PAPER Special Section on Parallel and Distributed Computing and Networking

Performance Evaluation of a 3D-Stencil Library for Distributed Memory Array Accelerators

Yoshikazu INAGAKI[†,††a)], ***Nonmember***, **Shinya TAKAMAEDA-YAMAZAKI**[††b)], **Jun YAO**[††c)], ***and*** **Yasuhiko NAKASHIMA**[††d)], ***Members***

SUMMARY The Energy-aware Multi-mode Accelerator eXtension [24], [25] (EMAX) is equipped with distributed single-port local memories and ring-formed interconnections. The accelerator is designed to achieve extremely high throughput for scientific computations, big data, and image processing as well as low-power consumption. However, before mapping algorithms on the accelerator, application developers require sufficient knowledge of the hardware organization and specially designed instructions. They also need significant effort to tune the code for improving execution efficiency when no well-designed compiler or library is available. To address this problem, we focus on library support for stencil (nearest-neighbor) computations that represent a class of algorithms commonly used in many partial differential equation (PDE) solvers. In this research, we address the following topics: (1) system configuration, features, and mnemonics of EMAX; (2) instruction mapping techniques that reduce the amount of data to be read from the main memory; (3) performance evaluation of the library for PDE solvers. With the features of a library that can reuse the local data across the outer loop iterations and map many instructions by unrolling the outer loops, the amount of data to be read from the main memory is significantly reduced to a minimum of 1/7 compared with a hand-tuned code. In addition, the stencil library reduced the execution time 23% more than a general-purpose processor.

key words: *CGRA, coarse grained reconfigurable architecture, accelerator, library, stencil, optimization*

1. Introduction

1.1 Performance Enhancement of Computations

To speed up the scientific and technological computations required in such fields as image processing and 3-dimensional simulation, many studies and developments have been reported from the views of hardware and software. From the view of the former especially in the HPC field, computer systems have employed high-performance processors equipped with SIMD units, such as SPARC64 with HPC-ACE [7], Intel with SSE/AVX extensions [3], and ARM with Neon extensions [4], or such general-purpose accelerators as GPGPU [6], [9] and Xeon Phi [8]. To improve the execution efficiency of these computer systems, new algorithms, efficient compilers, and libraries have been proposed to make the best use of each CPU architecture [10], [15], [16].

1.2 Partial Differential Equations (PDE)

In general, SPEC [1] and NPB [2] are widely used to represent the performance of computer systems. However, in the HPC field, the importance is growing of measuring the performance obtained by several standard partial differential equations (PDE) [11]. The PDE solver's kernel code, which is obtained by some finite differential method, is called stencil computation because of the memory access pattern that forms predefined stencils across each dimension of the data array. Since stencil computation fundamentally has both spatial and temporal localities, the performance depends significantly on the quality of the instruction scheduling where the size of the data does not fit in the cache memories. However, it is difficult to schedule complicated patterns in the memory access so that traditional cache-based multi-core systems can avoid the contentions of cache lines. To address this problem, optimization schemes have been proposed for parallel resource allocation, such as data reuse [12] and a domain-specific language compiler for stencil computing [13], [14], [16].

1.3 CGRA for Stencil Computations

In contrast to previous studies, we proposed for stencil computations Coarse Grained Reconfigurable Architectures (CGRAs) [20]–[26], which have many processing elements, local memories, and inter-connection networks so that many operations can be executed simultaneously on several data streams. We also focused on how to design general-purpose stencil libraries on plural specific CGRAs. The optimization scheme of stencil libraries for CGRA depends closely on the stencil pattern and the sequence of calculations, the frequency and width of the memory system, and the frequency of PEs.

1.4 3D-Stencil Library for EMAX

In this paper, we propose a 3D-Stencil Library that can receive input parameters, such as the number of CGRA columns and the degree of stencil calculation, and generate

Manuscript received January 8, 2015.
Manuscript revised May 11, 2015.
Manuscript publicized September 15, 2015.

[†]The author is with Fujitsu Computer Technologies Limited, Kawasaki-shi, 211–8588 Japan.
[††]The authors are with Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
a) E-mail: y.inagaki@jp.fujitsu.com
b) E-mail: shinya@is.nasit.jp
c) E-mail: yaojun@is.naist.jp
d) E-mail: nakashim@is.naist.jp

DOI: 10.1587/transinf.2015PAP0015

Copyright © 2015 The Institute of Electronics, Information and Communication Engineers

optimized code. For quantitative evaluation, we assume a specific CGRA that we call the Energy-aware Multi-mode Accelerator eXtension (EMAX), which is equipped with distributed single-port local memories and a 2-dimensional interconnection network. To understand easily such a stencil library, it is necessary to grasp the inter-PE structure of CGRA, including columns (horizontal connections), rows (vertical connections), and memory hierarchy. The EMAX structure will be described below. The structure of this paper is as follows. In Sect. 2, an overview of stencil computing is given. In Sect. 3, some key features of EMAX are shown. In Sect. 4, the techniques in a 3D-Stencil Library for optimizing and generating codes for EMAX are described. In Sect. 5, the performance as measured by certain benchmarks is presented, and we conclude in Sect. 6.

2. Stencil Computation

This section describes the structure of stencil computing that is supported by the 3D-Stencil Library.

2.1 3D-Stencil Kernels

Figure 1 shows two examples of 3D-stencil kernels represented by C language. Each of the kernels has neighbors that spread from the center in three directions along the X, Y, and Z axes. The difference between these two examples is the number of elements in each of the three directions. In this paper, the number of elements (the distance) from the center is called the "degree". A 7-point stencil kernel has *degree* = 1, and a 13-point stencil kernel has *degree* = 2 (Fig. 1). A 3D-stencil kernel with "*degree* = 1" expresses a 3D Jacobi solver from the Rodinia benchmark suite [17], which is used for the evaluation of heterogeneous computing [18], [19].

