Empirical Study for Optimization of Power-Performance with On-Chip Memory

Chikafumi TAKAHASHI Mitsuhisa SATO Daisuke TAKAHASHI Taisuke BOKU
Center for Computational Sciences, University of Tsukuba
{takahasi, msato, daisuke, taisuke}@hpcs.cs.tsukuba.ac.jp

Hiroshi NAKAMURA Masaaki KONDO Motonobu FUJITA[†]
Research Center for Advanced Science and Technology, The University of Tokyo
{nakamura, kondo, mfujita}@hal.rcast.u-tokyo.ac.jp

Abstract

Recently, power-performance (performance per uniform power consumption) has become a more important factor in modern high-performance microprocessors. In processor design, it is a well-known that off-chip memory access has a large impact on both performance and power consumption. On-chip memory is one solution for this problem, so that many processors such as the Renesas SH-4 and some ARM architecture type processors adopt on-chip memory, which resides on the same layer as the cache memory. In this study, the effectiveness of the on-chip memory in an SH-4 processor was quantitatively examined by directly measuring the real power of the processor. For these experiments, we proposed a method that made use of the on-chip memory for power reduction. The experimental results show that the optimization of data transfer using on-chip memory reduces EDP(energy delay product) by up to 15.2%. As an extension of on-chip memory, we have proposed an on-chip RAM architecture called SCIMA (software controllable integrated memory architecture) which enables DMA (direct memory access) transfer to the on-chip memory. According to the empirical data from the SH-4 processor, it was found that the additional DMA transfer using SCIMA reduces EDP by up to 26.3%.

1. Introduction

In recent years, the clock frequency of modern microprocessors has been increased, due to advances in device technology and architecture. However, a high clock frequency causes extreme heat generation so that greater improvement of performance is prevented. Therefore, it is important to pursue improvement of performance related to power consumption (power performance), not only computational power.

In processor design, it is a well-known problem that the off-chip memory access has a large impact on both performance and power consumption. Modern processors have a cache used to transfer data between the register and the main memory. Cache utilization may reduce power consumption due to the reduction of off-chip main memory access. However, the cache holds data that is automatically selected by the hardware according to an access pattern. Furthermore, the cache holds data in a fixed unit size. These limitations cause useful data to drop from the cache and non-useful data to be acquired from the main memory. Transfer of non-useful data requires extra execution time, resulting in unnecessary energy consumption. Use of an on-chip memory is one solution to this problem. On-chip memory resides on the same layer as the cache memory, and is accessed as a part of the main memory. Many processors, such as the Renesas SH-4 and some ARM architecture processors adopt on-chip memory.

In this study, we quantitatively examine the effectiveness of the on-chip memory by directly measuring the real power of the processor. We then propose a method to use on-chip memory for the power reduction. Utilizing on-chip memory can improve power performance, because of software controlled memory access. As an extension of the on-chip memory, we have proposed an on-chip RAM architecture called SCIMA (software controllable integrated memory architecture) which enables DMA (direct memory access) transfer to the on-chip memory. Power performance improvement for this SCIMA architecture is then predicted based on the experimental data.

The contributions of this paper are as follows:

- We quantitatively examine the effectiveness of the on-chip memory in a real processor, SH-4, by directly measuring the real power of the processor.

[†]He is presently with the Japan Defense Agency.

- We propose a method to make use of on-chip memory for power reduction.
- We demonstrate that SCIMA can provide greater power-performance improvement by addition of DMA transfer from off-chip to on-chip memory.

Note that many quantitative evaluations of power performance have been performed by simulation, but few have been reported using real power measurements within processor architecture research.

The next section provides an overview of on-chip memory architecture as the background of our research, in addition to a method for use of on-chip memory. Section 3 details the experimental settings used for the evaluation of power performance of the SH-4 processor. The experimental results are given in section 4, and in section 5, improvement of power performance is discussed with consideration for the SH-4 processor with DMA as an assumption of SCIMA. Concluding remarks and future work are described in section 6.

