社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－ＡＲＣ－168（18）
2006／6／9

画像処理プロセッサのための最適メモリアロケーション

張山 昌論† 小林 康浩†† 亀山 充隆†

† 東北大学大学院情報科学研究科 〒980-8579 仙台市青葉区荒巻字青葉 6-6-05
†† 小山工業高等専門学校 〒323-0806 栃木県小山市大字中久喜 771
E-mail: †{hariyama,kameyama}@ecei.tohoku.ac.jp, ††y-kobayashi@oyama-ct.ac.jp

あらまし 画像処理プロセッサの設計においては，簡単な総合結合網で並列アクセスを可能とするメモリシステムの設計が重要となる．本稿では，メモリ・演算器間の通信を局所化したロジックインメモリアーキテクチャを対象とした最適設計法を提案する．ロジックインメモリアーキテクチャでは，使用されるメモリモジュール数に比例して，ハードウェア量が増加する．本稿では，処理時間制約下でメモリモジュール数を最小化するメモリアロケーション手法を提案する．

キーワード ハイレベルシンセシス，メモリアロケーション，スケジューリング，画像処理

Optimal Memory Allocation for Image Processor

Masanori HARIYAMA†, Yasuhiro KOBAYASHI††, and Michitaka KAMEYAMA†

† Graduate School of Information Sciences, Tohoku University Aoba 6-6-05, Aramaki, Aoba, Sendai, 980-8579 Japan
†† Nakakuki 771, Oyama, 323-0806 Japan
E-mail: †{hariyama,kameyama}@ecei.tohoku.ac.jp, ††y-kobayashi@oyama-ct.ac.jp

Abstract One major issue in designing image processors is to design a memory system that supports parallel access with a simple interconnection network. This paper presents an efficient memory allocation to minimize the number of memory modules and processing elements with a parallel access capability based on regularity of window-type image processing.

Key words high-level synthesis, memory allocation, scheduling, image processing

1. Introduction

Highly-parallel image processors requires a complex interconnection network between memory modules and processing elements(PEs) for parallel memory access. Figure 1 shows an architecture model of parallel image processors. It consists of a lot of memory modules, PEs, interconnection network between memory modules and PEs, and an inter-PE network. The complexity of the interconnection network the memory modules and PEs increases with the number of memory modules. The number N_{BUS} of the buses is equal to the number of memory modules. The number of inputs of each multiplexor also tends to increase with the number of memory modules. The complex interconnection network causes significant overhead in delay and power in deep-submicron and more advanced technologies since the delay and the power of interconnection units are more dominant than those of logic units. The interconnection problem is also serious in image processors using field-programmable gate arrays that have large interconnection delays because of complex programmable switch blocks. Memory allocation has a great impact on the number of memory modules. Therefore, this paper presents memory allocation to minimize the number of memory modules with a parallel access capability.

This paper targets window-type image processing. The window-type image processing is widely used in practical applications. Its examples include filtering, template matching and morphology. An application usually require several types of windows. The memory allocation must support parallel access for all types of windows that is given as a specification.

There are a number of research works which have addressed memory allocation [1]- [8]. They are general-purpose

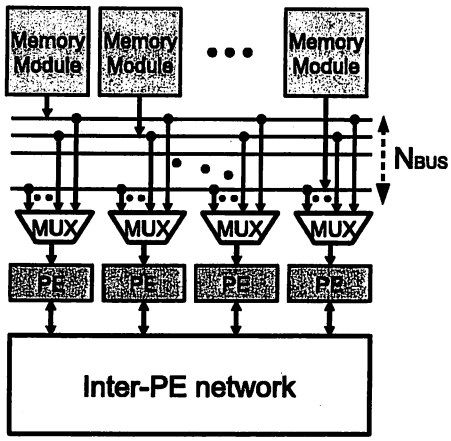

Fig. 1 Overall architecture of parallel image processors.

approaches, that is, not limited to image processing. However, they do not take into account regularity of image processing. From a practical point, a memory allocation result should have a simple address function that calculates which memory module a pixel is stored in. Hence, we consider a periodical memory allocation based on regularity of image processing, where a memory allocation for a whole image is given by repeating a memory allocation for a partial image. A periodical memory allocation gives a simple address function because of its regularity. An efficient search method is presented based on the regularity of a periodical memory allocation.

