MEMORY-AWARE DYNAMIC FREQUENCY SCALING FOR SAVING ENERGY IN REAL-TIME SYSTEMS

[1]YONG-WEI ZHAN, [2]CHING-WEN CHEN

Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 40724, Taiwan
E-mail: [1]m0503866@fcu.edu.tw, [2]chingwen@fcu.edu.tw

Abstract - In this study, a Memory-Aware Dynamic Frequency Scaling (MDFS) mechanism is proposed to efficiently reduce energy consumption in a real-time system. To minimize energy consumption and meet the deadline of a time-constrained application, a mathematical model is proposed to show at which frequency a processor should be run to reduce the maximum energy consumption when the cache miss rate and the relax time of the application are given. Accordingly, the proposed MDFS mechanism predicts the number of potential execution instructions of an application with the cache miss rate to evaluate the proper frequency of a processor to save energy. In addition, to guarantee the deadline requirement of an application, a relax time timer (RTT) is used in the proposed MDFS mechanism. The simulation results showed that our proposed MDFS mechanism and RTT consumed less approximately 22% of energy than the processor that used the highest frequency, and satisfied the real-time requirement for time-constrained applications.

Index Terms - Dynamic voltage and frequency scaling, unavailable time, cache miss, real-time applications.

I. INTRODUCTION

Dynamic voltage and frequency scaling (DVFS) is usually used in a state-of-the-art system to reduce the energy consumption of a processor. The frequency of a processor can be adjusted to a lower state to reduce the energy consumption when the system load is light. However, in a real-time system where the applications are time-constrained, the result of completing an application by its deadline is more important than reducing energy consumption. Therefore, to use DVFS in a real-time system for reducing the energy consumption, the deadline of time-constrained applications should be considered first, followed by the reduction in energy consumption.

Cache memory mainly affects the performance and energy consumption of a processor. When a processor wants to fetch an instruction or data from a cache and a miss occurs, the processor stalls until the desired instruction or data is loaded from a lower-level cache or main memory. Therefore, if the processor uses a lower frequency when it waits for the desired instruction or data, the energy consumption can be reduced. However, switching the frequency of a processor leads to the unavailable time problem that affects the performance and energy consumption; the unavailable time problem means that a processor cannot work for a certainperiod of time (about 10 μs to 650 μs) [1][2], which is much greater than the cache miss penalty.

In this study, a Memory-Aware Dynamic Frequency Scaling (MDFS) mechanism is proposed to efficiently reduce the energy consumption in a real-time system. The contributions of this study are described as follows: to make a time-constraint application meet its deadline and reduce energy consumption, a mathematical model is proposed to show at which frequency a processor should be run to reduce the maximum energy consumption when the cache miss rate and the relax time of the application are given; further, to reduce the number of times of switching the frequency of a processor, two frequencies of a processor are used to exhaust the relax time of an application for minimizing the energy consumption and satisfying the deadline requirement of the time-constrained application.

II. PROPOSED METHOD

In this section, an MDFS mechanism is proposed for time-constrained applications to reduce energy consumption and meet deadline requirements. When a cache miss occurs, the energy consumption can be reduced by lowering the frequency of a processor. However, frequently switching a processor's frequency will cause the unavailable time problem which makes a time-constrained application miss its deadline. Therefore, the proposed MDFS mechanism can solve the abovementioned issue.

A. Analysis

In this section, we analyse the frequency at which a processor can be run to have the maximum energy reduction and make an application meet its deadline. Let us assume that a processor has n frequencies, $F=\{f_0, f_1, \ldots f_{n-1}\}$, where f_0 is the highest frequency. For an application, we assume that T_{f_0} is the execution time of the application by a processor with frequency f_0 and $T_{Deadline}$ is the deadline of the application. Therefore, the relax time ΔT_{RT} of the application is shown in (1) and Fig. 1, when f_0 is used.

$$\Delta T_{RT} = \left(T_{Deadline} - T_{f_0}\right) \quad (1)$$

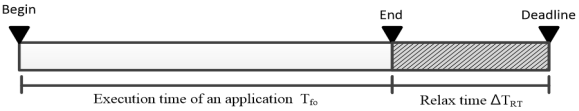


Fig. 1 The relax time of an application when a processor uses the highest frequency, f_0

