2007－EMB－6 (1)
2007／12／4

A Dynamic Algorithm for Energy Savings in DEPS Framework

Gang Zeng, Hiroyuki Tomiyama and Hiroaki Takada
Graduate 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 has been proposed in previous work to explore application-specific energy-saving potential in hard real-time embedded systems. The framework employed static algorithm to select the optimal DEPS configuration off line because it assumed stable workload with worse case execution time (WCET) requirement. However, workload usually exhibits fluctuant in practice due to data dependence, which results in early completion over WCET. To take advantage of the run-time slack for additional energy savings, a dynamic algorithm is proposed for selecting the DEPS configurations with less energy consumption on line. Through a case study, its efficiency 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-centric low power designs, software-centric energy performance tradeoff approach has 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]. Furthermore, program behaviors with respect to access of I/O devices (e.g. external memory) are also application-dependent. This fact manifests the energy saving potential during the execution of program by dynamically tuning hardware resource, which corresponds to different power consumption. (2) In real-time systems the utilization of processor is always 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. The fact of existing slack in real-time system reveals the chance to tradeoff performance for energy since the highest performance is not always required if the deadline can be met. To take advantage of this application-specific potential for energy savings, software-centric approach attempts to select the optimal DEPS configurations for different applications or different program phases to minimize total energy consumption while meeting the deadline constraints simultaneously.

There are two kinds of commonly used power control technologies for energy and performance tradeoff. 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. (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 relation for different hardware configuration is also difficult to predict. The second technology for energy performance tradeoff is dynamic voltage frequency scaling (DVFS) [1, 2, 3, 4, 5, 6, 7]. Because the dynamic power consumption of CMOS circuits is proportional to its clock frequency and its voltage square, DVFS saves energy by lowering both frequency and voltage of processor subject to deadline constraint. 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. Moreover, the variation of execution time and energy consumption can be estimated by simple calculations. For example, most DVFS algorithms assume that the energy is proportional to the square of the supply voltage and the execution time is inversely proportional to

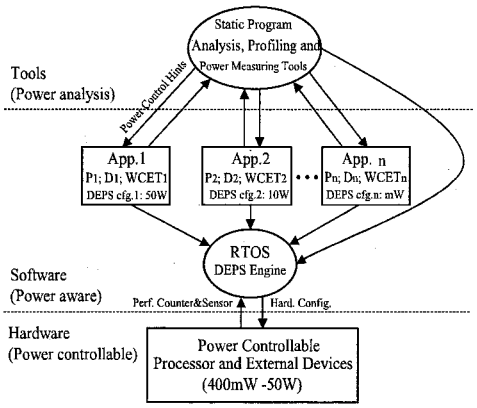

Figure 1. DEPS framework

clock frequency.

Although these existing technologies are very effective for energy and performance tradeoff alone, 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 deadline for DHRC application, the only way to obtain the energy time relation 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 [23], a dynamic energy performance scaling (DEPS) framework has been proposed for energy savings by combining the above two power control technologies. The framework employed static algorithm to select the optimal DEPS configuration off line. While the static scheme is effective for stable workload with WCET requirement, it cannot deal with variable workload to further save energy. In practice, embedded systems often exhibit fluctuant workload 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 dynamic scheme of DEPS for additional energy savings by reclaiming runtime slack.

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

2 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 to assist the selection of candidate DEPS configurations which have higher energy efficiency than other configurations. The power measurement tool then is employed to obtain the specific energy time relations under each selected configuration. By this way, the measurement time can be reduced to some content. 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 also crucial for the DEPS framework. Generally, the more power controllable mechanisms the hardware provides, the more energy savings potential there is. However, for specific applications the effective mechanism may be different, and the overhead for power control should also be considered. Note that the framework can support both inter-task and intra-task based applications with coarse or fine granularity, although we only consider the inter-task and task-level application in this work.

2.1 System Model

This work focuses on embedded system and assumes a DHRC and DVFS enabled embedded processor. The DVFS is assumed to operate at a finite set of supply voltage levels, each with an associated speed, as seen in most commercial processors.

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 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 utilize measurement to obtain this application-specific energy time relation under selected DEPS configuration. There are two reasons for this. First, as described in Section 1, the energy consumption and execution time of 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 comprise external memory or other I/O devices access behaviors as well as leakage power consumption, etc., which are generally neglected in simple calculation.

