IPSJ Transactions on System LSI Design Methodology Vol. 2 167–179 (Aug. 2009)

Regular Paper

A Generalized Framework for Energy Savings in Hard Real-Time Embedded Systems

Gang Zeng,[†1] Hiroyuki Tomiyama[†1] and Hiroaki Takada[†1]

A dynamic energy performance scaling (DEPS) framework is proposed for energy savings in hard real-time embedded systems. In this generalized framework, two existing technologies, i.e., dynamic hardware resource configuration (DHRC) and dynamic voltage frequency scaling (DVFS) are combined for energy performance tradeoff. The problem of selecting the optimal hardware configuration and voltage/frequency parameters is formulated to achieve maximal energy savings and meet the deadline constraint simultaneously. Through case studies, the effectiveness of DEPS has been validated.

1. Introduction

Power and energy consumption has become one of the major concerns in today's embedded system design. Reducing power or energy consumption can extend battery lifetime of portable systems, decrease chip cooling costs, as well as increase system reliability. In contrast to the conventional hardware-centric low power designs, the software-centric energy performance tradeoff approach has attracted much attention recently due to its flexibility and easy implementation. This approach is based upon two observations. First, application needs for particular hardware resources such as caches, issue queues, and instruction fetch logic within an embedded processor can vary significantly from application to application [8]. This fact manifests the application-specific energy saving potential via dynamically turning off the unnecessary hardware resource according to the actual requirements of different applications. Second, in real-time systems the utilization of processor is generally less than 100% even if all tasks run at the worst case execution time (WCET). The fact of existing slack in real-time system reveals the chance for trading off performance for energy savings since the highest performance is not always necessary if the deadlines can be met. To take advantage of the above energy saving potential, software-centric approach attempts to provide "just fit" power and performance for different applications to minimize total energy consumption while meeting the deadline constraints simultaneously.

There are two kinds of commonly used power control technologies for exploiting the above energy saving potential. One is dynamic hardware resource configuration (DHRC), such as adaptive-issue queue [13], adaptive branch prediction [10], selective cache way [11], etc.. This technology tries to improve processor energy efficiency by dynamically tuning major processor resources in accordance with varied needs of applications [8]. In general, the more hardware resource the application requires, the higher power the processor dissipates. Therefore, DHRC saves energy by turning off the unnecessary hardware resource over application needs. DHRC is effective for exploring application-specific energy saving potential as the above first observation. However, its effectiveness on different applications is difficult to predict for two reasons. (1) DHRC is application-dependent, i.e., a specific DHRC technique may be effective for some applications, but may be ineffective for other ones [9]. (2) Even for a DHRC-effective application, the specific energy and performance relations under different hardware configurations are difficult to predict. The second technology for energy savings is dynamic voltage frequency scaling (DVFS) [1)–7),26),27]. Because the dynamic power consumption of CMOS circuits is proportional to its clock frequency and its voltage square, DVFS can save energy effectively through lowering both frequency and voltage of processor. Unlike DHRC, DVFS generally has similar effectiveness on different applications. That is, lowering frequency and voltage in a range always leads to longer execution time and less energy consumption. Therefore, DVFS is an effective method for energy savings by trading off energy and performance as the above second observation. Moreover, the variation of execution time and energy consumption for different voltage/frequency settings can be estimated by simple calculations. For example, most DVFS algorithms assume that the energy is proportional to the square of supply voltage and the execution time is inversely proportional to clock frequency.

†1 Graduate School of Information Science, Nagoya University

It is desirable to save more energy by combining the above technologies. This is because combining them can provide more chances for energy and performance tradeoff than DVFS or DHRC alone. Moreover combining them can overcome the limitation of each technology. For example, for system with high CPU utilization, the DVFS may be useless since there is not adequate slack to degrade performance for energy savings. In this case DHRC however can still save energy due to its less performance degradation. Unfortunately, it is not a trivial problem to combine the two technologies, especially for the hard real-time systems. The reasons are as follows: (1) While the energy consumption and execution time can be estimated by calculation after voltage/frequency scaling; they cannot be done so after hardware configuration is changed. Thus to guarantee deadline for DHRC application, the only way to obtain the energy time relation under a specific hardware configuration is measurement. (2) Combining DVFS and DHRC may result in so many possible configurations that the total measurement and computation time is unaffordable. (3) Consider the fact that one kind of hardware resource configuration may be effective for some applications, but may be useless for other applications. Thus a framework should have the capability to accommodate different hardware configuration mechanisms for various applications.

Based on different criteria, the software-centric energy performance tradeoff approaches can be classified into different categories. For example, according to the granularity at which the technologies are applied, they can be classified into inter-task and intra-task approaches. While the inter-task approach is targeted at different tasks or different jobs of the same task; the intra-task approach is applied for periodic intervals [26], program phases [11),12)] or subroutines [9] within one task. In addition, they can be classified into static (off-line) and dynamic (on-line) approaches according to when the decisions are made.

In this work, we propose a generalized framework, i.e., dynamic energy performance scaling (DEPS), to save more energy by combining two existing technologies. This framework is targeted at hard real-time embedded systems with preemptive scheduling policy. As a first step, we discuss its static inter-task based application in this paper. Generally, the static and inter-task based approach has global view of program power behaviors, low runtime overhead, simple implementation, and it is particularly suitable for task with known and stable workload. Through analysis of an actual DVFS application, Ref. 25) suggested that while simple dynamic DVFS might miss deadline during extreme workload variations, static DVFS generally is sufficient. In addition, it is shown in Ref. 9) that static application of DHRC achieved better energy savings than dynamic one due to its global information of program behaviors. Furthermore, though off-line static approach cannot handle dynamic variations of workload, it can often be used as a complement to on-line dynamic approach. The main contributions of this work are as follows: (1) Formulate the problem of selecting the optimal hardware configuration and CPU voltage/frequency to achieve the maximal energy savings and meet the deadline requirements simultaneously. (2) Propose a inter-task based static application scheme of DEPS and corresponding decision algorithm for selecting effective DEPS configurations. (3) Construct a simulation environment for evaluating the DEPS framework, and demonstrate its effectiveness through case studies.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 presents the proposed DEPS framework. Section 4 gives case studies. Finally, Section 5 summarizes the paper.