The energy consumption of a processor that runs an application with frequency f_x is evaluated as follows. First, the ideal cycles of executing an instruction are denoted as CPI_{ideal}, which means the cycles of executing an instruction and cache access are ideal. In addition, we have to consider the time penalty of one cache miss. The cycles of the cache miss penalty vary with the processor frequency. As shown in(2), The time penalty of one cache miss is denoted as $Cycle_{penalty}(f_x)$, where $CycleTime_{f_x}$ is cycle time with frequency f_x. When a cache miss situation is considered, the energy consumption of executing an instruction by a processor with frequency f_x is shown in (3), where the miss ratios of the instruction cache and the data cache are denoted as MissRate(ins) and MissRate(data) , respectively, the rate of load/store instructions is denoted as N, and the energy consumption of executing an instruction can be represented as $Energy_{(ins,f_x)}$.

$$Cycle_{penalty}(f_x) = Delay_{penalty} \times f_x \quad (2)$$

$$Energy_{(ins,f_x)} = Energy_{(cycle,f_x)} \times (CPI_{ideal} + (\text{MissRate(ins)} + N \times \text{MissRate(data)}) \times Cycle_{penalty}(f_x)) \quad (3)$$

When a cache miss is occurred and the processor uses a higher frequency to execute an application, it has to consume more energy to receive the desired instruction or data. Therefore, If the processor lowers its frequency at the moment, the energy consumption can be reduced efficiently. However, there are two issues may have to be considered:

1)How to exhaust the relax time to reduce the energy consumption.

2)How to ensure an application can be completed by its deadline.

For the first issue, as shown in (4), we can determine that a processor with frequency f_x can execute $InsCount(f_x, f_0)$ instructions to exhaust the relax time ΔT_{RT}.

$$InsCount(f_x, f_0) = \frac{\Delta T_{RT}}{CPI_{ideal} \times (\frac{1}{f_x} - \frac{1}{f_0})} \quad (4)$$

However, the total instructions of an application may be less or more than$InsCount(f_x, f_0)$.To obtain the proper frequency of the processor to exhaustΔT_{RT},we estimate the total instructions of an application with (5).

$$InsCount_{all} = \frac{T_{f_0}}{(CPI_{ideal} + (MissRate(ins) + N \times MissRate(data)) \times Cyclepenalty(f_x)) \times CycleTime_{f_0}} \quad (5)$$

With (4) and (5), the instructions executed by a processor with frequency f_x is$InsCount_{cover}(f_x, f_0)$, as shown in (6), which is equal to the minimal one of $InsCount(f_x, f_0)$ and $InsCount_{all}$.

$$InsCount_{cover}(f_x, f_0) = Min(InsCount(f_x, f_0), InsCount_{all}) \quad (6)$$

According to (6), the frequencies can be classified into two categories:

First category: The estimated instructions executed by a processor with frequency f_x are more than the executed instructions of an application. Therefore, $InsCount_{cover}(f_x, f_0)$ is equal to $InsCount_{all}$; however, ΔT_{RT} may just or not be exhausted, as shown in Fig. 2(a). In this case, the processor can use frequency f_x to execute all instructions of an application to reduce the energy consumption, but ΔT_{RT} may not be exhausted. In order to exhaust ΔT_{RT}, the processor can use a frequency lower than f_x to execute a part of the instructions as well as use f_x to execute the remaining. Therefore, the energy consumption can be reduced in advance.

Second category:The estimated instructions executed by a processor with frequency f_x cannot cover all the instructions of an application. This means that $InsCount_{cover}(f_x, f_0)$ is equal to $InsCount(f_x, f_0)$, but ΔT_{RT} is exhausted. In this case, the processor with frequency f_x runs a part of the instructions and the remaining instructions have to be executed by the highest frequency f_0 to ensure the application can meet the deadline, as shown in Fig. 2(b). Therefore, energy consumption cannot be reduced efficiently in this case. If a processor can use a frequency higher than f_x but lower than f_0 to execute a part of the instructions, the situation in which the instructions are executed with the highest frequency f_0 can be avoided. In addition, in order to consider the unavailable time problem, execute all the instructions of an application with proper frequency, and exhaust ΔT_{RT} of an application, we determine two frequencies that can execute all of the instructions of an application to reduce the energy consumption.