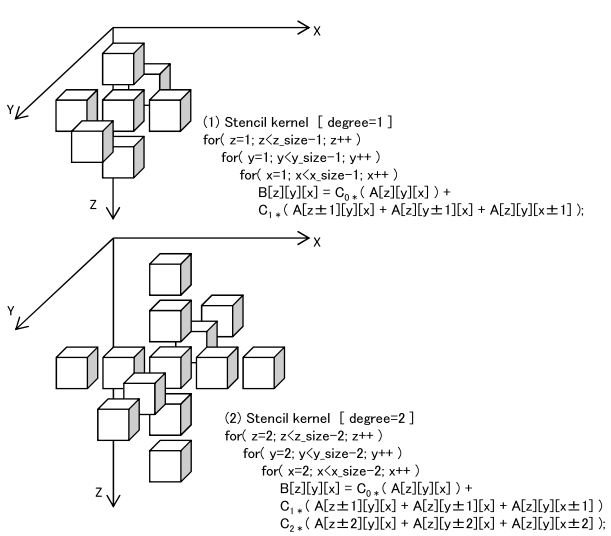

Fig. 1 3D-Stencil kernels.

3. Overview of EMAX

In this section, such general features of CGRA as columns, rows, mnemonics, and memory hierarchy are described as is our proposed EMAX configuration [24], [25].

3.1 System Diagram of EMAX

As shown in Fig. 2, EMAX consists of two or more basic processing elements (PEs) arranged in the shape of a matrix. Each PE has several arithmetic logical units (e.g., EX1, EX2), distributed single-port local memories (LMM), an effective address generator (EAG), and several FIFOs that can hold a certain amount of recent data read from LMM. In EMAX, the LMM and FIFOs in the same row are connected with a common data path. In the case of the configuration shown in Fig. 2, the data in an LMM can be sent to a maximum of 8 FIFOs in the same row. Additionally, the PEs at the top are connected with the PEs at the bottom in a ring fashion. The number of rows is defined by the number of stages required by specific application programs.

3.2 Interface with Host Computer

In general, accelerators have a local DDR3 memory and are connected to host computers through an external I/O bus (Fig. 2). Consequently, the overhead for sending instructions and data to the accelerators should be taken into consideration for modeling performance.

3.3 EMAX Mnemonics

For describing the 3D-Stencil Library, it is convenient to prepare some typical mnemonics for controlling the EMAX. The framework of the mnemonics is shown in Fig. 3. The preceding combination of *row* and *col* specifies the logical location of PE to be assigned with the succeeding function. To explain the EMAX operation, we describe three load instructions in the same Z coordinates (Fig. 4 (1) at $Y = 0, 1, 2$ and Fig. 4 (2) at $Y = 1, 2, 3$). Each load instruction can incre-

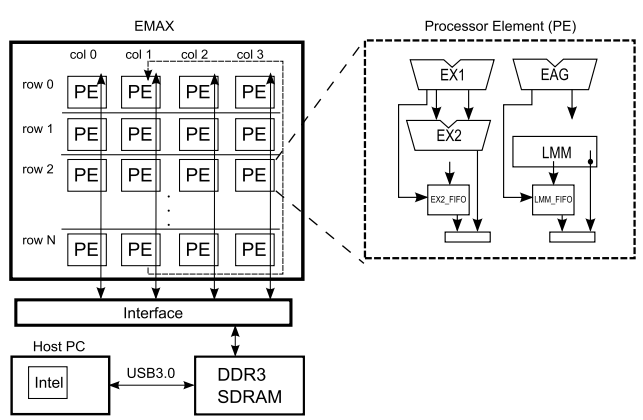

Fig. 2 EMAX configuration.

Case 1:@row#, col#, dist [count] ALU_OP RGI & MEM_OP RGI LMM_CONTROL
Case 2:@row#, col#, dist [count] ALU_OP RGI
Case 3:@row#, col#, dist [count] & MEM_OP RGI LMM_CONTROL

(a) Instruction Format

32 bit operations 16 bit[2] operations misc operations load from FIFO floating-point operations	add/add3/sub/sub3 mauh/mauh3/msuh3 mluh/mmrg3/msad/minl/minl3/mh2bw/mcas/ mmid3/mmax/mmax3/mmin/mmin3 ldb/ldub/ldh/lduh/ld fmul/fma3/fadd

(b) EX1 operations

32 bit operations 16 bit[2] operations	and/or/xor mauh/mauh3/msuh3

(c) EX2 operations

load from LMM or LMM_FIFO store to LMM	ldb/ldub/ldh/lduh/ld stb/sth/st/cst

(d) Memory operations

Fig. 3 EMAX mnemonics.

ment the load address in the X axis *count* times. Therefore, one load instruction can read the data of the *count* points from the stream data along the X axis. In the 1st processing on EMAX at $Y = 0, 1, 2$ (Fig. 4 (1)), three load instructions read the *count* points data from the stream data along the X axis (three stream data are $Y = 0$, $Y = 1$, and $Y = 2$). Then these three load instructions are mapped on PE at *rows 0, 2, 4* (Fig. 4 (3)). After the 1st processing is completed, the next processing starts to read the *count* points data from the three stream data at $Y = 1, 2, 3$ (Fig. 4 (2)). Then the two data streams at $Y = 1$ and $Y = 2$ on the X axis are used on PE at *rows 2 and 4* in the 1st processing. However, because these data exist in the LMM of another PE when the instruction is similarly arranged, it cannot be read. To reuse these data streams, the *Dist* field exists in the EMAX mnemonics. *Dist* specifies the distance of the vertical location between LMMs that hold neighbor streams for each other for the stencil computation. After EMAX has finished processing on a stencil stream at $Y = 0, 1, 2$ along the X axis (Fig. 4 (1)), some LMMs can supply data for the next processing on the next stencil stream at $Y = 1, 2, 3$, because the neighbor stencil streams overlap each other at $Y = 1$ and $Y = 2$ (Fig. 4 (2)). If the LMM that holds the stream corresponding to $Y = 1$ is located on *row 2*, and $Y = 2$ is located on *row 4*, then *dist* should be 2. For the next processing with $Y = 1, 2, 3$, the previous contents of LMMs are kept and the mapping of instructions is shifted by *dist*, as shown in Fig. 4 (4). Consequently, the next processing can reuse the streams at $Y = 1$ and $Y = 2$ (black LMMs in Fig. 4 (4)). The operations for EX1 and EX2 are specified by *ALU_OP*, as shown in Figs. 3 (b) and (c). The load operation from EX2_FIFO can be specified as EX1 operations. The *MEM_OP* (Fig. 3 (d)) is the load/store operations from LMM or LMM_FIFO. The initial value of the register for *ALU_OP* and *MEM_OP* can be specified by *RGI*.

