A Software Controllable Variable Line Size Cache Exploiting High On-Chip Memory Bandwidth for Low Power Embedded SoCs

Takatsugu Ono1
1Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan
ono@c.csce.kyushu-u.ac.jp

Koji Inoue2
Kazuaki Murakami2
2Faculty of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Japan
{inoue, murakami}@i.kyushu-u.ac.jp

Kenji Yoshida3
3System Fabrication Technologies, Inc.
Yokohama, Japan
yoshida@s-f-t.co.jp

Abstract—This paper proposes a software-controllable variable line-size (SC-VLS) cache architecture for low power embedded systems. High bandwidth between logic and a DRAM is realized by means of advanced integrated technology. In the merged logic/DRAM SoCs, it is important to reduce the DRAM energy consumption. The specific DRAM needs a small cache memory to improve the performance. We exploit the cache to reduce the DRAM energy consumption. The SC-VLS cache is able to change a line size to an adequate one at runtime with a small area and power overheads. We analyze the adequate line size and insert line size change instructions at the beginning of each function of a target program before executing the program. In our evaluation, it is observed that the SC-VLS cache reduces the DRAM energy consumption up to 50%, compared to a conventional cache with fixed 256B lines.

I. INTRODUCTION

In recent years, high memory bandwidth is available [9]. “System-in-Silicon (SiS)” is one of the architecture to realize the wide bandwidth which is 2.13 GB/s between logic and relatively large memory [9]. An ASIC and a specific DRAM (SiS-DRAM) are mounted onto a silicon interposer. Each chip is connected to the silicon interposer by a eutectic solder bump array. Through SiS fabrication, the micro bump formation and the chip mount are implemented in a mature and conventional IC fabrication process.

To keep the versatility of the SiS-DRAM, we connect the SiS-DRAM to standard bus systems. In the SiS architecture, it is important to reduce the SiS-DRAM energy consumption for an SoC energy reduction. The SiS-DRAM is connected to a small cache memory (SiS-cache) to improve the chip performance. Fig.1 shows a schematic diagram of assumed memory system. The SiS-cache size is smaller than L1 cache size under an area constraint. The bandwidth between the SiS-cache and the SiS-DRAM is high. On the other hand, the bandwidth between the SiS-cache and the microprocessor is limited by the poor bus performance.

The high bandwidth achieved by the SiS-DRAM makes it possible to reduce the cache miss penalty because large amount of data can be transferred at a time. If a large cache line size is employed, we can expect the effect of prefetching. However, it might worsen the system performance if programs do not have enough spatial localities of memory references. Furthermore, larger cache line sizes tend to consume more energy than small lines, because many DRAM banks need to be activated. If we use a SiS-cache line size which is not suitable for an application program, the cache increases not only the execution time but the SiS-DRAM energy consumption. Therefore, an intelligent control system for the SiS-cache are needed to achieve high performance and low energy consumption.

Several approaches have so far been proposed to optimize the cache line size [3][7][8][10][12]. They attempt to find the best line size by employing hardware monitors, which estimate the amount of spatial locality during the program execution. However, the extra mechanisms are needed for online profiling, resulting in a power and area overhead which is prohibitive in embedded systems. Most dynamic selection techniques have this weak point. Static selection techniques have been studied. Agarwal et al. [1] present a software approach for controlling memory bandwidth consumption. They target memory references that are likely to exhibit sparse memory access patterns. They employ the static analysis of source codes to identify memory references that have the potential to access memory sparsely. Vleet et al. [11] proposed using off-line profiling to determine the line sizes which are normal or large.
sizes. They have also proposed dynamic selection algorithm. These studies have looked at improving cache performance. They do not describe any mechanism to reduce energy consumption. Grun et al. [5] customize the local memory architecture suitable for both the diverse access patterns and locality types presented in an application program. They achieve to decrease the main memory bandwidth requirement thus generating power savings. However, their approach cannot change the cache line size during a program execution.