Let us assume that there are two frequencies: a high frequency $f_H$and a low frequency f_L. In addition, assume that a processor uses frequency f_H to execute an application and the entire instructions of the application can be executed by frequencyf_H, but ΔT_{RT} is not exhausted. In contrast, assume that the processor only uses frequency f_L to execute the application and only partial instructions of the application are executed by the processor with frequency f_L and ΔT_{RT} can be exhausted. However, the remaining instructions have to beexecuted with the highest frequency f_0 to ensure the application can completed before its deadline. Therefore, the sum of$Ins(f_H)$and $Ins(f_L)$ is equal to the total executed instructions $InsCount_{all}$of an application, where $Ins(f_H)$ and $Ins(f_L)$ are the number of instructions of the application executed by a processor with frequencies f_H andf_L, respectively, as shown in (7).

When a processor with frequency f_x that is lower than the highest frequency f_0 executes an instruction, it takes more time to execute that instruction than when using frequencyf_0, as shown in (9), where $ETPI(f_x, f_0)$ indicates the time difference for executing an instruction with frequencies f_x andf_0. Let us assume that the unavailable time of switching a processor

frequency once is U. Therefore, when a processor uses two frequencies f_H and f_L to execute an application and the relax time of the application is exhausted, the sum of $Ins(f_H) \times ETPI(f_H, f_0)$, $Ins(f_L) \times ETPI(f_L, f_0)$, and U is equal to the relax time of the application, as shown in (8), where $Ins(f_H)$ and $Ins(f_L)$are the numbers of instructions executed by a processor with frequencies f_H and f_L, respectively. With (7) and (9), we can obtain $Ins(f_H)$ and $Ins(f_L)$ when the relax time of an application and the unavailable time are given.

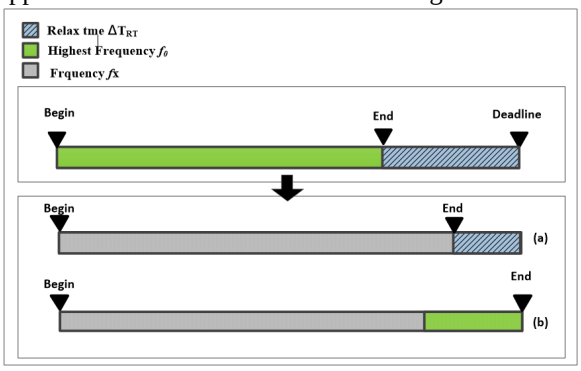


Fig. 2 All frequencies can be classified into two groups: (a) $InsCount_{cover}(f_x, f_0)$ is equal to $InsCount_{all}$, but ΔT_{RT}may just or not be exhausted. (b) $InsCount_{cover}(f_x, f_0)$ is equal to $InsCount(f_x, f_0)$, but ΔT_{RT} is exhausted

$$InsCount_{all} = Ins(f_H) + Ins(f_L) \quad (7)$$

$$ETPI(f_x, f_0) = CPI_{ideal} \times (\frac{1}{f_x} - \frac{1}{f_0}) \quad (8)$$

$$\Delta T_{RT} = (Ins(f_H) \times ETPI(f_H, f_0)) + (Ins(f_L) \times ETPI(f_L, f_0)) + U \quad (9)$$

B. Memory-Aware Dynamic Frequency Scaling (MDFS)

In this section, the proposed MDFS design is introduced. When the relax time and execution time of an application are given, there are two steps in the proposed MDFS mechanism:

Step 1: All frequencies of a processor is classified into two groups f_H and f_L. The highest frequency f_0 is used to execute a period of time to know the instruction and data cache miss ratios MissRate(ins) and MissRate(data), respectively, and the rate of load/store instructions N. In addition, assuming that the ideal cycles of executing an instruction including the hit of the cache access cycle areCPI_{ideal}. Equation (3) and (6) can be applied to estimate the potential number of instructions to be executed for an application,$InsCount_{all}$.Subsequently, (5)can be used to know the number of executed instructions by the processor with the remaining n-1 frequencies, excluding the highest frequency f_0, when the relax time of the application is exhausted. Finally, to exhaust the relax time of the application and run lower frequencies than the highest one so as to cover all the instructions of the application and to reduce the energy consumption efficiently, these n-1 frequencies are classified into groups f_H and f_L according to the following rules: if the number of the instructions executed by a processor with frequency f_x is less than the number of the estimated total instructions of the application and the relax time is exhausted, then frequency f_x belongs to group f_L. Otherwise, frequency f_x belongs to group f_H.

