Real-Time Power Management for a Multi-Performance Processor

Ishihara, Tohru
System LSI Research Center, Kyushu University

出版情報：Proceedings of International SoC Design Conference. 2009, pp.147-152, 2009-11. IEEE
バージョン：
権利関係：

Real-Time Power Management for a Multi-Performance Processor

Tohru Ishihara
System LSI Research Center, Kyushu University, JAPAN
ishihara@slrc.kyushu-u.ac.jp

ABSTRACT

This paper presents an energy efficient embedded processor which can be used as a design alternative for the dynamic voltage scaling (DVS) processors in embedded real-time system design. The processor consists of multiple same-ISA PE (processing element) cores and a selective set-associative cache memory. The PE-cores differ in their clock speeds and energy consumptions. Only a single PE-core is activated at a time and the other PE-cores are deactivated using clock gating and signal gating. The major advantage over the DVS processors is a small overhead for changing its performance. Our processor can change its performance within 1.5 microsecond and dissipates about 10 nano-joule while conventional DVS processors need hundreds of microseconds and dissipate a few micro-joule for each voltage transition. Our processor makes it possible to use the DVS control in embedded real-time systems and to perform more sophisticated dynamic power management.

Keywords: Microprocessor, Low-power design, Embedded systems, Real-time systems

I. INTRODUCTION

Dynamic voltage scaling (DVS) is one of the most popular approaches for reducing the energy consumption of processors. The DVS-capable processor dynamically lowers the supply voltage to the lowest possible value which ensures a correct operation of the processor under a given clock frequency. Since the dynamic energy consumption of CMOS circuits, which dominates total energy consumption, is quadratically proportional to the supply voltage, significant energy reduction can be achieved by the DVS scheme. In past years, a lot of DVS processor architectures have been proposed [1, 2, 3, 4]. However, only a few of them are used in embedded systems. One major reason is that many DVS processors involve large mass production cost including test cost, design cost and the cost of on-chip DC-DC converters. The other reasons are delay and energy overheads for dynamically changing the supply voltage and the clock frequency. In this paper, a real-time power management technique for our originally developed processor named multi-performance processor is presented. The processor can be used as a design alternative for the DVS processors. This paper is an extension of our previous work presented in [5]. Although our previous work is focused on presenting the multi-performance processor architecture, this paper presents how the processor can be applied to real-time embedded systems. This paper also quantitatively evaluates the energy consumptions of three types of processors; 1) a conventional DVS processor, 2) a heterogeneous multi-processor, and 3) our multi-performance processor.

The rest of the paper is organized as follows. In section 2, related work and an idea of our multi-performance processor is presented. Section 3 presents an overview and specifications of our prototype processor which integrates the minimal functionality of the multi-performance processor architecture. Section 4 shows how our processor can be effectively applied to real-time embedded applications for reducing the energy consumption without violating real-time constraints. The paper concludes in Section 5.

II. RELATED WORK AND OUR APPROACH

A. Voltage and Task Scheduling on DVS Processors

In the past, a lot of DVS scheduling algorithms have been proposed for real-time systems. For given multiple tasks, these algorithms dynamically assign the proper clock frequency and operating voltage to each task while guaranteeing all their deadlines. In real-time systems, since the actual execution time of each task may be much smaller than the worst-case execution time (WCET), CPU slack times are generated at run time even though the worst-case CPU utilization is 1. However, it is usually difficult to exploit the CPU slack times because we cannot know the actual amount of slack time before the completion of a task. Therefore, most of DVS scheduling algorithms transfer the slack time to later tasks which can use it [6, 7, 8]. These techniques determine the supply voltage on task-by-task basis. More specifically, only one operating voltage is assigned to each active task and it is not changed during the task execution. This policy is not very effective because the next tasks are not always ready to be executed when a task completes much earlier than its deadline.

Lee and Sakurai proposed a technique which can use CPU slack times within the current task [9]. This technique first divides each task into fixed-length segments. After the completion of each segment, the optimal operating voltage is selected depending on the slack time made by the previous segments. In this case, the duration of the segment is typically a few milliseconds. However, as shown in table 1, the transition time required for changing the operating voltage of DVS processors is hundreds of microseconds [11]. More specifically, 500 microseconds are supposed for the voltage transition time in [9].

