A Generalized Framework for Energy Savings in Real-Time Multiprocessor Systems

Gang Zeng, Tetsuo Yokoyama, Hiroyuki Tomiyama, Hiroaki Takada,
Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
{sogo, yokoyama, 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 multiprocessor systems. This software-centric framework 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. The problem of determining the optimal task allocation and DEPS configurations is formulated as an integer linear programming (ILP) problem. Several practical issues such as how to reduce measurement and computation time and how to reduce the configuration overhead are also addressed. The effectiveness of DEPS is validated through a case study.

Keywords: embedded real-time systems, energy-aware multiprocessor scheduling, dynamic hardware resource configuration, dynamic voltage frequency scaling, dynamic power management.

I. 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 [1]. 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 generally less than 100% even if all tasks run at the worst case execution time (WCET). The fact of existing slack in real-time system reveals the chance for trading off performance for energy savings since the highest performance is not always required if the deadline can be met.

There are three kinds of commonly used power control technologies for energy performance tradeoff. One is dynamic hardware resource configuration (DHRC), such as adaptive branch predictor, selective cache way etc.. This technology tries to improve processor energy efficiency by dynamically tuning major processor resources in accordance with varied needs of applications [1]. 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 [2]. The second technology is dynamic voltage frequency scaling (DVFS). Because the dynamic power consumption of CMOS circuits is proportional to its clock frequency and its voltage square, DVFS can save energy effectively through lowering both frequency and voltage of processor. Unlike DHRC, DVFS generally has similar effectiveness on different applications. That is, lowering frequency and voltage in a range always leads to longer execution time and less energy consumption. Moreover, 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.

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 DHRC application, the only way to obtain the energy time relation under a hardware configuration is actual or simulation measurement (measurement for short, hereinafter).

978−1−4244−2599−0/08/$25.00 ©2008 IEEE

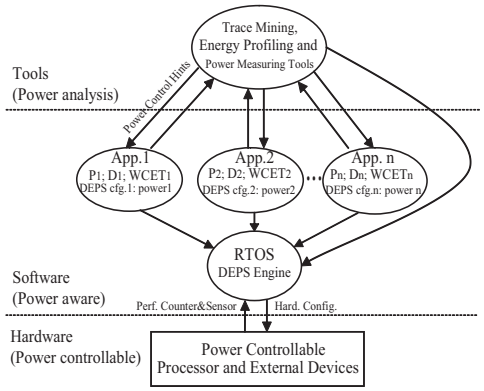

Fig. 1. DEPS framework.

(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.

In this work, we propose a generalized framework, called dynamic energy performance scaling (DEPS), to achieve the maximal energy savings in real-time multiprocessor systems by combining three existing power control technologies. The rest of the paper is organized as follows. Sec. 2 presents the proposed DEPS framework. Sec. 3 gives a case study and simulation results. Sec. 4 discuses the reduction of configuration overhead. Finally, Sec. 5 summarizes the paper.

II. DEPS Framework

Our 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 a software-centric approach, the DEPS engine is implemented in the scheduler of OS. The power analysis tools are employed for analysis and extraction of power relative information. The power measurement tool is utilized to obtain the energy time relations under selected configuration.

A. System Model

Consider a homogeneous multiprocessor composed of s identical cores denoted by the set $\Phi = \{core_1, \ldots, core_k, \ldots, core_s\}$. Each core is equipped with the DVFS and DHRC capabilities, and the multiprocessor is equipped with DPM-enabled external device.

We consider hard real-time applications including a set of independent n periodic real-time tasks, represented as $\Gamma = \{\tau_1, \ldots, \tau_i, \ldots, \tau_n\}$. 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 $\Psi_i = \{C_{i1}, \ldots, C_{ij}, \ldots, C_{im_i}\}$ 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 processors and external devices during T_{ij}.

There are two main approaches to schedule real-time tasks on the multiprocessor: global scheduling and partitioned scheduling. The global scheme uses a global scheduler to assign tasks to the processors online, whereas the partitioned scheme uses a dedicated scheduler for each processor, and tasks are assigned to particular processors offline without runtime migration. In this work, we focus on the partitioned scheme due to its simplicity and ease of implementation.

B. Problem Formulation

We assume that the overhead for task switching and DEPS reconfiguration is negligible for simplicity, and the energy time relations of each effective DEPS configuration have been obtained in advance. *hyperperiod* denotes the least common multiple of all task periods. Then, the energy optimization problem is to determine the optimal allocation and DEPS configuration for each task such that the total energy consumption over the hyperperiod is minimized and all deadline constraints are met.