2 Optimization of Power Performance with On-Chip Memory

2.1 On-chip Memory Architecture

The on-chip memory is a type of memory located on the same silicon chip as the processor core. Many on-chip memory architectures have been proposed. We proposed an on-chip memory architecture called SCIMA[6, 8]. SCIMA is an architecture which has SRAM on the same memory hierarchy layer as the cache. Kogge et al. [12] proposed a SIMD architecture called PIM (Processor-In-Memory), that has a processing unit on the memory module. Draper et al. [4] proposed to connect many PIM, in the so-called Data-IntensiVe Architecture (DIVA). Patterson et al.[10] proposed Intelligent RAM (IRAM), that has on-chip DRAM. In addition, some production processors have on-chip memory. For example, the L3 cache of a PowerPC processor in BlueGene/L can be used as on-chip memory[1]. The Renesas SH-4 processor has an on-chip memory mode, which allows half of the data cache to be reconfigured as on-chip memory[11]. There are also several chips that store intermediate data on the scratch pad memory[3, 14].

In this paper, we propose to optimize power performance with on-chip memory that resides on the same layer as the cache. The on-chip memory model is shown in Figure 1. Efficient management of data transfers between the on-chip memory and main memory decreases power consumption. On-chip memory has the following advantages.

(1-1) By explicitly specifying the transferred data necessary for computation, unexpected cache misses and non-useful data transfers caused by fixed cache line size data transfer can be avoided.

(1-2) Prefetching the data transfer enables overlap with computation by the DMA function, so that data transfer time can be hidden as in cache prefetch.

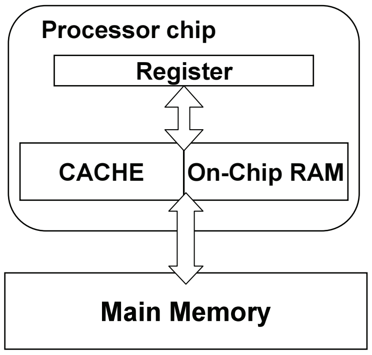

Figure 1. On-chip memory model

On the other hand, there are disadvantages of the on-chip memory, as follows:

(2-1) The requirement for software support to insert codes for explicit data transfer.

(2-2) The data transfer codes may increase the number of instructions.

To make use of on-chip memory, a programmer or compiler must insert data transfer instruction(s) explicitly.

2.2 Metrics of Power Performance

The power consumed in a processor can be divided into two parts. One is static power consumption, which is derived from leakage power consumption and is consumed constantly. The other is dynamic power consumption, which arises from switching activity on the bus and memory, and is consumed when the processor is working.

In this paper, we define the power consumption in the idle state as static power in order to simplify the discussion. This is different from the above definition. For example, although the power consumption in the clock line should be classified as dynamic power consumption, according to our definition, it is classified as static power consumption. Therefore, static power consumption is defined as stationary consumed power, and the dynamic power consumption is that after subtraction of static power from the working power.

The total power consumption is defined by:

$$E_{all} = P_{static} * t + E_{dynamic}$$

where P_{static} is the static power, $E_{dynamic}$ is the dynamic energy, and t is working time.

Reduction of data transfer by optimizing memory access results in a decrease of the switching activities of the bus and memory modules. Therefore, optimization of memory access results in the reduction of dynamic power consumption $E_{dynamic}$. In addition, the improvement of performance by (1-1) and (1-2); that is, shortening the working time t, can reduce the static power consumption P_{static}. Therefore, it is expected that improved use of on-chip memory will enable reduction of power consumption.

The model used in this study is similar to the SCIMA model. The power performance of SCIMA has already been evaluated by simulation, and the effectiveness is reported in [3, 7]. In this study, we examine the effectiveness of on-chip memory usage with a real processor.