2. Problem formulation

2.1 Target algorithms

We consider window-type algorithms. In these algorithms, the output/intermediate output depends on a small neighborhood of an input image, where the neighborhood size is fixed and given as a window as shown in Fig. 2. Algorithms of this type frequently appear in practical situations: spatial filter, morphology, and image matching, and so on. Moreover, they usually have high degree of parallelism, and are suited for VLSI implementations.

We use window-serial-and-pixel-parallel (WSPP) scheduling as shown in Fig. 3. In this scheduling, operations are performed in parallel with pixels in a window, whereas operations are performed in a serial manner with windows. Figure 3(a) shows the location of the window at each step. The thick line denotes a window. Figure 3(b) shows the scheduled data-flow graph(SDFG) corresponding to Fig. 3(a). A node in the SDFG denotes a operation. The labels **A** and **B** on operations denote the operation types of the nodes. There is an edge from node V_1 to node V_2 when the output of V_1 is used as an input of V_2. At Step 1 in Fig. 3, pixels:$(0,0)$, $(0,1)$, $(0,2)$ and $(1,1)$ are used as inputs of operations of type **A**. The pixels must be accessed in parallel since the **A**-type operations are performed in parallel. Their results are used as inputs of a **B**-type operation. As the location of the window changes, the input pixels change.

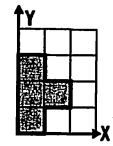

Fig. 2 Window.

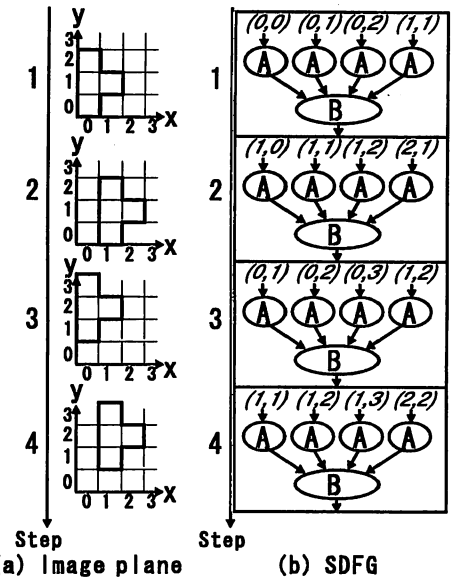

Fig. 3 Data-flow graph of a pixel-parallel-and-window-serial schedule.

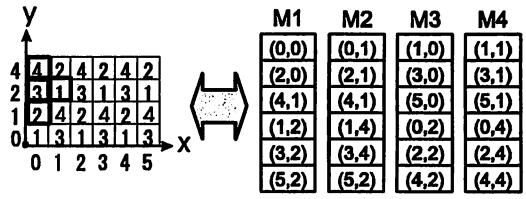

Fig. 4 Memory allocation.

2.2 Memory allocation for image processing

Memory allocation is a task that assigns pixels to memory modules. Figure 5 shows an example of memory allocation for the window in Fig. 2. The label on each pixel denotes the memory module which the pixel is assigned to For example, pixels: $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ are assigned to memory module: M1, M2, M3 and M4, respectively.

To meet the timing constraint of the WSPP scheduling, all the pixels in a window must be accessed in parallel for all

possible locations of the window. In other words, all the pixels in a window must be distributed among different memory modules for all possible locations of the window. Readers can examine that the memory allocation shown in Fig. 4 enables the parallel memory access.