Step 2: Two frequencies are chosen separately from f_H and f_L to execute the application to minimize the energy consumption and to exhaust the relax time of an application. Each combination of two frequencies is selected from f_H and f_L by applying (7), (8), and (9) to determine the number of executed instructions with the two frequencies. By applying (4), all possible combinations are evaluated to determine the lowest energy consumption. Finally, the least energy consumption combination can be obtained so as to execute the application.

C. RTT

In the proposed MDFS mechanism, the number of instructions of an application is predicted by monitoring the cache miss ratio at the beginning phase of executing the application. However, the cache miss ratio may vary during execution, thereby leading to a situation where the relax time of the application may be exhausted and the application is not completed. However, when the situation occurs, the application may not satisfy the deadline requirement. Accordingly, an RTT is added in MDFS to trace the remainder of the relax time of an application to avoid of missing the deadline, as shown in Fig. 3. To guarantee that an application can be completed by its deadline, an extra unit of unavailable time is reserved to make the processor switch its frequency to the highest one. This means that when the relax time of an application remains $\Delta T_{RT} - U$ only, the processor switches to the highest frequency to execute the remaining instructions of the application to satisfy the deadline requirement.

III. SIMULATION RESULTS

In this section, the simulation environments and the simulation results are shown. The Simics [3] full system simulator with a Solaris installation executed the SPLASH-2 benchmark suite [4]. The deadline of each application is the execution time and the extra 10% to 15% of execution time of the application where the execution time is the time that an Intel i5 processor runs an application with the highest frequency.

Intel i5 processor with P-state is used for the simulation [5], where the operation frequency, consumed power, and the energy consumption of each clock cycle are given in Table I. A trace-driven simulator is developed to analyze the trace files of the SPLASH-2 benchmark suite to evaluate MA-DVFS [6], our proposed MDFS and MDFS with RTT (MDFS+RTT) mechanisms, and Intel i5 with different frequencies in terms of the number of times of scaling a processor frequency, execution time, and energy consumption. In MDFS and MDFS+RTT, in order to

estimate the number of executed instructions of an application, the first 10% of execution time of an application is executed by the highest frequency (f_0) to obtain the miss ratios of the instruction cache and the data cache by the hardware performance counters (HPCs) [7][8]. The simulation environment and parameters are listed in Table II.