2.3 Optimization Methods for On-chip Memory

In this subsection, methods to optimize programs for on-chip memory on the same layer as the cache are presented. The basic strategy of the optimization is similar to that for the cache, which employs the concept of temporal locality. The steps for optimization of on-chip memory are as follows:

1. Find data which can be re-used, and transfer the data to the on-chip memory from the main memory.

2. Execute computation using data in the on-chip memory.

3. Restore dirty data from the on-chip memory to the main memory.

If the data set size is larger than the on-chip memory size, the data should be divided into several chunks of smaller data.

Another possible optimization method is to hide the data transfer time. If the on-chip memory supports Direct Memory Access (DMA) and it can be overlapped with the computation, data pre-fetching can avoid stalling in the pipeline by waiting for data transfer completion. In this situation, performing software pipelining for data transfer and computation is effective.

The optimizations are effective when the access pattern for the main memory is predictable. It is difficult to optimize unpredictable memory access, because it is not possible to specify data transfer instruction(s) at the programming or compilation phase. Therefore, on-chip memory is used only for memory access with a predictable access pattern, and unpredictable memory access is performed by the traditional cache.

In this work, the optimizations are coded manually by hand. Fujita presented the automatic optimization by a compiler for SCIMA, which can be applied to our model. [5]

3 Experimental Setting

3.1 Target Processor

We have examined the power performance of the Renesas SH7751R SH-4 32-bit super scalar RISC processor, which is a commodity product. The SH7751R can use the 32 KB 2-way set associative data cache as 16 KB on-chip memory and 16 KB direct map cache. There are two modes in this processor: “OCR mode” (on-chip RAM mode) is a mode where on-chip memory is available, and “CACHE mode” is a mode where all the memory is used as a cache. The detailed specifications for the SH4 on-chip memory are provided in Table 1. In OCR mode, half of the data cache is mapped on a specific address area. This area can be accessed by the move instruction and so on. A Hitachi Ultra LSI system Solution Engine was used as the mother board for the evaluation. The installed OS is Linux. Details of the processor are given in Table 2.

3.2 Power Measurement Environment

A power measurement environment was built using a Hall device, which measures the electric current on a conducting wire by observing changes in the magnetic field. Sensors were set on the ATX power supply output line, input and output of a 3-terminal regulator. Two types of power data can be obtained as follows:

CORE This is the power supplied to the SH-4 processor core. It is measured at the output of the 3-terminal regulator(1.5 V), which connects to the processor core. This power is consumed at the registers, cache, on-chip memory, and other processor core, except the chip I/O.

MEMORY&BUS This is the power for memory and I/O. It is calculated by subtracting the 3-terminal regulator input from the ATX 3.3 V output. This 3.3 V power is supplied to the memory modules, the processor I/O

Table 1. SH4 (SH7751R) On-chip Memory

Mode	D-Cache	On-chip memory
CACHE mode	32 KB (2- way)	0 KB
OCR mode	16 KB (1-way)	16 KB

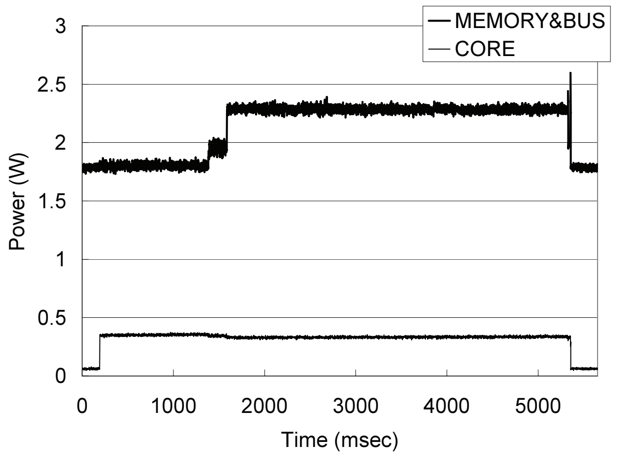

Figure 2. Power Consumption from the Data Scan Program

(bus), and other on board devices. Changes in power consumption for these memory related devices are observed from this power data.

