2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

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

Gang Zeng, Hiroyuki Tomiyama, Hiroaki Takada,
Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
{sogo, tomiyama, hiro}@ertl.jp

Tohru Ishihara
System LSI Research Center, Kyushu University
3-8-33, Momochihama, Sawara-ku, Fukuoka, 814-0001, Japan
ishihara@slrc.kyushu-u.ac.jp

Abstract

A generalized dynamic energy performance scaling (DEPS) framework is proposed for exploring application-specific energy-saving potential in hard real-time embedded systems. This software-centric framework focuses on system-wide energy reduction and takes advantage of possible power control mechanisms to trade off performance for energy savings. Three existing technologies, i.e., dynamic hardware resource configuration (DHRC), dynamic voltage frequency scaling (DVFS), and dynamic power management (DPM) have been employed in this framework to achieve the maximal energy savings. Static and dynamic schemes of DEPS are proposed to deal with stable or variable workload in the embedded systems. Through a case study, its effectiveness 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 [4]. Furthermore, program behaviors with respect to access of I/O devices (e.g. external memory) are also application-dependent. 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 always less than 100% even if all tasks run at the worst case execution time (WCET). Moreover, the actual workload of the same task may vary from instance to instance, which depends on the specific input data and execution path. The fact of existing slack in real-time system reveals the chance to trade off performance for energy savings since the highest performance is not always required if the deadline can be met.

There are three kinds of frequently used power control technologies for energy performance tradeoff. One is dynamic hardware resource configuration (DHRC), such as adaptive branch predictor [6], selective cache way [7] etc.. This technology tries to improve processor energy efficiency by dynamically tuning major processor resources in accordance with varied needs of applications [4]. However, its effectiveness for energy savings is application-dependent, i.e., a specific DHRC technique may be effective for some applications, but may be ineffective for other ones [5]. The second technology is dynamic voltage frequency scaling (DVFS) [1, 2, 3, 27, 19]. Because the dynamic power consumption of CMOS circuits is proportional to its clock frequency and its voltage square, DVFS can save energy effectively by lowering both frequency and voltage of the processor. Unlike DHRC, DVFS generally has similar effectiveness on different applications. That is, lowering frequency and voltage in a range always leads to degraded performance and reduced energy consumption. Moreover,

978-0-7695-3492-3/08 $25.00 © 2008 IEEE
DOI 10.1109/EUC.2008.101

the variation of execution time and energy consumption after DVFS can be estimated by simple calculations. The third one is dynamic power management (DPM) which is generally employed to reduce the energy consumption of processor or device in idle state by transferring them to a low power mode. However, the transition from low power mode to the normal mode consumes additional time and energy. The challenge of DPM is to balance the energy savings and performance degradation by determining when the processor or device should be transferred to the low power mode, and what low power mode should be entered [24].

It is desirable to save more energy by combining the above technologies. Unfortunately, it is not a trivial problem, 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 DVFS, they cannot be done so after reconfiguration of hardware. Thus to guarantee deadline for DEPS application, the only way to obtain the energy time relation under a hardware configuration is physical or simulation measurement (hereinafter measurement for short). (2) Combining them may result in so many possible configurations that the total measurement and computation time is unaffordable. (3) Consider the fact that the efficiency of DHRC is application-dependent, thus a framework should have the capability to accommodate different hardware configuration mechanisms for various applications.

It has been known recently that the energy savings in processor does not necessarily lead to energy savings in whole system including external devices [25, 26]. Therefore, it is necessary to consider the energy savings of whole system but not the processor alone. In this work, we propose a generalized framework, called dynamic energy performance scaling (DEPS), to achieve system-wide energy savings in hard real-time systems by combining three existing power control technologies. The main contributions of this work are as follows. (1) Propose static and dynamic schemes of DEPS corresponding to stable or variable workload to reduce the total energy consumption of processor and external devices. (2) Propose an algorithm to reduce both measurement and computation time by selecting the effective DEPS configurations. (3) Construct a generalized simulation environment to evaluate the DEPS framework, and demonstrate its effectiveness via a case study.

The rest of the paper is organized as follows. Sec. 2 introduces the related work. Sec. 3 presents the proposed DEPS framework. Sec. 4 gives a case study and simulation results. Finally, Sec. 5 summarizes the paper.