2.2 Static scheme of DEPS

Static scheme of DEPS has been proposed in [23] where the problem of selecting the optimal DEPS configuration for each task to achieve the maximal energy savings and meet the deadline constraints has been formulized as a typically multiple choice 0/1 knapsack problem. The static scheme solves this problem off line using common method. Note that the optimization can only be guaranteed when the workload is constant with WCET. More detailed descriptions about the implementation of static scheme can be found in [23].

3 Dynamic Scheme of DEPS

Some dynamic DVFS algorithms have been proposed in literatures to reclaim dynamic slack, 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 [1, 2, 3, 20], this assumption is not held for DEPS anymore. It is evident that when DEPS configuration is changed such as cache size, branch prediction etc., the number of cycles for task execution is changed also. As a result, the left execution time for task cannot be predicted exactly when its DEPS configuration is changed during execution, which may lead to miss of deadline. In fact, even if only changing the voltage and frequency, the number of cycles is also changed due to the variation of cycles required for external memory access. A recent paper solves this problem by distinguishing the total workload as on-chip, off-chip, frequency-dependent, and frequency-independent, respectively [21]. However, it is impracticable to assume knowing the left each portion of workload during the execution of task. For this reason, our algorithm maintains the same DEPS configuration for each task during its entire execution. That is, even if the preempted task gets the slack from higher priority task, it cannot reclaim the slack to change its DEPS configuration. To guarantee the deadline, we impose this strong constraint on the algorithm at the cost of lost chance for energy savings. Figure 2 summarizes the definitions and notations used in the algorithm. As an example of notations, Fig.3 shows their relative time relation.

- Assume $task_i$ has been completed before dispatching $task_j$
- NTA: earliest arrival time of next task from current dispatch time of $task_j$ (t)
- Actual execution time (AET) of task measured by OS
- Definition of dynamic slack
  - Total slack: early completion of task than WCET
    - Total Slack = $WCET_i - AET_i$;
  - Useable slack: slack time can be used by next task
    - Useable Slack = Total Slack – CPU idle time;
- Effective execution time for $task_j$ when useable slack exists
  - $EET_j = WCET_j + Slack$;

Figure 2. Definitions and notations.

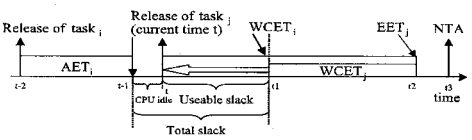

Figure 3. Relative time relations of notations.

In brief, the dynamic algorithm includes two steps, i.e., off-line step and on-line step as shown in Fig. 4. The detailed on-line slack reclaiming algorithm is also listed in this figure. The basic rules for this algorithm are as follows. (1) The default configuration is obtained using static scheme where the schedulability is guaranteed even for the WCET. (2) Only the slack generated by high priority task can be used by low priority task. (3) If only one task in the ready queue, the NTA time can be used for this task. As proved in [3, 20], these rules can guarantee the schedulability of algorithm.

As an example, Table 1 shows the DEPS configuration table of v42 benchmark. In practice, this table only contains two columns, i.e., hardware configuration parameters and associated WCET. Note that the first entry should be the optimal static DEPS configuration obtained as described in [23], the other entries are the effective configurations that are sorted as increasing WCET or decreasing energy. As can be seen, with increased execution time, the energy consumption is decreased. Dynamic scheme therefore attempts to find new DEPS configuration with less energy by absorbing dynamic slack. For example, when v42 task obtains a 20ms slack form higher priority task, then it can update its current DEPS configuration to 2 which consumes less energy than configuration one.

4 A Case Study

As mentioned earlier, since DEPS can adopt diverse DHRC and DVFS techniques, the achievable energy savings of DEPS are highly dependent on the em-

- Dynamic scheme of DEPS
  - STEP 1: static (off-line) optimal algorithm
    - Off-line implementation to obtain the static optimal DEPS configuration for each task
    - Construct the DEPS config. table for each task using the optimal and candidate DEPS configurations
  - STEP 2: dynamic (on-line) slack reclaiming algorithm
    - Implemented by OS scheduler when task is completed, or dispatched
- Dynamic (on-line) slack reclaiming algorithm
  - Completion of $task_i$
    - slack = $EET_i - AET_i$; // total slack
  - Dispatch $task_i$ at time t
    - $EET_i = WCET_i$; // WCET of $task_i$ under static optimal DEPS configuration
    - If only one task in the ready queue and NTA > t + $WCET_j$
      - EET_j = NTA – t;
    - If the priority of $task_i$ is higher than that of $task_j$
      - slack = slack – CPU idle time; // useable 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 in dynamic scheme.