The motherboard consumes ATX 5.0 V power. However, almost all of the power is consumed in unrelated devices. Therefore, only the CORE and MEMORY&BUS are discussed, not the ATX 5.0 V consumption.

3.3 Basic Power Characteristics

As a preliminary experiment, the power consumption was measured by executing a simple data scan program. The results are shown in Figure 2. The program sequentially scans the data in the memory area , and repeats the scan by gradually increasing the accessed area . Using this experiment, we can observe the behavior of power consumption related to the cache, because by gradually increasing the access area a cache miss is eventually caused.

As shown in Figure 2, the program starts at approximately 0.2 seconds, and the CORE increases to approximately 360 mW. Initially, the MEMORY&BUS is maintained at 1.6 W. However, after 1.7 seconds, the MEMORY&BUS increases to approximately 500 mW, and the CORE decreases to approximately 15 mW. The MEMORY&BUS is increased because cache misses occur frequently, resulting in many off-chip memory accesses, and the CORE is decreased due to the pipeline stall caused by the wait for data transfer.

The program stopped at 5.3 seconds, and the processor returned to idle state. At this time, the MEMORY&BUS is approximately 2.3 W, which is very large compared to the CORE 65 mW. This is due to the mother board used for development, which has many additional devices that constantly consume a large amount of power.

Table 2. Solution Engine detail

Name	SuperH Solution Engine
Type	MS7751RSE01
CPU	SH7751R (SH-4) 240 MHz
Memory	64 MB SDRAM 60 MHz
I/O	CompactFlash, IDE, etc
OS	Linux 2.4.18
Compiler	gcc 3.2.3

3.4 On-chip memory Optimization for SH-4 On-chip Memory

For the SH-4, which has no DMA support from main memory to on-chip memory, data transfer between the on-chip memory and main memory is performed by transfer instructions. In addition to this, the instructions support only transfer between registers and the main memory or registers and the on-chip memory, so that transfer of data between the on-chip memory and the main memory should occur via a register. Furthermore, the size of the data transfer from the main memory to the on-chip memory is fixed to 32 bytes on the bus, because transferred data must go through the cache. This may cause transfer of unnecessary data from the main memory to the cache in the case of data transfers smaller than 32 bytes.

In the experiment, we have optimized three programs: matrix multiplication, NPB CG (sequential version) and FFT. As shown in Table 1, the on-chip memory size is only 16 KB, which is half of the cache in CACHE mode. Consequently, in some cases, not only on-chip memory but also cache is used even in OCR mode.

3.4.1 Matrix Multiplication

We have optimized the matrix multiplication (MM) program, expressed by $C = A \times B$, where A, B and C are square matrices of the same size, with on-chip memory.

In the traditional cache architecture, cache blocking (or tiling) [9] is usually used to make the working set small enough to fit the cache size and to improve the data locality. However, unfortunate line conflicts may occur and degrade performance. In matrix multiplication, data access to non-contiguous addresses (row-ordered access to matrix B) may cause line conflicts. Therefore, these line conflicts should be avoided in order to obtain higher performance. Although this problem is partially solved by the set associativity of the cache, the data access pattern must be modified with software to completely avoid these conflicts. However, such optimization requires complicated programming to fit the properties of the cache size, line size, associativity, etc.

In OCR mode, the data of matrices A, B and C are blocked (as in the tiling transformation for cache). Next, the portion of matrices A and B are transferred to on-chip memory via registers. Therefore A and B do not cause cache conflicts. A portion of C is transferred to the cache to utilize the cache area. However, C has hardly any cache line conflict because the cache is only used for matrix C.

A matrix size of 500×500 was used for these experiments. The block size used in the blocking optimization was selected with consideration for the capacity of the on-chip memory or cache and the results obtained by the trial run. The following block sizes were carefully selected: 36×36 for CACHE mode and 32×32 for OCR mode.