2 Related work

Energy saving technologies using DVFS, DHRC and DPM have been extensively studied so far. Although these technologies are effective for energy savings, there are few papers considering the combination of them. Huang et al. first proposed the combination of DVFS and DHRC for energy and temperature management in which an on-line interval-based algorithm was presented for selecting the most energy-saving configuration subject to a given slowdown factor [9]. While their work is targeted at single-task application with the given slowdown factor, our approach is aimed at multi-task hard real-time application with given period and deadline. Fan et al. proposed the combination of DVFS and DPM for system-wide energy savings in which the DPM was employed to reduce the standby energy of external memory [25]. Nacul and Givargis proposed the combination of DVFS and cache reconfiguration for low power [10]. Unlike our off-line optimal global exploration algorithm for all tasks, [10] used an on-line algorithm for selecting the Pareto-optimal configuration that best fill the slack for the next task to be executed. Recently, Zeng et al. first proposed the combination of DVFS and DHRC for the fixed-priority real-time systems [11]. However, previous approach primarily focused on the processor energy reduction, and only dealt with the stable workload, which is different from this approach targeting at the system-wide energy reduction and dealing with both the stable and variable workload. Moreover, the paper did not discuss how to reduce the measurement and computation time, which makes it unrealistic for large task set.

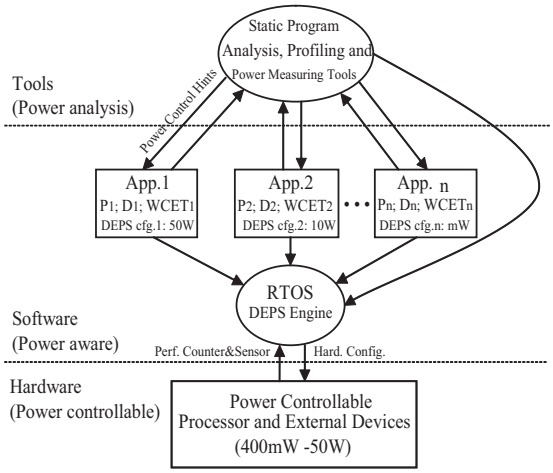

Figure 1. DEPS framework

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 the 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 mea-

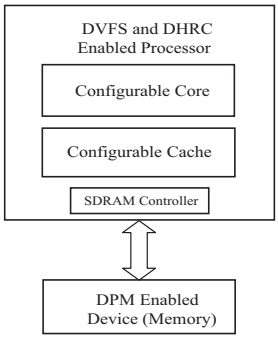

Figure 2. Target system

surement tool is utilized to obtain the energy time relations under each selected configuration.

3.1 System model

This work focuses on the embedded systems and assumes a DHRC and DVFS enabled embedded processor. Since our objective is to achieve system-wide energy savings, the main memory, a representative of external device, is included into our framework as shown in Fig.2. The energy model of external memory and DPM for the reduction of memory energy are given in Sec. 4.1 in detail. Note that other devices with DPM capability can also be incorporated into the model.

We consider hard real-time applications including a set of independent n periodic real-time tasks, represented as $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$. Each task τ_i has a period P_i and relative deadline D_i that is equal to P_i. A task τ_i has m_i effective DEPS configurations $C_{i1}, C_{i2}, ..., C_{im_i}$ consisting of DHRC configuration, DVFS parameters, and DPM policies. 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 the worst-case execution time under the C_{ij} configuration, and E_{ij} is the energy consumption of processor and external device during T_{ij}.

3.2 Problem formulation for static scheme

We assume that the overhead for task switching and DEPS reconfiguration is negligible for simplicity, and the energy time relations of the effective DEPS configurations have been obtained in advance. We denote $hyperperiod = LCM(P_1, P_2, ..., P_n)$, i.e., the least common multiple of all task periods. Then, the energy optimization problem is to determine the optimal DEPS configuration for each task such that the total energy consumption over a hyperperiod is minimized and all deadline constraints are met. We extend the formulation defined in [11] to include both fixed-priority and dynamic priority scheduling as follows:

Minimize energy:

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

subject to

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

or

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{T_{ij}}{P_i}x_{ij} \leq 1 \quad (3)$$

where

$$\sum_{j=1}^{m_i} x_{ij} = 1,\ i = 1, 2, ..., n,\ x_{ij} \in \{0, 1\},\ \forall i, j \quad (4)$$

In the above equation (1), the W_{idle} denotes the idle power of processor and device. The equations (2) and (3) represent utilization-based schedulability test for rate monotonic (RM) and EDF scheduling, respectively [13]. Equation (4) indicates that for one task, only one DEPS configuration can be selected 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 NP-hard problem [12]. 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 study, we use LPSolve tool [20], a free mixed integer linear programming solver, to solve this energy optimization problem. Although we do not consider the configuration overhead in the above formulation for simplicity, they can be included as discussed in [11].

