IPSJ Transactions on System LSI Design Methodology Vol.8 100–104 (Aug. 2015)

[DOI: 10.2197/ipsjtsldm.8.100]

Short Paper

An Allocation Optimization Method for Partially-reliable Scratch-pad Memory in Embedded Systems

Takuya Hatayama[1,a) Hideki Takase[1] Kazuyoshi Takagi[1] Naofumi Takagi[1]

Received: December 5, 2014, Revised: March 13, 2015,
Accepted: April 29, 2015, Released: August 1, 2015

Abstract: In this paper, we propose the use of a memory system which has a partially reliable scratch-pad memory (SPM). The reliable region of the SPM employing the ECC is higher soft error tolerant but larger energy consumption than the normal region. We propose an allocation method in order to optimize energy consumption while ensuring required reliability. An allocation method about instruction and data to proposed memory system is formulated as integer linear programming, where the solution archives optimal energy consumption and required reliability. Evaluation result shows that the proposed method is effective when overhead for error correction is large.

Keywords: energy minimization, reliability, scratch-pad memory

1. Introduction

Scratch-Pad Memory (SPM) is an on-chip SRAM mapped into an address space that is disjoint from the off-chip memory. SPM is efficient in terms of area, energy and predictability compared with the cache [1]. Therefore, SPM is often employed as on-chip memory in modern embedded systems. However, designers or software must determine the allocation of instruction and data to SPM.

In recent years, the rise in memory soft error rate (SER) has been a major concern with increasing the demands on the reliability of embedded systems. A soft error means a transient fault in the circuits, which is caused by alpha ray or neutron. Frequently accessed instructions and data needs to enhance soft error tolerance. Especially, the instruction memory needs to enhance soft error tolerance because instruction memory is accessed in almost every cycle. One way to enhance the soft error tolerance of memory is by using a parity code for detecting soft error. An error-correcting code (ECC) is employed when error correction is required.

In modern semiconductor manufacturing technology, the SERs in FIT/bit for DRAM with ECC, SPM with ECC and SPM without ECC are 10^{-12}, 10^{-7} and 10^{-4}, respectively [2], [3]. However, employing an ECC in a simplistic form may excessively increase overheads such as the energy consumption, area and execution time.

There are some previous studies about reliable memory systems. Reference [4] uses a parity code to error detection and a data duplication method to error correction. Reference [5] proposes 2-bit interleaved-parity per word to error detection. Reference [6] proposes the Partially Protected Caches (PPC) architecture in which ECC is applied to a part of the cache. The size of cache applied ECC is enough small for the access latency not to exceed the access latency of normal cache. The PPC is utilized to enhance soft error tolerance in multimedia applications [7]. Reference [8] utilizes the normal cache and SPM with ECC. The SPM is enough small for the access latency not to exceed the access latency of cache.

The purpose of this paper is energy optimization of embedded systems while ensuring the required reliability. We propose a use of a partially reliable SPM that an ECC is applied to part of the SPM region. The region of SPM where the ECC is applied is higher soft error tolerant but larger energy consumption than the region of SPM without ECC. The allocation optimization method for partially reliable SPM is formulated as an integer linear programming (ILP). The objective function of the ILP is energy consumption. The constraints are the SPM capacities and the vulnerability of the whole system. An optimal solution of the proposed ILP problem corresponds to the allocation which minimizes the energy consumption of the system while the ensuring required reliability.

2. Partially-reliable Memory System

Figure 1 shows the proposed memory system which has partially reliable SPM. We employ its SPM to organize the memory system as an on-chip memory. The partially reliable SPM is

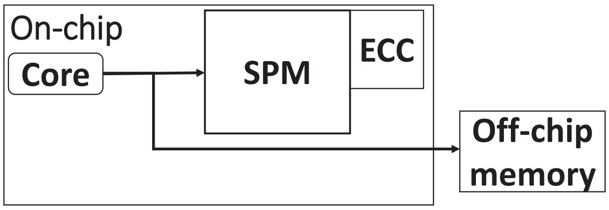


Fig. 1 Proposed memory system which has partially reliable SPM.