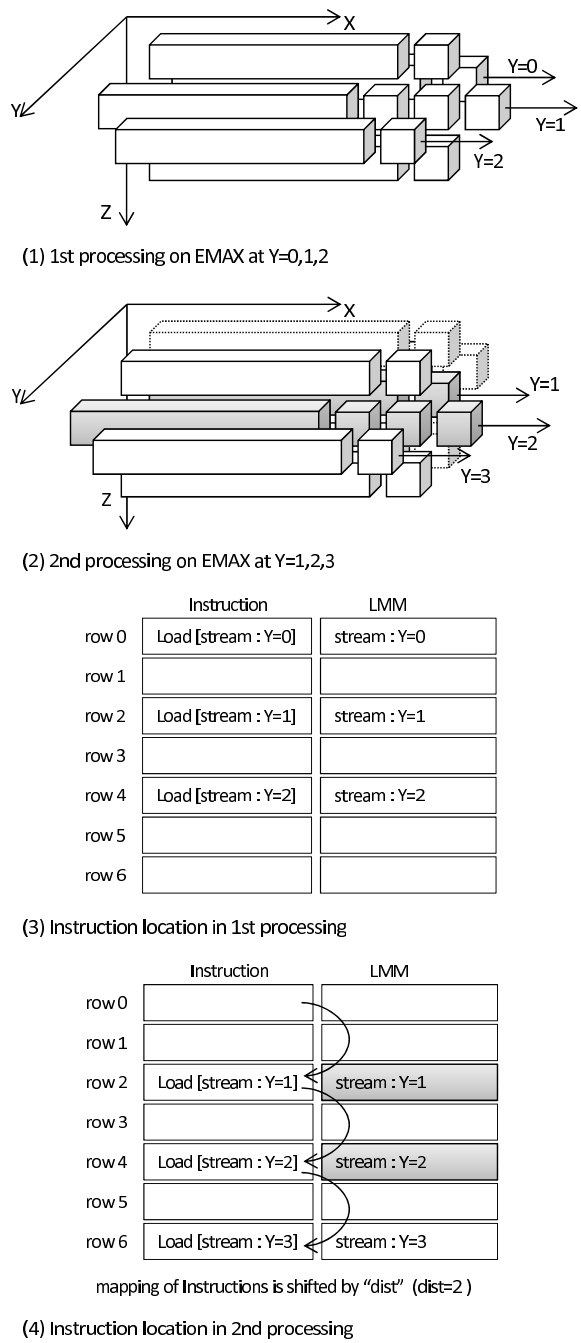

Fig. 4 Instruction shift by "dist".

4. 3D-Stencil Library for EMAX

This section describes the user interface of the 3D-Stencil Library and a technique for generating instructions for EMAX. For the best use of EMAX, we must reduce the amount of data transmission between LMM and the main memory (DDR3). The optimization scheme must focus on how to map local memory to the stencil streams.

4.1 User Interface of 3D-Stencil Library

As shown in Fig. 1, typical templates for stencil kernels are easily written in C language. The parameters for customizing the templates and implementation on EMAX are the "degree" of the stencil and the "number of columns" of EMAX. Figure 5 (1) is an example of a customizable stencil library. When the degree is one, the Jacobi 3D stencil computation is expressed, as described in Sect. 2. When the degree is three, a FD6 kernel is expressed. Application developers can embed various stencil computations on various EMAXs by customizing 3D-Stencil Libraries with specific parameters (Fig. 5 (2)).

4.2 Structure of 3D-Stencil Library

The structure of the 3D-Stencil Library is shown in Fig. 6. When an application program calls the stencil library, the library first generates EMAX instructions based on the given parameters. Then the instructions and the data are sent to EMAX through DDR3. Finally, EMAX executes the instructions mapped on PEs simultaneously and stores the result in DDR3. The following is the processing flow of using the 3D-Stencil Library:

(1) The stencil application allocates three memory arrays in the main memory of the host PC. *A* is the input 3D-array, *B* is the output 3D-array, and *C* is the symmetric-constant coefficients of the stencil [5].
(2) The stencil application calls the 3D-Stencil Library. Arguments `A`, `B`, and `C` are the address pointers allocated in the main memory by the stencil application (1), and `size_X`, `size_Y`, and `size_Z` are the dimension sizes of the 3D-array.
(3) The 3D-Stencil Library automatically generates instructions (mapping data) of the EMAX including input 3D-array data from the input parameter. If the data size of the X-direction exceeds the LMM capacity, the 3D-Stencil Library divides the 3D-Stencil space and executes EMAX multiple times. Figure 7 shows the divided image.
(4) The 3D-Stencil Library transmits the instruction data including the data for the EMAX activation to EMAX's DDR3 with DMA.
(5) When the activation data are written on DDR3, EMAX is activated automatically and prefetches the input 3D-array data from the DDR3 to the LMMs. Then each PE of EMAX executes the instructions and sends the result to DDR3.
(6) The result of the stencil computation is transmitted by DDR3 to the pointer of array *B* in the main memory of the host PC.

(1) Interface of 3D-Stencil Library

```
stencil_3d( double ***A, double ***B, double *C, int size_x, int size_y, int size_z,
            int degree, int e_stage );
```

(2) Parameters

double	***A	: input 3D array
double	***B	: output 3D array
double	*C	: symmetrical constant coefficient
int	size_X	: dimension size of X-direction
int	size_Y	: dimension size of Y-direction
int	size_Z	: dimension size of Z-direction
int	degree	: number of elements from center point
int	e_stage	: number of EMAX columns

Fig. 5 Interface of stencil kernel.

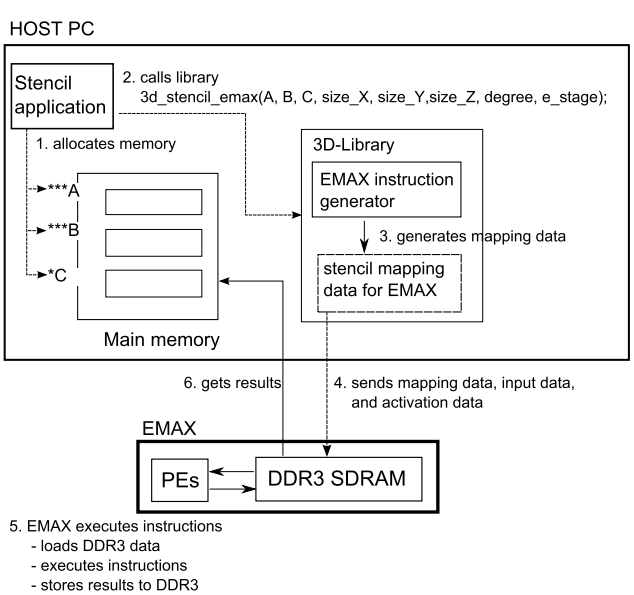

Fig. 6 Structure of 3D-Stencil Library.