2.3 Periodical memory allocation

From a practical point, a memory allocation should have a simple address function to map the coordinates of a pixel onto the memory module number. If the address function is complex, its overhead in area and delay become serious. For example, a look-up table is required for address mapping of all pixels in the worst case. Therefore, we consider a periodical memory allocation where a memory allocation for a whole image is given by repeating a memory allocation for a partial image. The periodical memory allocation has a simple address function because of its regularity. Let K be the number of memory modules. Let N_i be the number of pixels that are allocated to the memory module M_i for $1 \leqq i \leqq K$. Let (x_i^j, y_i^j) be the coordinates of a pixel that is allocated to the memory module M_i for $1 \leqq j \leqq N_i$. Then, we exactly define a periodical memory allocation as a memory allocation where the coordinates (x_i^j, y_i^j) of pixels allocated to M_i are expressed as

$$\begin{bmatrix} x_i^j \\ y_i^j \end{bmatrix} = s_i^j \begin{bmatrix} U_x \\ U_y \end{bmatrix} + t_i^j \begin{bmatrix} V_x \\ V_y \end{bmatrix} + \begin{bmatrix} x_i^0 \\ y_i^0 \end{bmatrix} \quad (1)$$

where $\mathbf{U} = [U_x\ U_y]^T$ and $\mathbf{V} = [V_x\ V_y]^T$ are vectors to represent periods(called period vectors); the variables s_i^j and t_i^j are integers; the coordinates (x_i^0, y_i^0) are those of the reference pixel allocated to M_i. Note that you can select an arbitrary pixel as the reference one from the ones allocated to the same memory module.

Fig. 5(a) shows an example of a periodical memory allocation for the SDFG shown in Fig. 3. The label on a pixel denotes the module number where the pixel is allocated. For example, the pixels: $(0,0)$,$(0,1)$,$(1,0)$ and $(1,1)$ are allocated to M_1, M_2, M_3 and M_4, respectively. From Fig. 5 (b), the coordinates of the pixels allocated to M_1 are given by

$$\begin{bmatrix} x_1^j \\ y_1^j \end{bmatrix} = s_1^j \begin{bmatrix} 1 \\ 2 \end{bmatrix} + t_1^j \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2)$$

where the coordinates of the reference pixel and the period vectors are $[0\ 0]^T$, $\mathbf{U} = [1\ 2]^T$ and $\mathbf{V}=[2\ 0]^T$, respectively. Each of the pixels with label 1 in Fig. 5 (b) are given by (s_1^j, t_1^j)=$(0,0), (1,0), (2,0), (0,1), (1,1)$ or $(2,1)$. Figures 5-(c), 4-(d) and 4-(e) show the pixels allocated to M_2, M_3 and M_4, respectively. These figures show that the same period vectors as M_1 are used for M_2, M_3 and M_4. They also show that the coordinates of the reference pixels for M_2, M_3 and M_4 are $[0\ 1]^T$, $[1\ 0]^T$ and $[1\ 1]^T$, respectively. The memory allocation shown in Fig. 5 satisfies Eq.((1)), that is, the definition of a periodical memory allocation.

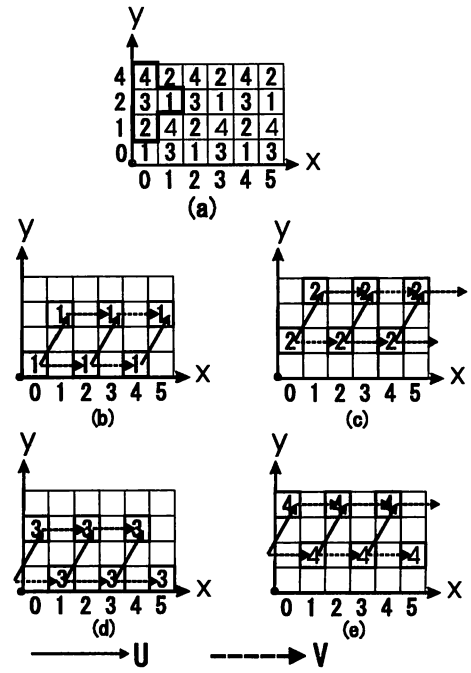

Fig. 5 Example of a periodical memory allocation.