[1] Graduate School of Informatics, Kyoto University, Kyoto 606–8501, Japan

a) hatayama@lab3.kuis.kyoto-u.ac.jp

SPM that the ECC is applied to a part of region. The SPM region where an ECC is applied is referred to as reliable SPM. In contrast, the SPM region without the ECC is referred to as normal SPM. The reliable SPM takes high soft error tolerant but large energy consumption. In contrast, the normal SPM takes small energy consumption but low soft error tolerant. Instructions and data which require high reliability can be allocated to the region of reliable SPM. In contrast, instructions and data not requiring high reliability are allocated to the region of normal SPM. The proposed memory system can contribute reduction of the energy consumption while the ensuring required soft error tolerance.

3. Allocation Optimization Problem

This section describes the problem definition of an allocation problem to partially-reliable SPM.

The purpose of this paper is the minimization of the energy consumption while ensuring the required reliability. We propose an allocation method about instructions and data to partially reliable SPM to achieve the purpose.

In this paper, the instruction and data allocation granularities as memory block are functions of source code and set of data (e.g., variable, array), respectively. The candidate memories where these memory blocks are allocated to are the normal SPM, reliable SPM and off-chip main memory. We define the allocation optimization problem as to determine an optimal instruction and data allocation in terms of energy minimization while ensuring a vulnerability of system.

The energy consumption of the memory is calculated by the energy consumption per memory access and the number of accesses. In this paper, instructions and data which require high reliability are defined as frequently accessed ones. The vulnerability of the whole system is determined by the vulnerability of instructions and data weighted by SER of memory. The vulnerabilities of instructions and data are defined by the proportion of the number of accesses, because errors in instructions and data frequently accessed widely propagates. Ensuring the required reliability means that the vulnerability of the whole system should not be more than a given value by the instructions and data allocation. It should be noted that we assume a unified memory architecture, where instructions and data are allocated to one SPM. However, our method can easily be applied to the harvard memory architecture by modifying some constraints of our ILP.

Reference [1] energy consumption was used as an objective function and the SPM capacity as a constraint, while the allocation problem was treated as the knapsack problem. The problem was then solved as an ILP problem. We add vulnerability as a constraint to the allocation problem; therefore, it is reasonable to formulate our allocation problem as an ILP problem.

4. Integer Linear Programming Problem

This section describes the formulation of our allocation optimization method as an ILP problem.

4.1 Preliminaries

Table 1 shows the constants in our ILP problem. In this paper, $inst_i$ and $data_j$ denote the i-th memory block of instruction and the j-th memory block of data, respectively. S_{inst_i}, N_{inst_i}, S_{data_j}, NR_{data_j} and NW_{data_j} are obtained from a statical analysis by task profiling, and the other constants are determined by the target system configuration. The decision variables in our ILP problem, which take the binary values shown in **Table 2**, indicate that the allocation in which memories $inst_i$ and $data_j$ is allocated to. In the rest of this paper, the RSPM means the region of the SPM where an ECC is applied.

Table 1 Constants.

Constant	Definition
S_{inst_i}, S_{data_j}	Memory size of $inst_i$ and $data_j$
N_{inst_i} NR_{data_j}, NW_{data_j}	The number of access for $inst_i$ The number of read/write access for $data_j$
$ER_{MM}, ER_{RSPM}, ER_{SPM}$ $EW_{MM}, EW_{RSPM}, EW_{SPM}$	Energy consumption per read access to main memory, RSPM and SPM Energy consumption per write access to main memory, RSPM and SPM
C_{RSPM}, C_{SPM}	The memory capacity of RSPM and SPM
$R_{MM}, R_{RSPM}, R_{SPM}$	SER of main memory, RSPM and SPM

Table 2 Binary variables.