4.3 Basic Instruction Mapping

As mentioned above, the amount of data, which is transmitted between LMMs and the main memory (DDR3), must be reduced to increase EMAX's execution efficiency. The stencil computation has spatial (horizontal) locality in the X-direction and temporal (vertical) locality in the Y-direction. Horizontal locality can be maximally utilized by employing FIFOs in the same row of the EMAX. The FIFOs are

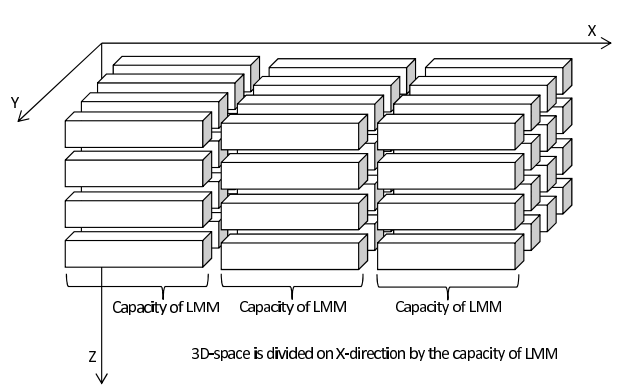

Fig. 7 Divided 3D-Stencil space.

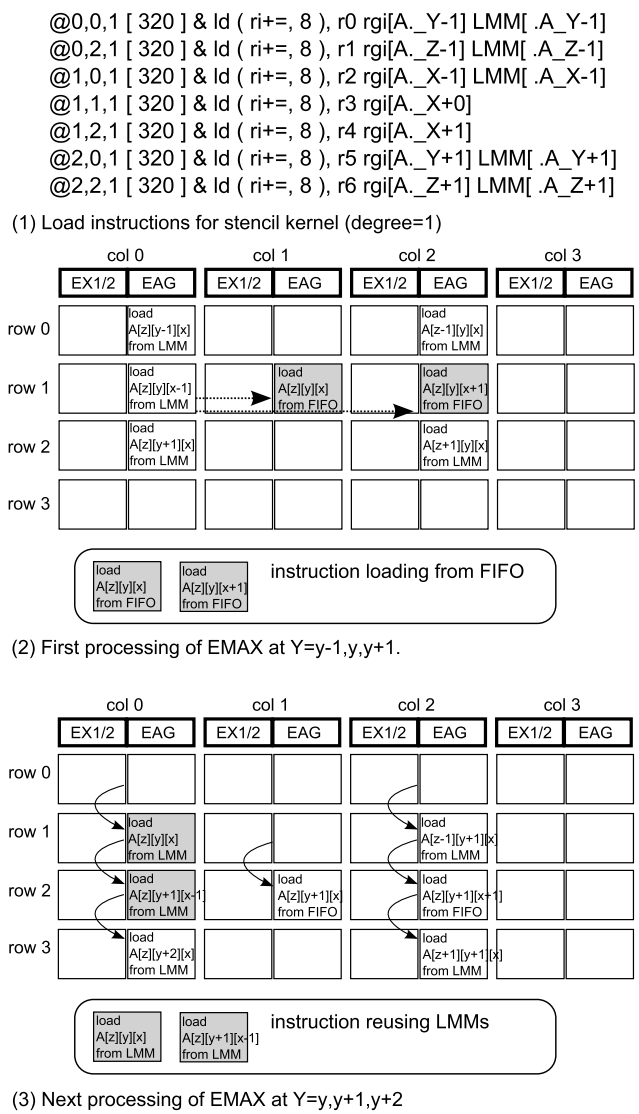

Fig. 8 Basic instruction mapping.

filled with the data from the LMM in the same row and can hold several neighbors in the X-direction. In contrast, vertical locality can be maximally utilized by reusing LMMs that hold different streams in the Y-directions. Figure 8 describes the basic memory mapping of stencil computation when the "degree" equals one. Figure 8 (1) shows the load instructions for a stencil kernel expressed by EMAX mnemonics. First, seven load instructions are mapped on corresponding EAGs (Fig. 8 (2)). Then for utilizing FIFOs to load the neighbor data in the X-direction, the load instructions from $A[z][y][x-1]$, $A[z][y][x]$, and $A[z][y][x+1]$ must be mapped in the same row, and the load instructions from $A[z][y-1][x]$, $A[z][y][x-1]$, and $A[z][y+1][x]$ must be mapped in the same column to reuse the data in the LMMs. By setting each *dist* field of the instructions to 1, subsequent processing can reuse the LMMs of $A[z][y][[x]$ and $A[z][y+1][x-1]$ (Fig. 8 (3)).

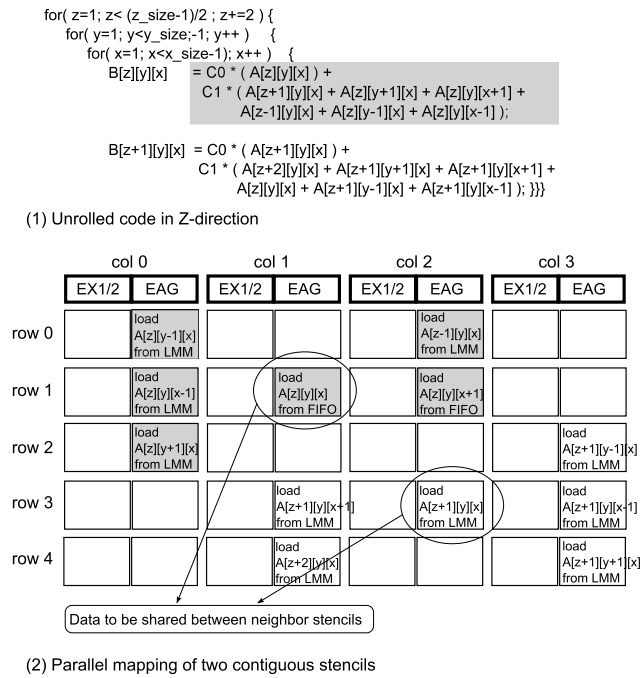

Fig. 9 Parallel mapping.

