A Software Framework for Energy and Performance Tradeoff in Fixed-Priority Hard Real-Time Embedded Systems

Gang Zeng, Hiroyuki Tomiyama, and Hiroaki Takada

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

Abstract. A dynamic energy performance scaling (DEPS) framework is proposed to save energy in fixed-priority 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) can be 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 a case study, the effectiveness of DEPS has been validated.

1 Introduction

Power consumption has become one of the major concerns in today's embedded system design. Reducing power consumption can extend battery lifetime of portable systems, decrease chip cooling costs, as well as increase system reliability. In contrast to the traditional hardware-based low power designs, software-based energy performance tradeoff approaches have attracted much attention recently due to its flexibility and easy implementation. This approach is based upon the following observations: (1) 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 and even within the different phases of a given application [8]. (2) In real-time systems the utilization of processor is less than 100% even if all tasks run at worse case execution time (WCET). Moreover, the actual workload even for the same task may vary from instance to instance, which depends on the specific input data and execution path. To take advantage of this application-dependent potential for energy and performance tradeoff, software-based approach tries to select the appropriate hardware resource for different applications or different program phases to save energy and meet the deadline constraint simultaneously.

There are two kinds of commonly used energy performance tradeoff technologies. 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]. However, its effectiveness on specific application is difficult to predict for two reasons. First, DHRC

T.-W. Kuo et al. (Eds.): EUC 2007, LNCS 4808, pp. 13–24, 2007.

is application-dependent, i.e., a specific DHRC technique may be effective for some applications, but may be ineffective for other ones [9]. Second, even for a DHRC-effective application, the specific energy and performance relation for different hardware configuration is also difficult to predict. Another technology for energy performance tradeoff is dynamic voltage frequency scaling (DVFS) [1-7]. Because the dynamic power consumption of CMOS circuits is proportional to its clock frequency and its voltage square, DVFS tries to save energy by lowering both frequency and voltage of processor subject to deadline constraint. In contrast to 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. Moreover, the variation of execution time and energy consumption can be estimated by simple calculations. For example, most DVFS algorithms assume the execution time is linear-inversely proportional to the processor frequency.

Based on different criteria, the software-based energy performance tradeoff approaches can be classified into different categories. First, 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 targets for different applications (tasks) or different jobs of the same task; the intra-task approach is applied on periodic intervals [24], program phases [11, 12] or subroutines [9] within one application. Second, they can be classified into static (off-line) and dynamic (on-line) approaches according to when the configuration decisions are made.

Although both DHRC and DVFS are very effective for energy and performance tradeoff, unfortunately, combining them to achieve more energy savings is not a trivial problem. The reasons are that (1) while the energy consumption and execution time can be predicted by calculation after voltage/frequency scaling; they cannot be done so after hardware configuration is changed. Thus to guarantee hard real time for DHRC application, the only way to predict execution time is measurement. (2) As a general energy performance tradeoff technology, DVFS can be effectively applied to various applications. On the contrary, 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.

In this work, we propose a generalized software framework, i.e., dynamic energy performance scaling (DEPS), to combine the two energy performance tradeoff technologies for more energy savings. This framework targets for hard real-time embedded systems with preemptive scheduling policy. As a first step, we discuss its static inter-task based application. In general, 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 stable workload. Through analysis of an actual DVFS application, it is suggested in [23] that while dynamic DVFS is of limited use in case of large DVFS overhead and without precise prediction of CPU load, static DVFS generally is sufficient. In addition, it is shown that static application of DHRC achieved better energy savings than dynamic one due to its global information of program behaviors [9]. Furthermore, though off-line approach cannot handle dynamic variations of workload, it can often be used as a complement to on-line approaches. 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) Proposes a static application scheme of DEPS. (3) Construct a simulation environment for evaluating the proposed framework, and demonstrate the effectiveness of DEPS by a case study.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 presents the proposed DEPS framework. Section 4 gives a case study. 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 little DVFS overhead [3]. Mochocki et al. introduced off-line DVFS algorithms for EDF scheduling policy considering time and energy overhead of DVFS [25]. Cho et al. first proposed DVFS algorithms considering measured energy performance relations for different applications [4]. In contrast to 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 [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 utilize temporality-based program phase information to switch the configurations. On the contrary, Huang et al. proposed position-based (subroutine) hardware configuration approach including off-line and on-line algorithms [9]. Note that all these DHRC approaches performed fine-granularity configuration and 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 [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 configuration 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 [26]. While this work targets for single-task application with given slowdown factor, our approach target for 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 [14].Their approach used an on-line algorithm for selecting the Pareto-optimal configuration that best fill the slack for the next task to be executed, which is different from our off-line optimal global exploration algorithm for all tasks. Moreover, our generalized framework can adopt various DHRC schemes, and not limited to cache reconfiguration.

