Energy Efficiency of Scratch-Pad Memory at 65 nm and Below: An Empirical Study

Hideki Takase Hiroyuki Tomiyama Gang Zeng Hiroaki Takada
Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

ABSTRACT

A number of approaches have been proposed so far for reducing the energy consumption of embedded systems by using scratch-pad memory. However, most of previous work focused on dynamic energy reduction, and did not take enough consideration of the leakage energy in their evaluations. As the technology scales down to the deep submicron domain, the leakage energy in memory devices could contribute to a significant portion of the total energy consumption. Therefore, evaluation of energy consumption including the leakage energy is necessary. In this paper, we investigate the effectiveness of scratch-pad memory on energy reduction considering both the dynamic and leakage energy. The experiments are performed for 65 nm, 45 nm, and 32 nm technologies. The results demonstrate the effectiveness of scratch-pad memory in deep submicron technology. It is also observed that the leakage energy becomes less significant along with the technology scaling.

Keywords

Energy consumption, scratch-pad memory, embedded systems, deep submicron

1. INTRODUCTION

Energy minimization has become one of the primary goals in the design of embedded systems. These days, cache memory is used not only in general-purpose processors but also in embedded processors in order to improve their performance. Cache also contributes to energy reduction because of decreased accesses to off-chip memory. Still cache is one of the most energy-hungry components in processors [1], so a large amount of studies have addressed cache energy minimization for more than a decade. More recently, Scratch-Pad Memory (SPM) has attracted attention as an alternative to, or to be used in combination with, cache memory due to its energy efficiency. SPM consumes less dynamic energy than cache mainly because no tag comparison is necessary. A number of techniques have been proposed for efficient usage of SPM in terms of energy consumption as well as performance, for example in [2]-[7].

It is well known that leakage energy becomes more dominant as the feature size shrinks. In terms of leakage energy, the effectiveness of SPM is not obvious. On one side, SPM consumes less leakage power than cache since no tag memory is necessary. On the other side, the use of SPM instead of cache often results in longer memory access latency, especially in case the size of working set is considerably larger than that of SPM. Thus, SPM has both positive and negative effects on leakage energy. However, most of past studies on SPM focus on dynamic energy with little consideration of leakage energy.

This paper presents an experimental study on the effectiveness of SPM with considering not only dynamic energy but also leakage energy. This study assumes 65, 45 and 32 nanometer technologies where leakage energy is thought to be significant. To our knowledge, this is the first paper which explicitly studies the energy effectiveness of SPM for the deep submicron era. As a first step, this paper focuses on the energy consumed for instruction accesses.

This paper is organized as follows. Section 2 provides a brief survey on related work. Section 3 describes how our experiments are conducted, and then Section 4 shows the results. Finally, Section 5 concludes this paper.

2. RELATED WORK

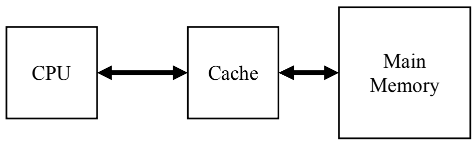
A considerable amount of research on SPM has been conducted so far for energy or performance optimization. The authors of [2] proposed a compiler optimization technique to allocate variables to SPM for performance improvement. They formulated the optimization problem as a 0/1 programming problem based on the number of memory accesses and memory access latency. In [3], allocation of program code to SPM is studied and a 0/1 programming formulation is presented. They demonstrated the effectiveness of SPM against cache memory in terms of dynamic energy. Since 0/1 programming is in general an NP problem, several heuristic algorithms were also proposed. A dynamic programming algorithm with a polynomial-time complexity is proposed in [4]. In [5], the authors proposed a data allocation method for SPM based on Total Conflict Factor (TCF). TCF indicates the possibility of data cache conflicts, and variables with a high TCF value are placed in SPM in order to minimize cache conflicts. The authors of [6] proposed a hardware mechanism for efficient overlay of SPM at runtime. In [7], the use of SPM in multiprocessor systems is studied. They assumed that each processor has a local SPM. Access to remote SPM is possible but its latency is longer. In addition to the papers mentioned above, there exist a number of papers which address SPM-based energy optimization. However, most of them take account of dynamic energy only and neglect the leakage energy in their evaluations. This might not be feasible in deep submicron era where leakage energy could be dominant.