x_{inst_i}	$x_{inst_i} = 1$	$\Longleftrightarrow$	$inst_i$ is allocated in main memory
x_{data_j}	$x_{data_j} = 1$	$\Longleftrightarrow$	$data_j$ is allocated in main memory
y_{inst_i}	$y_{inst_i} = 1$	$\Longleftrightarrow$	$inst_i$ is allocated in RSPM
y_{data_j}	$y_{data_j} = 1$	$\Longleftrightarrow$	$data_j$ is allocated in RSPM
z_{inst_i}	$z_{inst_i} = 1$	$\Longleftrightarrow$	$inst_i$ is allocated in SPM
z_{data_j}	$z_{data_j} = 1$	$\Longleftrightarrow$	$data_j$ is allocated in SPM

4.2 Vulnerability

We define the vulnerability of the whole system V_{system} as follows. V_{system} corresponds to the FIT of whole system.

$$V_{system} = \sum_i V_{inst_i} \cdot FIT(inst_i) + \sum_j V_{data_j} \cdot FIT(data_j) \quad (1)$$

$$V_{inst_i} = N_{inst_i} / \sum_i N_{inst_i} \quad (2)$$

$$V_{data_j} = (NR_{data_j} + NW_{data_j}) / \left(\sum_j (NR_{data_j} + NW_{data_j}) \right) \quad (3)$$

Here, V_{inst_i} and V_{data_j} denote the vulnerabilities of $inst_i$ and $data_j$, that are the ratio of access frequency for $inst_i$ and $data_j$ to all, respectively. $FIT(inst_i)$ and $FIT(data_j)$ represent the FIT of $inst_i$ and $data_j$, respectively. The instructions and data which are frequently accessed have high vulnerability. For example, if the number of instruction accesses with errors is twice, the probability of error propagation is twice. Therefore, $FIT(inst_i)$ and $FIT(data_j)$ are weighted by V_{inst_i} and V_{data_j}, respectively. V_{system} corresponds to a weighted average of the FIT for the instructions and data.

$$FIT(inst_i) = x_{inst_i} \cdot R_{MM} \cdot S_{inst_i} + y_{inst_i} \cdot R_{RSPM} \cdot S_{inst_i} + z_{inst_i} \cdot R_{SPM} \cdot S_{inst_i} \quad (4)$$

$$FIT(data_j) = x_{data_j} \cdot R_{MM} \cdot S_{data_j} + y_{data_j} \cdot R_{RSPM} \cdot S_{data_j} + z_{data_j} \cdot R_{SPM} \cdot S_{data_j} \quad (5)$$

The vulnerability of the whole system is high if the frequently accessed memory blocks are allocated to the memory which has high SER.

4.3 Objective Function

The objective function of our ILP is the energy consumption of the memory system. We aim to minimize energy consumption.

$$\textbf{minimize} : E_{system} = \sum_i E(inst_i) + \sum_j E(data_j) \qquad (6)$$

$E(inst_i)$ and $E(data_j)$ denote the energy consumption of $inst_i$ and $data_j$, respectively. These are calculated by energy consumption of memory access and the number of accesses as follows.

$$\begin{aligned} E(inst_i) = {} & x_{inst_i} \cdot ER_{MM} \cdot N_{inst_i} + y_{inst_i} \cdot ER_{RSPM} \cdot N_{inst_i} \\ & + z_{inst_i} \cdot ER_{SPM} \cdot N_{inst_i} \end{aligned} \qquad (7)$$

$$\begin{aligned} E(data_j) = {} & x_{data_j} \cdot (ER_{MM} \cdot NR_{data_j} + EW_{MM} \cdot NW_{data_j}) \\ & + y_{data_j} \cdot (ER_{RSPM} \cdot NR_{data_j} + EW_{RSPM} \cdot NW_{data_j}) \\ & + z_{data_j} \cdot (ER_{SPM} \cdot NR_{data_j} + EW_{SPM} \cdot NW_{data_j}) \end{aligned} \qquad (8)$$

4.4 Constraints

There are three constraints in our ILP problem.