Table 1: Commercial DVS processors

Processor	Voltage (V)	Transition Time
Transmeta Crusoe	1.1-1.65	300μs
AMD Mobile K6	0.9-2.0	200μs
Intel PXA250	0.85-1.3	500μs
Compaq Itsy	1.0-1.55	189μs
TI TMS320C55x	1.1-1.6	3.2 ms (1.6 $\rightarrow$ 1.1V) 300μs (1.1 $\rightarrow$ 1.6V)
UCB [13]	1.2-3.8	520μs

This is very large compared to the duration of the task segment, which reduces the chances of lowering the operating voltage of the processor. Shin and Kim [10] generalize the technique in [9] to non-periodic tasks as well as the periodic tasks presented in [9]. However, the large overhead for changing the voltage in DVS processors is still an critical issue for real-time applications since many real-time systems require less than a millisecond of task response time [12]. In addition to the transition time overhead, the energy overhead is very large as well. The energy consumed in DC-DC converter during voltage transition from V_{DD2} to V_{DD1} is:

$$E_{TRAN} = (1-\eta)\cdot C\cdot|V_{DD2}^2 - V_{DD1}^2| \quad (1)$$

where η is the efficiency of the DC-DC converter [13]. A typical capacitance of 100μF yields E_{TRAN} = 6.4μJ for 0.6V-1.0V voltage transition and η = 90%. This energy corresponds to the energy consumed during 25K cycles of task execution on a typical embedded processor.

B. Dynamic Task Scheduling on Multi Processors

Dynamic task migration on a heterogeneous multi-processor can reduce the energy consumption for task execution [14]. Suppose we have a heterogeneous dual-core multi-processor. Both of the MPUs employs scratchpad and cache memories which are tightly connected with the corresponding MPU-core. These two processors have the same instruction-set architecture but differ in their clock speeds and power consumptions like our multi-performance processor. More specifically, let us assume the clock frequencies of MPU1 and MPU2 are 200MHz and 100MHz, respectively, and the power consumptions are 8 and 1, respectively.

Consider a task whose execution time is 6. If the task runs on the MPU1 from the beginning to the end, the energy consumed is 8$\times$6 = 48. If some portion of the task is moved to the MPU2 and is executed on it so that the deadline of the task is not violated, the energy consumed is 36 as shown in Figure 1. In this case, the energy consumption is reduced by 25%. However, this task migration involves a large overhead depending on the size of data moved from MPU1 to MPU2. If a stack segment is allocated in a scratchpad memory (i.e., SPM), all data in the stack segment have to be moved along with the task migration. For example in JPEG encoder which is used in our experiments, the sizes of stacks used for several functions are more than 2,000 bytes. For those functions, transferring the

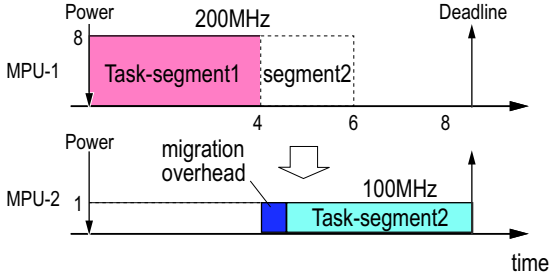

Figure 1: An Example of Dynamic Task Migration

stack data from MPU1 to MPU2 takes more than 30μ seconds and consumes about 3μJ in our prototype multi-processor presented in the following section. This overhead is comparable with the overhead of DC-DC converters used in DVS processors. Even if all data in the scratchpad memory is moved from MPU1 to MPU2 without penalty, the number of cache misses in MPU2 may drastically increase after the task migration.

C. Our Approach

Our processor consists of multiple processing element (PE) cores and a selectable-way cache memory as shown in Figure 2. The PE-cores are functionally equal to each other but have different clock speeds and energy consumptions. Only a single PE-core is activated at a time and the other PE-cores are deactivated using clock gating and signal gating. The PE-core and cache ways which should be activated can be changed at run time by software running on the processor. On-chip memories including cache and scratchpad memories are shared by PE-cores. However, since only a single PE-core is activated at a time, a single port SRAM is used for constructing those on-chip memories. If designers need more task level parallelism, we can easily implement a chip multi-processor (CMP) by integrating some of our processor cores on a chip.