This paper, on the contrary, assumes deep submicron technologies at 65 nm and below, and evaluates the effectiveness of SPM with considering not only dynamic energy but also leakage energy.

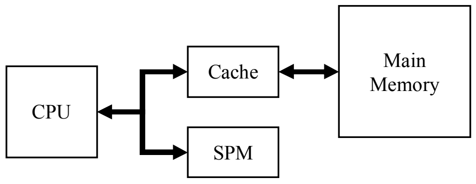
3. EXPERIMENTAL SETUP

This section describes how our experiments are conducted.

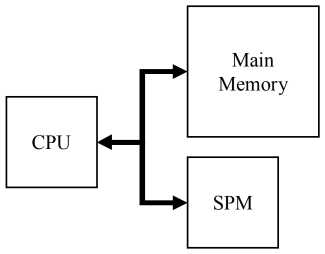
978-0-7695-3287-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICESS.2008.60



(a) Cache and main memory



(b) Cache, SPM and main memory



(c) SPM and main memory

Figure 1. Instruction memory system organization

3.1 Experimental Procedure and Tools

This paper focuses on energy consumption for instruction accesses. Several instruction memory organizations are tested as shown in Figure 1. The memory system in Figure 1 (a) does not have SPM, that in Figure 1 (b) has both cache and SPM, and that in Figure 1 (c) does not have cache. Total memory capacity is kept same among these systems. In Figure (a), all the program code is placed on the main memory. On a cache hit, the CPU fetches an instruction from the cache in a single cycle. On a cache miss, however, it takes multiple cycles in order to transfer the memory block from the main memory to the cache. In Figure (b), a part of the program code is placed in the SPM, and the other part is in the main memory. Instructions in the SPM can be fetched by the CPU in a single cycle. In Figure 1 (c), a part of the program code is placed in the SPM and the other part is in the main memory, which is similar to Figure 1 (b). The difference is that Figure 1 (c) does not have cache, so the instructions in the main memory cannot be accessed in a single cycle.

Our experimental procedure is depicted in Figure 2. On the left side of the figure, dynamic and leakage power for cache, SPM and main memory are calculated based on CACTI 5.0 [8]. Benchmark programs are cross-compiled into binary code, which is fed by an instruction-set simulator to generate instruction access traces. We assume a single-issue ARM processor as our target CPU, and the SimpleScalar/ARM simulator [9] is used for the instruction-level simulation. Based on the memory access traces, code allocation is decided. In this work, code allocation is done at the function-level granularity based on the method in [2]. The code allocation method is summarized in Section 3.3. Then, cache simulation is performed to calculate the dynamic and leakage energy as well as the execution cycles. An in-house cache simulator is used for the purpose. In our experiments, we use six benchmark programs which are selected from the MiBench suite [10].

3.2 Energy Calculation

In this work, energy consumed in the memory system is calculated as follows.

The total energy consumption E_{Total} is define as

$$E_{Total} = E_C + E_S + E_M$$

where E_C, E_S and E_M are the energy of cache, SPM and main memory, respectively. The energy of each memory component consists of dynamic energy and static one as follows.

$$E_C = E_{C\_dyn} + E_{C\_lkg}$$

$$E_S = E_{S\_dyn} + E_{S\_lkg}$$

$$E_M = E_{M\_dyn} + E_{M\_stby}$$

Here, E_{C_dyn}, E_{S_dyn} and E_{M_dyn} denote the dynamic energy of cache, SPM and main memory, respectively. E_{C_lkg} and E_{S_lkg} denote the leakage energies of cache and SPM, respectively, and E_{M_stby} is the stand-by energy of main memory. Let E_{C_hit}, E_{C_miss}, N_{C_hit} and N_{C_miss} be the cache energy consumed at cache hits, that at cache misses, the number of cache hits, and that of cache misses, respectively. Also, let E_{C_read} and E_{C_write} be the cache energy on a read access and that on a write access, respectively. Then, E_{C_dyn} is defined as follows.

$$E_{C\_dyn} = E_{C\_hit} + E_{C\_miss}$$

$$E_{C\_hit} = E_{C\_read} \times N_{C\_hit}$$

$$E_{C\_miss} = (E_{C\_read} + E_{C\_write}) \times N_{C\_miss}$$

The dynamic energy of SPM is defined as

$$E_{S\_dyn} = E_{S\_read} + N_{S\_hit}$$

where E_{S_read} and N_{S_hit} denote the energy on a read access to SPM and the number of accesses to SPM, respectively.