Minimize energy:

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

subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{T_{ij}}{P_i}x_{ijk} \le n(2^{\frac{1}{n}} - 1), \qquad \forall core_k \in \Phi; \quad (2)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{T_{ij}}{P_i}x_{ijk} \le 1, \qquad \forall core_k \in \Phi; \quad (3)$$

$$\sum_{j=1}^{m_i}\sum_{k=1}^{s}x_{ijk} = 1, \qquad \forall \tau_i \in \Gamma; \quad (4)$$

$$x_{ijk} = \{0,1\}, \forall \tau_i \in \Gamma, \forall C_{ij} \in \Psi_i, \forall core_k \in \Phi. \qquad (5)$$

In the above equation (1), the W_{idle} denotes the idle power of processors and devices. The equations (2) and (3) represent utilization-based schedulability test for rate monotonic (RM) and EDF scheduling, respectively. Equation (4) indicates that for each task, only one DEPS configuration can be selected on all cores where $x_{ijk} = 1$ denotes that task τ_i is allocated to $core_k$ with configuration C_{ij}, otherwise $x_{ijk} = 0$. Note that the above formulation can be extended to heterogeneous multiprocessor if replacing the above T_{ij}, E_{ij} with T_{ijk}, E_{ijk} respectively, which means that the energy time relations of a configuration are different from core types in heterogeneous case.

It is known that the above ILP problem is an NP-hard problem. While there is no polynomial-time exact method for this

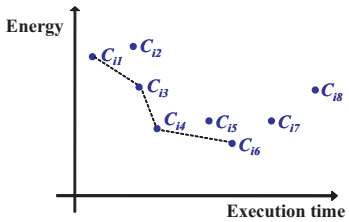

Fig. 2. An example to illustrate the effective configurations.

problem, we can use common methods for solving any reasonable size by off-line computation. In the following case study, we use LPSolve tool [7], a free mixed integer linear programming solver, to solve this energy optimization problem.

C. Selection of 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 $(1 \leq j \leq K)$ configurations), it thus needs to perform $L \times Q \times F_1 \times F_2 \times \cdots \times F_K$ times measurements to obtain all possible energy time relations for one task. Furthermore, the same number of variables is required for one task in the optimization computation. Obviously, this problem makes the framework unsuitable for practice. Fortunately, we can reduce both measurement and computation time without waste of energy by only selecting some energy efficient configurations. Figure 2 gives an example to show the different energy efficiency of each DEPS configuration where measured execution time and energy consumptions associated with 8 possible DEPS configurations for task τ_i are presented. It is clear that only 4 out of 8 configurations are effective for energy and performance tradeoff which are connected by the dotted line in the figure. If all DEPS configurations are considered as a partially ordered set, then the effective configurations are a set of minimal elements.

Instead of exhaustively exploring all possible configurations, we can predict the effective configurations by searching the effective parameters in each power control method separately. First, as discussed in Sec. 1, since DVFS is effective for any applications, we consider all DVFS parameters effective. Then, to find the effective parameters 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 [2]. For example, the cache miss rate will not change even the processor is set with different voltage and frequency. Therefore, the search can be performed separately, each time for one kind of DHRC or DPM policy and with specified voltage/frequency. As a result, only $Q + F_1 + F_2 + \cdots + F_K$ times measurements are needed to find all effective configurations and policy in DHRC and DPM. After that, if each DHRC has H_j $(1 \leq j \leq K)$ effective configurations where $H_j \leq F_j$, and only one DPM policy is effective, then total $L \times (H_1 + H_2 + \cdots + H_K)$ 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, configurations with increased execution time and decreased energy consumption 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. For example, the C_{i3}, and C_{i4} are more effective than C_{i6} in Fig. 2 through the evaluation of energy efficiency which is defined as reduced energy / increased execution time when comparing with the configuration with the shortest execution time, i.e., the C_{i1} in this example.

D. Implementation of DEPS

The implementation of DEPS may be static or dynamic which depends on actual workload characteristics. In general, while the static one is for stable workload and maintains the optimal DEPS configuration for entire execution, the dynamic one is for unstable workload and may change the DEPS configuration online based on actually measured slack time. A key difference between the dynamic DEPS and dynamic DVFS is that the assumption for most existing dynamic DVFS algorithms that the total number of cycles of task is constant even if voltage and frequency are changed during the execution of task is no longer held for dynamic 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 beginning of execution in the dynamic scheme. More detailed information about the dynamic DEPS schemes can be found in [4].

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. 2.3.

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. 2.2 to obtain the optimal allocation and DEPS configuration for each task.

4. Store the optimal DEPS configurations and the associated configuration parameters into a static configuration table for each processor core.

TABLE I
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 32bytes; 4ways
L2 cache	None
Memory bus width	4bytes

TABLE II
BRANCH PREDICTION CONFIGURATION.

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

5. For each context switch or dispatch of task, the OS scheduler of each core 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.