As mentioned at the beginning of this section, the advantage of the periodical memory allocation is its simple address function. For the periodical memory allocation shown in Fig. 5(a), the address function that maps the coordinates (x, y) of a pixel to the module number is given by

$$f(x,y) = (2x + y) \bmod 4 \ + \ 1 \quad (3)$$

2.4 Optimal memory allocation

Let us minimize the number of memory modules when windows:W_1, W_2 and W_n and SDFGs are specified. The SDFGs impose the degree of parallelism in memory access. For example, in the SDFG shown in Fig. 3, the degree of parallelism in memory access is 4. Our optimization problem is formulated as finding a memory allocation that minimizes the number of memory modules under a specified degree of parallelism in memory access. The optimal memory allocation is defined as the memory allocation that satisfies the following conditions:

C1: For an arbitrary location of the windows:W_1, W_2 and W_n, pixels in the window can be retrieved in parallel. In other words, the pixels in the window are allocated to different memory modules. Each pixel is allocated to a single memory module. This condition ensures that the total memory capacity is minimized. The number of memory mod-

ule is minimized. This condition ensures that the hardware amount is minimized in the logic-in-memory architecture. The memory allocation is a periodical one. This condition ensures that the hardware for the address function is small.

For example, Fig. 5(a) is the optimal memory allocation for the window shown in Fig. 2 from the following reasons. The condition C1 is satisfied since pixels in the window are allocated to different pixels for an arbitrary location of the window. The condition C2 is satisfied since each pixel has a single label. The condition C3 is satisfied since the number of memory module is 4 and is equal to the minimum number of memory modules required for parallel access, i.e. the number of pixels in the window. The condition C4 is satisfied since the memory allocation is periodical as mentioned in section 2.3.

3. Search algorithm and design examples

3.1 Overview of search method

In the periodical memory allocation, the period vectors **U** and **V** determine which pixels are allocated to the same memory module as shown in Fig. 5. The period vectors make a parallelogram. Given the period vectors, the number of memory modules is estimated by the area of the parallelogram. The area S of the parallelogram is given by

$$S = |U_x \cdot V_y - U_y \cdot V_x|. \tag{4}$$

For example shown in Fig. 5,

$$S = |1 \cdot 0 - 2 \cdot 2| = 4,$$

and S is exactly same as the number of memory modules. This is because the memory allocation for a whole image is given by repeating the one for the parallelogram, and the parallelogram must be filled with pixels allocated to different memory modules. From these observations, finding the optimal memory allocation is reduced to finding period vectors that make the minimum parallelogram still satisfying the parallel access condition.

To deal with multiple windows: $W_1, W_2 \cdots$ and W_n, a super window is generated by a union of the windows. Figure 6(a),(b) and (c) show the given windows. Figure 6(d) shows the super window generated by the windows. Basically, the search for the optimal period vectors for $\mathbf{U}_{min}$ and $\mathbf{V}_{min}$ is performed by repeating following steps for **U** and **V** within possible coordinates of pixels.

Step 1 Check the parallel access condition, that is, whether the super window includes the pixels allocated to the same memory module.

Step 2 If the parallel access condition is satisfied, calculate the area S by (4). Otherwise, go to Step 1.

Step 3 If S is smaller than current minimum area S_{min}, then $S_{min} \leftarrow S$, $\mathbf{U}_{min} \leftarrow \mathbf{U}$, $\mathbf{V}_{min} \leftarrow \mathbf{V}$.