2. Related Work

There have been a large number of publications using DHRC or DVFS for energy and performance tradeoff in recent years. Pillai and Shin proposed off-line and on-line DVFS algorithms under fixed-priority and earliest deadline first (EDF) scheduling policy, respectively [1]. Kim, et al. evaluated various existing DVFS algorithms including both fixed-priority and EDF scheduling algorithm for hard real-time systems [2]. Saewong and Rajkumar proposed several off-line and on-line DVFS algorithms for fixed-priority real-time systems corresponding to processors with large or small DVFS overhead [3]. Mochocki, et al. introduced off-line DVFS algorithms for EDF scheduling policy considering time and energy overhead of DVFS [27]. Cho, et al. proposed DVFS algorithms considering measured energy performance relations for different applications [4]. Unlike the above inter-task approaches; Choi, et al. presented a fine-grained intra-task DVFS algorithm for memory bound application using performance counter for runtime

measurement [5]. In Ref. 6), Shin and Kim also proposed intra-task DVFS algorithm using control flow information for hard real-time systems. Recently, Yuan, et al. proposed cross-layer adaptation DVFS algorithm combining both inter-task and intra-task scaling for energy savings in a soft real-time application [7].

As far as DHRC is concerned, Albonesi proposed selective cache ways by using off-line program profiling and runtime program phase-based configuration [11]. Banerjee, et al. proposed completely dynamic cache ways configuration using hardware-based program phase detector [12]. Both the above approaches focused on data cache and utilized program phase information. In contrast, Huang, et al. proposed general subroutine-based hardware configuration approach [9] in which both off-line and on-line algorithms were proposed. In Ref. 10), Chaver, et al. also proposed subroutine-based branch predictor reconfiguration using access gating and structure resizing. Note that all these DHRC approaches performed fine-granularity configuration and were not targeted for hard real-time systems, which is different from the proposed approach. Albonesi, et al. summarized recent dynamically tuning processor resources approaches in Ref. 8).

Although the two technologies are effective for energy savings, there are few papers considering the combination of them due to the reasons discussed in Section 1. Huang, et al. first proposed the combination of DVFS and hardware resource reconfiguration for energy and temperature management in which an on-line interval-based algorithm was presented to select the most energy-saving configuration subject to a given slowdown factor [14]. While their work was targeted at single-task application with given slowdown factor, our approach is aimed at multi-task hard real-time application with given period and deadline. Recently, Nacul and Givargis proposed combination of DVFS and cache reconfiguration for low power [15]. Unlike our off-line optimal global exploration algorithm for all tasks, Ref. 15) used an on-line algorithm for selecting the Pareto-optimal configuration that best filled the slack for the next task to be executed. Moreover, our generalized framework can adopt various DHRC schemes, and not limited to cache reconfiguration.

3. DEPS Framework

The entire DEPS framework includes three layers, i.e., power controllable hardware, power aware software, and power analysis tools. **Figure 1** shows the framework and interactions between three layers. As software-centric approach, the DEPS engine is implemented in the scheduler of OS. The power analysis tools are employed to analyze and extract the power relative information. The power measurement tool is employed to obtain the specific energy time relations under each selected configuration. In addition, power analysis tools can insert power control hints into program to support fine-grained energy saving algorithm. Meanwhile, the power controllable hardware is critical. However, for specific applications the effective mechanism may be different, and the overhead for power control should also be considered. The framework can support both inter-task and intra-task based applications with coarse or fine granularity, although we only consider the inter-task based application with coarse granularity in this work.

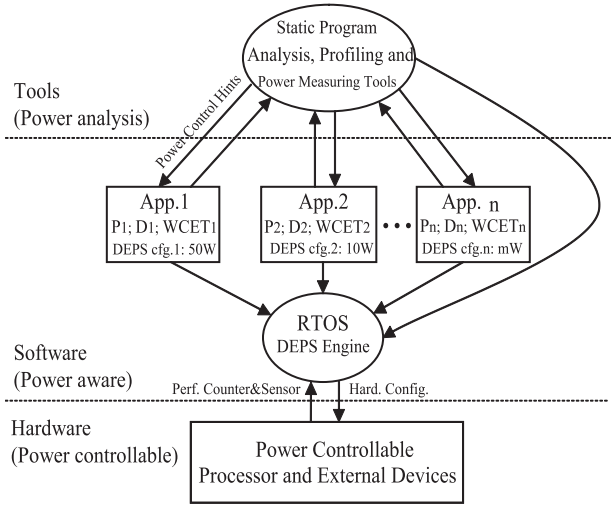

Fig. 1 DEPS framework.

3.1 System Model

This work focuses on embedded system and assumes a DHRC and DVFS enabled embedded processor. The DVFS can operate at a finite set of supply voltage levels, each with an associated speed.

We consider hard real-time applications consisting of a set of independent n

periodic real-time tasks, represented as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task τ_i has a period P_i and relative deadline D_i that is equal to the P_i. A task τ_i has m_i effective DEPS configurations $C_{i1}, C_{i2}, \ldots, C_{im_i}$ consisting of DHRC configuration and DVFS parameters. Each DEPS configuration C_{ij} is associated with a specific energy time relation, which can be represented by a pair of values (T_{ij}, E_{ij}) where T_{ij} is its worst-case execution time under this DEPS configuration, and E_{ij} is its energy consumption corresponding to the T_{ij}.

As mentioned earlier, this work mainly focuses on the static application of DEPS and assumes known and stable workload. This applies to many hard real-time systems which are the main target of this work. Therefore, we can utilize measurement to obtain the application-specific energy time relation under selected DEPS configuration and given input data. There are two reasons for the use of measurement. First, as described in Section 1, the energy consumption and execution time of a program is difficult to predict under different DEPS configurations. Second, while most DVFS papers use calculation for prediction after voltage/frequency scaling, recent research shows application-specific energy time relation through actual measurements, which can be exploited to further save energy over normal DVFS application [4),5)]. These application-specific power characteristics include external memory or other I/O devices access behaviors as well as leakage power consumption, etc., which are generally ignored in simple calculation.