The dynamic energy of main memory is calculated differently depending on whether cache memory exists or not. In case cache exists as shown in Figures 1 (a) and (b), E_{M_dyn} is defined as

$$E_{M\_dyn} = E_{M\_burst\_read} \times N_{C\_miss}$$

where $E_{M\_burst\_read}$ denotes the energy on a burst read access to main memory. On the other hand, in case cache does not exist as shown in Figure 1 (c), E_{M_dyn} is defined as

$$E_{M\_dyn} = E_{M\_random\_read} \times N_{S\_miss}$$

where $E_{S\_random\_read}$ denotes the energy on a random read access to main memory and N_{S_miss} denotes the number of SPM misses. In other words, N_{S_miss} is the number of random accesses to main memory.

The leakage energy depends not only on the leakage power but also on the execution time. The leakage energies of cache and SPM are given by

$$E_{C\_lkg} = P_{C\_lkg} \times EC/f$$

$$E_{S\_lkg} = P_{S\_lkg} \times EC/f$$

while the stand-by energy of main memory is given by

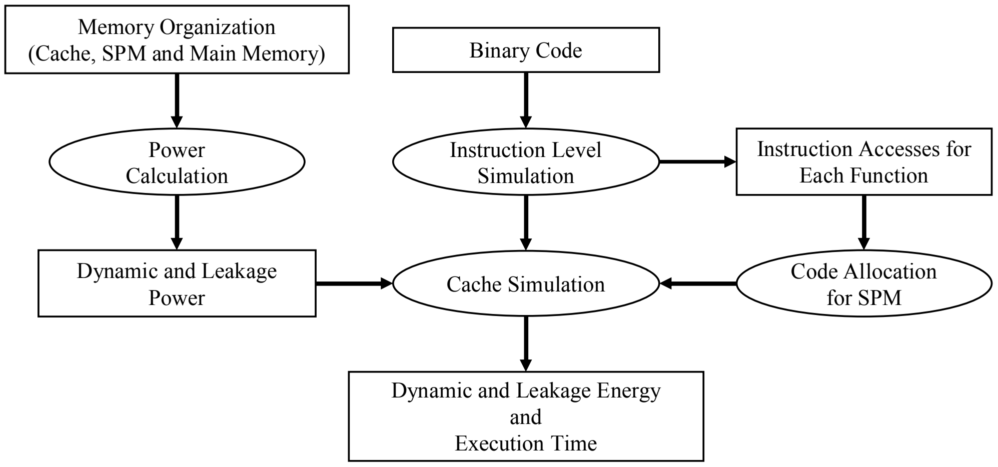

Memory Organization (Cache, SPM and Main Memory)
Binary Code
Power Calculation
Instruction Level Simulation
Instruction Accesses for Each Function
Dynamic and Leakage Power
Cache Simulation
Code Allocation for SPM
Dynamic and Leakage Energy and Execution Time

Figure 2. Experimental procedure

$$E_{M\_stby} = P_{M\_stby} \times EC / f$$

where P_{C_lkg} and P_{S_lkg} denote the leakage power of cache and SPM, respectively, and P_{M_stby} denote the stand-by power of main memory, and EC and f denote the execution cycles and the clock frequency, respectively.

3.3 Code Allocation to SPM

This work assumes that allocation of program code to SPM is static. This means that the contents of SPM are not changed during an execution of a program. Code allocation is performed at a function-level granularity. Our allocation method is based on the work in [2] where the allocation problem is formulated as a knapsack problem.

Let $func_i$ be the i-th function in a given program, $C(func_i)$ the code size of $func_i$, and $N(func_i)$ the dynamic number of executed instructions in $func_i$. Let x_i be a 0/1 variable whose value is 1 if $func_i$ is allocated to SPM, otherwise 0. Also, let S be the size of SPM. Then, the optimal allocation is obtained by finding x_i's that

maximize: $\sum_i N(func_i) \times x_i$

subject to: $\sum_i C(func_i) \times x_i \leq S$

In our experiments, the knapsack problem is solved with an open-source ILP solver lp_solve [11].