Table 1. DEPS configuration table of v42.

No.	Hardware configuration parameters	WCET (ms)	Energy (mJ)
1	220MHz/1.8V; EBP; 8k I cache	45.06	11.46
2	160MHz/1.6V; EBP; 8k I cache	60.17	9.22
3	100MHz/1.4V; EBP; 8k I cache	95.42	7.49

ployed DHRC and DVFS. Therefore, it is difficult to evaluate the exact energy savings of general DEPS. 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) [11], configurable branch predictor (CBP) for DHRC in the DEPS framework. In [3], it is shown 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 [19]. The reason for selecting SCW and CBP is due to their easy implementation and low configuration overhead. The detailed implementations of SCW and configuration overhead can be found in [11, 12]. Note that our DEPS framework is general and independent on the employed DHRC and DVFS technologies. We simple choose the above technologies as an example of DEPS.

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

Table 3. Branch prediction configuration.

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

4.1 Simulation Environment Setup

As we focus on embedded systems, a SimpleScalar/ARM [14] based power simulator, Sim-Panalyzer [15], is employed to measure the energy and time in 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 2. The configurations of branch predictor are denoted in Table 3. Note that we only implement the SCW on instruction cache to avoid large configuration overhead for keeping data coherence. The possible configurations for L1 instruction cache are summarized in Table 4. In addition to the above configurations for SimpleScalar, Sim-Panalyzer retains its default configuration. Furthermore, we incorporate DVFS capability into the Sim-Panalyzer as shown in Table 5. For simplicity, we assume zero power consumption of processor during idle state of OS. Actually, the idle power can be considered as constant when using the proposed accurate DPM method in [22]. Some benchmark programs from Mibench [16], Mediabench [17] and Powerstone [18] are used for this evaluation. A synthetic task set consisting of these benchmark programs is assumed to run on this ARM simulator using fixed-priority scheduling with specified periods as given in Table 6. The DEPS results using static scheme are denoted in Table 7 which will be utilized as the default configurations in the dynamic scheme.

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

4.2 Simulation Results of Dynamic Scheme of DEPS

Instead of assuming stable WCET workload in static scheme, workload is assumed to be varied from 20 to 100 percent of the WCET in dynamic scheme. To calculate energy consumption, we assume that the average power maintains constant for each DEPS configuration even if the actual execution time is variable. The results of static scheme are utilized as the default configurations, and other effective configurations with less energy are employed to construct the DEPS configuration table which is similar to the Table 1. We build an evaluation environment to simulate the execution and fixed-priority scheduling of tasks as well as the proposed dynamic scheme as described in Fig.4. To evaluate the maximal potential for energy savings we also implement a oracle algorithm in the experiment. We assume that the oracle algorithm knows the precise execution time for each task instance in advance. Obviously, this assumption is unrealistic and it is only used for the evaluation. The experimental results are given in Fig.5. From these results, we can obtain the following observations.

- Dynamic algorithm always achieves less energy consumption than static algorithm if task completes early than WCET.
- Generally, the more dynamic slacks in the system, the more energy savings potential there is.
- The oracle algorithm indicate the maximal theoretical possibility for energy savings in the systems although it is impossible in practice.

Specifically, dynamic algorithm shows the maximal 4.7% improvement over static algorithm, and oracle algorithm shows the maximal 6% improvement than

Table 6. Synthetic task set.

Task name	Period (ms)	WCET(ms) at 280MHz HRC cfg.1	Total CPU uti.
sha	400	64.9	93.6 %
v42	200	36.7	
engine	100	8.7	
g3fax	100	15.6	
cjpeg	400	95.2	
tiff2rgba	3200	435.2	

Table 7. DEPS results of static scheme.