3.2 Problem Formulation For Static Application of DEPS

We assume the overhead for task switching and DEPS configuration is negligible for simplicity, and denote *hyperperiod* as the least common multiple of all task periods. The energy optimization problem is to determine the set of DEPS configurations that minimize the energy consumption over a hyperperiod while meeting the deadline constraints. This problem can be formulated as follows:

Minimize energy:

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{hyperperiod}{P_i}(E_{ij} - T_{ij}W_{idle})x_{ij} \tag{1}$$

subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{T_{ij}}{P_i}x_{ij} \le n(2^{\frac{1}{n}} - 1) \tag{2}$$

or

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{T_{ij}}{P_i}x_{ij} \le 1 \tag{3}$$

and

$$\sum_{j=1}^{m_i}x_{ij} = 1, i = 1, 2, \ldots, n \tag{4}$$

where

$$x_{ij} \in \{0, 1\}, \forall i, j. \tag{5}$$

The above Eq. (1) is deduced from the following equations.
Task running energy in a hyperperiod is:

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{hyperperiod}{P_i}E_{ij}x_{ij} \tag{6}$$

CPU idle energy in a hyperperiod is:

$$\left(hyperperiod - \sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{hyperperiod}{P_i}T_{ij}x_{ij}\right)W_{idle} \tag{7}$$

The total energy consumption in the hyperperiod is thus the sum of task running energy and CPU idle energy where W_{idle} denotes the idle power of processor. Because the $hyperperiodW_{idle}$ is constant in Eq. (7), minimizing the total energy is equal to minimizing Eq. (1). Equations (2) and (3) represent utilization-based schedulability test for rate monotonic (RM) and EDF scheduling [17)], respectively. Note that more complex schedulability test such as response time analysis (RTA) [18)] can also be used for fixed-priority based scheduling at the expense of higher computational complexity. Equation (4) indicates that only one DEPS configuration can be selected for one task where $x_{ij} = 1$ denotes that the configuration C_{ij} has been selected for task τ_i, otherwise $x_{ij} = 0$.

It is clear that the problem for selecting the optimal DEPS configuration is

a typically multiple choice 0/1 knapsack problem, which is a known NP-hard problem [16]. While there is no polynomial-time exact method for this problem, we can use common methods for solving any reasonable size by off-line computation. In the following case studies, we use LPSolve tool [28], a free mixed integer linear programming solver, for solving this energy optimal problem.

Although we do not consider the configuration overhead for simplicity, they can be included in the above formulation. This is because the number of hardware configurations and DVFS settings occurred in one hyperperiod is known. Thus, if the DEPS overhead in terms of time latency and energy consumption for once hardware configuration and DVFS setting is also known, their influences can be incorporated into Eqs. (1), (2) or (3). The overhead of DVFS is generally higher than that of DHRC. While DVFS overhead is micro second level, DHRC is cycle level. A detailed discussion on the overhead of DHRC and DVFS configuration can be found in Refs. 9) and 27), respectively.

3.3 Decision Algorithm For Selecting Effective DEPS Configurations

Consider a DEPS framework with L voltage levels, and K DHRC configurations, we thus need perform $L \times K$ times measurement to obtain all possible energy time relations for one task. Also, $L \times K$ variables are involved in the optimal computation for one task. Fortunately, not all configurations are effective for trading off execution time for energy savings. Because even if these configurations are employed in the optimal computation, additional energy savings can not be achieved. For example, the configurations with increased execution time and energy consumption are ineffective. Therefore, we can reduce both measurement and computation time without sacrificing energy savings by only selecting the effective DEPS configurations which consume less energy than other configurations by prolonging execution time. As discussed in Section 1, since DVFS is effective for any applications, we retain all DVFS parameters directly. Then, to find the effective DHRC configurations, we assume that DVFS and DHRC do not interfere with each other. This assumption is reasonable because software generally exhibits similar behaviors (e.g., cache hit or miss) even if the CPU's frequency and voltage are different. Therefore, the following decision algorithm can be performed with specified DVFS parameters instead of all possible ones. As a result, only K times measurement are needed to find all effective DHRC configurations. After that, if DHRC has H effective configurations where $H \leq K$, then total $L \times H$ measurements are required. Finally, these obtained energy time relations at effective DHRC configurations and different DVFS parameters are employed in the optimal calculation.

The decision algorithm for finding the effective DHRC configurations for a given task is described as follows.

(1) Conduct K measurements to obtain all possible energy time relations in DHRC under any fixed DVFS parameter (e.g., the highest frequency and voltage).
(2) Sort the configurations as increasing execution time order.
(3) Select the configurations such that the selected ones have increasing execution time but with decreasing energy consumption. Obviously, those configurations are effective for energy savings by degrading performance.
(4) When two configurations have the same execution time, only the one with less energy consumption is selected.

Example for selecting the effective DEPS configuration is given in Section 4.2.

3.4 Implementation of Static DEPS

The implementation procedure of static DEPS mainly includes the following steps:

(1) Select effective DEPS configurations for each task as the decision algorithm given in Section 3.3.
(2) Obtain energy time relations associated with each effective DEPS configurations by measurement.
(3) Solve the energy optimal problem using the formulation described in Section 3.2 to obtain the optimal DEPS configuration for each task.
(4) Store the optimal DEPS configuration and corresponding hardware parameters into a static task configuration table.
(5) For each context switch, OS scheduler sets corresponding DEPS configuration for the next running task according to the static configuration table.