3.3 Algorithm for selecting effective DEPS configurations

Consider a DEPS framework with L voltage levels, Q DPM policies, as well as K kinds of DHRC (each DHRC has $F_j (j = 1 \sim K)$ configurations), we thus need to perform $L \times Q \times F_1 \times F_2 \times ... \times F_K$ times measurements to obtain all possible energy time relations for one task. Furthermore, the same number of variables will be employed for one task in the optimization computation. Fortunately, not all configurations are energy efficient. That is, only the DEPS configurations those have less energy consumption than any other configurations with same or shorter execution time are effective for energy and performance tradeoff. Therefore, we can reduce both measurement and computation time by only selecting effective DEPS configurations. As discussed in Sec. 1, since DVFS is effective for any applications, we retain all DVFS parameters directly. Then, to find the effective configurations in one kind of DHRC, we assume that different DVFS parameters or DPM policies do not affect the selection of effective DHRC parameters, which has been confirmed in our case study and also

was suggested in [5]. For example, the appropriate instruction cache size is dependent on the characteristics of program but not voltage/frequency of the processor. Therefore, the measurements can be performed separately, each time for one kind of DHRC or DPM policy and with fixed the other parameters, thus only $Q + F_1 + F_2 + ... + F_K$ times measurements are needed to find all effective configurations and policy in DHRC and DPM. After that, if each DHRC has $H_j (j = 1 \sim K)$ effective configurations where $H_j <= F_j$, and only one DPM policy is effective, then the total $L \times (H_1 + H_2 + ... + H_K)$ times measurement are required under different DVFS parameters for each task in the optimization computation.

In the above procedure, we use the following algorithm to find the effective DHRC parameter in one kind of DHRC. First, we conduct F_j measurements to obtain all possible energy time relations in one kind of DHRC. Second, sort the configurations in increasing execution time order. Third, check the configurations from the first one to the last one, and configurations that have less energy consumption than any previous configurations are selected as the effective configurations. The above algorithm can also be used for the selection of effective DPM policies. Note that we can further reduce measurement and computation time at the cost of more energy consumption over the optimal one by only selecting the most effective DEPS configurations and ignoring other ones. The energy efficiency of each effective configuration can be evaluated as reduced energy / increased execution time, which is compared with the configuration with the shortest execution time.

3.4 Implementation of static scheme

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

1. Select effective DEPS configurations for each task as the algorithm given in Sec. 3.3.
2. Obtain energy time relations associated with each effective DEPS configurations by measurement.
3. Solve the energy optimization problem using the formulation described in Sec. 3.2 to obtain the optimal DEPS configuration for each task.
4. Store the optimal DEPS configurations and the associated configuration parameters into a static configuration table.
5. For each context switch or dispatch of task, the OS scheduler sets the optimal DEPS configuration for the next task to run according to the static configuration table when the current DEPS configuration is not the expected one.

3.5 Dynamic scheme

As described earlier, the static scheme is suitable for stable WCET workload. However, embedded systems often exhibit fluctuation of workload in practice due to data dependence of program behaviors. In the case of early completion of task, dynamic slack can be employed to further save energy. To this end, we propose a dynamic scheme of DEPS for additional energy savings by reclaiming runtime slack. Some dynamic DVFS algorithms have been proposed in literature to reclaim dynamic slack [1, 2, 3, 21]; they however cannot be directly applied to DEPS due to the difference between them. The key difference is that while most existing DVFS algorithms assume constant number of cycles for each task even if voltage and frequency have been changed during the execution of task, this assumption is no longer held for DEPS. It is evident that when DEPS configuration is changed such as cache size, branch prediction etc., the number of cycles required for task execution is also changed. As a result, the left execution time of task will become unpredictable if its DEPS configuration is changed during execution, and ignoring this fact may lead to miss of deadline. For this reason, DEPS configurations are merely allowed to change at the first dispatch time after release of the task in our dynamic scheme. In other words, even if the preempted task gets the slack from the early completion of higher priority task, it cannot change its DEPS configuration for the remaining execution time.