3.4.2 NPB Kernel CG

The second workload is the CG (Conjugate Gradient) kernel of the NPB (NAS Parallel Benchmarks) [2]. This is a kernel loop to compute the eigenvalues of a large-scale symmetric random sparse matrix using the CG method. The kernel consists of double-nested loops, which include the multiplication of a sparse matrix and a vector. This part dominates the total execution time.

To perform multiplication of a sparse matrix and dense vector efficiently, the actual computation is expressed as $q = \sum_i (A(i) \times p(colidx(i)))$, where q is an element of the resultant vector, A is a packed array of non-zero elements of the target matrix, $colidx$ is an array for match-making between elements of the matrix and vector, and p is the target vector. As for the data access characteristics , A and $colidx$ are sequentially accessed, whereas p is randomly accessed.

Because matrix A and $colidx$ are sequentially accessed, they have spatial locality. However, they do not have data reusability. On the other hand, vector p has significant reusability. Data access to A and $colidx$ strongly degrades the cache-hit ratio for data access of vector p, since it is accessed randomly under capacity pressure on the cache. To improve the cache-hit ratio for p, one solution is to use cache blocking for p. By the application of blocking, vector p is divided into several portions, and each portion is calculated with partial data of matrix A that is associated to that portion. The elements of A and $colidx$ can be pre-sorted before multiplication to make a set of data that restricts the data accesses of p to each block. This is possible because the contents of A and $colidx$ are never modified through calculation in the CG method and this modification is only required once in the entire calculation.

In OCR mode, the same blocking optimization can be applied. The block of p is transferred to on-chip memory. In CACHE mode, the cache miss caused by the random access of vector p may degrade performance. However, in OCR mode, such an unexpected data swap-out never occurs and the reusability of vector p can be utilized. Each of the blocks of A and $colidx$ are transferred using only the cache. There is the possibility that A and $colidx$ are in conflict with each other. However, the conflict may be insignificant, because the access patterns of these matrices are sequential. Furthermore, the effect of cache miss is very small, because these matrices have no reusability.

A problem size of class-W was used. With a problem size of class-W, the size of the original (not-packed) matrix is 7000×7000 and the sparsity (ratio of non-zero elements) is approximately 1%. The block size of vector p is set at 1750 elements (elements of vector p are divided into four portions). The optimal size was determined by trial run.

3.4.3 Fast Fourier Transform

The third workload is the Fast Fourier Transform (FFT). The FFT is an algorithm used to compute the discrete Fourier transform quickly, and is used for various purposes. In this study, a three-dimensional double precision complex FFT program was used.

The 3-D FFT is computed by data pair of each dimensional axis. In this computation, stride memory access is caused at two directions of the dimensional axis . The stride access affects only 16 Bytes, which is smaller than the cache line size. This fine grain data access is inefficient, because half of the transferred data is non-useful data, and therefore unusable for the computation. To improve this problem, blocking optimization is applied for the sequential address direction of both modes. The optimization packs data to a coarse grain data size, which is larger than the cache line size[13]. Cache line conflicts caused by stride memory access can be improved by padding with non-useful data. However, this padding is not a perfect solution for improvement. In OCR mode, using on-chip memory clearly improves the cache line conflicts. However, OCM mode has a disadvantage because the on-chip memory size is half of the cache in CACHE mode, so that the blocking size is reduced.

A data size of 64×64×64 was used, and repeated twice. The block size used in the blocking optimization was selected by preliminary experiment. The following data sizes are selected: 16 for CACHE mode and 8 for OCR mode.

4 Power-Performance on SH-4

In this paper, the Energy Delay Product (EDP) was employed as an index of power performance. EDP is a widely used index in the field of low power research. EDP is defined by $EDP = P \times t \times t$, where P is average power, and t is time.