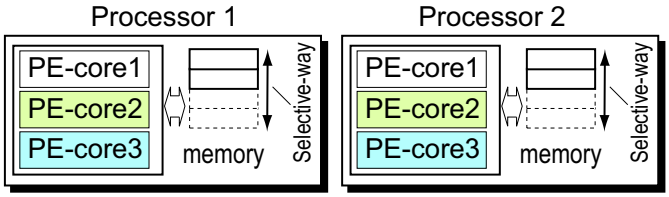

Figure 2: Architecture of Multi-Performance Processor

III. Multi-Performance Processor

A. Architecture

Figure 3 shows a prototype of our multi-performance processor. This prototype has three MPUs and these are connected through AMBA AHB^{TM} bus. The MPU is based on Media embedded Processor (MeP) developed by Toshiba [15]. The clock frequency of the bus is a 67MHz and those of PE-cores are multiple of the bus clock frequency. More specifically, a high-end PE, a middle-end PE, and a low-end PE are operated with 1V/200MHz, 0.68V/133MHz, and 0.52V/67MHz,

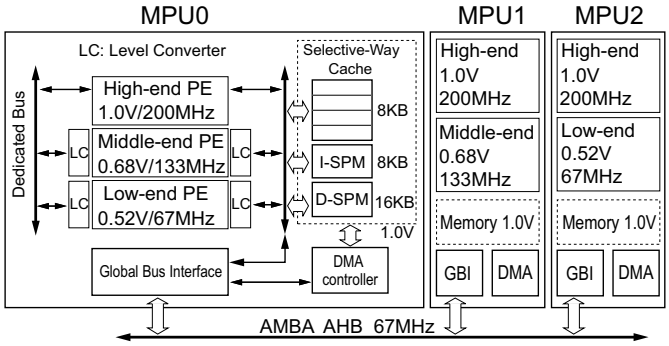

Figure 3: An Example of the Multi-Performance Processor

respectively. An 8K-byte instruction cache, an 8K-byte instruction scratchpad and a 16K-byte data scratchpad are employed by each of those MPUs. Since every on-chip memory uses 1.0V voltage supply, level converters are needed between the middle-end PE-core and the on-chip memories, and between the low-end PE-core and the memories. Only a single PE-core in each MPU is activated at a time and the other PE-cores are deactivated by clock and signal gating. Once a PE-core is activated, an entire clock frequency of the MPU is set to the frequency of the active PE-core. A PE-core which should be activated can be selected by storing a specific value to a special purpose register. Therefore, programmers can explicitly specify the PE-core to be used by using a STORE instruction. Before switching the active PE-cores, the values of internal registers are transferred from the currently activated PE-core to the following PE-core through a dedicated bus and a stack. In our prototype, the data resided in the general purpose registers is transferred through a stack which is allocated in the data scratchpad memory. The data in the special purpose registers is moved directly through the dedicated internal bus.

Each MPU employs a 4-way set-associative instruction cache which is based on a cache architecture proposed in [16]. The cache memory has an extra flag bits to indicate active cache-ways as shown in Figure 4. In our prototype, the flags can be set by storing a specific value to the special purpose register. Therefore, programmers can explicitly specify cache-ways to be used. If the flag bit of way1 is "0", sense amplifier circuits in the corresponding cache-way are deactivated. In this case, the way1 will not be used for replacement even in case of a cache miss and accessing to the way1 will always cause a cache miss. We can trade cache associativity for energy saving.

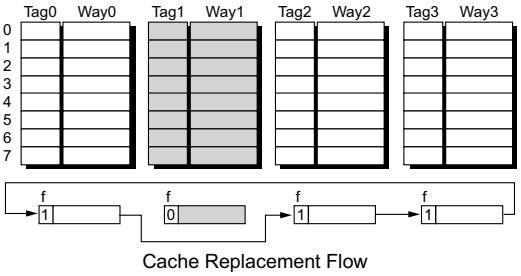

Figure 4: Selective Way Cache Memory

This mechanism is implemented on an instruction cache only. Applying the technique to a data cache is our future work.

B. Transition Overhead

Table 2 shows transition time and energy overheads required for changing PE-cores. High, Middle and Low in the table represent PE-cores using 1.0V, 0.68V and 0.52V, respectively. The values in the table include time required for storing register values of the currently activated PE-core into stack and time required for reading them back from the stack to the PE-core activated next. As one can see from Table 1 and 2, transition overheads required for our multi-performance processor is more than two orders of magnitude smaller than those of commercial DVS processors. These small overheads make it possible to apply our multi-performance processor to real-time systems and to perform finer grained dynamic voltage control.