Some definitions and notations used in the algorithm are given as follows:

- NTA_i: the earliest arrival time of next task from current dispatch time of task τ_i.
- AET_i: the actual execution time of task τ_i measured by OS.
- Usable slack: the slack time can be used by next task. It is equal to the total slack minus the CPU idle time.
- EET_i: the effective execution time for task τ_i running without miss of deadline.

As an example, Fig.3 shows the relative time relation of the notations. The complete algorithm for dynamic DEPS is presented in Fig.4. In brief, the dynamic scheme includes two steps, i.e., off-line static optimization algorithm and on-line dynamic slack reclaiming algorithm. The on-line algorithm can be further divided into two processes, i.e., slack detection process and DEPS configuration updating process. The basic rules of the algorithm are described as follows. (1) The default configuration is obtained by using static DEPS scheme in which the schedulability is guaranteed even for the WCET. (2) When a task is released, it is assumed that at least the WCET of the default configuration is required to execute. (3) Only the slack generated by high priority task can be used by low priority task. (4) If only one task exists in the ready queue, the NTA time can be used for this task. As proved in [3, 21], these rules can guarantee the schedulability of the dynamic scheme.

To illustrate how to update the DEPS configuration using the detected slack, we utilize the configuration table of

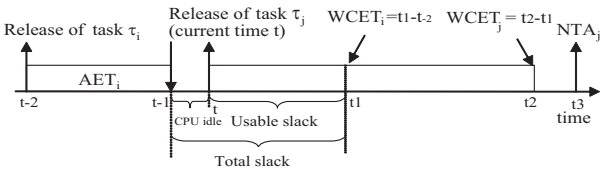

Figure 3. Relative time relations of notations.

Dynamic scheme of DEPS

- ☐ STEP 1: static (off-line) optimization algorithm
  - off-line implementation to obtain the static optimal DEPS configurations which are used as the default configurations
  - construct the DEPS config. table for each task using the optimal and other effective DEPS configurations
- ☐ STEP 2: dynamic (on-line) slack reclaiming algorithm
  - implemented by OS scheduler when task is completed or released
  - assume task τ_i has been completed before release of task τ_j

Dynamic (on-line) slack reclaiming algorithm

- ☐ Completion of task τ_i
  - slack = EET_i – AET_i; // total slack
  - record the priority of task τ_i
- ☐ Dispatch of task τ_j at time t (excluding the redispatch after preemption)
  - EET_j = $WCET_j$; //WCET of task τ_j under static optimal DEPS configuration
  - if only one task in the ready queue and NTA_j > t + $WCET_j$ then
    - ☐ EET_j = NTA_j – t;
  - if the priority of task τ_i is higher than that of task τ_j then
    - ☐ slack = slack – CPU idle time; // usable slack
    - ☐ if slack < 0, then slack=0;
    - ☐ if $WCET_j$ + slack > EET_j then EET_j = $WCET_j$ + slack
  - if EET_j > $WCET_j$ then search DEPS config. table to find new config. $WCET_j$* such that $WCET_j$* is the largest one but <= EET_j
    - ☐ $WCET_j$ = $WCET_j$*;
    - ☐ configure the processor hardware as the updated DEPS config.

Figure 4. Algorithm of dynamic scheme.

cjpeg benchmark in Table 1 as an example. The first entry of the table should be the optimal static DEPS configuration obtained as described in Sec. 3.2, and the others are some of effective configurations that have longer execution time and less energy consumption than the optimal configuration. These configurations are sorted in increasing WCET or decreasing energy order. Dynamic scheme attempts to find new DEPS configuration with less energy through absorbing dynamic slack. For example, when cjpeg task obtains a 100ms slack from the higher priority task, then it can update its current DEPS configuration to 3 which consumes less energy than the configuration one.

4 A case study

As mentioned earlier, the achievable energy savings of DEPS are highly dependent on the employed technologies. For this reason, we use a case study to demonstrate the effectiveness of DEPS. In this case study, we choose a 4-level voltage processor for DVFS and the selective cache way (SCW) [7], configurable branch predictor (CBP) for DHRC in the DEPS enabled processor. And, the processor is supposed to equip with two 256MB mobile DDR

Table 1. DEPS configuration table of cjpeg.