This paper proposes a Software-Controllable Variable Line-Size cache (SC-VLS cache) to reduce on chip DRAM energy consumption without sacrificing the performance. The SC-VLS cache is able to select an adequate line size which is determined before executing a program. We exploit memory mapped I/O instructions to specify the current cache line size and the instructions are inserted into the original program code. Unlike the previous hardware approaches, the SC-VLS cache does not require any hardware monitor, thus we are able to reduce energy consumption with trivial hardware overhead. In our evaluation, it is observed that the SC-VLS cache reduces the SiS-DRAM energy consumption up to 50%.

The rest of this paper is organized as follows. Section 2 describes a software-controllable variable line-size cache. We show evaluation results of our approach in Section 3 and Section 4 summarizes our work.

II. A SOFTWARE-CONTROLLABLE VARIABLE LINE-SIZE CACHE

A. Architecture

To reduce the SiS-DRAM energy consumption, we attempt to optimize the SiS-cache line size based at runtime. A Software-Controllable Variable Line-Size Cache (SC-VLS cache) is able to select a data transfer size, which it is indicated in a status register. In this paper, we call the SiS-cache which has ability to select a line size at runtime the SC-VLS cache. Fig.2 illustrates the block diagram of a direct-mapped SC-VLS cache. In the SC-VLS cachce, an SRAM cell array and a DRAM cell array are divided into several subarrays. Data transfer for cache replacement is performed between corresponding SRAM and DRAM subarrays. The SRAM and DRAM subarrays are 32B, respectively. The minimum line size is 32B, and four line sizes (32B, 64B, 128B and 256B) are provided. If we need to replace data of 64B line size, the data is transferred between two adjacent subarrays.

The line size when the SiS-cache access takes place is always the minimum line size, i.e. 32B. On cache hit, the required data is also read in the same manner as conventional cache. Otherwise, i.e. a cache miss occurred, the line which size is indicated in the status register is replaced.

In order to set the status register to an adequate line size, we exploit memory-mapped I/O instructions. A memory address is assigned to the status register. Therefore, by executing a store instruction, we can control the size of the SC-VLS cache. Note that the performance overhead caused by executing the store instructions is trivial.

B. Adequate Line Sizes Analysis and Code Generation

We execute cache simulation with each line size independently to determine an adequate line size. For instance, if the SC-VLS cache supports four line sizes, e.g. 32B, 64B, 128B and 256B, four times cache simulations are performed.

To analyze an adequate line size, the following approaches are performed.

1. An average cache miss rate of each function is calculated.
2. We compare the average cache miss rates with all line size candidates. A line size which the cache miss rate is the smallest is determined as an adequate line size. It is not easy to decide on adequate line size for each function because memory access behavior is changed intra program executions. However, we need to set an adequate line size for the function, because line size change instructions are inserted to a program code. Thus we determine the adequate line size based on average cache miss ratios.

We explain the analysis algorithm by using Fig.3. We assume that the SC-VLS cache supports 32B and 64B line sizes. First, we measure cache miss rates in each function using a cache simulator. The foo1() causes 10 misses out of 200 accesses. Second, we calculate an average miss rate of foo1(). An average miss rate of foo1() is 5.0% in case of 64B line size and is 2.0% in case of 32B line size. Finally, we compare the average cache miss rate with each line size. We decide an
adequate line size for the foo1() by comparing the average cache miss rate of 64B line size with that of 32B line size. In this example, the adequate line size is 32B, because the 32B lines produce a lower cache miss rate in average than the 64B lines. We decide the adequate line sizes of other functions in the same way as the foo1().

Line size change instructions are inserted into start of functions in original program code after the adequate line size analysis. Before a function is executed, the instruction sets the status register to indicate an adequate line size.

C. Energy Model

Total SiS-DRAM energy consumption \( E_{\text{mem}} \) is calculated by the following equation:

\[
E_{\text{mem}} = \sum_{i=1}^{\text{access}} \left( E_{\text{bank}} \times \frac{\text{LineSize}_{\text{SCVLSi}}}{\text{BankSize}} \right)
\]  

(1)

where, \( E_{\text{bank}} \) is energy consumption per a bank access, \( \text{LineSize}_{\text{SCVLSi}} \) is the SiS-cache size, \( \text{BankSize} \) is the bitwidth of the SiS-DRAM banks, and \( \text{access} \) is the total number of accesses to SiS-DRAM. We ignore the SC-VLS cache energy consumption because it is much smaller than the energy dissipated for SiS-DRAM accesses. As the same reason, we do not take the energy for status register access into account.