3 Proposed 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 $\Gamma = \{\tau_1, \tau_2 ..., \tau_n\}$. 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 candidate DEPS configurations $\{C_{i1}, C_{i2}, \ldots, C_{imi}\}$ consisting of both DHRC configuration and DVFS parameters. Each DEPS configuration C_{ij} is associated with a specific energy time (performance) 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}.

Note that we employ measurement to obtain this application-dependent energy time relation for each DEPS configuration. There are two reasons for this. First, as described in Section 1, the only way for prediction of energy and time relation after DHRC configuration change is measurement. Second, although most DVFS papers use calculation to predict energy and time relation after voltage/frequency scaling, recent research reveals 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 memory or I/O access behaviors as well as leakage power consumption, etc., which is generally neglected by 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 $hyperperiod = LCM(P_1, P_2, \ldots, P_n)$, i.e., the least common multiple of all task periods. The problem is to determine the set of optimal DEPS configurations that minimize the energy consumption over a hyperperiod while meeting the deadline constraints. This problem can be formulated as follows:

Minimize energy:

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

subject to

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

and

$$\sum_{j=1}^{m_i}C_{ij}=1 \quad i=1,2,\cdots,n \tag{3}$$

where $C_{ij} \in \{0,1\}, \forall i, j$.

In the above formulation, W_{idle} denotes the idle power of processor. The constraint (2) represents utilization-based schedulability test for RM scheduling [16]. Note that more complex schedulability test such as response time analysis (RTA) [17] can also be used for fixed-priority based scheduling at the expense of higher computational complexity. Although we only give the schedulability test for fixed-priority based scheduling, it is straightforward to extend it to EDF based scheduling. Constraint (3) indicates that for one task, only one DEPS configuration can be selected where C_{ij} = 1 denotes that the configuration C_{ij} has been selected for task τ_i in DEPS framework, otherwise C_{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 known as a NP-hard problem [15]. Although there is no polynomial-time exact method for this problem, we can use common dynamic programming or mixed-integer linear programming method for solving any reasonable size by off-line computation.

Note that although we do not consider the configuration overhead in the above formulation for simplicity, they can be incorporated easily. This is because in one hyperperiod, the occurred number of hardware configuration and DVFS settings 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 formula one and two. A detailed discussion on the overhead of DHRC and DVFS configuration can be found in [9] and [25], respectively.

3.3 Decision Algorithm for Selecting Candidate DEPS Configurations

Actually, a processor may have many DEPS configurations consisting of different DHRC and DVFS parameters. To reduce the computational complexity we only select some of them as candidates in the above optimal computation. As discussed in Section 1, because DVFS is effective for any applications, we retain all DVFS parameters as candidates directly. And then, to select effective DHRC configuration under the same DVFS parameters, first, we conduct measurement to obtain energy time relation for all possible DEPS configurations. Second, the maximal energy consumption E_{max} and the minimal execution time T_{min} from the above results are selected as comparative objects. Third, different DHRC configurations C_{ij} (T_{ij}, E_{ij}) with the same DVFS parameters are compared with each other by calculating its energy improvement over performance

degradation, which is represented by $(E_{max}-E_{ij})/(T_{ij}-T_{min}+1)$. Finally, the DHRC configurations with higher energy improvement rate will be selected as candidates in the optimal computation.