We use an example to illustrate the application of static DEPS. This simple example includes two periodic tasks and 7 effective DEPS configurations as shown in **Fig. 2**, where $C_{11}(1.0, 9)$ indicates that for DEPS configuration C_{11} of task 1,

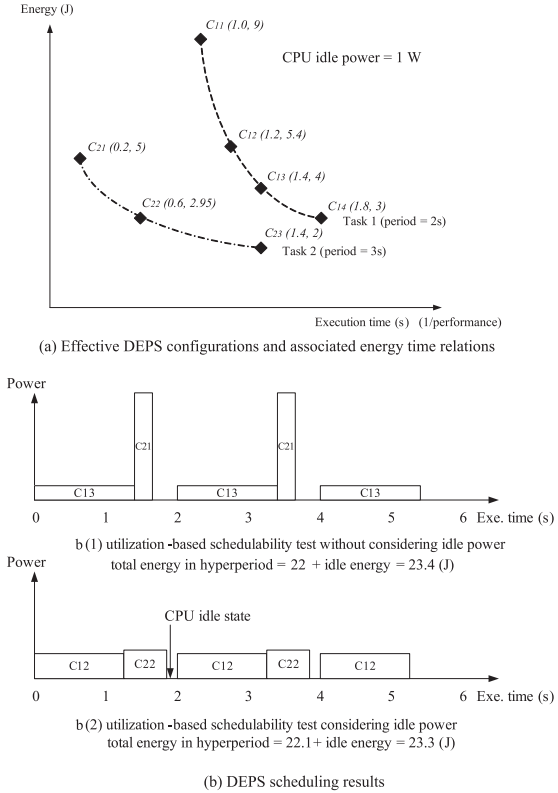

Fig. 2 An example for static DEPS application.

its corresponding worst-case execution time and energy consumption are 1.0 s and 9 J, respectively. The idle power of processor is assumed to be 1 W. The DEPS results for one hyperperiod scheduling are given in the same figure in which Fig. 2 b (1) and Fig. 2 b (2) denote the DEPS results by assuming W_{idle} is zero and 1 W, respectively, in Eq. (1). As can be seen, the selected optimal configurations can achieve the minimal energy consumption and meet the deadlines. Moreover, considering idle power in the formulation can lead to more energy savings than disregard of idle power.

4. Case Studies

As mentioned earlier, the achievable energy savings of DEPS are highly dependent on the employed DHRC and DVFS because DEPS can adopt diverse DHRC and DVFS techniques. Therefore, it is difficult to evaluate the absolute energy savings of general DEPS. For this reason, we use case studies to demonstrate the effectiveness of DEPS. In these case studies, we choose a 4-level voltage processor for DVFS, the selective cache way (SCW) [11] and configurable branch predictor (CBP) for DHRC in the DEPS framework. In Ref. 3), the relations between the energy savings and the number of voltage/frequency levels were investigated, and it was revealed that limited voltage/frequency levels will result in less energy savings for DVFS applications. However, while most general-purpose commercial DVFS processors can provide more voltage levels, embedded processors typically have less ones due to its relatively low running frequency. For example, the evaluation board of TMS320C5509 only provides 3-voltage levels [24]. The reason for selecting SCW is due to its easy implementation and low configuration overhead. SCW exploits the subarray partitioning of set associative caches in order to provide the capability to disable ways of the cache during periods where full cache functionality is not required to achieve energy savings. The detailed implementations of SCW, configuration overhead, as well as method for keeping data coherency can be found in Refs. 11) and 12). A configurable branch predictor is also assumed for the DHRC. The CBP can trade off power consumption for performance improvement by selecting different prediction methods. Generally, dynamic prediction can improve performance at the cost of extra hardware and power consumption. Note that our DEPS framework is general and independent of the employed DHRC and DVFS technologies. We simply choose the above technologies as examples of DEPS.

4.1 Simulation Environment Setup

As we focus on embedded systems, a SimpleScalar/ARM [19] based power simulator, Sim-Panalyzer [20], is employed to run power simulation in our experiments. Sim-Panalyzer is an infrastructure for microarchitectural power simulation considering both dynamic and leakage power. The ARM configurations are listed in **Table 1**. Note that we only implement the SCW on instruction cache to reduce

Table 1 SimpleScalar/ARM configuration.

Fetch queue	2
Branch Predictor	CBP
Fetch, Decode width	1
Issue width	1 (in-order)
Functional units	1 int ALU, 1 int Multiplier 1 FP ALU, 1 FP Multiplier
Instruction L1 Cache	Selective cache way (SCW)
Data L1 Cache	Size 8 KB; sets 64 block size 32-byte; 4-way
L2 Cache	None
Memory bus width	4-byte

Table 2 Instruction L1 cache SCW configurations.

Parameters	cfg.1	cfg.2	cfg.3
Cache size (KB)	8	4	2
Num. of sets	64	64	64
Block size	32	32	32
Associativity	4	2	1
Replacement policy	LRU	LRU	-

Table 3 Configurable Branch Predictor (CBP).

Dynamic Branch Prediction (DBP)	Bimodal 2 K entries 3 cycles extra penalty
Static Branch Prediction (SBP)	Not-taken

Table 4 DVFS parameters.

Processor frequency (MHz)	280	220	160	100
Processor voltage (V)	2.0	1.8	1.6	1.4

the configuration overhead associated with keeping data coherence. The possible configurations of SCW on L1 instruction cache are summarized in **Table 2**. The CBP has two configurable modes with different prediction accuracy and power consumption as shown in **Table 3**. In addition to the above configurations, special configurations for Sim-Panalyzer are retained. Furthermore, we incorporate the DVFS capability into the Sim-Panalyzer as the given parameters in **Table 4**.

Table 5 Benchmarks information.

Name of benchmark	Instruction count	Description
sha	13,537,444	secure hash algorithm
v42	5,876,830	modem encoding/decoding
engine	1,661,192	engine control application
g3fax	2,456,034	group three fax decode
cjpeg	15,945,744	jpeg encoder
tiff2rgba	36,946,334	image format conversion

Table 6 Synthetic task set 1 with high CPU utilization.

Task name	Period (ms)	WCET (ms) at 280 MHz 8 KB Icache and SBP	Total CPU uti.
g3fax	40	15.6	99.6%
engine	60	8.7	
v42	150	36.7	
sha	300	64.9	

Table 7 Synthetic task set 2 with low CPU utilization.