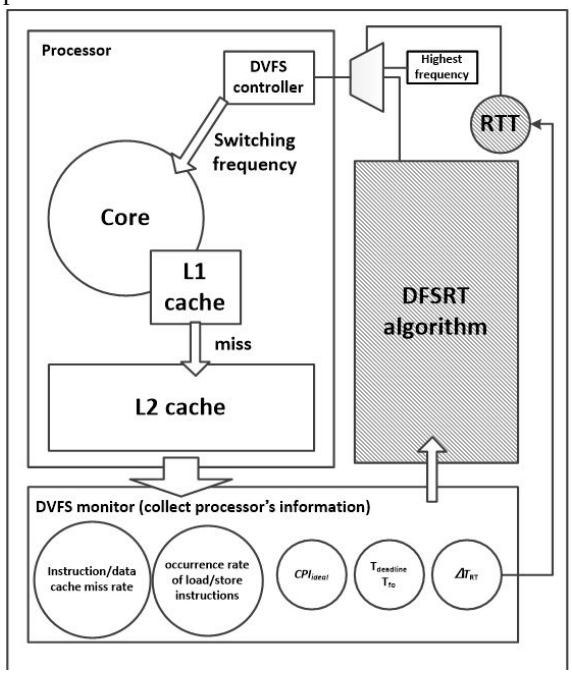


Fig. 3 Proposed DFSRT and RTT

A. The number of scaling processor's frequency

Table III shows the simulation results of the number of scaling of processor's frequency in MA-DVFS, our proposed MDFS, and MDFS+RTT mechanisms. MA-DVFS switches the processor's frequency by monitoring the miss ratio of the cache while executing an application. Therefore, MA-DVFS switches the processor's frequency more times than our proposed MDFS and MDFS+RTT mechanisms. However, switching the frequency of a processor leads to the unavailable time problem. MA-DVFS may miss the deadline after considering the unavailable time problem. In our proposed MDFS mechanism, the processor uses the highest frequency to collect the miss ratios of the instruction and data caches at the beginning phase of executing an application to evaluate the potential number of execution instructions. Accordingly, two frequencies are used to execute the application to minimize energy consumption. Therefore, there are, at most, two instances where the processor's frequency is switched in MDFS. However, the situation of missing the deadline may occur in MDFS because the number of the application's execution instructions is not accurately predicted. Therefore, the RTT design can be augmented in MDFS to satisfy the deadline requirement of an application. Therefore, MDFS+RTT switches the frequency of the processor at most three times in some benchmarks including LU, OCEAN, VOLREND, and WATER_NSQ, as shown in Table III.

B. Execution time and Deadline Requirement

Fig. 4 shows the simulation results of the execution time of our proposed MDFS, MDFS+RTT, and MA-DVFS mechanisms, and the conventional Intel i5 processor with different frequencies, where the dashed line represents the deadline of that application, and there are 11 frequencies in Intel i5 processor. With regard to the Intel i5 processor with different frequencies, the four lowest frequencies of Intel i5 processor always miss the deadline for all benchmarks and the four highest frequencies of Intel i5 processor satisfy the deadline for all benchmarks. Therefore, we show the execution time of Intel i5 with the remaining frequencies, that is, frequencies f_3, f_4, f_5, f_6, and f_7 in Fig. 4.

MA-DVFS switches the processor's frequency according to the miss rates of the cache during the execution of an application. Therefore, the execution time of an application in MA-DVFS increases due to the unavailable time problem and this leads a higher probability of missing the deadline. The proposed MDFS mechanism without RTT predicts the number of total execution instructions of an application by monitoring the cache miss ratios during the beginning phase of an application execution to select the suitable frequencies so as to exhaust the relax time of an application as far as possible for reducing energy consumption and meeting the deadline. However, when the prediction of the number of the execution instructions is not precisely accurate, the proposed MDFS mechanism without RTT may miss the deadline requirement. In contrast, MDFS+RTT mechanism can trace the remaining relax time of an application to guarantee the deadline requirement and exhaust the relax time for energy consumption, as shown in Fig. 4.

C. Energy consumption

In the simulation results of energy consumption, the designs that can satisfy the deadline of the benchmarks are shown in Fig. 5. Therefore, some designs are absent from Fig. 5 because they miss the deadline of the benchmarks. The miss rate shown in each subgraphs of Fig. 5 is the sum of the miss rates of the instruction cache and data cache.

In Fig. 5, our proposed MDFS+RTT mechanism shows energy consumption less than Intel i5 with fixed frequencies, f_0, f_1, f_2, f_3, f_4, f_5, and f_6, where frequencies f_5 and f_6 are absent from some benchmarks because of missing the deadline. Below, we discuss the simulation results with three kinds of cache miss rates of the benchmarks.

In the first case, the benchmark has high miss ratios of the cache, for example, WATER_NSQ has 49.8% cache miss rates, as shown in Fig. 5 (a). In this kind of benchmark, the miss penalty takes the most of the

execution time. Therefore, lowering the frequency of a processor can efficiently reduce the energy consumption. In the simulation results, frequencies f_0 to f_6 can satisfy the deadline requirement of WATER_NAQ and frequency f_6 shows the lowest energy consumption than the other fixed frequencies. Our proposed MDFS and MDFS+RTT mechanisms can satisfy the deadline of WATER_NAQ and can consume less energy consumption than Intel i5 with frequency f_6 by 10% to 7%.