III. A CASE STUDY

We use a case study to demonstrate the effectiveness of DEPS because the achievable energy savings of DEPS are highly dependent on the employed technologies. In this case study, a four-core ARM-based homogeneous multiprocessor is assumed. Each core is equipped with independent DVFS and DHRC capabilities. The DVFS has 4-level selectable voltage/frequency, and the DHRC consists of the selective cache way (SCW) [3] and configurable branch predictor (CBP). Additionally, the multiprocessor is supposed to equip with two 256MB mobile DDR SDRAM chips with 32-bit width. The reason for selecting SCW and CBP is their easy implementation and low configuration overhead. 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.

A. Simulation Environment Setup

A SimpleScalar/ARM [5] based power simulator, Sim-Panalyzer [6], is employed to measure energy and time in our experiments. Default configuration is used for Sim-Panalyzer. The configuration of SimpleScalar/ARM is listed in Table I. The configurations of CBP, SCW, and DVFS are given in Table II, Table III, and Table IV, respectively. The SCW is implemented only on instruction cache to avoid large configuration overhead for keeping data coherence.

TABLE III
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 IV
DVFS PARAMETERS.

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

The employed SDRAM chip is supposed to be able to 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 be implemented by the SDRAM controller. Table VI summarizes the energy model, DPM policies, and 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, MediaBench and Powerstone are selected for the evaluation. A synthetic task set consisting of these benchmark programs is assumed to run on the ARM-based multiprocessor with different average utilizations as shown in Table V.

B. Simulation Results

According to the above Table II, III, IV and VI, 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. The number of selected effective configurations for each benchmark is given in the last column of Table V. In summary, total 178 instead of 792 measurements are required for 11 tasks in this case study. Execution time of LPSolve [7] in all experiments is between 1 second and 27 minutes on a computer equipped with a 3.6GHz Pentium processor and 2GB RAM.

In general, DHRC is effective even for systems with high CPU utilization, while DVFS is effective for systems with low CPU utilization. Through the combination of these technologies, DEPS is more effective than either DHRC or DVFS in isolation since it can provide more chances for energy and

TABLE V
SYNTHETIC TASK SET.

Task name	WCET (ms)	Period1 (ms)	Period2 (ms)	Period3 (ms)	Eff. conf.
sha	63.0	200	300	600	6
v42	35.7	100	200	400	8
engine	8.8	40	50	100	8
g3fax	14.6	40	50	200	8
cjpeg	92.2	200	400	800	8
susan	12.1	40	50	100	8
ispell	93.1	200	400	800	4
dgsm	122.4	300	500	1000	12
mad	164.1	600	700	1200	8
djpeg	40.6	200	300	400	12
adpcm	130.0	300	600	1000	8

TABLE VI
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 [8, 9]	
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

performance tradeoff, and overcome the inherent limitation of each technology. Detailed results for comparison of DEPS with existing technologies can be found in [4].

Figure 3 shows the normalized energy consumption with respect to the peak power of the multiprocessor at different average CPU utilizations, in which the scheme of power-off represents the full speed running for task execution and power off for idle state. As can be seen, with the reduced CPU utilization, more energy can be saved. The energy difference between the two schemes indicates the capability of DEPS to trade off performance for energy savings. However, when CPU utilization is 30%, this difference becomes small. Actually, we observe that almost all tasks have selected the configurations with the minimal energy consumption in this case. In other words, even if the CPU utilization is further reduced, no more energy can be saved. We discuss this problem in the following subsection.

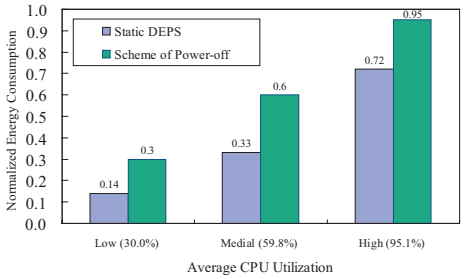

Fig. 3. Normalized energy consumption at different CPU utilizations.

C. Energy Savings Potential of DEPS

To predict the maximal energy savings potential of DEPS, it is necessary to evaluate the maximal energy scalability of DEPS which is defined as the maximal energy / the minimal energy in all effective DEPS configurations. As an example, Table VII shows all effective DEPS configurations and associated information of adpcm benchmark. These configurations are sorted as increased execution time order. It is clear from the table that with delayed execution time the energy consumption can be reduced. Specifically, prolonging the execution time to three times, the energy consumption will become one third of original one. However, it is impossible to achieve more energy savings than this one even the execution time is prolonged to more than 3 times. This is because the maximal energy scalability is limited to 3 times in this case study. To give an insight into the 3 times energy scalability, we evaluate each power control mechanism alone. The results reveal that DVFS, SCW, and CBP achieve 1.86, 1.29, and 1.09 times energy scalability alone, respectively. These results suggest that higher energy scalability is required to obtain more energy savings potential, which can be achieved either by adding new power control mechanisms to the DEPS or improving the energy scalability of DVFS. For example, an XScale processor may have 5.7 times energy scalability because it has wider voltage and frequency range (0.75V to 1.8V) than the employed one (1.4V to 2.0V). Meanwhile, it is also important to note that the specific achieved energy savings are dependent on the CPU utilization and program characteristics besides the given maximal energy scalability.