Task name	Period (ms)	WCET (ms) at 280 MHz 8 KB Icache and SBP	Total CPU uti.
engine	300	8.7	33.2%
g3fax	400	15.6	
cjpeg	800	95.2	
tiff2rgba	3,000	435.2	

Some benchmark programs from Mibench [21], Mediabench [22] and Powerstone [23] which exhibit different hardware resource requirements are used for the evaluations, and associated information is given in **Table 5**. Three synthetic task sets that correspond to high, low and medial CPU utilization are used to run on this ARM simulator with specified periods as given in **Tables 6**, **7** and **8**, respectively. The total CPU utilization of one task set is the sum of WCET/period of each task. Since the EDF scheduling has higher utilization bound than fix-priority scheduling as shown in Section 3.2, we only consider the EDF based scheduling in the case studies to evaluate the maximal potential for energy saving in DEPS.

To investigate the effectiveness of DEPS, we conduct experiments under six

Table 8 Synthetic task set 3 with medial CPU utilization.

Task name	Period (ms)	WCET (ms) at 280 MHz 8 KB Icache and SBP	Total CPU uti.
engine	100	8.7	67.4%
g3fax	200	15.6	
sha	400	64.9	
v42	400	36.7	
cjpeg	800	95.2	
tiff2rgba	3,200	435.2	

Table 9 Processor configurations in the case studies.

No.	Configuration name	Hardware configurations
1	DEPS 1	DVFS + SCW + CBP
2	DEPS 2	DVFS + SCW + SBP
3	DEPS 3	DVFS + 8 K Icache + CBP
4	DVFS alone	DVFS + 8 K Icache + SBP
5	SCW alone	280 MHz + SCW + SBP
6	CBP alone	280 MHz + 8 K Icache + CBP

kinds of processor configurations in the case studies as given in **Table 9**. The method is to run the above three task sets under these configurations and evaluate their average power consumptions during the hyperperiod. All energy results are obtained from the simulation results of Sim-Panalyzer by running individual benchmark program under different hardware configurations. In other words, we ignore the effect of OS scheduling and the configuration overhead of DVFS and DHRC during task switching as mentioned earlier. Moreover, we assume that the disabled parts of hardware consume negligible power via clock gating or power gating when full hardware resource is not required. For example, in case of SCW we obtain its energy consumption from Sim-Panalyzer by configuring the instruction cache as shown in Table 2. Note that the Sim-Panalyzer cannot run multiple tasks simultaneously, and cannot change its configuration during the execution of task. However, we can use the combinational results of individual benchmark for representing the running of multiple tasks because our DEPS framework does not change the configuration during the execution of same task.

4.2 Experimental Results

To illustrate the efficiency of the decision algorithm, we consider the most complex case, i.e., the DEPS 1 in Table 9. According to the above Tables 2, 3 and 4, there are 6 configurations for DHRC and 4 configurations for DVFS. The DEPS 1 can thus provide total 24 configurations. To investigate all possible energy time relations and validate the proposed decision algorithm, each benchmark is simulated 24 times using Sim-Panalyzer, which corresponds to 24 DEPS configurations. As can be seen from the simulation results in **Table 10**, for DVFS, lowering processor frequency and voltage always leads to longer execution time and less energy consumption. However, for DHRC, the energy performance tradeoff is highly dependent on program behaviors. For example in SCW, while the v42 requires large instruction cache (8 KB) to achieve better energy performance results, g3fax prefers small instruction cache (2 KB) because it leads to negligible degradation of performance but with significant energy savings. Similarly, application-dependent characteristic can be observed in CBP. For example, while all benchmarks except the engine can achieve high branch prediction accuracy (above 93%) and improved performance with DBP, the engine is relatively difficult to predict (86% accuracy) with DBP, resulting in worse performance due to frequent miss prediction penalty.

To select the effective DHRC configurations, the decision algorithm proposed in Section 3.3 is applied as the following steps. (1) 280 MHz/2.0 V is selected as the fixed DVFS parameters for simulation. (2) 6 DHRC configurations including different SCW and CBP are simulated for each benchmark respectively to obtain corresponding energy time relations. (3) Sort the configurations as increasing execution time order. (4) The configurations with increased execution time and decreased energy consumption are selected as effective configurations which are marked with * in the Table 10. Note that for configurations with the same execution time, only the one with less energy consumption is selected. Actually, if we repeat the above experiments with specified other DVFS parameters, similar results to the given one can be obtained for most benchmarks. This fact implies that the DHRC exhibits similar behaviors under different frequency and voltage settings. In summary, the DEPS with configuration 1 requires total 36 measurements to find the effective configurations for 6 tasks. Then, 68 rather

Table 10 Power simulation results of individual benchmark under possible DEPS configurations.