3.4 Memory Parameters

As described in Section 3.1, several memory system organizations are tested. In any organization, the total size of cache and SPM is fixed to 8K bytes. This size is determined based on the sizes of the benchmark programs so that the total size of cache and SPM is approximately 10 to 20% of the code size of the individual programs. The cache size is changed from 0 to 8K bytes, and the SPM size is determined accordingly. For example, if the cache size is 1K bytes, the SPM size becomes 7K bytes. The size of main memory is 2M bytes.

The values of read access energy and leakage power for cache, SPM and main memory are obtained with CACTI [8]. We assume that cache and SPM is made of SRAM while main memory is of DRAM. The technology size is varied from 65, 45 to 32 nm. The energy and power values also depend on the temperature, and in our experiment, the typical room temperature of 27 Celsius degree is assumed. The burst and random access times of main memory are also derived from CACTI, and their values are 18 and 4 cycles, respectively.

As seen in Section 3.2, the leakage energy depends on the clock frequency, and in turn, the clock frequency should be changed depending on the technology size. Considering the roadmaps of state-of-the-art embedded processors in industry, we assume that the clock frequency at 45 nm is 1 GHz, and those at 65 and 32 ns are derived based on the technology scaling.

4. EXPERIMENTAL RESULTS

In this work, we conduct two sets of experiments. First, we test the effectiveness of SPM at the 45 nm technology which is the state-of-the-art as of the publication date of this paper. Then, we scale the technology size.

4.1 Effectiveness of SPM at 45 nm

Figure 3 shows the energy and performance results at 45 nm. The lines indicate the normalized execution times. The bars show the energy consumption which are analyzed into four factors; stand-by energy of main memory, dynamic energy of main memory, leakage energy of SRAM (including both cache and SPM), and dynamic energy of SRAM.

From the figure, the use of cache only results in the largest energy consumption for all the programs. This demonstrates the effectiveness of SPM at 45 nm.

If we carefully look at a breakdown of the energy consumption, we notice that the DRAM stand-by energy is the most dominant. The DRAM dynamic energy is trivial except for patricia. Since patricia features less locality in memory accesses, which translates to more cache misses.

It is observed that the SRAM leakage energy is not trivial. For patricia, the SRAM leakage energy larger than its dynamic energy. Therefore, it is not feasible to ignore the leakage energy when we evaluate the energy consumption of memory systems.

