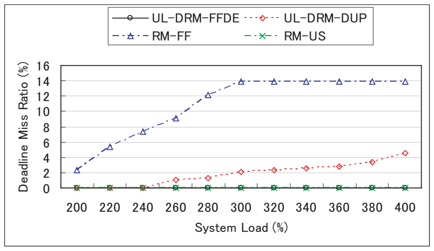

Figure 7. Deadline miss ratio (with schedulability test)

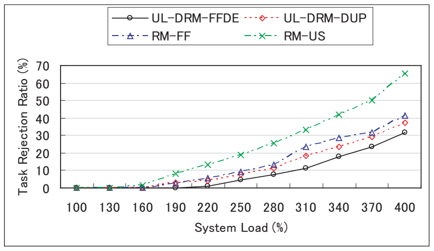

Figure 8. Task rejection ratio (with schedulability test)

UL-DRM-FFDE could reduce was also larger.

4.2 Real-Time Processing Evaluation

As shown in Figure 6, UL-DRM-FFDE scheduled the tasks without missing the deadline till system load 220% while the deadline miss ratio was 4.31% in UL-DRM-DUP, 5.39% in RM-FF and 4.80% in RM-US. Since UL-DRM-FFDE concerns the compatibility among the co-scheduled tasks, the execution times of the tasks were reduced as we discussed in Section 4.1 and, as a result, it succeeded in scheduling more tasks than the other algorithms without missing the deadline.

Figure 7 depicts the deadline miss ratio with the schedulability tests. Both UL-DRM-FFDE and UL-DRM-DUP test Equation (13) and (14). Note that UL-DRM-FFDE does not need to test Equation (15) due to the pseudo-task transformation. In this experiment, no deadline miss was observed in UL-DRM-FFDE and RM-US, which means the schedulability test of these algorithms were sufficient enough to meet the deadline. Meanwhile some deadline misses were observed in UL-DRM-DUP and RM-FF. Since UL-DRM-DUP estimates WCET at runtime, some tasks missed the deadline before WCET was estimated. In addition, UL-DRM-DUP does not conduct the pseudo-task transformation. Therefore some tasks missed the deadline when the actual execution time was larger than estimation. The deadline miss ratio went up to 4.54% in the end. The deadline miss ratio of RM-FF was the largest and was constantly about 13.90% after there was no LP to be assigned.

As we explained, it turned out that UL-DRM-FFDE and RM-US can guarantee the tasks to meet the deadline using EET described in Section 3.1. However, as shown in Figure 8, the rejection ratio of RM-US was the largest. RM-US, in other words, avoided missing the deadline instead rejected many tasks instead because its schedulability test is quite strict. Meanwhile UL-DRM-FFDE accepted the most tasks in any system load. It accepted the tasks 5.81% over UL-DRM-DUP, 9.89% over RM-FF and 33.78% over RM-US.

5 Conclusion

This paper described the extension of U-Link Scheduling. We defined the execution efficiency and described how to estimate the execution time for SMT real-time systems. Then we proposed UL-FFDE algorithm that establishes the co-scheduled sets where the execution efficiency can be increased. Finally we provided the schedulability analysis of UL-DRM that is the algorithm where RM is combined with the extended U-Link Scheduling.

In the experiments, we showed that UL-DRM with UL-FFDE, i.e. UL-DRM-FFDE, increased the execution efficiency about 4% ~ 30%. As a result, UL-DRM-FFDE succeeded in scheduling about 6% ~ 35% more tasks than the other algorithms without missing the deadline.

Acknowledgement

This research is conducted through the fund of JSPS. Also this research is supported in part by CREST, JST.

References

[1] B. Andersson, S. Baruah, and J. Jonsson. Fixed-priority scheduling on multiprocessors. In *Proc. of Real-Time Systems Symposium*, pages 193–202, 2001.

[2] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26:127–140, 1978.

[3] S. Kato, H. Kobayashi, and N. Yamasaki. U-Link Scheduling: Bounding Execution Time of Real-Time Tasks with Multi-Case Execution Time on SMT Processors. In *Proc. of Intl. Conf. on Embedded and Real-Time Systems and Applications*, pages 193–197, 2005.

[4] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proc. of Annual Intl. Symp. on Computer Architecture*, pages 392–403, 1995.