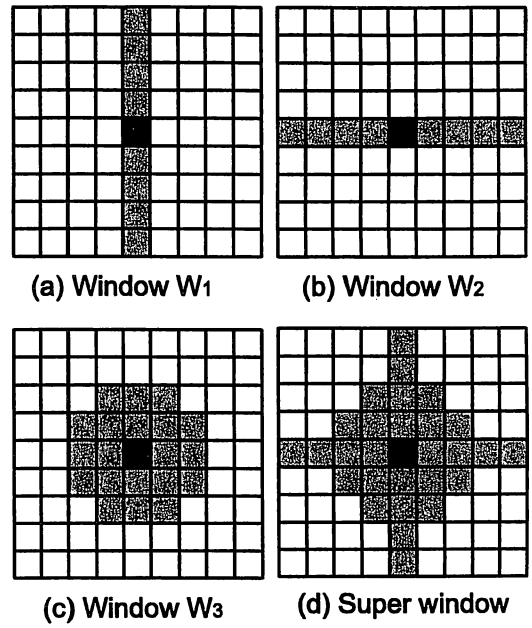

Fig. 6 Super window (Example 1).

To reduce the computational amount, we can limit the reference pixel to $[0\ 0]^T$ as shown in Fig. 5(b). Moreover, we can limit the search space using the branch-and-bound method such that

$$S = |U_x \cdot V_y - U_y \cdot V_x| < S_{min},$$

where S_{min} is the current minimum area of the parallelogram made by the current period vectors.

3.2 Design examples

3.2.1 Example 1

Firstly, let us consider the windows shown in Fig. 6. The windows W_1, W_2 and W_3 are used for vertical edge detection, horizontal edge detection and smoothing in inspection of LSI chips Figure fig:result-exp1 shows the memory allocation result with $\mathbf{U} = [23\ 0]^T$ and $\mathbf{V} = [5\ 1]^T$. The required number of memory modules is 23. The search time is less than 1msec@Pentium4(2GHz) for an image of size 512×512.

Let us compare the proposed method with the conventional approach from the point of the number of memory modules. For the window-type image processing, one efficient memory allocation method is a rectangular memory allocation [10] as shown in Fig. 8. The minimum bounding rectangle approximates a given super window(Fig. 8(a)). The optimal memory allocation for the rectangular window is obtained by the rectangular memory allocation(Fig. 8(b)). The number of memory modules is $K \times L$ for a rectangular window of size $K \times L$. The super window shown in Fig. 6(d) is approximated by a 9×9 bounding rectangular window. Then, the required

10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9
15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14
20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6
12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11
17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8
14	15	16	17	18	19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13
19	20	21	22	23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Fig. 7 Memory allocation result for the example 1.

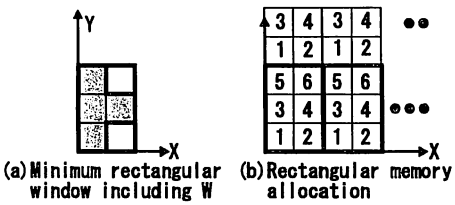

Fig. 8 Rectangular memory allocation.

number of memory modules is 81(= 9 × 9). As a result, the number of memory modules of the proposed method is reduced to 28% in comparison with the rectangular memory allocation.

3.2.2 Example 2

Image processing with multi-resolution images is one efficient method to reduce the computational amount. Figures 9 (a),(b) and (c) show the windows for template matching using multi-resolution images. Figures 9 (a),(b) and (c) are rectangular windows with size of 3 × 3 for sampling period 1, 2 and 3, respectively. Figure 9 (d) is the super windows for them. Figure 10 shows the result of memory allocation with $\mathbf{U} = [11\ 0]^T$ and $\mathbf{V} = [3\ 1]^T$. The required number of memory modules is 11 in the proposed method. The super window requires 49 memory modules in the rectangular memory allocation. Then, the number of memory modules is reduced to 22%. The search time is less than 1msec@Pentium4(2GHz) for an image of size 512 × 512.