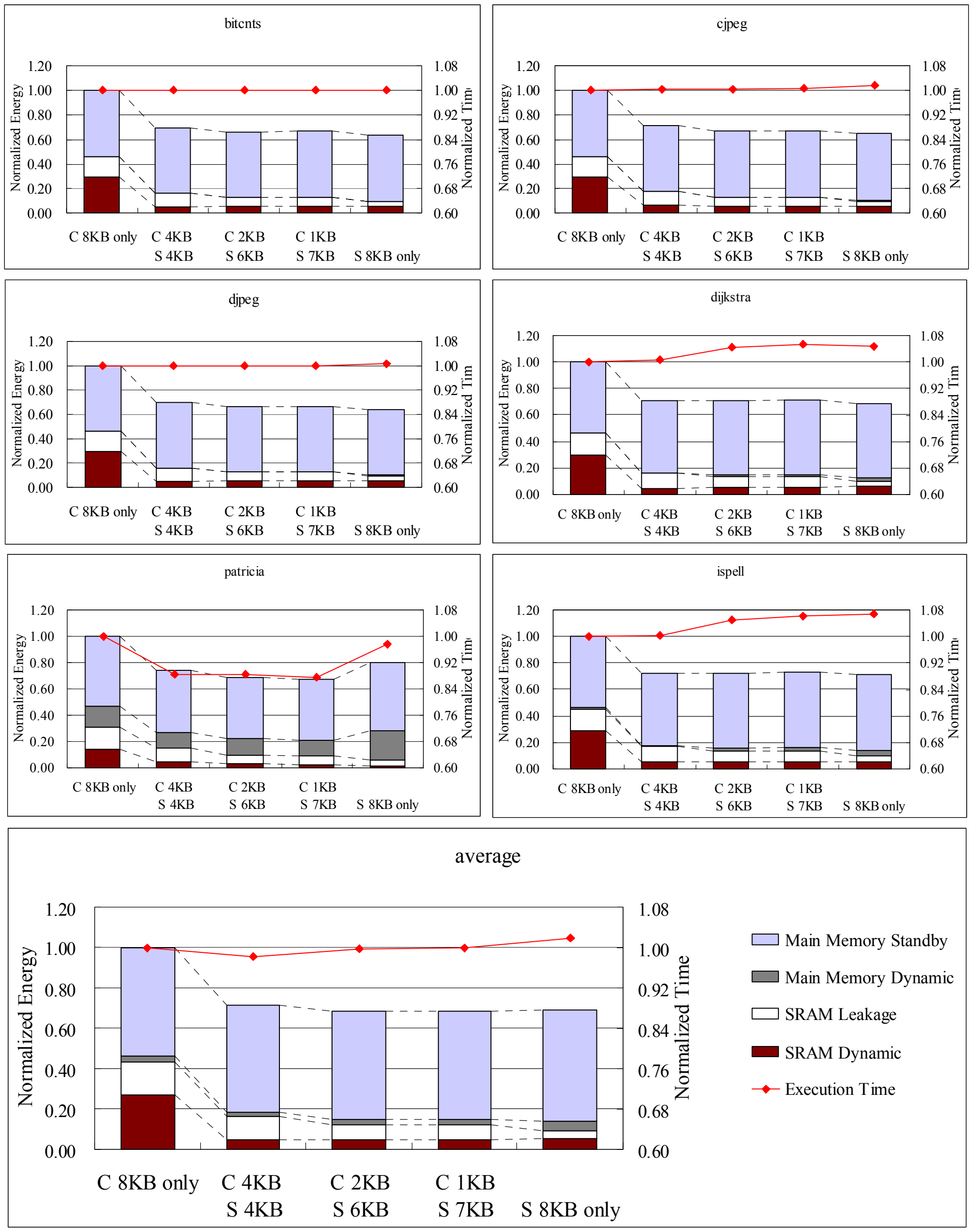


Figure 3. Energy consumption and performance at 45 nm

The SRAM leakage energy tends to decrease as the SPM size increases. This demonstrates the effectiveness of SPM on the leakage energy reduction. The execution time tends to become worse as the cache size decreases. This is because the decreased cache size brings more cache misses. However, the increase in the execution time is not significant, i.e., an average of 1.2%. Again, patricia is an exception where combination of cache and SPM gives the highest performance as well as the lowest energy.

4.2 Effects of Technology Scaling

We investigate the effect of technology scaling on the energy consumption. Figure 4 shows the average energy and execution time over the six benchmark programs for 65, 45 and 32 nm technologies. As the technology shrinks, both the energy consumption and the execution time are reduced. An interesting observation in this figure is that the percentage of the SRAM leakage energy is decreased along with the technology scaling. To our knowledge, this trend has not widely been recognized so far. One of the major reasons for the trend is due to the clock frequency. As the technology shrinks, the clock frequency is improved. Then, an application program finishes its execution in a shorter time, and thus, the leakage energy consumed during the execution is reduced. This implies that the memory should be