In the second case, the cache miss rates of the benchmarks are 10% to 30%, for example, FMM and LU belong to this kind, as shown in Fig. 5 (b). The lowest energy consumption of these fixed frequencies for these benchmarks are different, as shown in Fig. 5. The main reason is that using a lower frequency can save more energy but it may result in an application to miss its deadline. In these fixed frequencies of Intel i5, f_4 can satisfy the deadline requirement of these benchmarks and achieve the lowest energy consumption. In our proposed MDFS and MDFS+RTT mechanisms, energy consumption lesser than Intel i5 with frequency f_4 can be achieved. With regard to the proposed MDFS mechanism, because the number of execution instructions of benchmarks LU, OCEAN, VOLREND, and WATER_SPA, is not predicted precisely, the proposed MDFS mechanism misses the deadline for these benchmarks. To meet the deadline requirement, MDFS+RTT mechanism can ensure satisfying the deadline requirement for all of this kind of benchmarks and consume less energy consumption than Intel i5 with frequency f_4 by 10%.
In the third case, the benchmarks have lower miss rates of the cache, for example, RAYTRACE has 8.3% cache miss rates, as shown in Fig. 5 (c). In the fixed frequencies of Intel i5, using

Table I. P-states in Intel i5 processor

P-state	Frequency (MHz)	Power (mW)	Energy of each cycle (nJ)
f_0	2400	25000	10.42
f_1	2399	25000	10.42
f_2	2266	23316	10.29
f_3	2133	21689	10.17
f_4	1999	20116	10.06
f_5	1866	18531	9.93
f_6	1733	17021	9.82
f_7	1599	15517	9.70
f_8	1466	14068	9.60
f_9	1333	12640	9.48
f_{10}	1199	11250	9.38

Table II. Simulation environment and system parameters

Processor core	Intel i5 processor, in-order, issue width 4
L1 instruction cache	Split, each is 4 KB, 2 way set-associative, 64-bytes line size, miss penalty: 8.33 ns
L2 cache	Shared, 32 KB, 8 way set-associative, 64-bytes line size, miss penalty: 41.6 ns
Unavailable time	300 μs
Relax time	10% of execution time of each application run by the processor with frequency f_0.

Table III. The number of times a processor's frequency is switched

	MDFS	MADFS +RTT	MA-DVFS
FMM	2	2	28
LU	2	3	131
OCEAN	2	3	49
RADIX	2	2	38
RAYTRACE	2	2	17
VOLREND	2	3	53

WATER_NSQ	2	3	66
WATER_SPA	2	2	19

frequency f_4 meets the deadline and shows the lowest energy consumption of using the other frequencies which meet the deadline.In this case, because the cache miss rates are low, the energy savings achieved by switching the processor's frequency are limited. Fig. 5 shows that our proposed MDFS mechanism consumes less energy than Intel i5 with frequency f_4 by 1%. The main reason is that the proposed MDFS mechanism can efficiently exhaust the relax time of the benchmarks with two frequencies. However, to ensure the benchmarks can meet their deadline, an RTT is used with the proposed MDFS mechanism. When the RTT is used, a slight overhead is added which results in the proposed MDFS+RTT mechanism consuming more energy than Intel i5 with frequency f_4 by 1% for RAYTRACE.On comparing Intel i5 with the highest frequency, our proposed MDFS+RTT mechanism gives an average of 22% energy consumption, as shown in Fig 5 (d).