4. Conclusion

This paper presents an interconnection-aware design methodology for image processors. A key to success is a optimal memory allocation for a logic-in-memory architecture. The method is also useful for FPGA implementations where interconnection overhead in delay is significant large.

Table 1 Comparison between the proposed method and the rectangular allocation.

	Proposed	Rectangular Allocation
Number of memory modules	121	625
Number of AD units	121	625
Number of adders	123	624

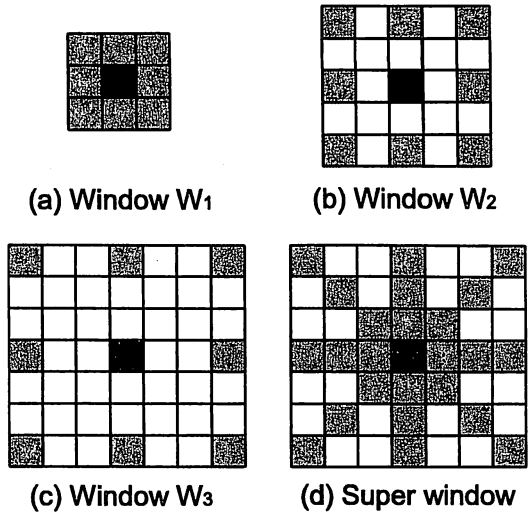

Fig. 9 Example2: Windows and the super window for multiresolution images

10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1
5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4
8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7
11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2
6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5
9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11
4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3
7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6
10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1
5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4
8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7
11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2
6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5
9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11

Fig. 10 Memory allocation result for the example 2

References

[1] D. Gajski, N. Dutt, A. Wu, and S. Lin, "HIGH-LEVEL SYNTHESIS -Introduction to Chip and System Design," Kluwer Academic Publishers, pp.277-278(1992).

[2] W.T. Shiue, S. Tadas and C. Chakrabarti, "Low power multi-modules, multi-port memory design for embedded systems", in Proc. Signal Processing Systems, pp.529-538(2000).

[3] N. Holmes and D. Gajski, "Architectural exploration for datapaths with memory hierarchy", in Proc. European Design Automation Conf., pp.340-344(1994).

[4] L. Ramachandran, D. Gajski, and V. Chaiyakul, "An algorithm for array variable clustering", in Proc. European Design Automation Conf., pp.262-266(1994).

[5] P.R.Panda, "Memory bank customization and assignment in behavioral synthesis", in Proc. Int. Conf. Computer-

Aided Design, pp. 477-481(1999).

[6] S. Wuytack, F. Catthoor, D. De Jong, and H. De Man, "Minimizing the required memory bandwidth in VLSI system relizations", IEEE Trans. VLSI Syst., Vol.7, pp.433-441(1999).

[7] P. R. Panda, F. Catthoor, N. Dutt, L.Danckaert, E. Brockmeyer, C.Kulkarni, A. Vandercappelle, and P.G.Kjeldsberg, "Data and memory optimization techniques for embedded system", ACM Trans. Design Automation Electron. Syst., vol.6, no.2, pp.149-206(2001).

[8] J. Seo, T. Kim, P. R. Panda, "Memory Allocation and Mapping in High-Level Synthesis -An Integrated Approach", IEEE Trans. VLSI Syst. , vol.11, no.5, pp.928-938(2003).

[9] M. Hariyama, H. Sasaki, and M. Kameyama, "Architecture of a Stereo Matching VLSI Processor Based on Hierarchically Parallel Memory Access", IEICE Trans. Info. and Syst., Vol. E88-D, No.7,pp.1486-1491(2005).

[10] M. Hariyama, S. Lee, and M. Kameyama, "Highly-Parallel Stereo Vision VLSI Processor Based on an Optimal Parallel Memory Access Scheme", IEICE Trans. Electron., Vol.E84-C, No.5, pp.382-389(2003).