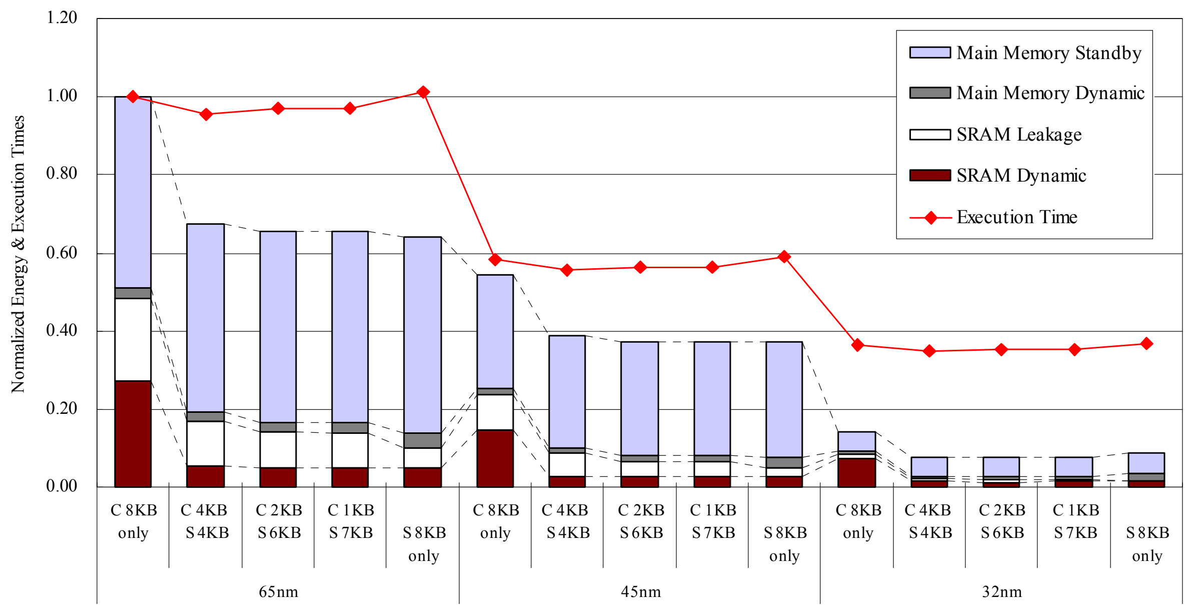


Figure 4. Effects on technology scaling on energy consumption and performance

powered-off when no task is ready to run. Otherwise, the leakage energy will be a serious problem.

5. CONCLUSIONS

Scratch-Pad Memory (SPM) has been considered as a promising solution for energy optimization in embedded systems and a large amount of literature has been published which address the energy optimization techniques using SPM. However, most of them focused on dynamic energy reduction, and neglected the leakage energy in their evaluations. As technology scales down, the leakage energy in memory devices could contribute to a significant portion of the total energy consumption. This paper first presents experimental studies on the energy effectiveness of SPM at 65, 45 and 32 nm with consideration of not only the dynamic energy but also the leakage energy. The results demonstrate the effectiveness of SPM in the deep submicron technology. It is also observed that the leakage energy becomes less significant along with the technology scaling.

Since our experimental results show that the DRAM stand-by energy is the most dominant, our future work will focus not only on SPM but also on the DRAM energy optimization.

6. ACKNOWLEDGMENTS

We thank Prof. Tohru Ishihara for suggesting our experiments. This work is supported in part by Core Research for Evolutional Science and Technology (CREST) from Japan Science and Technology Agency.

REFERENCES

[1] J. Montanaro, et al., “A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor,” *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 11, 1996.

[2] O. Avissar and R. Barua, “An Optimal Memory Allocation Scheme for Scratch-Pad-Based Embedded Systems,” *ACM Trans. on Embedded Computing Systems*, Vol. 1, No. 1, 2002.

[3] S. Steinke, et al., “Assigning Program and Data Objects to Scratchpad for Energy Reduction,” *Design, Automation and Test in Europe*, 2002.

[4] F. Angiolini, L. Benini, and A. Caprara, “An Efficient Profile-Based Algorithm for Scratchpad Memory Partitioning,” *IEEE Trans. on Computer-Aided Design*, Vol. 24, No. 11, 2005.

[5] P. R. Panda, A. Nicolau, and N. Dutt, *Memory Issues in Embedded Systems-On-Chip*, Kluwer Academic Publishers, 1998.

[6] A. Janapsatya, A. Ignjatovic, and S. Parameswaran, “Exploiting Statistical Information for Implementation of Instruction Scratchpad Memory in Embedded System,” *IEEE Trans. on VLSI Systems*, Vol. 14, No. 8, 2006.

[7] M. Kandemir et al., “Compiler-Directed Scratch Pad Memory Optimization for Embedded Multiprocessors”, *IEEE Trans. on VLSI Systems*, Vol. 12, No. 3, 2004.

[8] S. J. E. Wilton and N. P. Jouppi, “CACTI: An Enhanced Cache Access and Cycle Time Model,” *IEEE Journal of Solid State Circuit*, Vol. 31, No. 5, 1996.

[9] SimpleScalar LLC, http://www.simplescalar.com/”.

[10] M. R. Guthaus, et al., “MiBench: A Free, Commercially Representative Embedded Benchmark Suite”, *International Workshop on Workload Characterization*, 2001.

[11] S. E. Buttrey, “Calling the lp_solve Linear Program Software from R, S-PLUS and Excel”, *Journal of Statistical Software*, Vol. 14, Issue 4, 2005.