4.4 Packing of Instructions

After basic mapping is completed, the packing of instructions should be considered by unrolling in the Z-direction. As mentioned above, the X-direction is mapped on the sequential access with FIFOs, and the Y-direction is mapped on the reusing of LMMs. Moreover, we have another chance to parallelize the Z-direction by unrolling so that two contiguous stencil computations can be performed simultaneously. An example of the program and an arrangement for unrolling is shown in Fig. 9. The load instructions from $A[z][y][x-1]$, $A[z][y][x]$, $A[z][y][x+1]$, $A[z][y-1][x]$, and $A[z][y+1][x]$ are defined as a fixed pattern in the 3D-Stencil Library due to utilizing FIFOs and reusing LLMs. The neighbor stencil on the Z-direction is mapped on empty PEs in a mirrored fashion (Fig. 9 (2)). By having unrolled codes share two load instruction ($A[z][y][x]$, $A[z+1][y][x]$), two load instructions can be eliminated. Therefore, we expect that the packing of instructions reduces the total amount of data to be transmitted between LMMs and the main memory (DDR3).

4.5 Evaluation of Packing

If we assume that the target EMAX has an infinite number of rows, the number of parallel mappings in the Z-direction can theoretically be increased infinitely. To compute the optimal degree of parallelizing in the Z-direction based on the number of EMAX rows, the amount of data to be transmitted in each case is evaluated. The flow of prefetching data to LMM in the case of stencil computation is shown in Fig. 10, where each block corresponds to a stream in the X-direction

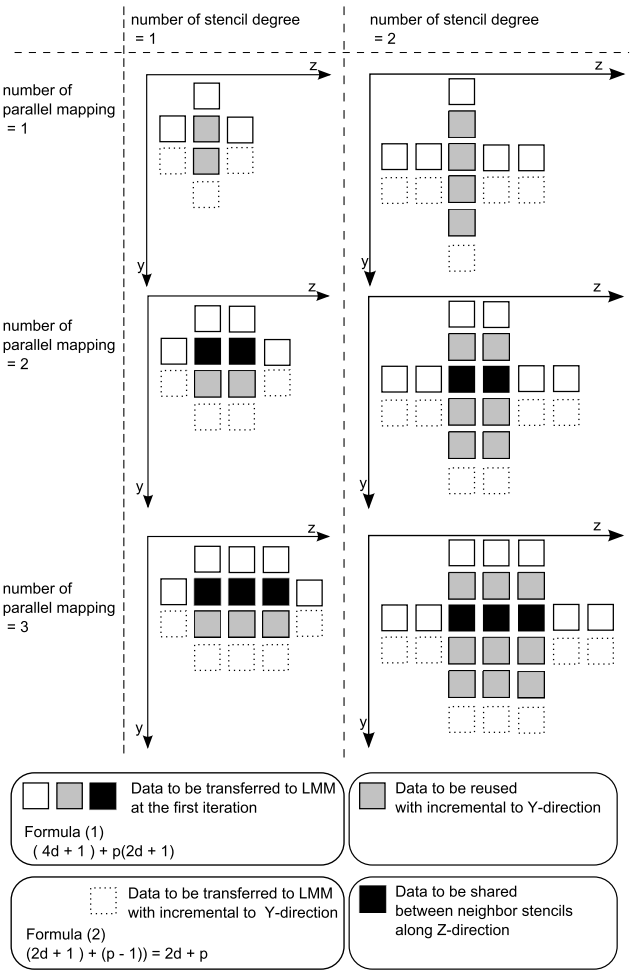

Fig. 10 Data transmission to LMM on stencil computing.

and is stored in each LMM. The two dimensions of Y and Z in Fig. 10 correspond to the Y- and X-directions. The amount of data to be transmitted to LMM can be estimated by the following formulas: p: number of parallel mappings, d: number of stencil degrees, y: size of Y-direction, and z: size of Z-direction.

(1) First required bit of data: $(4d + 1) + p(2d + 1)$
(2) Subsequent required data: $(2d + 1) + (p - 1) = 2d + p$
(3) Total data in xy-iteration: $y(2d + p)$
(4) Total data in xyz-iteration: $z((1) + (3))/p$
(5) Total data (approximate value): $yz(2d + p)/p$

Formula (1) is the amount of data to be transmitted to LMM at the first iteration of the Y-direction, and formula (2) is the amount of data required by the subsequent incremental stencil in the Y-direction. The total amount of data with the incremental to the Y-direction is expressed by formula (3) (i.e., $(2) * y$). Since the number of iterations in the Z-direction (z) is divided by the number of parallel mappings (p), formula (4) becomes the total amount of data to be transmitted to LMM. Since the stencil calculation has many dimensions ($p < y$), the value of (1) can be ignored, and the amount of required data can be approximated as the value of