Bench.	SCW	CBP	280 MHz/2.0 V		220 MHz/1.8 V		160 MHz/1.6 V		100 MHz/1.4 V	
	conf.	conf.	T (ms)	E (mJ)	T (ms)	E (mJ)	T (ms)	E (mJ)	T (ms)	E (mJ)
sha	8 KB*	DBP *	**63.00**	**24.88**	**80.21**	**20.40**	**109.87**	**16.35**	**175.66**	**13.01**
		SBP	64.88	24.34	82.60	19.93	113.16	15.94	180.91	12.62
	4 KB*	DBP*	**63.02**	**21.38**	**80.24**	**17.55**	**109.90**	**14.09**	**175.69**	**11.27**
		SBP	64.91	20.99	82.64	17.23	113.19	13.83	180.96	11.06
	2 KB*	DBP*	**64.44**	**20.30**	**81.89**	**16.77**	**111.46**	**13.49**	**177.85**	**10.91**
		SBP*	**66.92**	**19.37**	**84.98**	**16.09**	**115.40**	**12.93**	**184.01**	**10.48**
v42	8 KB*	DBP*	**35.71**	**13.83**	**45.06**	**11.46**	**60.17**	**9.22**	**95.42**	**7.49**
		SBP*	**36.72**	**13.40**	**46.35**	**11.10**	**61.94**	**8.93**	**98.24**	**7.23**
	4 KB	DBP	42.45	15.03	52.89	12.83	67.47	10.44	105.43	8.87
		SBP	44.48	14.66	55.37	12.60	70.43	10.28	109.95	8.79
	2 KB	DBP	71.18	26.29	86.25	23.63	98.64	19.88	148.28	18.17
		SBP	72.90	25.82	88.36	23.54	101.28	19.71	152.42	18.15
engine	8 KB	DBP	8.83	3.43	11.23	2.80	15.41	2.25	24.64	1.79
		SBP	8.69	3.22	11.05	2.63	15.17	2.11	24.25	1.67
	4 KB*	DBP	8.83	2.98	11.23	2.44	15.41	1.96	24.64	1.57
		SBP*	**8.69**	**2.72**	**11.05**	**2.22**	**15.17**	**1.78**	**24.25**	**1.42**
	2 KB	DBP	14.19	4.94	17.46	4.37	21.25	3.59	32.68	3.18
		SBP	14.10	4.70	17.33	4.17	21.03	3.36	32.30	2.99
g3fax	8 KB	DBP	14.61	5.87	18.59	4.80	25.52	3.85	40.82	3.06
		SBP	15.56	6.00	19.80	4.90	27.19	3.92	43.48	3.10
	4 KB*	DBP*	**14.61**	**5.21**	**18.59**	**4.27**	**25.52**	**3.43**	**40.82**	**2.73**
		SBP	15.56	5.13	19.80	4.20	27.19	3.36	43.48	2.66
	2 KB*	DBP*	**14.62**	**4.89**	**18.60**	**4.01**	**25.53**	**3.22**	**40.83**	**2.57**
		SBP*	**15.58**	**4.71**	**19.82**	**3.85**	**27.20**	**3.09**	**43.50**	**2.46**
cjpeg	8 KB*	DBP*	**92.17**	**36.18**	**116.58**	**30.00**	**155.56**	**24.07**	**246.81**	**19.52**
		SBP	95.16	35.31	120.35	29.27	160.80	23.47	255.19	18.97
	4 KB*	DBP*	**94.54**	**32.91**	**119.82**	**27.65**	**158.13**	**22.13**	**250.33**	**18.18**
		SBP*	**97.66**	**31.17**	**123.70**	**26.24**	**163.49**	**20.97**	**258.88**	**17.22**
	2 KB	DBP	114.59	39.73	143.38	34.46	179.89	28.15	280.26	24.32
		SBP	119.37	37.81	149.18	33.13	187.24	26.86	291.67	23.32
tiff2rgba	8 KB*	DBP*	**432.25**	**148.39**	**509.94**	**128.15**	**603.14**	**102.49**	**909.07**	**89.84**
		SBP*	**435.20**	**141.56**	**513.68**	**123.72**	**608.28**	**98.16**	**917.45**	**86.82**
	4 KB*	DBP*	**438.90**	**140.99**	**517.70**	**122.64**	**610.52**	**98.41**	**919.28**	**87.21**
		SBP*	**441.99**	**133.34**	**521.59**	**117.54**	**615.86**	**93.54**	**928.00**	**83.75**
	2 KB	DBP	453.77	142.12	535.16	124.40	626.84	100.26	941.72	89.66
		SBP	457.34	134.09	539.56	119.31	632.81	95.17	951.33	85.98

Table 11 DEPS results using task set 1 (99.6% CPU utilization).

Task name and experimental results	DEPS 1		DEPS 2		DEPS 3	
	DHRC	F/V	DHRC	F/V	DHRC	F/V
sha	2 KB SBP	280/2.0	4 KB SBP	280/2.0	8 KB DBP	280/2.0
v42	8 KB DBP	280/2.0	8 KB SBP	280/2.0	8 KB DBP	280/2.0
engine	4 KB SBP	280/2.0	4 KB SBP	280/2.0	8 KB SBP	220/1.8
g3fax	2 KB SBP	280/2.0	2 KB SBP	280/2.0	8 KB DBP	280/2.0
Average power	319.85 mW		322.38 mW		365.72 mW	
Power reduction	20.2%		19.6%		8.8%	

Table 12 DEPS results using task set 2 (33.2% CPU utilization).

Task name and experimental results	DEPS 1		DEPS 2		DEPS 3	
	DHRC	F/V	DHRC	F/V	DHRC	F/V
engine	4 KB SBP	100/1.4	4 KB SBP	100/1.4	8 KB SBP	100/1.4
g3fax	2 KB SBP	100/1.4	2 KB SBP	100/1.4	8 KB DBP	100/1.4
cjpeg	4 KB SBP	100/1.4	4 KB SBP	100/1.4	8 KB SBP	100/1.4
tiff2rgba	4 KB SBP	100/1.4	4 KB SBP	100/1.4	8 KB SBP	100/1.4
Average power	60.33 mW		60.33 mW		65.87 mW	
Power reduction	85.0%		85.0%		83.6%	

than 144 simulations whose results are denoted in boldface in the Table 10 are performed. Finally, these measured results are used in the optimal computation. LPSolve tool [28] is employed for solving the energy optimal problem. Execution time of LPSolve for all experiments is less than 0.08 second in our computer with a 1.6 GHz Pentium processor and 1 GB RAM.

The evaluation of DEPS includes the same algorithm with three different configurations (i.e., DEPS 1, 2 and 3) as given in Table 9. DEPS results corresponding to high and low CPU utilization are reported in **Tables 11** and **12**, respectively, in which DHRC denotes the SCW and/or CBP configurations as given in Tables 2 and 3, and the F/V denotes the frequency voltage parameters as given in Table 4. It is clear from the results that more energy can be saved for lower CPU utilization since more slacks can be used for trading off performance for energy savings.