The experimental results are shown as Table 3. The EDP ratio, which is normalized by the EDP of CACHE mode, is

Table 3. Power-Performance of SH-4

		Matrix Multiplication		NPB CG		3D FFT	
		CACHE	OCR	CACHE	OCR	CACHE	OCR
EDP $(W * sec^2)$	CORE	35.5	31.4	761.0	699.8	2.892	2.561
	MEMORY&BUS	207.0	181.0	4615	4258	17.10	14.38
	Total	242.5	212.5	5376	4958	19.99	16.95
$E_{dynamic}$ $(W * sec)$	CORE	2.848	2.675	14.16	13.48	0.826	0.801
	MEMORY&BUS	1.876	1.470	22.54	20.89	0.766	0.669
	Total	4.723	4.145	36.70	34.36	1.592	1.471
E_{static} $(W * sec)$	CORE	0.621	0.585	2.781	2.681	0.177	0.164
	MEMORY&BUS	18.32	17.27	80.20	77.44	5.163	4.751
	Total	18.94	17.86	82.98	80.13	5.340	4.915
Execution time (sec)		10.25	9.66	44.93	43.30	2.88	2.66

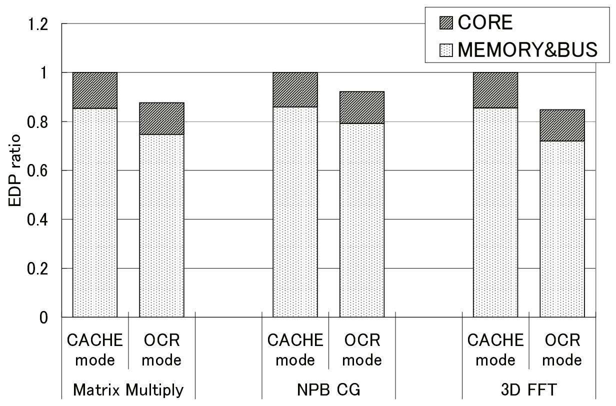

Figure 3. EDP Ratio of SH-4

shown as Figure 3. Figure 3 shows that EDPs are decreased when using on-chip memory for any application. The EDP is decreased to approximately 12.4% for MM, 7.4% for CG, and 15.2% for FFT.

Table 3 shows that the dynamic power consumption $E_{dynamic}$ is reduced in OCR mode for all applications. A reduction of MEMORY&BUS was observed, which results from the reduction of transferred data. It is expected that this will effect an improvement of cache line conflicts. The reduction of the CORE power is considered to be a result of decreasing cache renewal caused by cache misses. A reduction of the EDP results from these effects and results in the shortening of execution time.

The percentage of power reduction for E_{static} is larger than that for $E_{dynamic}$. This is because of the larger static power of the MEMORY&BUS, which is referred to in the previous section. If a different production motherboard (not the evaluation board) was used, the contribution to dynamic power reduction would be larger, because the power consumption of external devices would be smaller than the current environment.

5 Discussion

5.1 Improvement of Power Performance by SH-4 On-chip Memory

The experimental results show that utilizing on-chip memory improves power performance for the SH-4 processor. Considering the number of execute instructions, OCR mode may have some disadvantage for power consumption, because the OCR mode requires extra instructions to transfer data between the on-chip memory and main memory, and this causes an increase in the dynamic power consumption of the CORE. However, Table 3 shows a power consumption that is smaller than that of the CACHE mode . This indicates that the OCR mode performs better than CACHE mode, because off-chip memory access is reduced and the program runs faster due to less off-chip memory accesses.

The CORE dynamic power consumption shown in Figure 3 indicates that OCR mode has a lower value compared with CACHE mode. The on-chip memory reduces non-useful data transfer caused by cache misses. However, since data transfer between on-chip memory and main memory must go through the cache, the CORE power on OCR mode should include power consumed at the cache. This indicates the gain from removing non-useful data transfer exceeds the loss caused by passing the cache. As a result, the OCR mode has better power-performance than the CACHE mode.