Task name	DEPS results: energy = 994.8 mJ	
	HRC	F/V
sha	2K I cache + EBP	280MHz/2.0V
v42	8K I cache + EBP	220MHz/1.8V
engine	4K I cache + EBP	280MHz/2.0V
g3fax	2K I cache + EBP	280MHz/2.0V
cjpeg	4K I cache + EBP	280MHz/2.0V
tiff2rgba	4K I cache + DBP	280MHz/2.0V

dynamic algorithm. Note that dynamic algorithm actually achieves decreased improvement than static algorithm when the actual execution time of task is less than 0.6 WCET. The reason is that the provided maximal energy scalability in DEPS framework determines the possible maximal energy savings. For example, as shown in Table 1, even if the v42 task obtains runtime slack larger than 51 ms, it can only select the configuration 3, which indicates that it cannot achieve less energy consumption than 7.49mJ even though there is enough dynamic slack. Therefore, the larger energy scalability the DEPS framework provides, the more energy saving potential can be obtained in the dynamic algorithm.

5 Conclusion

We proposed a dynamic scheme to improve the previous static scheme in the DEPS framework. The dynamic scheme targets at the fluctuant workload in embedded systems to achieve additional energy savings. To this end, it attempts to select the new DEPS configuration with less energy consumption on line by considering the runtime slack. Through a case study, it is validated that the dynamic scheme always achieves less energy consumption than static scheme. Moreover, experimental results also suggest that to guarantee the efficiency of dynamic scheme it is important to provide sufficient energy scalability in the DEPS framework.

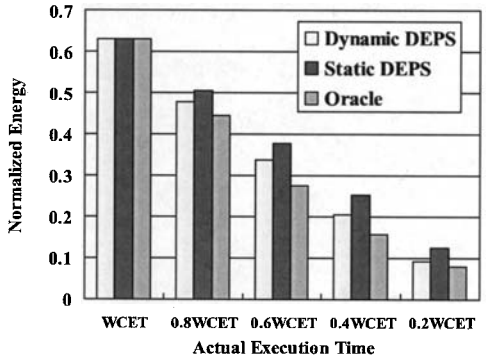

Figure 5. Results of dynamic scheme.

Acknowledgments

The authors thank Professor Tohru Ishihara for his valuable comments. This work is supported by the Core Research for Evolutional Science and Technology (CREST) project of Japan Science and Technology agency.

References

[1] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," Proc. ACM Symposium Operating Systems Principles, pp. 89–102, 2001.

[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," Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 219–228, 2002.

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

[4] Y. Cho, N. Chang, C. Chakrabarti and S. Vrudhula, "High-Level Power Management of Embedded Systems with Application-Specific Energy Cost Functions," Proc. Design Automation Conference (DAC), pp. 568–573, 2006.

[5] K. Choi, R. Soma, and M. Pedram, "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, Jan. 2005.

[6] D. Shin and J. Kim, "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, Oct. 2005.

[7] W. Yuan, K. Nahrstedt, S. V. Adve, D. L. Jones, and R. H. Kravets, "GRACE-1: Cross-layer adaptation for multimedia quality and battery energy," IEEE Tans. Mobile Computing, Vol. 5, No. 7, pp. 799–815, July 2006.

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

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

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

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

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

[13] A. Buyuktosunoglu et al., "A circuit-level implementation of an adaptive-issue queue for power-aware microprocessors," Proc. Great Lakes Symp. VLSI, ACM Press, pp. 73–78, 2001.

[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," IEEE Annual Workshop on Workload Characterization, Dec. 2001.

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

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

[19] Texas Instruments, Application Report, SPRA848A, "Using the power scaling library," 2004.

[20] 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, pp.584-600, May 2004.

[21] H. Aydin, V. Devadas, and D. Zhu, "system-level energy management for periodic real-time tasks," Proc. of RTSS pp.313-322, 2006.

[22] G. Zeng, H. Tomiyama and H. Takada, "Power optimization for embedded system idle time in the presence of periodic interrupt services," Proc. of International Embedded Systems Symposium, pp.241-254, 2007.

[23] G. Zeng, H. Tomiyama and H. Takada, "A generalized framework for energy and performance tradeoff," Proc. of Embedded Systems Symposium, pp.74-81, 2007.