Table 2: Performance Transition Overheads

Direction	time [ns]	energy [nJ]
High → Middle	1,113	11.84
High → Low	1,290	11.25
Middle → High	968	13.04
Middle → Low	1,443	9.27
Low → High	1,205	13.39
Low → Middle	1,286	8.93
Cache Way	690	10.35

C. Performance of Low-V_{DD} Operation

Figure 5 shows voltage-delay curve for a critical path extracted from the high-end PE core in the prototype. A static timing analysis function of design_compiler is used for extracting the critical path. For calculating delay values in different voltage settings, HSPICE of SYNOPSYS is used. In this example, the path delay at 0.68V is about 12.5ns which corresponds to an 80MHz clock frequency. This means that a typical DVS processor designed with 1.0V logic cells targeting a 200MHz runs, in the best case, at an 80MHz for 0.68V voltage supply. Our multi-performance processor runs at a 133MHz for 0.68V, which is 1.7 times higher performance than the typical DVS processor. In case of 0.52V, the DVS processor runs

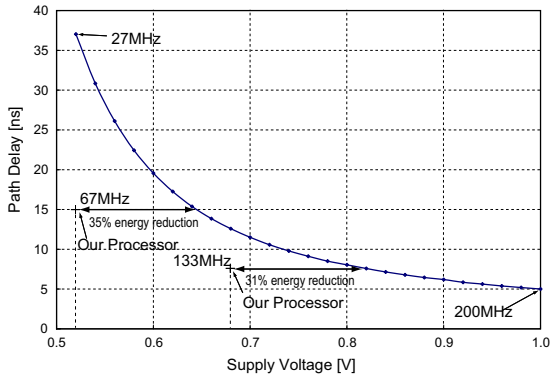

Figure 5: Voltage-Delay Curve

at a 27MHz while ours runs at a 67MHz, which is 2.5 times higher performance than the DVS processor. More precisely, the critical path for a 1.0V operation may not be a critical path of the processor in lower voltage operations. Therefore, the performance of the DVS processor in lower operating voltage might be worse than the curve shown in Figure 5. This is because the delays of several cells using high V_{th} or cells having many inputs rapidly increase along with the voltage downscaling. This issue can be partially resolved by using a dynamic body biasing technique [17]. However it is expensive and may reduce latch-up immunity.

D. Energy and Performance Specifications

Figure 6 shows the energy consumptions and the execution times for processing a fixed amount of data. Bar charts and line charts represent energy consumptions and execution times, respectively. As one can see from the figure, using a 67MHz clock with 0.52V voltage supply does not have any advantages in terms of both energy consumption and execution time. This is because some portions of the power consumption are independent from the clock frequency and these cause an increase of energy consumptions when the execution time increases.

Since the process technology used in this prototype is a low standby-power process where the threshold voltage of transistors is high and therefore the leakage energy is very low at the expense of a switching speed of transistors. The target clock frequency is a few hundreds of MHz while many high-performance process technologies target a few GHz which results in a large amount of leakage energy consumption. For example, a percentage of leakage energy in our processor is less than 3% when the clock frequency is 200MHz.

The optimal cache associativity value depends on an application program. For example, a direct mapped cache is the best for ADPCM decoder while a 2-way set-associative cache is the optimal for JPEG and MPEG2 encoder with respect to the energy efficiency. In MPEG2 encoder, 2x energy scalability is achieved by our multi-performance processor.

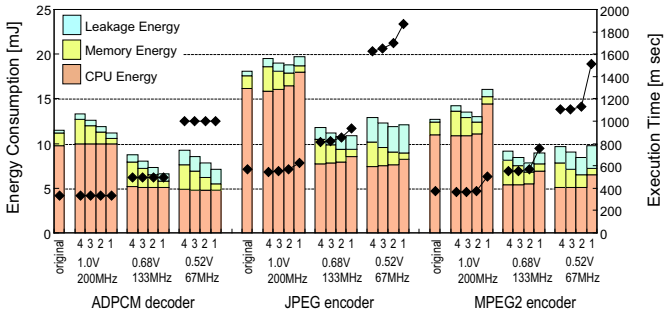

Figure 6: Energy Consumption and Execution Time

IV. Application

A. Real-Time Power Management

A major application of our multi-performance processor is a real-time system. According to the survey results published