It should be noted that the block size for the OCR mode became smaller than the CACHE mode, because the size of the on-chip memory is smaller than the full size of the cache. For example, for MM, while the maximum block size is 36 $\times$ 36 in in CACHE mode, it is only 32 $\times$ 32 in OCR mode. That is approximately 79% of the block size compared to the CACHE mode. For FFT, the OCR mode has only 50% block size. This is unavoidable because sufficient data size must be guaranteed for unpredictable memory access. However, the results of our experiments show that using on-chip memory has sufficient benefits when appropriately optimized.

5.2 Power Performance on SH-4 with DMA

5.2.1 Assumption of SH-4 with DMA

As shown in the previous section, data transfer between the on-chip memory and main memory must pass the cache and register. This limitation forces the OCR mode to consume more power by non-useful data transfer and execution of pipeline stall caused by non-useful data transfer. In addition, passing via the registers may obstruct the computation. It is expected that the adaptation of a DMA controller for on-chip memory will avoid these problems.

DMA support removes the obstructing computations, and enables the overlap of computations with data transfer in some situations. That is, it can be used in software pipelining. This is not only advantageous, but it also allows selection of data granularity and flexible transfer of data. Coarse grain data transfer can reduce the data transfer overhead, and fine grain data transfer reduces non-useful data transfer. These advantages might reduce power consumption through the reduction of data transfer and cause shorter execution time. The power performance using FFT was investigated under the assumption that DMA support is available for the SH-4 processor.

5.2.2 Assumption and Method

We assumed DMA support for the SH4 (SH7751R) processor, and DMA supports direct data transfer between the on-chip memory and main memory with block stride data transfer. Preliminary examination confirmed the data transfer time was sufficiently shorter than the computation time for the FFT program. Therefore the program was optimized by software pipelining. For pipelining, the on-chip memory is divided into two portions, which are alternately used for computation or transfer. This optimization makes the block size small, and it is only "2". The rate of pipelined data transfer is shown as:

$$\frac{\begin{array}{c}5*NX*\log_2 NX*(NY-NBLK*2)*NZ+\\ 5*NY*\log_2 NY*(NX-NBLK*2)*NZ+\\ 5*NZ*\log_2 NZ*(NX-NBLK*2)*NY\end{array}}{5*NX*NY*NZ*\log_2(NX*NY*NZ)}$$

where the element size is $NX \times NY \times NZ$, and the block size is $NBLK$.

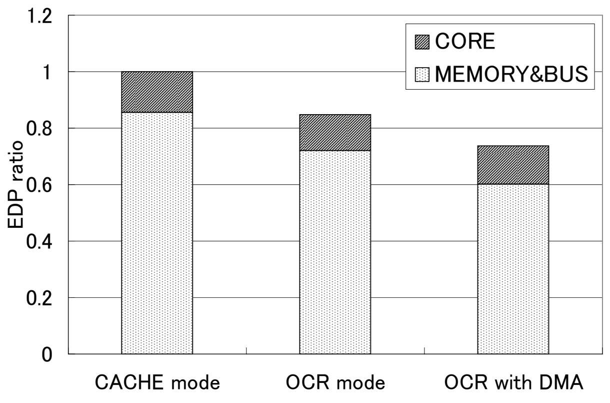

Figure 4. EDP Ratio of SH-4 with DMA

This equation shows that 93.75% of data transfer can be pipelined for this evaluation. Power performance is calculated by this ratio and the evaluation results of the actual SH-4 processor. We assume that the static power consumption can be reduced by shortening the execution time, which is enabled by software pipelining. The time required for data transfer on the SH-4 processor with DMA is calculated by the time difference between the pipelined program and the program without data transfer.

5.2.3 Results

The calculated EDP ratio is shown as Figure 3. The figure shows that OCR mode with adopted DMA decreases EDP by 26.3% compared with the CACHE mode, and by 13.1% EDP compared with OCR mode without DMA. As this estimation does not regard the influence of memory traffic reduction caused by variable grain data transfer, the power consumption for a real DMA is expected to be smaller than this result.