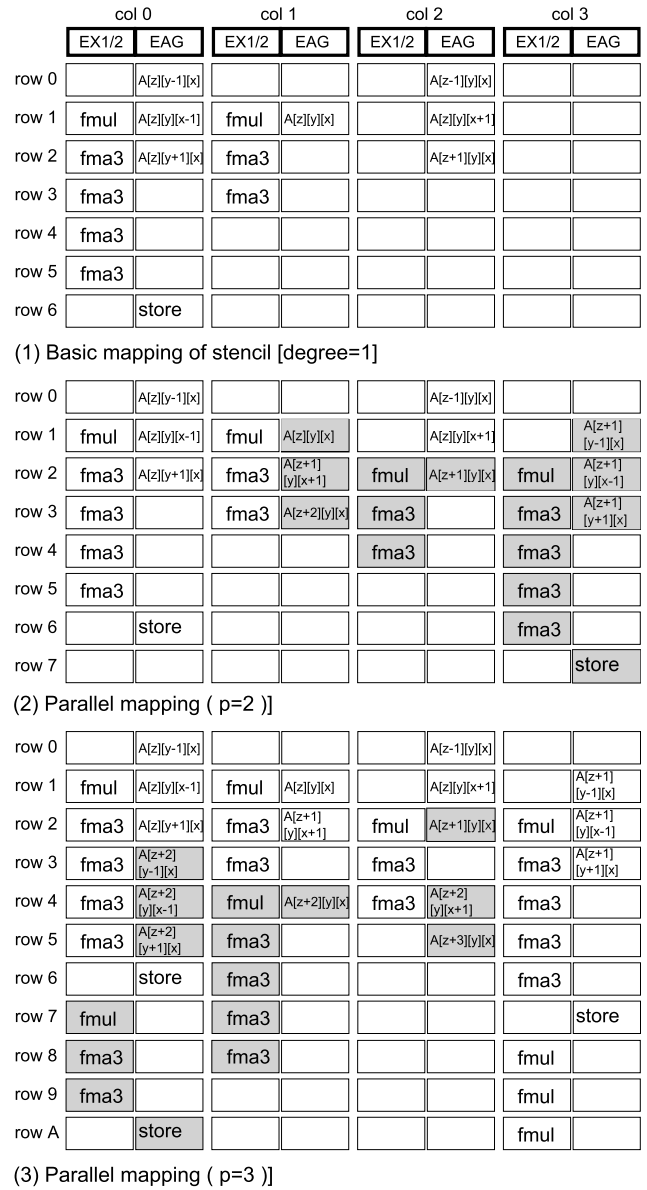

Fig. 11 Instruction mapping of stencil [$degree = 1$].

formula (5). Therefore, the more p increases, the more the required data are reduced. In the case of $degree = 1$, the ratio of without parallelization ($p = 1$) to with parallelization becomes $3 : (2 + p)/p$. When p is assumed to be infinite, the ratio becomes $3 : 1$. Therefore, by parallelization, the amount of data to be transmitted is reduced to a maximum of 1/3 of that without parallelization, and for $degree = 2$ and $degree = 3$, it is reduced to a maximum of 1/5 and 1/7, respectively.

4.6 Parallel Mapping in 3D-Stencil Library

It has become obvious that parallelization can reduce the amount of data transmission based on the above evaluation. Therefore, the 3D-Stencil Library must automatically generate instructions to increase the number of parallel mappings

at least to the maximum number of EMAX rows. First, the 3D-Stencil Library maps the instructions according to the basic pattern. When $degree = 1$, the basic pattern is copied in seven rows (Fig. 11 (1)). If the number of EMAX columns is seven or less, the 3D-Stencil Library maps the instructions in the form of Fig. 11 (1) (i.e., $p = 1$). If the number of EMAX columns is eight or more, for computing the neighbor stencil in the Z-direction, the basic pattern occupies the flipped horizontal location next to the original pattern (Fig. 11 (2) (i.e., $p = 2$)). When the number of sets for parallel mapping is 3 or 4, the basic pattern is copied and put in the lower location. Then the added number of the EMAX rows becomes four because the empty units are used (Fig. 11 (3)). For example, when the number of EMAX columns is 11 or 12, the number of parallel mappings becomes three or four, respectively. On the other hand, if $degree = 3$, since the PEs in the same row are occupied by the basic pattern using eleven rows, all of the basic patterns should be located in subsequent rows. The number of rows, generated by the 3D-Stencil Library according to the degree of the stencils and the number of parallel mappings, can be defined as the following formulas (p: number of parallel mappings):

- $degree = 1$: $8 + 4((p/2) + (p \bmod 2) - 1) - (p \bmod 2)$
- $degree = 3$: $11p$

5. Results and Analysis

In this section, the execution time of the 3D-Stencil Library is estimated by a clock-accurate EMAX simulator. The results are compared with general-purpose processors.

5.1 Simulation Model for Performance Measurement

For an accurate estimation of the execution time in EMAX, we developed a clock-accurate simulator that represents the activities of the memories and the registers in EMAX. The assumptions of the frequency in each component, the memory bandwidth, and the host bandwidth are shown in Table 1. After performing circuit composition by CAD with 28-nm technology using design data that became LSI by Rohm 0.18 (it operates at 52.6 MHz), we learned that EMAX operates at an internal clock speed of 252 MHz. Therefore, in the simulator condition, the EMAX frequency is assumed to be 200 MHz. When the 3D-Stencil Library executes 3D stencil computation with a dimension size of $320 \times 320 \times 320$, the 3D-Stencil Library activates EMAX 320×320 times (when not using parallel mapping). For $degree = 1$ in the first execution, five data streams along the X axis are transmitted to DDR3 to get the results of 320-points data with the same X axis. This execution pattern resembles the processing in Fig. 4 (1), and the unit of the execution corresponds to the *stream* in Fig. 12. To obtain all the stencil computing results, the 3D-Stencil Library activates EMAX in increments with the Y and Z axes, and the total activated time becomes 320×320. Figure 12 describes the timing chart of EMAX. Each execution is composed of five states.

- State 1
  Data transmission is executed between the main memory of the host PC and the DDR3 in EMAX.
- State 2
  Data prefetching is executed from DDR3 to LMM.
- State 3
  The instructions mapped on the PEs are executed simultaneously on the data stream.
- State 4
  The result stored in LMM is transmitted to DDR3.
- State 5
  Data transmission is executed from DDR3 to the main memory of the host PC.
- [Other conditions]
  - Each state can be started after the previous state when identical execution is completed.
  - States 1 and 5 and States 2 and 3 use the same data path. Therefore each state exclusively uses the data path.
  - Each state can be overlapped by another state.
  - Since the data size of the control information including the operation of the EMAX activation is much smaller than the 3D-array, it is ignored.

Table 1 Simulation parameters.

EMAX frequency	200 MHz
HOST-DDR3 bandwidth (USB 3.0)	400 Mbyte/sec
DDR3-LocalMEM bandwidth	800 Mbyte/sec
DDR3 SDRAM capacity	256 Mbyte
Local MEM capacity	8 Kbyte

5.2 Evaluation of 3D-Stencil Library

We measured the execution time of the 3D-Stencil Library using the simulator. Figure 13 presents the execution time for running stencil kernels with $degree = 1$. The results with $degree = 3$ are shown in Fig. 14. The size of the X, Y, and Z-directions in the 3D space is fixed to 320, and each point has a double precision floating point value. The "stage count" shows the number of EMAX rows. In Fig. 13, "stage count = 7" corresponds to a hand-tuned code (the

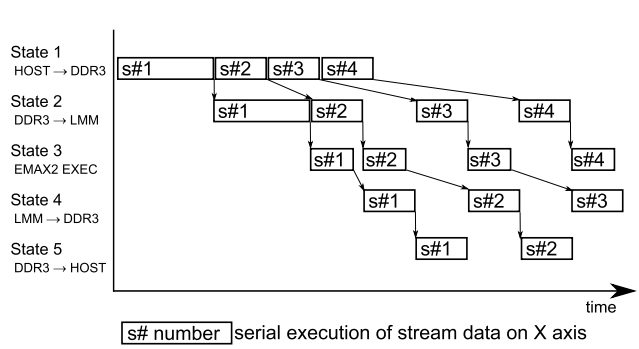

Fig. 12 Execution sequence of EMAX.