Table 13 compare the DEPS with other power saving methods using three task sets. Because the proposed DEPS is an inter-task based static method, we also select the inter-task based static application of DVFS and DHRC for fair comparison. In Table 13, the DVFS alone represents the optimal speed assignment algorithm proposed in Ref. 3) where different speeds are assigned statically to different tasks to achieve the maximal energy savings. The SCW and CBP alone represents the off-line application of DHRC where the hardware configurations with less energy consumption but can still meet the deadline constraints are selected statically for each task. Meanwhile, we assume that DVFS alone utilizes full hardware resource, SCW and CBP alone utilize the highest processor performance, which correspond to the hardware configurations with the same name in Table 9. Because the absolute energy consumption depends on the run time of application, we compare the average power of various methods to the maximal power consumption on this ARM-based simulator, i.e., 401 mW when running g3fax at 280 MHz, 8 KB instruction cache and DBP. As can be seen from Table 13, the DEPS with configuration 1 to 3 achieves 55.3%, 54.9%, and 49.4% power reduction, respectively. The results imply that the combination of DVFS and SCW is more effective for energy savings than combination of DVFS and CBP, and additional use of CBP can only achieve limited energy reduction than the combination of DVFS and SCW. Furthermore, the DEPS achieves consistently more energy savings than DVFS alone and DHRC alone. Specifically, the DEPS with configuration 1 achieves average 6.7%, 8.5%, and 16.1% improvement over DVFS alone, SCW alone and CBP alone respectively for various CPU utilizations. **Table 14** gives detailed results of comparison using task set 3.

Table 13 Power comparisons of different methods using three task sets.

Methods and configurations	Average power (mW) of different task sets			Average percentage power reduction
	High (99.6%) uti.	Medial (67.4%) uti.	Low (33.2%) uti.	
DEPS 1	319.85	157.02	60.33	55.3%
DEPS 2	322.38	159.28	60.33	54.9%
DEPS 3	365.72	177.26	65.87	49.4%
DVFS alone [3]	374.13	178.06	65.97	48.6%
SCW alone	322.38	213.31	104.25	46.8%
CBP alone	370.88	244.28	116.73	39.2%

Through the above case studies, we can derive the following conclusions from energy savings point of view.

Table 14 Configuration and power result of different methods using task set 3 (67.4% CPU utilization).

Task name and	DVFS alone [3]	SCW alone	CBP alone	DEPS 1		DEPS 2		DEPS 3	
experimental results	F/V	conf.	conf.	DHRC	F/V	DHRC	F/V	DHRC	F/V
sha	220/1.8	2 KB	SBP	2 KB SBP	220/1.8	2 KB SBP	220/1.8	8 KB SBP	220/1.8
v42	160/1.6	8 KB	SBP	8 KB DBP	160/1.6	8 KB SBP	160/1.6	8 KB SBP	160/1.6
engine	160/1.6	4 KB	SBP	4 KB SBP	220/1.8	4 KB SBP	160/1.6	8 KB SBP	160/1.6
g3fax	160/1.6	2 KB	DBP	2 KB DBP	160/1.6	2 KB SBP	160/1.6	8 KB DBP	220/1.8
cjpeg	220/1.8	4 KB	SBP	4 KB SBP	160/1.6	4 KB SBP	220/1.8	8 KB SBP	160/1.6
tiff2rgba	160/1.6	4 KB	SBP	4 KB SBP	160/1.6	8 KB SBP	160/1.6	8 KB SBP	160/1.6
Hyperperiod energy	569.80 mJ	682.58 mJ	781.68 mJ	502.46 mJ		509.68 mJ		567.24 mJ	
Average power	178.06 mW	213.31 mW	244.28 mW	157.02 mW		159.28 mW		177.26 mW	
Power reduction	55.6%	46.8%	39.1%	60.8%		60.3%		55.8%	

- DHRC is effective especially for systems with high CPU utilization since it can save energy with less performance degradation than DVFS. However, DHRC is ineffective in the case that all applications require the largest hardware resource (e.g., the v42 for SCW). Moreover, DHRC is less efficient in reclaiming slack time than DVFS in case of medial and low CPU utilization.
- DVFS is effective for systems with large slack time, because it can effectively reduce power consumption by reclaiming slack and degrading performance. On the other hand, DVFS may be useless for energy savings when system is with high CPU utilization and stable workload since the permitted performance degradation is limited.
- DEPS is more effective than either DHRC or DVFS in isolation since it can provide more chances for energy and performance tradeoff whatever the CPU utilization is high or low. The achievable energy savings of DEPS are highly dependent on the employed hardware configuration mechanisms and characteristics of system workload. In general, the more power controllable mechanisms the hardware provides, the more energy saving potential can be explored, and the more slack time in the systems, the more energy can be saved.

5. Conclusion

We proposed a generalized framework, i.e., DEPS: dynamic energy performance scaling for energy savings targeted at hard real-time embedded systems. It integrates two existing energy performance tradeoff technologies, i.e., dynamic hardware resource configuration and dynamic voltage frequency scaling into this framework. We formulate the problem of selecting the optimal DEPS configuration to achieve maximal energy savings and meet the deadline constraint simultaneously. As a first step, we propose static task-level application of DEPS. Through case studies, DEPS with different configurations shows consistently better energy savings than DVFS alone and DHRC alone under various CPU utilizations. For future work, we plan to evaluate the proposed DEPS on a more realistic platform with the consideration of DEPS configuration overhead, and to explore the possible dynamic application of DEPS for coping with varied workload.

Acknowledgments The authors would like to thank the anonymous reviewers for their constructive comments which helped improving the quality of this paper. This work is supported in part by the fund of Core Research for Evolutional Science and Technology, Japan Science and Technology Agency.

References

1) Pillai, P. and Shin, K.G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *Proc. ACM Symposium Operating Systems Principles*, pp.89–102 (2001).
2) Kim, W., Shin, D., Yun, H., Kim, J. and Min, S.L.: Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems, *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium* (*RTAS*), pp.219–228 (2002).