No.	DEPS configuration parameters	WCET(ms)	Energy(mJ)
1	160MHz/1.6V; EBP; 4k Icache; DPM2	159.32	26.21
2	100MHz/1.4V; EBP; 8k Icache; DPM2	247.54	24.04
3	100MHz/1.4V; EBP; 4k Icache; DPM2	251.22	23.32

Table 2. SimpleScalar/ARM configuration.

Fetch queue	2
Branch Predictor	configurable
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 8KB; sets 64 block size 32-byte; 4-way
L2 Cache	None
Memory bus width	4-byte

SDRAM chips with 32-bit width. We select the 4-level DVFS because embedded processors typically have less voltage levels than general-purpose processors. For example, the TMS320C5509 only provides 3-level voltages. The reason for selecting SCW and CBP is their easy implementation and low configuration overhead. The detailed description of SCW and its configuration overhead can be found in [7, 8]. Note that our DEPS framework is general and independent of the employed DHRC and DVFS technologies. We simply choose the above technologies as an example of DEPS.

4.1 Simulation environment setup

A SimpleScalar/ARM [14] based power simulator, Sim-Panalyzer [15], is employed to measure energy and time in our experiments. Sim-Panalyzer is an infrastructure for microarchitectural power simulation considering both dynamic and leakage power. Default configuration is used for Sim-Panalyzer. The ARM configuration of SimpleScalar is listed in Table 2. The configurations of CBP, SCW, DVFS are given in Table 3, Table 4, and Table 5, respectively. We only implement the SCW on instruction cache to avoid large configuration overhead for keeping data coherence. Furthermore, DVFS capability has been incorporated into the Sim-Panalyzer.

Figure 5 shows the power state transition graph of the employed SDRAM chip. As can be seen, this chip can provide multiple low power modes for different power-saving levels. An access count based energy model is employed to calculate the energy consumption of external memory which is composed of standby energy and access energy. To save the standby energy, we propose two DPM policies using different low power modes. Both DPM policies can

Table 3. Branch prediction configuration.

Enable Branch Prediction (EBP)	Bimodal 2K entries; 3 cycle penalty
Disable Branch Prediction (DBP)	Not-taken; 3 cycle penalty

Table 4. SCW configurations for L1 Icache.

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	LRU

Table 5. DVFS parameters.

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

Table 6. Synthetic task set.

Task name	Period (ms)	WCET(ms) at 280MHz & EBP	Total CPU uti.
sha	200-800	63.0	26.2-95.8 %
v42	200-800	35.7	
engine	100-200	8.8	
g3fax	100-400	14.6	
cjpeg	400-1600	92.2	

be implemented by the SDRAM controller. Table 7 summarizes the energy model, DPM policies, as well as the associated parameters used in the simulation. This energy model including DPM capability has been integrated into the original Sim-Panalyzer to calculate the runtime energy consumption of external memory in cycle accuracy. For simplicity, the power consumption of processor and memory is assumed to be zero during idle state of OS. Some benchmark programs from MiBench [16], MediaBench [17] and Powerstone [18] are selected for the evaluation. A synthetic task set consisting of these benchmark programs is assumed to run on the ARM simulator using fixed-priority scheduling with specified periods as given in Table 6.

4.2 Simulation results of static scheme

According to the above Table 3, 4, and 7, there are 6 configurations for DHRC, 4 configurations for DVFS, and 3 policies for DPM. The DEPS framework can thus provide total 72 configurations in this case study. To observe all possible energy and time relations under different configurations, each benchmark is simulated 72 times using Sim-Panalyzer. Due to the space limitation, only partial simulation results are given in Table 8. Since DPM2 achieves consistently better results than DPM1, only the results of DPM0 and DPM2 are given in the table. As can be seen, different benchmarks present distinct hardware requirements. For example, V42 requires large size of instruction cache. Otherwise, small cache will lead to more energy due to remarkably increased cache misses and execution time. In contrast, small cache is better for g3fax, since small cache results in negligible performance degradation and significant energy reduction. The selected effective DEPS configurations for each benchmark as the proposed algorithm are denoted in boldface in the table. As can be seen in the Table 8, only 8 and 4 measurements are necessary for g3fax and v42, respectively, rather than the 72 measurements. In summary, total 36 instead of 360 measurements or variables are required for 5 tasks in the optimization computation of the case study. Execution time of LPSolve [20] in all experiments is less than 0.02 second on a computer equipped with a 1.6GHz Pentium processor and 1GB RAM.