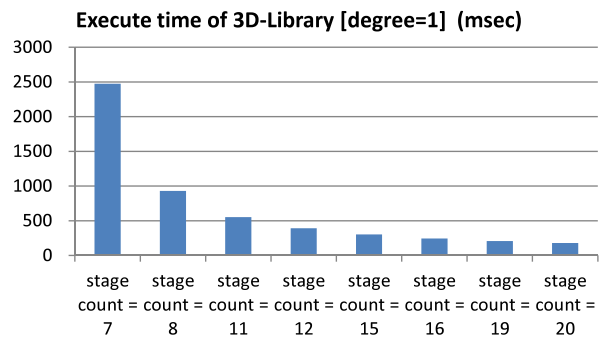

Fig. 13 Execution time of 3D-Library [degree=1].

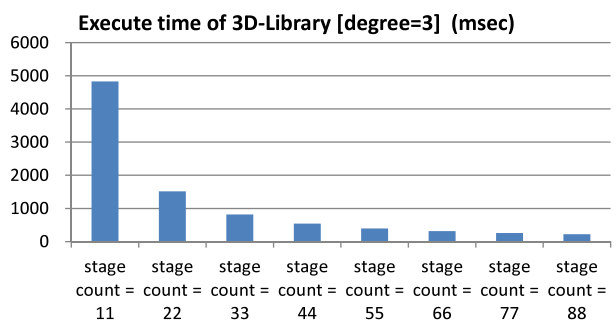

Fig. 14 Execution time of 3D-Library [degree=3].

Table 2 General-purpose processors for comparison.

CPU	Compiler	Compile option
Intel (R) Xeon (R) CPU E5405 2.00 GHz	gcc 4.1.2	-O3 -msse2 -ffast -math
Intel (R) Core (TM) i5-4670 CPU@3.40 GHz (Haswell)	gcc 4.6.3	-O3 -msse2 -ffast -math

number of parallel mappings = 1) and the others correspond to the 3D-Stencil Library. In the same manner, in Fig. 14, "stage count = 11" corresponds to a hand-tuned code, and the others correspond to the 3D-Stencil Library. The results in each case show that, as the degree of parallelization increases, the 3D-Stencil Library can reduce the execution time by more than 90% compared with a hand-tuned code. Our result shows that the evaluation in Sect. 4.5 is correct, and the execution time can be reduced by more than the ratio of the amount of data transmission. However, this is an over-estimation; when the size of the data increases, many conflicts occur on the communication path between DDR3 and the main memory of the host PC. Such a model is not included in the current simulator yet. However, in general, commercial accelerators have many banks to increase the memory throughput. EMAX can also increase the performance in the same manner.

5.3 Comparison with General-Purpose Processors

Finally, the execution times of the stencil computation of EMAX are compared with general-purpose CPUs. The processors, the version of compilers, and the compiling options are shown in Table 2. We also use the same stencil kernels

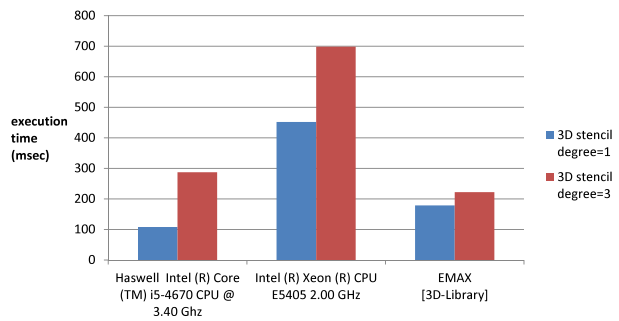

Fig. 15 Comparison of execution between CPUs and EMAX.

with the same size (the size of the X, Y, and Z-directions is 320), where $degree = 1$ and $degree = 3$. The results are shown in Fig. 15. In the case of $degree = 1$, Haswell is faster than the other processors including EMAX. However, when $degree = 3$, EMAX is faster. For reducing memory traffic, fitting many stencil points into FIFOs and LMMs works better than a normal shared cache system.

6. Conclusion

In this paper, we proposed a 3D-Stencil Library that automatically generates an instruction sequence that efficiently employs EMAXs. Using a 3D-Stencil Library, application developers can easily connect stencil computation on EMAX with C programs. Moreover, EMAX's execution time can be significantly reduced by eliminating the data transmission by parallelizing instruction mapping. In addition, with a performance simulator, the 3D-Stencil Library reduced the execution time 23% more than a general-purpose processor. The practicality of EMAX and the 3D-Stencil Library is in sight. We plan to enhance the library features to cope with many-point stencils, such as 27-point stencils.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers 24240005 and 24650020.

References

[1] Standard Performance Evaluation Corporation, http://www.spec.org/
[2] NAS Parallel Benchmarks, http://www.nas.nasa.gov/publications/npb.html
[3] Intel Instruction Set Architecture Extensions, https://software.intel.com/en-us/intel-isa-extensions
[4] NEON - ARM, http://www.arm.com/products/processors/technologies/neon.php
[5] 3D Finite Differences on Multi-core Processors, https://software.intel.com/en-us/articles/3d-finite-differences-on-multi-core-processors/
[6] High Performance Computing (HPC)-Supercomputing with NVIDIA Tesla GPU, http://www.nvidia.com/object/tesla_computing_solutions.html
[7] T. Maruyama, T. Yoshida, R. Kan, I. Yamazaki, S. Yamamura, N. Takahashi, H. Mikio, and H. Okano, "SPARC64 VIIIfx: A new-