3) Saewong, S. and Rajkumar, R.: Practical Voltage Scaling for Fixed-Priority RT-Systems, *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium* (*RTAS*), pp.106–114 (2003).
4) Cho, Y., Chang, N., Chakrabarti, C. and Vrudhula, S.: High-Level Power Management of Embedded Systems with Application-Specific Energy Cost Functions, *Proc. Design Automation Conference* (*DAC*), pp.568–573 (2006).
5) Choi, K., Soma, R. and Pedram, M.: Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Tradeoff Based on The Ratio of Off-Chip Access to On-Chip Computation Times, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.24, No.1, pp.18–28 (2005).
6) Shin, D. and Kim, J.: Intra-Task Voltage Scheduling on DVS-Enabled Hard Real-Time Systems, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.24, No.10, pp.1530–1549 (2005).
7) Yuan, W., Nahrstedt, K., Adve, S.V., Jones, D.L. and Kravets, R.H.: GRACE-1: Cross-Layer Adaptation for Multimedia Quality and Battery Energy, *IEEE Trans. Mobile Computing*, Vol.5, No.7, pp.799–815 (2006).
8) Albonesi, D.H., Balasubramonian, R., Dropsbo, S.G., et al.: Dynamically Tuning Processor Resources with Adaptive Processing, *IEEE Computer*, pp.49–58 (2003).
9) Huang, M., Renau, J. and Torrellas, J.: Positional Adaptation of Processors: Application to Energy Reduction, *Proc. IEEE International Symposium Computer Architecture*, pp.157–168 (2003).
10) Chaver, D., Pinuel, L., Prieto, M., Tirado, F. and Huang, M.: Branch Prediction on Demand: An Energy-Efficient Solution, *Proc. International Symposium on Low-Power Electronics and Design*, pp.390–395 (2003).
11) Albonesi, D.H.: Selective Cache Ways: on-Demand Cache Resource Allocation, *Proc. International Symposium on Microarchitecture* (*MICRO*), pp.248–259 (1999).
12) Banerjee, S., Surendra, G. and Nandy, S.K.: Program Phase Directed Dynamic Cache Way Reconfiguration for Power Efficiency, *Proc. Asia and South Pacific Design Automation Conference* (*ASPDAC*), pp.884–889 (2007).
13) Buyuktosunoglu, A., et al.: A Circuit-Level Implementation of an Adaptive-Issue Queue for Power-Aware Microprocessors, *Proc. Great Lakes Symp. VLSI*, pp.73–78 (2001).
14) Huang, M., Renau, J., Yoo, S.M. and Torrellas, J.: A Framework for Dynamic Energy Efficiency and Temperature Management, *Proc. International Symposium on Microarchitecture (MICRO)*, pp.202–213 (2000).
15) Nacul, A. and Givargis, T.: Dynamic Voltage and Cache Reconfiguration for Low Power, *Proc. Design Automation and Test in Europe* (*DATE*), pp.1376–1377 (2004).
16) Martello, S. and Toth, P.: Knapsack Problems: Algorithms and Computer Implementations, Wiley (1990).
17) Liu, C.L. and Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, *Journal of the ACM*, Vol.20, No.1, pp.40–61 (1973).
18) Lehoczky, J.P., Sha, L. and Ding, Y.: The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior, *Proc. IEEE Real Time Systems Symposium* (*RTSS*), pp.166–171 (1989).
19) SimpleScalar Tools. http://www.simplescalar.com/
20) Sim-Panalyzer Project. http://www.eecs.umich.edu/~panalyzer/
21) Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T. and Brown, R.B.: MiBench: A Free, Commercially Representative Embedded Benchmark Suite, *IEEE Annual Workshop on Workload Characterization* (2001).
22) Lee, C., Potkonjak, M. and Mangione-Smith, W.H.: Mediabench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *Proc. International Symposium on Microarchitecture* (*MICRO*), pp.330–335 (1997).
23) Scott, J., Lee, L., Arends, J. and Moyer, B.: Designing the Low-Power M*CORE Architecture, *Proc. International. Symposium. on Computer Architecture Power Driven Microarchitecture Workshop*, pp.145–150 (1998).
24) Texas Instruments, Application Report, SPRA848A: Using the Power Scaling Library (2004).
25) Texas Instruments, Application Report, SPRAA19A: Power Management in an RF5 Audio Streaming Application Using DSP/BIOS (2005).
26) Lee, S. and Sakurai, T.: Run-Time Voltage Hopping for Low-Power Real-Time Systems, *Proc. Design Automation Conference* (*DAC*), pp.806–809 (2000).
27) Mochocki, B.C., Hu, X.S. and Quan, G.: A Unified Approach to Variable Voltage Scheduling for Nonideal DVS Processors, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.23, No.9, pp.1370–1377 (2004).
28) LPSolve tool: http://sourceforge.net/projects/lpsolve/

(Received November 17, 2008)
(Revised February 20, 2009)
(Accepted April 13, 2009)
(Released August 14, 2009)

(Recommended by Associate Editor: *Shinji Kimura*)

Gang Zeng graduated from Hunan University, China, with B.E., M.E. degrees in 1993, 2001, respectively. From 1993 to 1998, he was a lecturer at the Institute of Electric and Information Engineering, Hunan University. He received his Ph.D. degree in information science from Chiba University in 2006. He is currently an assistant professor at the Graduate School of Information Science, Nagoya University. His research interests include power-aware computing, embedded system design, design for testability, system-on-a-chip testing. He is a member of the IEEE.

Hiroyuki Tomiyama received his Ph.D. degree in computer science from Kyushu University in 1999. From 1999 to 2001, he was a visiting postdoctoral researcher with the Center of Embedded Computer Systems, University of California, Irvine. From 2001 to 2003, he was a researcher at the Institute of Systems & Information Technologies/KYUSHU. In 2003, he joined the Graduate School of Information Science, Nagoya University, as an assistant professor, where he is now an associate professor. His research interests include design automation, architectures and compilers for embedded systems and systems-on-chip. He currently serves as an editorial board member of IPSJ Transactions on SLDM, IEEE Embedded Systems Letters, and International Journal on Embedded Systems. He has also served on the organizing and program committees of several premier conferences including ICCAD, ASP-DAC, DATE, CODES+ISSS, and so on. He is a member of ACM, IEEE, IPSJ and IEICE.

Hiroaki Takada is a professor at the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. He received his Ph.D. degree in Information Science from University of Tokyo in 1996. He was a research associate at University of Tokyo from 1989 to 1997, and was an assistant professor and then an associate professor at Toyohashi University of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IEEE, IPSJ, IEICE, and JSSST.