IV. MULTI-PERFORMANCE PROCESSOR FOR CONFIGURATION OVERHEAD REDUCTION

As described in Sec. 2.4, the DEPS requires one configuration for one task. In worst case, each task context switch may require configuration change. For conventional DVFS processors which require hundreds of microseconds for voltage and frequency transition, this configuration overhead is too large to be ignored. To solve this problem, we proposed a multi-performance processor (MPP) which can be used as a design alternative for the conventional DVFS processors in embedded

TABLE VII
MAXIMAL SCALABILITY OF ADPCM.

Effective DEPS config.	Energy (mJ)	Exe. Time (ms)	Ave. Power (mW)
8k I cache 280MHz EBP	57.8	130.0	445.1
2k I cache 280MHz EBP	38.9	130.2	298.5
2k I cache 280MHz DBP	35.7	142.4	251.0
2k I cache 220MHz EBP	31.9	165.6	192.5
2k I cache 220MHz DBP	29.3	181.2	161.8
2k I cache 160MHz EBP	25.8	227.6	113.3
2k I cache 160MHz DBP	23.7	249.0	95.2
2k I cache 100MHz EBP	20.9	364.0	57.3
2k I cache 100MHz DBP	19.2	398.4	48.1
Maximal scalability	3	3	9

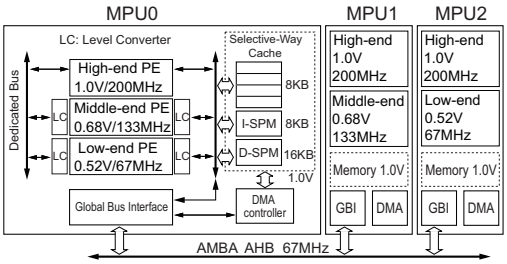

Fig. 4. Prototype of multi-performance processor [10].

system design. The processor consists of multiple PE (processing element) cores, and each PE-core has the same instruction set architecture but differ in their clock speeds and energy consumptions. Only a single PE-core is activated at a time and the other PE-cores are deactivated using clock gating and signal gating techniques. To evaluate the effectiveness of DHRC, we also implemented the SCW instruction cache in this MPP. Figure 4 shows the prototype of MPP.

The most significant advantage of MPP is a small overhead for changing its configuration. The gate-level simulation demonstrates that the MPP can change its configuration including both voltage/frequency transition and cache way selection within 1.5 microsecond and dissipates about 10 nanojoule while conventional DVFS processors need hundreds of microseconds and dissipate a few microjoule for the performance transition. For more detailed information of the MPP, refer to [10].

V. CONCLUSION

We proposed a generalized dynamic energy performance scaling (DEPS) framework for energy savings in real-time multiprocessor systems. This framework integrates three existing power control technologies, i.e., DHRC, DVFS, and DPM to effectively trade off performance for energy savings. Its effectiveness has been validated through a case study. Several practical issues such as how to select the effective configurations, as well as how to reduce the configuration overhead have also been addressed. For future work, we plan to evaluate the DEPS framework on an in-house developed multiple performance processor chip.

ACKNOWLEDGMENTS

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

REFERENCES

[1] D. H. Albonesi, R. Balasubramonian, S. Dropsho, *et al.*, "Dynamically tuning processor resources with adaptive processing," *IEEE Computer*, vol. 36, no. 12, pp. 49–58, 2003.

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

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

[4] G. Zeng, H. Tomiyama, H. Takada, and T. Ishihara, "A generalized framework for system-wide energy savings in hard real-time embedded systems," in *Proc. IFIP/IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 2008. (to appear)

[5] "Simplescalar tools." [Online]. Available: http://www.simplescalar.com/

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

[7] "lp_solve." [Online]. Available: http://sourceforge.net/projects/lpsolve/

[8] "Mobile DDR SDRAM MT46H16M16LF," Micron Technology Inc.

[9] "Calculating memory system power for DDR," Micron Technology Inc., Technical Note, TN-46-03, 2001.

[10] T. Ishihara, S. Yamaguchi, Y. Ishitobi, T. Matsumura, Y. Kunitake, Y. Oyama, Y. Kaneda, M. Muroyama and T. Sato, "AMPLE: An adaptive multi-performance processor for low-energy embedded applications," in *Proc. IEEE Symposium on Applicatioin Specific Processors*, 2008, pp. 83–88.