Figure 6 shows the comparisons of DEPS and the existing methods at different CPU utilizations. Because the proposed DEPS is an inter-task based static method, we also select the inter-task based static DVFS [1, 3] and static DHRC for fair comparison. In addition, we assume that both methods are without using DPM for external memory; the static DVFS utilizes full hardware resource; and the static DHRC utilizes the highest processor performance. Because the energy consumption is dependent on the run time of application, we compare the average power of several methods over the hyperperiod to the maximal power consumption in this ARM-based simulator, i.e., 475.2 mW when running cjpeg at 280 MHz on 8k Icache with EBP. As can be seen from Fig.6, the DEPS can achieve average 62.7% power reduction, and average 13.6% and 13.7% improvement over the DVFS alone and the DHRC alone respectively. Also, the results indicate that the DEPS can achieve consistently better results than DHRC or DVFS alone for any CPU utilization. This is because the DEPS can provide more chances for energy performance tradeoff than any existing technology alone. To evaluate the effect of DPM, we perform additional experiment without using any DPM. Results show that DEPS without DPM achieves 10% less energy savings than DEPS with DPM2 at medial CPU utilization.

4.3 Simulation results of dynamic scheme

To evaluate the efficiency of the dynamic scheme, we utilize the same synthetic task set as shown in Table 6. Unlike the static scheme, workload is supposed to vary from 20 to 100 percent of the WCET in the dynamic scheme. For the calculation of energy, we assume that the average power remains constant for each DEPS configuration whatever the actual execution time is. The results of static scheme are employed as the default configurations, and the dynamic DEPS configuration table is constructed as described in Sec. 3.5. We build an evaluation environment to simulate the ex-

Table 8. Partial simulation results of g3fax and V42.

Name of benchmark	DEPS configuration parameters		280MHz/2.0V		220MHz/1.8V		160MHz/1.6V		100MHz/1.4V	
			T(ms)	E(mJ)	T(ms)	E(mJ)	T(ms)	E(mJ)	T(ms)	E(mJ)
g3fax	4K 2-way I cache	DBP+DPM0	20.04	6.60	25.49	5.94	35.02	5.64	56.01	6.17
		DBP+DPM2	20.06	5.44	25.51	4.47	35.04	3.61	56.03	2.92
		EBP+DPM0	14.61	6.10	18.59	5.38	25.52	4.95	40.82	5.16
		EBP+DPM2	**14.63**	**5.25**	**18.61**	**4.31**	**25.54**	**3.48**	**40.84**	**2.80**
	2k 1-way I cache	DBP+DPM0	20.05	6.17	25.51	5.60	35.03	5.37	56.03	5.96
		DBP+DPM2	20.07	5.01	25.53	4.12	35.05	3.33	56.04	2.71
		EBP+DPM0	14.62	5.78	18.60	5.13	25.54	4.75	40.84	5.01
		EBP+DPM2	**14.64**	**4.93**	**18.62**	**4.05**	**25.55**	**3.27**	**40.85**	**2.64**
V42	8K 4-way I cache	DBP+DPM0	43.22	17.11	54.62	15.41	73.31	14.30	116.44	15.12
		DBP+DPM2	43.53	14.98	54.89	12.94	73.66	10.61	116.73	9.42
		EBP+DPM0	35.71	16.69	45.06	14.86	60.17	13.52	95.42	13.87
		EBP+DPM2	**35.99**	**14.95**	**45.31**	**12.89**	**60.49**	**10.53**	**95.68**	**9.29**
	4K 2-way I cache	DBP+DPM0	51.14	20.56	63.81	19.18	81.88	17.90	128.19	19.13
		DBP+DPM2	52.10	18.06	64.63	16.26	82.99	13.82	129.09	12.89
		EBP+DPM0	42.45	19.80	52.89	18.20	67.47	16.67	105.44	17.35
		EBP+DPM2	43.29	17.73	53.60	15.84	68.44	13.32	106.22	12.28

Table 7. Energy model of external memory.