III. EVALUATION

A. Experimental Setup

In this section, we quantitatively evaluate the efficiency of our approach. The processor configuration assumed in this paper is shown in Tab.1. We assume a SC-VLS cache with the ability to dynamically choose four line sizes, 32B, 64B, 128B and 256B. We modified the SimpleScalar tool set [2] in order to measure cache miss rates.

We use five programs in MiBench [6] and a H.264 decode program [4]. Since the adequate line sizes depend on inputs, we used two types of input data sets in this evaluation. To analyze the adequate line size (analysis phase), we use a small and akiyo (QCIF) input sets for MiBench and H.264 decoder, respectively. A large and salesman (QCIF) input sets are used for evaluating our approach (execution phase).

B. Energy Reduction and SC-VLS Cache Miss Ratios

Here, we discuss the efficiency of the SC-VLS cache in terms of energy and cache miss reduction. Fig.4 shows the SiS-DRAM energy consumption and miss ratios for each benchmark program. The x-axis shows benchmark programs and the y-axis describes the energy consumption normalized to the result of 256B in each program. The y-axis also described the miss ratios.

We can observe that our approach reduces the SiS-DRAM energy comparing with the fixed 256B line size. The SC-VLS cache reduces the SiS-DRAM energy by more than 45% for jpeg, qsort, rijndael, sha and H.264 decoder. On the other hand, in susan, the reduction rate of the energy is about 3%. To maintain same miss rates as 256B line size, we need to select the large line size. Thus, the energy consumption is not able to reduce comparing with 256B line size.

The cache miss rate is reduced about 3% comparing with 256B in sha and susan. In particular, the miss rate is lower than the fixed 128B which indicates the lowest miss rate in sha. In jpeg, the miss rate of SC-VLS cache is the same as the fixed 256B. The SC-VLS cache reduces the SiS-DRAM energy consumption without sacrificing the miss rate. In other programs, however, the SC-VLS cache increases the miss rate about 3% comparing with the fixed 256B line size.
SC-VLS cache changes the line size to not suitable one since the adequate line size differs from the cache line size decided based on the analysis phase.

Fig.5 shows the SiS-cache miss rates for two input data sets. The FIX256B is the cache with the fixed 256B line size. The SCVLS-SI means we use the same input sets as the analysis phase in the execution phase. On the other hand, different input sets are used in the SCVLS-DI. The miss rate of the SCVLS-SI is lower than SCVLS-DI. In qsort and H.264 decoder, we consider that the adequate line sizes are different from each input set. In these programs, we need to analyze the adequate line size with more input sets. In rijndael, however, SCVLS-SI and SCVLS-DI show higher miss rate than FIX256B. In the execution phase, it is not always true that the line size is the as an assumption since the SC-VLS cache changes it. The cache hits depend on the previous cache states. The miss rate is increased since the SC-VLS cache changes the line size to not adequate one.

C. Performance Overhead

To set an adequate line size for the status register, our approach needs to execute extra store instructions. Executing the extra store instructions may worsen the processor performance, because it increases the total number of instructions executed. Tab.2 shows the instructions ratios of all executed instructions. It is observed that the overhead is lower than 1% in most benchmark programs.

IV. CONCLUSIONS

We have proposed a Software-Controllable Variable Line-Size cache (SC-VLS cache) to reduce on chip DRAM energy consumption for low power embedded systems. The cache line sizes are dynamically changed during a program execution with few hardware and performance overheads. In our evaluation, it is observed that the SC-VLS cache reduces SiS-DRAM energy consumption up to 50%, compared to a conventional cache which line size fixed to 256B.

In this work we need to analyze the adequate line sizes using software cache simulation. If the line size candidates are four, we need to simulate four times. In our feature work we plan to reduce the analysis costs.

ACKNOWLEDGEMENTS

The computation was mainly carried out using the computer facilities at Research Institute for Information Technology, Kyushu University. This work has been supported by System Fabrication Technologies, Inc.

REFERENCES