3.4 Implementation of Static DEPS

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

1. Obtain application-dependent energy time relation under all possible DEPS configurations by simulation or actual measurement.
2. Select candidate DEPS configurations for the optimal computation as the proposed decision algorithm.
3. Solve the energy optimal problem using the above formulation and obtain the optimal DEPS configuration for each task.
4. Store the optimal DEPS configuration including corresponding hardware parameters into a static configuration table.
5. OS scheduler sets corresponding DEPS configuration based on the static configuration table for next task to run at every context switch.

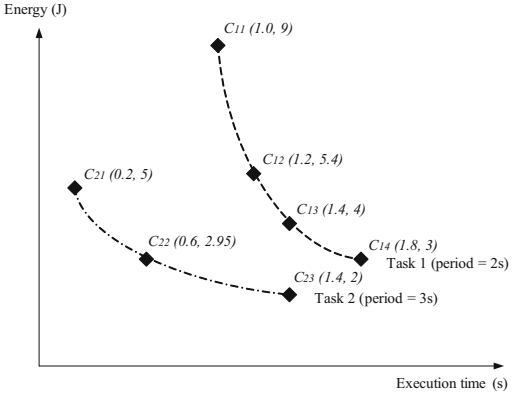

Fig. 1. An example for DEPS including two tasks and 7 selected DEPS configurations

We use the following example to illustrate the application of DEPS. This simple example includes two periodic tasks and 7 candidate DEPS configurations as shown in Fig.1, where $C_{11}(1.0, 9)$ indicates that for DEPS configuration C_{11} of task1, its corresponding worse case execution time and energy consumption are 1.0s and 9J, respectively. The idle power of processor is assumed to be 1 W. As the above formulation, the objective of DEPS is to find the optimal configuration combination for two tasks that can achieve the minimal energy and meet the deadline constraints simultaneously. The DEPS results for one hyperperiod scheduling are given in Fig.2. As can be seen, RTA-based method has more potential on energy savings, and considering idle power in the formulation can lead to more energy savings than without consideration of idle power.

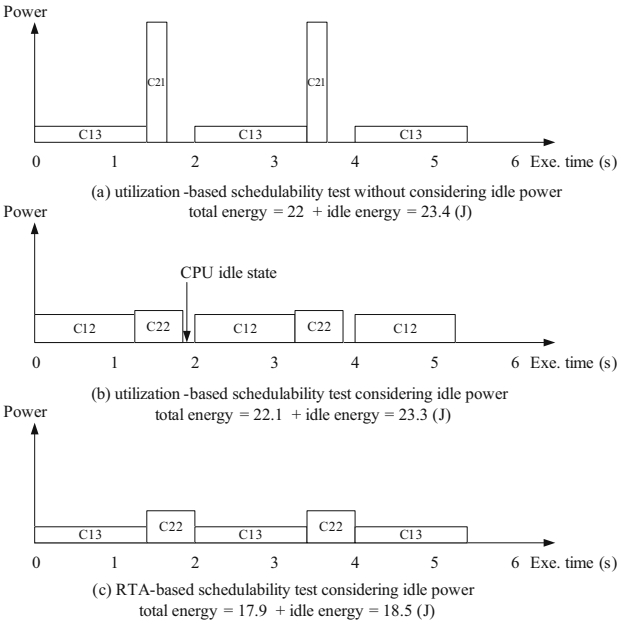

Fig. 2. DEPS results using different schedulability test methods with and without considering idle power

4 A Case Study

As mentioned earlier, because DEPS can adopt various DHRC and DVFS techniques, the achievable energy savings of DEPS are highly dependent on the employed DHRC and DVFS. Therefore, it is difficult to evaluate the absolute energy savings of general DEPS. For this reason, we demonstrate the effectiveness of DEPS through a case study.

We choose a 4-level voltage DVFS and the selective cache way (SCW) [11] as DHRC for our DEPS framework in this case study. In [3], it is shown that limited voltage/frequency level will result in more energy consumption 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 [22]. 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 detail implementations of SCW, configuration overhead, as well as method for keeping data coherency can be found in [11] and [12]. Note that our DEPS framework is general

and independent of the employed DHRC and DVFS technologies. We simple choose the above technologies as an example of DEPS.