Energy model	
Standby energy	exe. time × standby power
Access energy	access count × energy per access (excluding standby power)
DPM policy for standby power reduction	
DPM0	without using any DPM in standby state
DPM1	transition the memory into standby power down mode immediately after each access operation
DPM2	transition the memory into self refresh mode when no access during specified time window
Parameters [22, 23]	
Active read energy	burst read: 27.15 nJ /access
Active write energy	burst write: 20.17 nJ /access
Standby power	DPM0: 59.1mW; DPM1: 8.9mW; DPM2: 0.65mW
Time window size	1700 cycles

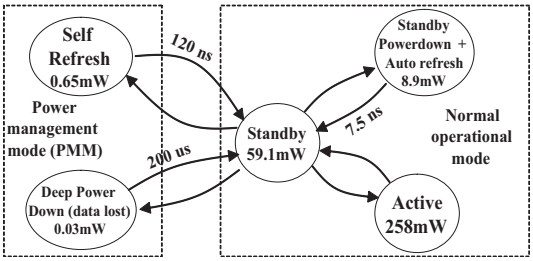

Figure 5. Mobile DDR SDRAM power state transition graph [22, 23].

ecution of tasks and the proposed dynamic scheme as described in Fig. 4. The simulation results are given in Fig. 7 where dynamic DVFS alone represents the same algorithm as the dynamic DEPS with disabled DHRC and DPM. As can be seen, the dynamic DEPS achieves the maximal 5.7% and 15.2% improvement over the static DEPS and the dynamic DVFS, respectively. The oracle that is assumed to know the precise execution time in advance shows the maximal 7% improvement over the dynamic DEPS. This method, however, is not realizable in practice. We also evaluate the dynamic DEPS without DPM, and results show that it can only achieve a slightly better result than dynamic DVFS alone. This result also suggests that the power reduction of the external memory in this framework is very important to exploit the potential of DEPS completely.

5 Conclusion

We proposed a generalized software framework DEPS: dynamic energy performance scaling, for system-wide energy reduction in hard real-time embedded systems. It integrates three existing energy performance tradeoff technologies, i.e., DHRC, DVFS, and DPM into the framework to achieve the maximal energy savings. Static and dynamic schemes of DEPS are proposed to cope with stable and fluctuant workload, respectively. In the case study, simulation results show that static DEPS achieves 13.6% and 13.7% improvement over DVFS alone and DHRC alone on average, and dynamic DEPS achieves the maximal 5.7% improvement over static DEPS. For future work, we plan to evaluate the DEPS framework on a multiple performance processor chip [28] that requires only about 1 us transition time for DVFS and SCW.

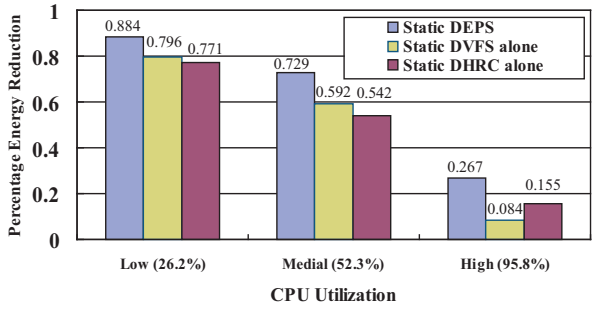

Figure 6. Comparison of static schemes at different CPU utilizations.

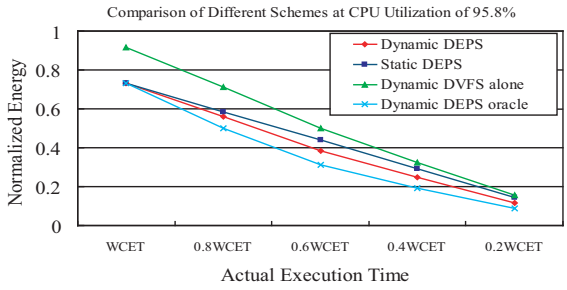

Figure 7. Results of dynamic scheme.

Acknowledgments

This work is supported in part by the CREST ULP project of JST.

References

[1] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," In *Proc. ACM Symposium Operating Systems Principles*, 2001, pp. 89–102.

[2] W. Kim, D. Shin, H. Yun, J. Kim, and S. L. Min, "Performance comparison of dynamic voltage scaling algorithms for hard real-time systems," In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2002, pp. 219–228.