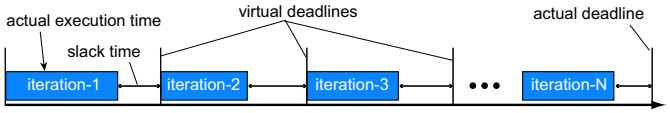

Figure 7: Intra-Task Voltage Scheduling

in [12], most real-time applications demand responses on the order of tens of microseconds, and many others require hundreds of microseconds and at most thousands of microseconds. For the applications which require responses of a few tens of microseconds, conventional DVS processors cannot be applied since the voltage transition time of the DVS processors is more than a few tens of microseconds. Dynamic task migration for a heterogeneous multi-processor is also ineffective for reducing the energy consumption since the task migration also takes a few tens of microseconds. Even for applications which require hundreds or thousands of microseconds, time and energy overheads in conventional approaches for changing the processor speed cannot be neglected since the processors have to change their speeds every thousands of microseconds for exploiting the processor slack time without any deadline misses. The following subsection shows how our multi-performance processor exploits the processor slack time in real-time applications which require responses on the order of thousands of microseconds.

B. Intra-Task Voltage Scheduling

A good application of our multi-performance processor is a media application where a single main loop spends the most of the execution time. The energy consumption can be drastically reduced by dynamically selecting the operating voltage and the clock frequency for each iteration of the loop according to proximity to a virtual deadline of each iteration. The virtual deadlines of the loop iterations have to be set beforehand as shown in Figure 7. Actual execution time for the single iteration can be measured by a timer module of the processor. If cumulative slack time is large enough for completing the next loop iteration by the virtual deadline even if the lower clock frequency is used, the processor uses the lower operating voltage and the clock frequency for the next loop iteration.

This voltage scheduling can be done by embedding a checkpoint at the top of the loop as shown in Figure 8. Every time a program reaches to the checkpoint, a timer module is accessed and the actual execution time of the loop iteration is obtained. Then the cumulative slack time is calculated. Pro-

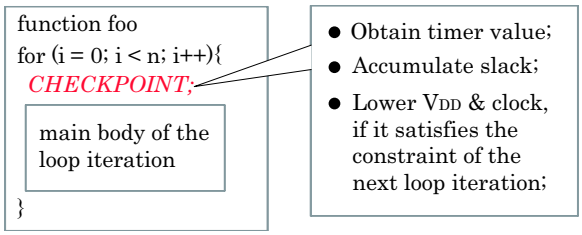

Figure 8: DVS Control using Program Checkpoints

cessor speed and operating voltage for the next loop iteration are lowered if they still satisfy the time constraint of the next loop iteration. The idea of intra-task voltage scheduling using program checkpoints is proposed in [18]. The main focus of [18] is a compilation technique for generating the program checkpoints automatically for reducing the energy consumption in DVS processors. This paper is focused on the evaluation of our multi-performance processor when a program with checkpoints runs on the processor.

C. Experimental Setup

The energy and time overheads required for changing the clock frequency of processors are shown in Table 3. These values are based on results obtained by our original multi-processor designed using 90nm process technology. In [13], an energy efficient DC-DC converter is presented. Internal capacitance of the DC-DC converter is $5\mu F$ while typical internal capacitance is $100\mu F$. The delay and energy overheads required for changing the output voltage of the converter are 20x smaller than those of the typical DC-DC converter. DVS-2 in Table 3 corresponds to a DVS processor with this DC-DC converter. Heterogeneous MP represents a heterogeneous multi-processor where the performances and energy consumptions of the CPU cores are different from each other as presented in subsection II.B. In the heterogeneous MP, if there is enough slack time to slow down the CPU speed, a task is migrated from a high performance CPU to a lower performance CPU for saving the energy consumption.

Table 3: Voltage and Clock Transition Overheads

Processor	Transition	time	energy
DVS-1 [9]	High → Middle	$500\mu s$	3,276nJ
	Middle → High	$500\mu s$	3,276nJ
DVS-2 [13]	High → Middle	$26\mu s$	164nJ
	Middle → High	$26\mu s$	164nJ
Heterogeneous MP	Task Migration	$30\mu s$	3,000nJ
Multi-Performance Processor	High → Middle	$1.1\mu s$	11.84nJ
	Middle → High	$1.0\mu s$	13.04nJ

The average and the worst case execution times (WCETs) for each iteration of main loops in ADPCM decoder, JPEG encoder and MPEG2 encoder are shown in Table 4, respectively. HMP and MPP represent a heterogeneous multi-processor and multi-performance processor, respectively. We assume that the WCET is 1.2 times of the longest measured execution time for the loop iterations. The measured execution times of the target loop iterations are obtained by RTL simulations and the longest ones are selected for ADPCM, JPEG and MPEG2, respectively for estimating the worst case execution times.