4.1 Simulation Environment Setup

As we focus on embedded systems, a SimpleScalar/ARM [18] based Sim-Panalyzer [19] power simulator is employed to run the power simulation for our experiments. Sim-Panalyzer is an infrastructure for microarchitectural power simulation considering both dynamic and leakage power. The ARM configuration for SimpleScalar is listed in Table 1. Note that we only implement the SCW on instruction cache to further reduce the configuration overhead associated with writing cache operations. The possible configurations for SCW on L1 instruction cache are summarized in Table 2. In addition to the above configurations for SimpleScalar, Sim-Panalyzer uses its default configuration. Furthermore, we incorporate the DVFS capability into the Sim-Panalyzer as shown in Table 3. Some benchmark programs from Mibench [20] and Powerstone [21] that have distinct power characteristics are selected for this evaluation. A task set including these benchmark programs is assumed to run on this ARM simulator using fixed-priority scheduling with specified periods in Table 4.

Table 1. Configuration for SimpleScalar/ARM

Fetch queue	8
Branch Predictor	Not-taken
Fetch & Decode width	1
Issue width	1 (in-order)
Functional units	1 int ALU, 1 int Multiplier 1FP 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

Table 2. Instruction L1 cache SCW configurations

Parameters	Config. 1	Config. 2	Config. 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 3. Configuration for DVFS

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

Table 4. Task set for experiments

No.	Task name	Period (ms)	WCET(ms) : 280MHz; HRC config.1	Total CPU utilization
1	sha	400	64.9	59%
2	v42	200	36.7	
3	engine	100	8.7	
4	g3fax	100	15.6	

4.2 Experimental Results

According to the above Table 2 and 3, there are 3 configurations for DHRC and 4 configurations for DVFS. Therefore, this framework can provide total 12 possible DEPS configurations for each task. Each benchmark is simulated 12 times using Sim-Panalyzer, which corresponds to 12 DEPS configurations. The simulation results are summarized in Table 5, in which the HRC denotes the hardware resource configuration as shown in Table 2, and VF denotes the voltage frequency parameters as shown in Table.3. As these results show, DVFS can provide an identical energy performance tradeoff for all benchmarks. That is, lowering processor frequency and voltage leads to longer execution time and less energy consumption. However, for DHRC, the energy performance tradeoff is highly dependent on program behaviors. For example, while the large instruction cache (HRC config.1: 8KB) can achieve better energy performance results for v42 benchmark; small instruction cache (HRC config. 3: 2KB) is the better choice for g3fax benchmark because it leads to negligible variation of execution time but with less energy consumption.

Table 5. Simulation Results for Benchmarks

Name	VF / HRC	280 MHz		220MHz		160 MHz		100MHz	
		E(mJ)	T(ms)	E(mJ)	T(ms)	E(mJ)	T(ms)	E(mJ)	T(ms)
sha	config.1	24.34	64.88	19.93	82.60	15.94	113.16	12.62	180.91
	config.2	**20.35**	**64.90**	**16.69**	**82.63**	**13.37**	**113.19**	**10.64**	**180.95**
	config.3	**19.37**	**66.92**	**16.09**	**84.98**	**12.93**	**115.40**	**10.48**	**184.01**
v42	config.1	**13.40**	**36.72**	**11.10**	**46.35**	**8.93**	**61.94**	**7.23**	**98.24**
	config.2	14.66	44.48	12.60	55.37	10.28	70.43	8.79	109.95
	config.3	25.82	72.90	23.54	88.36	19.71	101.28	18.15	152.42
engine	config.1	3.22	8.69	2.63	11.05	2.11	15.17	1.67	24.25
	config.2	**2.72**	**8.69**	**2.22**	**11.05**	**1.78**	**15.17**	**1.42**	**24.26**
	config.3	4.70	14.10	4.17	17.33	3.36	21.03	2.99	32.30
g3fax	config.1	6.00	15.56	4.90	19.80	3.92	27.18	3.10	43.48
	config.2	5.13	15.56	4.20	19.80	3.36	27.19	2.66	43.48
	config.3	**4.71**	**15.58**	**3.85**	**19.82**	**3.09**	**27.20**	**2.46**	**43.50**