[3] S. Saewong, and R. Rajkumar, "Practical voltage scaling for fixed-priority rt-systems," In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2003, pp. 106–114.

[4] D. H. Albonesi, R. Balasubramonian, S. G. Dropsbo, et al., "Dynamically tuning processor resources with adaptive processing," In *IEEE Computer*, 2003, pp. 49–58.

[5] M. Huang, J. Renau, and J. Torrellas, "Positional adaptation of processors: application to energy reduction," In *Proc. IEEE International Symposium Computer Architecture*, 2003, pp. 157–168.

[6] D. Chaver, L. Pinuel, M. Prieto, F. Tirado, and M. Huang, "Branch prediction on demand: an energy-efficient solution," In *Proc. International Symposium on Low-Power Electronics and Design (ISLPED)*, 2003, pp. 390–395.

[7] D. H. Albonesi, "Selective cache ways: on-demand cache resource allocation," In *Proc. International Symposium on Microarchitecture*, 1999, pp. 248–259.

[8] S. Banerjee, S. G. and S. K. Nandy, "Program phase directed dynamic cache way reconfiguration for power efficiency," In *Proc. Asia and South Pacific Design Automation Conference (ASPDAC)*, 2007, pp. 884–889.

[9] M. Huang, J. Renau, S. M. Yoo, and J. Torrellas, "A framework for dynamic energy efficiency and temperature management," In *Proc. International Symposium on Microarchitecture (MICRO)*, 2000, pp. 202–213.

[10] A. Nacul, T. Givargis, "Dynamic voltage and cache reconfiguration for low power," In *Proc. Design Automation and Test in Europe (DATE)*, 2004, pp. 1376–1377.

[11] G. Zeng, H. Tomiyama, and H. Takada, "A software framework for energy and performance tradeoff in fixed-priority hard real-time embedded systems," In *Proc. IFIP International Conference on Embedded and Ubiquitous Computing (EUC)*, 2007, pp. 13–24.

[12] S. Martello and P. Toth, Knapsack problems: algorithms and computer implementations, Wiley, 1990.

[13] C. L. Liu, and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM* 20(1), 1973, pp. 40–61.

[14] SimpleScalar Tools, http://www.simplescalar.com/

[15] Sim-Panalyzer Project, http://www.eecs.umich.edu/ panalyzer/

[16] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," In *IEEE Annual Workshop on Workload Characterization*, 2001.

[17] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: A tool for evaluating and synthesizing multimedia and communications systems," In *Proc. of the 30th Annual International Symposium on Microarchitecture (MICRO)*. 1997, pp. 330–335.

[18] J. Scott, L. Lee, J. Arends, and B. Moyer, "Designing the low-power M*CORE architecture," In *Proc. International. Symposium. on Computer Architecture Power Driven Microarchitecture Workshop*, 1998, pp. 145–150.

[19] B. C. Mochocki, X. S. Hu, and G. Quan, "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, Sep. 2004, pp. 1370–1377.

[20] LPSolve tool: http://sourceforge.net/projects/lpsolve/

[21] H. Aydin, R. Melhem, D. Mosse, P. M. Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Trans. on Computers*, Vol.53, No.5, May 2004, pp. 584-600.

[22] Micron Technology Inc., Data Sheet, "Mobile DDR SDRAM MT46H16M16LF".

[23] Micron Technology Inc., Technical Note, TN-46-03, "Calculating memory system power for DDR".

[24] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. on Very Large Scale Integration Systems (VLSI)*, Vol. 8, No. 3, June 2000, pp. 299–316.

[25] X. Fan, C. S. Ellis, and A. R. Lebeck, "The synergy between power-aware memory systems and processor voltage scaling," In *Proc. Power-Aware Computer Systems*, 2003, pp. 164–179.

[26] R. Jejurikar, and R. Gupta, "Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems," In *Proc. International Symposium on Low Power Electronics and Design (ISLPED)*, 2004, pp. 78–81.

[27] Y. Cho and N. Chang, "Memory-aware energy-optimal frequency assignment for dynamic supply voltage scaling," In *Proc. International Symposium on Low Power Electronics and Designs (ISLPED)*, 2004, pp. 387–392.

[28] T. Ishihara et al., "AMPLE: An adaptive multi-performance processor for low-energy embedded applications," In *Proc. IEEE Symposium on Applicatioin Specific Processors*, 2008, pp. 83–88.