The average power consumption of ADPCM decoder, JPEG encoder and MPEG2 encoder executed on a conventional DVS processor, heterogeneous multi-processor, and our multi-performance processor are shown in Table 5. Note that the DVS processor uses 0.82V for 133MHz operations while our multi-performance processor uses 0.68V for 133MHz operations. These values are based on the results presented in Figure 5. Since each CPU core of the heterogeneous multi-processor can be optimally designed for a specific supply voltage, it is also supposed to use 0.68V for 133MHz operational clock.

Table 4: Average and Worst Case Execution Time [ms]

	DVS & HMP		MPP			
clock [MHz]	200	133	200		133	
# ways	2	2	2	1	2	1
	Average Case Execution Time [ms]					
ADPCM	0.36	0.54	0.36	0.36	0.54	0.54
JPEG	1.8	2.8	1.8	2.0	2.8	3.1
MPEG2	2.3	3.4	2.3	3.0	3.4	4.5
	Worst Case Execution Time [ms]					
ADPCM	0.45	0.67	0.45	0.45	0.67	0.67
JPEG	2.4	3.6	2.4	2.7	3.6	4.0
MPEG2	3.3	4.9	3.3	4.7	4.9	7.1

At the checkpoint of the program, the processors select one of clock frequencies so that the energy consumption is minimized with satisfying the time constraint of the next loop iteration even if the worst case occurs. Our multi-performance processor also selects one of cache associativity values so that the energy consumption is minimized. For example in ADPCM, our processor selects a direct map cache while the other processors use a fixed 2-way set-associative cache.

Table 5: Average Power Consumption [mW]

	DVS		HMP		MPP			
clock [MHz]	200	133	200	133	200		133	
# ways	2	2	2	2	2	1	2	1
ADPCM	34	18	34	14	35	32	14	12
JPEG	32	17	33	13	33	31	13	11
MPEG2	34	18	35	14	35	32	14	12

D. Experimental Results

Figure 9 shows the energy comparison results. TCs shown at the bottom of the figure represent time constraints. ORIG represents a processor always using a 1.0V voltage supply and a 200MHz clock. We suppose that every processors can be immediately shutdown if a task running on the processors is completed. As can be seen from the results, the energy consumption of our multi-performance processor is smallest of all in every cases. More specifically, our multi-performance processor reduces the energy consumption by 33% compared to the DVS processor having an energy efficient DC-DC converter. Although the heterogeneous multi-processor is energy efficient, its large time overhead required for changing the clock frequency prevents us to use it in real-time systems. Unlike the conventional heterogeneous multi-processor, our multi-performance processor realizes very low energy with short response time.

V. Conclusions

Intel recently has produced a multi-core processor which employs 80 CPU cores on a chip. This is a proof of the high

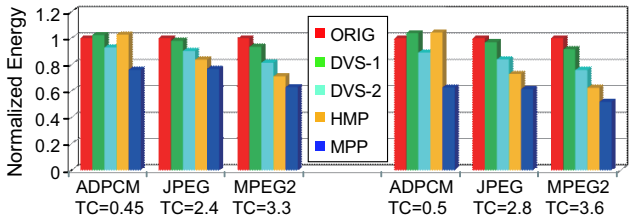

Figure 9: Energy Consumption Results

quality manufacturing process technology which makes it possible to integrate many cores on a chip. However, the problem is that there are not so many applications which require 80 parallel processing cores. Especially for embedded applications, since they do not always need their peak performances, multi-performance heterogeneous characteristics and quickly transitioning their performances are preferred rather than many parallel processing cores.

This paper presents a multi-performance processor which can be used as an alternative for the DVS processors. The processor realizes 2x energy scalability with preserving the peak performance of the processor. The gate-level simulation results demonstrate that our processor can change its performance within 1.5 microsecond and dissipates only about 10 nano-joule for the performance transition. These small overheads are more than two orders of magnitude less than those of conventional DVS processors. This makes it possible to apply our processor to many real-time systems and to perform finer grained and more sophisticated dynamic voltage control. The paper also presents how the multi-performance processor can be effectively used in embedded real-time applications. The experimental results obtained using ADPCM decoder, JPEG and MPEG2 encoders demonstrate that the multi-performance processor reduces the energy consumption by 25% compared to the conventional DVS processors. Our future work will be devoted to come up with more general algorithms for selecting optimal PE-core and cache ways statically by a compiler and dynamically by a real-time OS. Investigating new multi-performance architectures which increase energy and performance scalability is our future work as well.