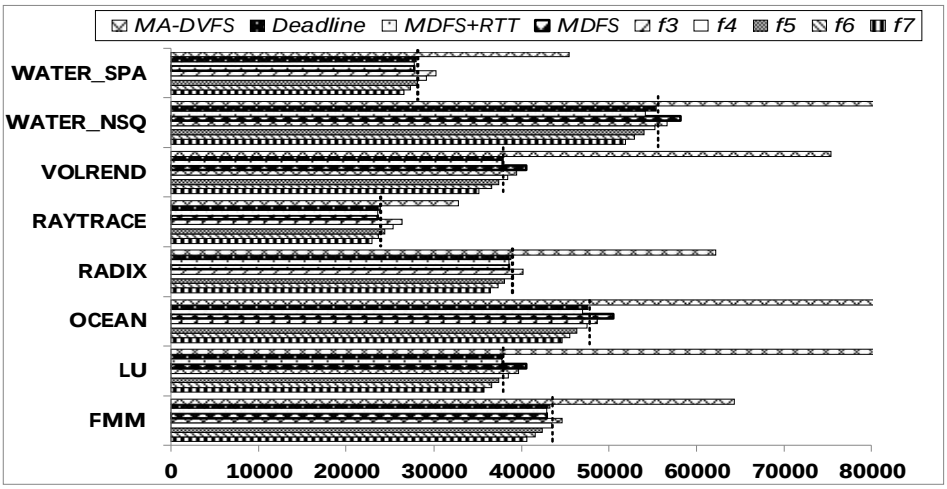
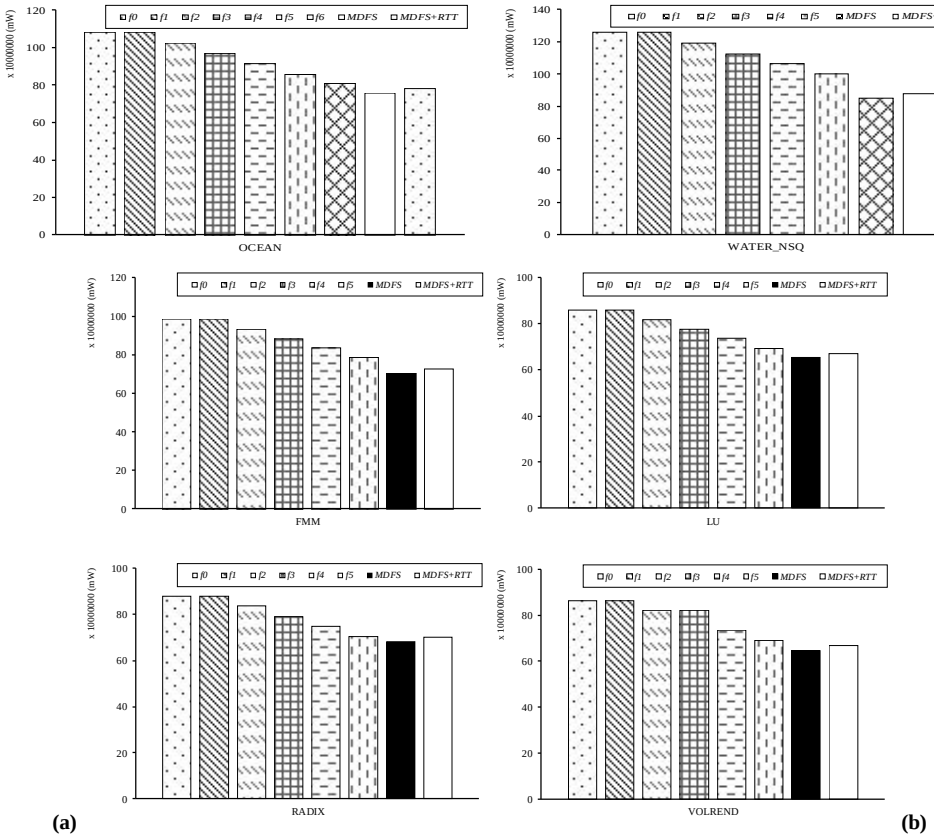
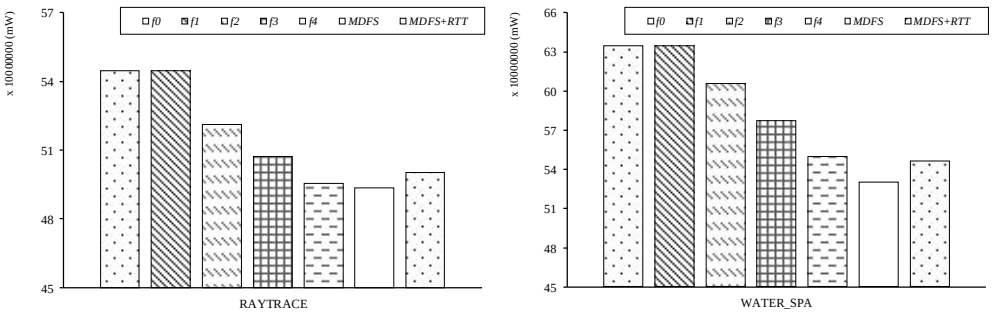