The selected candidate DEPS configurations for each benchmark as the proposed decision algorithm are denoted in boldface in the table. In this case study, we use LPSolve tool [27], a free mixed integer linear programming solver, to solve the energy optimal problem as described in Section 3.2. DEPS results corresponding to different schedulablity test methods are reported in Table 6 and 7. It is clear that DEPS can

Table 6. DEPS results for fixed-priority scheduling using utilization-based schedulability test

No.	Task name	DEPS results: total energy 62.57 mJ	
		HRC	VF
1	sha	config. 3	220 MHz
2	v42	config. 1	220 MHz
3	engine	config. 2	220 MHz
4	g3fax	config. 3	220 MHz

Table 7. DEPS results for fixed-priority scheduling using RTA-based schedulability test

No.	Task name	DEPS results: total energy 52.03 mJ	
		HRC	VF
1	sha	config. 3	160 MHz
2	v42	config. 1	160 MHz
3	engine	config. 2	220 MHz
4	g3fax	config. 3	160 MHz

Table 8. Comparison with other methods

Task name & Results	SVFS [1][3]		Opt-clock [3]		Static DHRC		DEPS	
	HRC	VF	HRC	VF	HRC	VF	HRC	VF
sha	config.1	220	config.1	160	config.3	280	config.3	160
v42	config.1	220	config.1	160	config.1	280	config.1	160
engine	config.1	220	config.1	220	config.2	280	config.2	220
g3fax	config.1	220	config.1	160	config.3	280	config.3	160
Energy of hyperperiod	72.3 mJ		60.0 mJ		75.9 mJ		**52.0 mJ**	
Ave. power	180.6 mW		150.0 mW		189.7 mW		**130.1 mW**	
Power red.	53.1 %		61.0 %		50.7 %		**66.2 %**	

achieve the minimal energy consumption and meet the deadline simultaneously by selecting the optimal DEPS configuration.

Table 8 compares the DEPS with other power saving methods. Note that 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 addition, we assume that static application of DVFS utilizes full hardware resource, and static application of DHRC utilizes the highest processor performance. In Table 8, the column denoted as SVFS represents the static voltage frequency scaling methods proposed in [1] and [3], in which identical speed is assigned to all tasks to reduce the energy loss caused by large DVFS overhead. The column denoted as Opt-clock represents the optimal speed assignment method proposed in [3]. This method statically assigns different speed for different tasks to achieve the maximal energy savings. 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 in this ARM-based simulator, i.e., 385 mW when running g3fax at 280 MHz on HRC config. 1. As can be seen from Table 8, the DEPS can achieve 66.2% power reduction and a 5%-15% improvement over previous methods when original task set has a total CPU utilization of 59%.

To verify the relation of the CPU utilization and power reduction rate, we extend the periods of sha and v42 in Table 4, to 600 and 300 ms, respectively, which means a lower CPU utilization, i.e., 47%. And then, the above experiments are conducted again, and results show a 75.7% reduction in power consumption, which is a significant improvement over the case of 59% CPU utilization.

5 Conclusion

We proposed a generalized software framework, i.e., DEPS: dynamic energy performance scaling for energy savings targeting for 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 a case study, DEPS shows 66% power reduction and a 5%-15% improvement over previous methods in the case of 59% CPU utilization.

Acknowledgments. The authors would like to thank Professor Tohru Ishihara at System LSI Research Center of Kyushu University for his valuable comments. This work is supported by the Core Research for Evolutional Science and Technology (CREST) from Japan Science and Technology Agency.