generation octocore processor for petascale computing," IEEE Micro, vol.30, no.2, pp.30–40, 2010.
[8] X. Liao, L. Xiao, C. Yang, and Y. Lu, "MilkyWay-2 supercomputer: System and application," Frontiers of Computer Science, vol.8, no.3, pp.345–356, 2014.
[9] T. Endo and S. Matsuoka, "Massive supercomputing coping with heterogeneity of modern accelerators," 2008 IEEE International Symposium on Parallel and Distributed Processing, pp.1–10, 2008.
[10] N. Sedaghati, R. Thomas, L.-N. Pouchet, R. Teodorescu, and P. Sadayappan, "StVEC: A vector instruction extension for high performance stencil computation," 2011 International Conference on Parallel Architectures and Compilation Techniques, pp.276–287, 2011.
[11] J. Cabezas, M. Araya-Plo, I. Gelado, N. Navarro, E. Morancho, and J.M. Cela, "High-performance reverse time migration on GPU," 2009 International Conference of the Chilean Computer Science Society, pp.77–86, 2009.
[12] T. Grosser, A. Cohen, J. Holewinski, P. Sadayappan, and S. Verdoolaege, "Hybrid hexagonal/classical tiling for GPUs," Proc. Annual IEEE/ACM International Symposium on Code Generation and Optimization, pp.66–75, 2014.
[13] T. Henretty, R. Veras, F. Franchetti, L.-N. Pouchet, J. Ramanujam, and P. Sadayappan, "A stencil compiler for short-vector SIMD architectures," Proc. 27th International Conference on Supercomputing, pp.13–24, 2013.
[14] M. Christen, O. Schenk, and Y. Cui, "PATUS for convenient high-performance stencils: Evaluation in earthquake simulations," 2012 International Conference for High Performance Computing, Networking, Storage and Analysis, pp.1–10, 2012.
[15] P. Barrio, C. Carreras, R. Sierra, T. Kenter, and C. Plessl, "Turning control flow graphs into function calls: Code generation for heterogeneous architectures," 2012 International Conference on High Performance Computing and Simulation, pp.559–565, 2012.
[16] T. Endo and G. Jin, "Software technologies coping with memory hierarchy of GPGPU clusters for stencil computations," 2014 IEEE International Conference on Cluster Computing, pp.132–139, 2014.
[17] S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," 2009 IEEE International Symposium on Workload Characterization, pp.44–54, 2009.
[18] G. Misra, N. Kurkure, A. Das, M. Valmiki, S. Das, and A. Gupta, "Evaluation of Rodinia codes on Intel Xeon Phi," 2013 4th International Conference on Intelligent Systems, Modelling and Simulation, pp.415–419, 2013.
[19] P. Di, D. Ye, Y. Su, U. Sui, and J. Xue, "Automatic parallelization of tiled loop nests with enhanced fine-grained parallelism on GPUs," 2012 41st International Conference on Parallel Processing, pp.350–359, 2012.
[20] J. Yao, M. Saito, S. Okada, K. Kobayashi, and Y. Nakashima, "EReLA: A low-power reliable coarse-grained reconfigurable architecture processor and its irradiation tests," IEEE Trans. Nucl. Sci., vol.61, no.6, pp.3250–3257, 2014.
[21] J. Yao, Y. Nakashima, M. Saito, Y. Hazama, and R. Yamanaka, "A flexibly fault-tolerant FU array processor and its self-tuning scheme to locate permanently defective unit," IEEE Symposium on Low-Power and High-Speed Chips XVII, pp.1–3, 2014.
[22] S. Kurebayashi, J. Yao, and Y. Nakashima, "A pipelined Newton-Raphson method for floating point division and square root on distributed memory CGRAs," IEEE Symposium on Low-Power and High-Speed Chips (poster), 2014.
[23] R. Shimizu, S. Takamaeda-Yamazaki, J. Yao, and Y. Nakashima, "High performance graph processing with a memory intensive array accelerator," IEICE Technical Report, CPSY2014-11, 2014.
[24] R. Shimizu, M.M. Tanomoto, S. Takamaeda-Yamazaki, J. Yao, and Y. Nakashima, "Implementation and evaluation of an accelerator based on manymemory network," IEICE Technical Report, CPSY2014-81, 2014.
[25] M. Tanomoto, S. Takamaeda-Yamazaki, J. Yao, and Y. Nakashima, "Convolutional neural network processing on an accelerator based on manymemory network," IEICE Technical Report, CPSY2014-82, 2014.
[26] S. Kurebayashi, S. Takamaeda-Yamazaki, J. Yao, and Y. Nakashima, "Parallelization of shortest path search on various platforms and its evaluation," IEICE Technical Report, CPSY2014-74, 2014.

Yoshikazu Inagaki received a B.E. degree in mechanical engineering from Saitama University in 2003 and an M.E. degree in computer engineering from the Tokyo Institute of Technology in 2005. He worked for FUJITSU Limited from 2005 to 2007. Since 2007, he has been an engineer at Fujitsu Computer Technologies Limited and since 2012, a Ph.D. candidate in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include processor architecture, high-performance computing, and verification of computer systems. He is a member of IPSJ.

Shinya Takamaeda-Yamazaki received B.E., M.E., and D.E. degrees from the Tokyo Institute of Technology in 2009, 2011, and 2014, respectively. From 2011 to 2014, he was a JSPS research fellow (DC1). Since 2014, he has been an assistant professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include memory systems, FPGA computing, high level synthesis, and processor architecture. He is a member of IEEE and IPSJ.

Jun Yao received B.E. and M.E. degrees from Tsinghua University in 2001 and 2004 and a Ph.D. degree from Kyoto University in 2009. Since 2013, he has been an associate professor in Graduate School of Information Science, Nara Institute of Science and Technology. His current researchers are in the computer architecture field, especially low-power consumption and high-reliable architectures. He is a member of IEEE, ACM, and IPSJ.

Yasuhiko Nakashima received B.E., M.E., and Ph.D. degrees in Computer Engineering from Kyoto University in 1986, 1988, and 1998, respectively. He was a computer architect in the Computer and System Architecture Department, FUJITSU Limited from 1988 to 1999. From 1999 to 2005, he was an associate professor in the Graduate School of Economics, Kyoto University. Since 2006, he has been a professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include processor architecture, emulation, CMOS circuit design, and evolutionary computation. He is a member of IEEE CS, ACM, and IPSJ.