The first is about allocation. Each memory object must be allocated to only one memory.

$$\forall i, \; x_{inst_i} + y_{inst_i} + z_{inst_i} = 1 \qquad (9)$$

$$\forall j, \; x_{data_j} + y_{data_j} + z_{data_j} = 1 \qquad (10)$$

The second is about the SPM capacities. Sum of S_{inst_i} and S_{data_j} allocated to RSPM or SPM exceed the capacities of them.

$$\sum_i y_{inst_i} \cdot S_{inst_i} + \sum_j y_{data_j} \cdot S_{data_j} \leq C_{RSPM} \qquad (11)$$

$$\sum_i z_{inst_i} \cdot S_{inst_i} + \sum_j z_{data_j} \cdot S_{data_j} \leq C_{SPM} \qquad (12)$$

The third is about vulnerability. Vulnerability of the system V_{system} should not be more than the given vulnerability $V_{\max}$.

$$V_{system} \leq V_{\max} \qquad (13)$$

Here, $V_{\max}$ is specified as the requirement of the system design.

5. Evaluation

We evaluated the effectiveness of proposed method in experiments.

We employed SkyEye [9] and TOPPERS/ASP kernel [10] as the simulator of ARM and real-time operating systems for the experimental environment, respectively. The benchmark programs were `basicmath`, `bitcount`, `susan` and `dijkstra` from MiBench [11]. We executed the benchmark programs and profiled the execution logs in order to obtain the constants in Table 1. We used CPLEX [12] as the ILP solver. The ILP problems of the benchmark programs comprised several hundred decision variables. In our computer environment, all the ILP problems of the benchmark programs were solved within 0.1 seconds. Therefore, we can conclude that the proposed ILP problems were solved within a reasonable amount of time.

As the amount of energy consumption model for each memory, we firstly used CACTI6.5 [13]. **Table 3** shows the energy consumption per access to each memory obtained by CACTI6.5. It should be noted that we employed the same value for write access as read access since CACTI6.5 only output the value of energy consumption for read access. An additional consideration about the use of CACTI6.5 is that it only calculates the influence of the addition of an ECC bit. The ECC coding and decoding processes are ignored to calculate energy consumption. Reference [14] reported that these processes consume 40.783 pJ in case of the 70 nm CMOS process [*1]. 40.783 pJ is approximately 10 times as large as energy consumption in Table 3. Therefore, we also evaluated the case for $ER_{RSPM} = ER_{SPM} * 10$ and $EW_{RSPM} = EW_{SPM} * 10$.

Table 3 Energy consumption obtained by CACTI.

Memory	Size	Energy [pJ]
Main Memory	4 MB	456.266
RSPM	2 KB	3.838
	4 KB	4.952
	6 KB	5.903
	8 KB	6.763
SPM	2 KB	3.604
	4 KB	4.714
	6 KB	5.584
	8 KB	6.383

Table 4 Results in case of values obtained by CACTI.

$V_{\max}/V_{origin}$	Configuration	Normalized energy consumption			
	RSPM/SPM	basicmath	bitcount	susan	dijkstra
1e0	8 KB/0 KB	1.00	1.00	1.00	1.00
1e1	6 KB/2 KB	1.01	0.93	1.16	0.97
	4 KB/4 KB	1.17	0.92	1.96	1.14
	2 KB/6 KB	1.80	4.23	4.01	1.95
	0 KB/8 KB	5.39	42.31	14.95	14.33
1e2	6 KB/2 KB	0.99	0.88	0.96	0.96
	4 KB/4 KB	0.98	0.84	0.94	0.94
	2 KB/6 KB	1.09	1.37	1.11	1.11
	0 KB/8 KB	4.67	33.55	12.02	11.84
1e3	6 KB/2 KB	0.97	0.72	0.95	0.91
	4 KB/4 KB	0.98	0.81	0.93	0.94
	2 KB/6 KB	0.97	0.74	0.95	0.92
	0 KB/8 KB	1.00	0.96	0.99	0.99

Table 5 Results in case of $E_{RSPM} = E_{SPM} * 10$.