ACKNOWLEDGMENT

This work is supported by Toshiba and VDEC, the Univ. of Tokyo with the collaboration of Renesas Technology, STARC, Panasonic, NEC Electronics, Toshiba, Synopsys, Cadence Design Systems and Mentor Graphics. This work is also supported by CREST ULP program of JST.

REFERENCES

[1] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processor," in Proc. of International Symposium on Low Power Electronics and Design, pp.197–202, Aug., 1998.

[2] T. Pering, T. Burd, and R. W. Brodersen, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," in Proc. of International Symposium on Low Power Electronics and Design, pp.76–81, Aug., 1998.

[3] T. Burd, T. Pering, A Stratakos and R. W. Brodersen, "A Dynamic Voltage Scaled Microprocessor System", IEEE Journal of Solid-State Circuits, vol.35, issue 11, pp.1571–1580, Nov., 2000.

[4] J. Pouwelse, K. Langendoen and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," in Proc. of International Conference on Mobile Computing and Networking, pp.251-259, July, 2001.

[5] T. Ishihara, S. Yamaguchi, Y. Ishitobi, T. Matsumura, Y. Kunitake, Y. Oyama, Y. Kaneda, M. Muroyama and T. Sato, "AMPLE: An Adaptive Multi-Performance Processor for Low-Energy Embedded Applications," in Proc. of International Symposium on Application Specific Processors, pp.83–88, June, 2008.

[6] T. Okuma, T. Ishihara, and H. Yasuura, "Real-Time Task Scheduling for a Variable Voltage Processor," in Proc. of International Symposium on System Synthesis, pp.24–29, Nov., 1999.

[7] Y. Shin, K. Choi and T. Sakurai, "Power Conscious Fixed Priority Scheduling for a Variable Voltage Processor," in Proc. of International Conference on Computer Aided Design, pp.365–368, Nov., 2000.

[8] R. Jejurikar, C. Pereira and R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems," in Proc. of Design Automation Conference, pp.275–280, June, 2004.

[9] S. Lee, and T. Sakurai, "Run-time Voltage Hopping for Low-power Real-time Systems," in Proc. of Design Automation Conference, pp.806–809, June, 2000.

[10] D. Shin and J. Kim, "Intra-task voltage scheduling on DVS-enabled hard real-time systems", IEEE Trans. on CAD of Integrated Circuits and Systems, Vol.24, Issue 10, pp.1530–1549, Oct., 2005

[11] N. Allah, Y. Wang, J. Xing, W. Nisar and A. Kazmi, "Towards Dynamic Voltage Scaling in Real-Time Systems - A Survey," International Journal of Computer Sciences and Engineering Systems, Vol.1, No.2, pp.93–104, April 2007

[12] J. A. Carbone, "RTOSes Balance Performance with Ease of Use," COTS Journal, November, 2004.

[13] T. Burd, and R. W. Brodersen, "Design Issues for Dynamic Voltage Scaling," in Proc. of International Symposium on Low Power Electronics and Design, pp.9–14, July, 2000.

[14] P. Yang and F. Katthoor, "Dynamic Mapping and Ordering Tasks of Embedded Real-Time Systems on Multiprocessor Platforms," in Proc. of International Workshop on Software and Compilers for Embedded Systems, pp.167-181, Sept., 2004.

[15] Toshiba Corp., "MeP Core (MeP-c4) User's Manual," http://www.semicon.toshiba.co.jp/eng/product/micro/

[16] D. H. Albonesi, "Selective Cache Ways: On-Demand Cache Resource Allocation," in Proc. of International Symposium on Microarchitecture, pp.248-259, Nov., 1999.

[17] S. M. Martin, K. Flautner, T. Mudge and D. Blaauw, "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads," in Proc. of International Conference on Computer Aided Design, pp.721-725, Nov., 2002.

[18] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A Veidenbaum, A. Nicolau "Profile-based Dynamic Voltage Scheduling using Program Checkpoints," in Proc. of Design Automation and Test in Europe, pp.168-178, March, 2002.