References

1. Pillai, P., Shin, K.G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In: Proc. ACM Symposium Operating Systems Principles, pp. 89–102 (2001)
2. Kim, W., Shin, D., Yun, H., Kim, J., Min, S.L.: Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems. In: Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 219–228 (2002)
3. Saewong, S., Rajkumar, R.: Practical Voltage Scaling for Fixed-Priority RT-Systems. In: Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 106–114 (2003)
4. Cho, Y., Chang, N., Chakrabarti, C., Vrudhula, S.: High-Level Power Management of Embedded Systems with Application-Specific Energy Cost Functions. In: Proc. Design Automation Conference (DAC), pp. 568–573 (2006)
5. Choi, K., Soma, R., 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 24(1), 18–28 (2005)
6. Shin, D., Kim, J.: Intra-Task Voltage Scheduling on DVS-Enabled Hard Real-Time Systems. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 24(10), 1530–1549 (2005)
7. Yuan, W., Nahrstedt, K., Adve, S.V., Jones, D.L., Kravets, R.H.: GRACE-1: Cross-Layer Adaptation for Multimedia Quality and Battery Energy. IEEE Trans. Mobile Computing 5(7), 799–815 (2006)

8. Albonesi, D.H., Balasubramonian, R., Dropsbo, S.G., et al.: Dynamically Tuning Processor Resources with Adaptive Processing. IEEE Computer, 49–58 (2003)
9. Huang, M., Renau, J., Torrellas, J.: Positional Adaptation of Processors: Application to Energy Reduction. In: Proc. IEEE International Symposium Computer Architecture, pp. 157–168 (2003)
10. Chaver, D., Pinuel, L., Prieto, M., Tirado, F., Huang, M.: Branch Prediction on Demand: An Energy-Efficient Solution. In: 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. In: Proc. International Symposium on Microarchitecture, pp. 248–259 (1999)
12. Banerjee, S., Nandy, G.S., Program, S.K.: Phase Directed Dynamic Cache Way Reconfiguration for Power Efficiency. In: 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. In: Proc. Great Lakes Symp. VLSI, pp. 73–78. ACM Press, New York (2001)
14. Nacul, A., Givargis, T.: Dynamic Voltage and Cache Reconfiguration for Low Power. In: Proc. Design Automation and Test in Europe (DATE), pp. 1376–1377 (2004)
15. Martello, S., Toth, P.: Knapsack problems: algorithms and computer implementations. Wiley, Chichester (1990)
16. Liu, C.L., Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM 20(1), 40–61 (1973)
17. Lehoczky, J.P., Sha, L., Ding, Y.: The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. In: Proc. IEEE Real Time Systems Symposium (RTSS), pp. 166–171 (1989)
18. SimpleScalar Tools, http://www.simplescalar.com/
19. Sim-Panalyzer Project, http://www.eecs.umich.edu/ panalyzer/
20. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., Brown, R.B.: MiBench: A Free, Commercially Representative Embedded Benchmark Suite. In: IEEE Annual Workshop on Workload Characterization (2001)
21. Scott, J., Lee, L., Arends, J., Moyer, B.: Designing the Low-Power M*CORE Architecture. In: Proc. International Symposium on Computer Architecture Power Driven Microarchitecture Workshop, pp. 145–150 (1998)
22. Texas Instruments, Application Report, SPRA848A: Using the Power Scaling Library (2004)
23. Texas Instruments, Application Report, SPRAA19A: Power Management in an RF5 Audio Streaming Application Using DSP/BIOS (August 2005)
24. Lee, S., Sakurai, T.: Run-Time Voltage Hopping for Low-Power Real-Time Systems. In: Proc. Design Automation Conference (DAC), pp. 806–809 (2000)
25. Mochocki, B.C., Hu, X.S., Quan, G.: A Unified Approach to Variable Voltage Scheduling for Nonideal DVS Processors. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 23(9), 1370–1377 (2004)
26. Huang, M., Renau, J., Yoo, S.M., Torrellas, J.: A Framework for Dynamic Energy Efficiency and Temperature Management. In: Proc. International Symposium on Microarchitecture (MICRO), pp. 202–213 (2000)
27. LPSolve tool: http://sourceforge.net/projects/lpsolve/