$V_{\max}/V_{origin}$	Configuration	Normalized energy consumption			
	RSPM/SPM	basicmath	bitcount	susan	dijkstra
1e0	8 KB/0 KB	1.00	1.00	1.00	1.00
1e1	6 KB/2 KB	0.94	0.86	0.94	0.90
	4 KB/4 KB	0.98	0.75	1.10	0.87
	2 KB/6 KB	1.28	1.09	1.65	1.03
	0 KB/8 KB	3.35	6.61	5.14	5.21
1e2	6 KB/2 KB	0.89	0.70	0.82	0.79
	4 KB/4 KB	0.84	0.61	0.71	0.72
	2 KB/6 KB	0.84	0.62	0.65	0.69
	0 KB/8 KB	2.91	5.24	4.14	4.30
1e3	6 KB/2 KB	0.69	0.19	0.48	0.39
	4 KB/4 KB	0.63	0.13	0.38	0.36
	2 KB/6 KB	0.62	0.14	0.35	0.35
	0 KB/8 KB	0.62	0.15	0.34	0.36

Tables 4 and **5** show the evaluation results with the values of CACTI and with the assumption of $E_{RSPM} = E_{SPM} * 10$. The baselines for energy consumption were the cases with 8 KB RSPM and 0 KB SPM. The vulnerability of the baselines was V_{origin}. We varied $V_{\max}$ from 10 times that of V_{origin} to 1,000 times that of V_{origin}. We assumed that the SER of the SPM with ECC was 1,000 times lower than the SER of the SPM without ECC. Even with

[*1] We gave parameters which corresponded to the configuration of Ref. [14] to CACTI.

1e3, $V_{\max}$ did not exceed the vulnerability when using the normal SPM. Therefore, these values satisfied the aims of this paper. For example, in Tables 4 and 5, 1e1 denotes $V_{\max} = V_{origin} * 10$.

From Tables 4 and 5, proposed method is effective in case overhead for coding and decoding are large in all programs. Therefore, the proposed method becomes effective in case overhead for the error detection and correction are large. Additionally, the proposed method is effective in programs which have high locality of memory access. In fact, the proportion of access to SPM and RSPM of `bitcount` is more than 90%.

From evaluation results, we can observe that the optimal size of ECC is varied by programs and $V_{\max}$. It is important to determine the appropriate configuration according to programs and $V_{\max}$.

6. Conclusion

In this paper, we proposed the memory system which has partially reliable SPM and a memory allocation method for its memory system. We defined the memory allocation problem and formulated the problem as ILP. the proposed method optimizes the energy consumption while ensuring the required reliability. From evaluation, the proposed method is effective in case the ECC overhead is large and programs have high spatial locality. However, our criteria of vulnerability ignores some factors such as life time of instructions and data. In future, we will define more appropriate criteria of vulnerability. It is also interesting that a searching method to optimize the size of applying ECC to the SPM region will be proposed.

Acknowledgments The part of this work was supported by JSPS KAKENHI Grant Number 26870303.

References

[1] Banakar, R., Steinke, S., Lee, B.-S., Balakrishnan, M. and Marwedel, P.: Scratchpad memory: A design alternative for cache on-chip memory in embedded systems, *Proc. 10th International Symposium on Hardware/Software Codesign*, pp.73–78 (May 2002).

[2] Slayman, C.W.: Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations, *IEEE Trans. Device and Materials Reliability*, Vol.5, No.3, pp.397–404 (Dec. 2005).

[3] Sugihara, M., Ishihara, T. and Murakami, K.: Task scheduling for reliable cache architectures of multiprocessor systems, *Design, Automation Test in Europe Conference & Exhibition* (*DATE*), pp.1–6 (Apr. 2007).

[4] Li, F., Chen, G., Kandemir, M. and Kolcu, I.: Improving scratchpad memory reliability through compiler-guided data block duplication, *IEEE/ACM International Conference on Computer-Aided Design*, pp.1002–1005 (Nov. 2005).