The result of the evaluation is worse than that for the SCIMA results. The SCIMA evaluation shows a 70% decrement of EDP for NPB CG [7], which is a larger effect than the result in this study. This is because SCIMA has a larger number of cache / on-chip memory way than SH-4 processor and it can switch every way to on-chip memory or cache. It reduces the disadvantage of OCR mode in SH-4 processor.

The technique of cache prefetching can achieve a similar efficiency to software pipelining for the on-chip memory. Cache prefetching can also reduce EDP in CACHE mode. However, to prefetch data precisely, it is necessary to execute the prefetch instruction with accurate timing. Poor execution may eliminate prefetched data from the cache, and a grain of data transfer is fixed to the cache line size. Therefore, we propose that use of an on-chip memory has more advantages than cache prefetch for power performance improvement.

6 Conclusion

In this paper, we have presented an empirical study for the optimization of power performance with an on-chip memory. We have proposed a method to make use of on-chip memory and quantitatively examined the effectiveness of the on-chip memory in a SH-4 processor by directly measuring the real power usage of the processor, The experimental results show that utilizing on-chip memory enables optimum data transfer, and reduces EDP by up to approximately 15.2%. According to our empirical evaluation of the SH-4 processor, we evaluate the effectiveness of DMA transfer between the on-chip memory and main memory. It was found that DMA transfer can reduce EDP by up to 26.3%.

Future work will include the investigation of other recent processors. Currently, we are planning to make a detailed examination of the SH-4A processor, which is a new processor that has DMA supported on-chip memory.

Acknowledgment.

This research is supported by “Research & Development Project for Enabling Technologies for Future Supercomputing” project in “Research and Development For Next-Generation Information Technology” program, which is funded by the Ministry of Education, Culture, Sports, Science and Technology.

References

[1] G. Almasi et al. Unlocking the performance of the blue-gene/l supercomputer. In *Proc. SC04*, page 57, 2004.

[2] D. Bailey et al. The NAS parallel benchmarks 2.0. In *NASA Ames Research Center Report*, NAS-05-020, 1995.

[3] K. Diefendorff. Sony’s emotionally charged chip. *Microprocessor Report*, 13(5), 4 1999.

[4] J. T. Draper et al. The architecture of the DIVA processing-in-memory chip. In *Proc. ICS 2002*, pages 14–25, 2002.

[5] M. Fujita et al. Data Movement Optimization for Software-Controlled On-Chip Memory. In *Workshop on Interaction between Compiler and Architecture (INTERACT-8)*, 2004.

[6] M. Kondo et al. SCIMA: Software controlled integrated memory architecture for high performance computing. In *Proc. ICCD2000*, pages 105–111, 2000.

[7] M. Kondo et al. Reducing Memory System Energy by Software-Controlled On-Chip Memory. *IEICE Trans. on Electronics*, E86-C(4):550–588, Apr. 2002.

[8] M. Kondo et al. Software-controlled on-chip memory for high-performance and low-power computing. *ACM SIGARCH Computer Architecture News*, 30:7–8, 2002.

[9] M. Lam et al. The cache performance and optimizations of blocked algorithms. In *Proc. ASPLOS-IV*, pages 63–74, 1991.

[10] D. Patterson et al. A Case for Intelligent RAM: IRAM. *IEEE Micro*, 17(2):34–44, Apr. 1997.

[11] Renesas. *SuperH RISC engine SH-4 Programming Manual*, 5th edition, 2001.

[12] T. Sunaga et al. A processor in memory chip for massively parallel embedded applications. *IEEE J. of Solid State Circuits*, pages 1556–1559, Oct. 1996.

[13] D. Takahashi. Efficient implementation of parallel three-dimensional FFT on clusters of PCs. *Computer Physics Communications*, 152:144–150, 2003.

[14] J. Turley. Strongarm speed to triple. *Microprocessor Report*, 13(6), 5 1999.