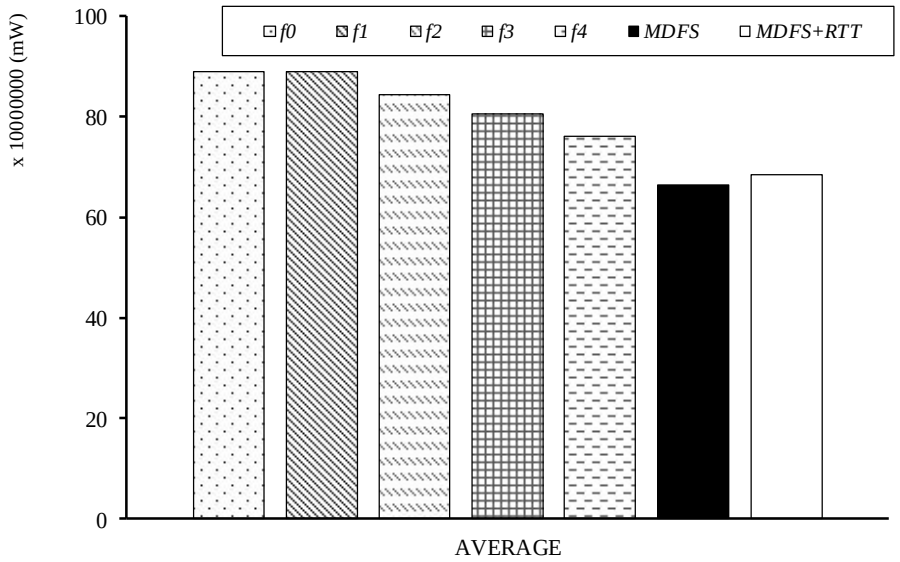
Fig. 4 Execution time



(a) **(b)**



(c)



(d)

Fig. 5 Energy consumption of different cache miss rates: (a) high miss rate (>30%), (b) middle miss rate (10% to 30%), (c) low miss rate (<10%), and (d) average

CONCLUSION

In this study, we proposed an MDFS mechanism to reduce energy consumption and satisfy the deadline of an application for a real-time system. To proposing the MDFS mechanism, a mathematical model is proposed to show the frequency at which the processor can be run to reduce the maximum energy consumption when the cache miss rate and the relax time of the application are given. In addition, to exhaust the relax time and reduce the number of switching a processor's frequency, the proposed mathematical model also considers two suitable frequencies to execute the application to reduce energy consumption in advance. With the mathematical model, the MDFS mechanism is proposed to reduce energy consumption. Finally, to satisfy the deadline of a real-time application, an RTT design is used in MDFS to trace the remainder of the relax time of an application.

The simulation results show that our proposed DFSRT+RTT mechanism can guarantee the deadline requirement of an application and consume less energy than Intel i5 with the highest frequency by 22% on average.

ACKNOWLEDGMENT

This research was supported by Ministry of Science and Technology Taiwan (MOST106-2221-E-035-023-)

REFERENCES

[1] Intel Corporation, "Enhanced Intel® SpeedStep® Technology for the Intel® Pentium® M Processor," http://www.intel.com/design/intarch/papers/301174.htm, March 2004.

[2] Kweon, H., Do, Y., Lee, J., Ahn, B.,"An Efficient Power-Aware Scheduling Algorithm in Real Time System,"IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 350-353, August 2007.

[3] Magnusson, P.S., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A., Werner, B., "Simics: A Full System Simulation Platform," IEEE Computer, Vol. 35, Issue 2, pp.50-58,February 2002.

[4] Woo, S.C., Ohara, M., Torrie, E., Singh, J.P., Gupta, A., "The SPLASH-2 Programs: Characterization and Methodological Considerations," Proceedings of the 22nd annual international symposium on Computer architecture (ISCA '95), pp. 24-36, June 1995.

[5] Saha, S., Ravindran, B., "An Experimental Evaluation of Real-Time DVFS Scheduling Algorithms," The Proceedings of the 5th Annual International Systems and Storage Conference (SYSTOR '12), 12 pages, Article 7, 2012.

[6] Liang, W., Chen, S., Chang, Y., Fang, J., "Memory-aware Dynamic Voltage and Frequency Prediction for Portable

Devices," The 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), August 2008, pp. 229-236.

[7] Weaver, V.M., Terpstra, D., Moore, S., "Non-Determinism and Overcount on Modern Hardware Performance Counter Implementations," IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 215-224, April 2013.

[8] Kim, H., Kim, Y., Yu, H., Suh, T. "Performance evaluation of many-core systems: case study with TILEPro64," IET Computer & Digital Techniques, Vol.7, Issue 4, pp. 143-154, July 2013.

★★★