[5] Farbeh, H., Fazeli, M., Khosravi, F. and Miremadi, S.G.: Memory mapped SPM: Protecting instruction scratchpad memory in embedded systems against soft errors, *2012 9th European Dependable Computing Conference* (*EDCC*), pp.218–226 (May 2012).

[6] Lee, K., Shrivastava, A., Dutt, N. and Venkatasubramanian, N.: Partitioning techniques for partially protected caches in resource-constrained embedded systems, *ACM Trans. Design Automation of Electronic Systems* (*TODAES*), Vol.15, No.4, pp.30:1–30:30 (Sep. 2010).

[7] Lee, K., Shrivastava, A., Issenin, I., Dutt, N. and Venkatasubramanian, N.: Partially protected caches to reduce failures due to soft errors in multimedia applications, *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems*, Vol.17, No.9, pp.1343–1347 (Sep. 2009).

[8] Morimoto, T., Kobayashi, R. and Sugihara, M.: A memory objects allocation scheme to improve soft errors tolerance on embedded systems with a scratch-pad memory, *ESS2011*, Vol.2011, pp.12-1–12-10 (Oct. 2011).

[9] Skyeye_wiki, available from ⟨http://skyeye.sourceforge.net/⟩.

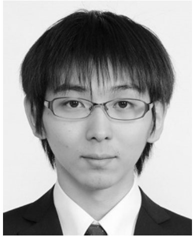
[10] TOPPERS Project, available from ⟨http://www.toppers.jp/⟩.

[11] Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M.. Mudge, T. and Brown, R.B.: Mibench: A free, commercially representative embedded benchmark suite, *IEEE 4th Annual Workshop on Workload Characterization*, pp.3–14 (Dec. 2001).

[12] IBM ILOG CPLEX optimization studio v12.5 documentation (2012).

[13] Muralimanohar, N., Balasubramonian, R. and Jouppi, N.P.: Cacti 6.0: A tool to understand large caches, Technical Report, University of Utah and Hewlett Packard Laboratories (2009).

[14] Degalahal, V., Li, L., Narayanan, V., Kandemir, M. and Irwin, M.J.: Soft errors issues in low-power caches, *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems*, Vol.13, No.10, pp.1157–1166 (2005).



Takuya Hatayama received his B.E. degree in informatics from Kyoto University, Kyoto, Japan in 2014. His research interests include low power design and system design methodology for embedded systems.

Hideki Takase is an assistant professor at the Graduate School of Informatics, Kyoto University. He received his Ph.D. degree in Information Science from Nagoya University in 2012. From 2009 to 2012, he was a research fellow of the Japan Society for the Promotion of Science (DC1). His research interests include compilers, real-time operating systems, and low energy design for embedded systems. He received the Incentive Award from Computer Science group of IPSJ in 2008. He is a member of IEICE.

Kazuyoshi Takagi received his B.E., M.E. and Dr. of Engineering degrees in information science from Kyoto University, Kyoto, Japan, in 1991, 1993 and 1999, respectively. From 1995 to 1999, he was a research associate at Nara Institute of Science and Technology. He had been an assistant professor since 1999 and promoted to an associate professor in 2006, at the Department of Information Engineering, Nagoya University, Nagoya, Japan. He moved to Department of Communications and Computer Engineering, Kyoto University in 2011. His current interests include system LSI design and design algorithms.

Naofumi Takagi received his B.E., M.E., and Ph.D. degrees in information science from Kyoto University, Kyoto, Japan, in 1981, 1983, and 1988, respectively. He joined Kyoto University as an instructor in 1984 and was promoted to an associate professor in 1991. He moved to Nagoya University, Nagoya, Japan, in 1994, and promoted to a professor in 1998. He returned to Kyoto University in 2010. His current interests include computer arithmetic, hardware algorithms, and logic design. He received Japan IBM Science Award and Sakai Memorial Award of the Information Processing Society of Japan in 1995, and The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology of Japan in 2005.

(Recommended by Associate Editor: *Mineo Kaneko*)