PAPER Special Section on Low-Power LSI and Low-Power IP

A Low-Power Systolic Array Architecture for Block-Matching Motion Estimation

Junichi MIYAKOSHI[†a)], **Yuichiro MURACHI**[†], **Koji HAMANO**[†], *Members*, **Tetsuro MATSUNO**[†], *Nonmember*, **Masayuki MIYAMA**[†], *and* **Masahiko YOSHIMOTO**[††], *Members*

SUMMARY This paper proposes a low-power systolic array architecture for a block-matching motion estimation processor IP for portable and high-resolution video applications. The architecture features a ring-connected processing element (PE) array to reduce both computation cycles and memory access cycles at the same time, allowing lower power characteristics. The feature of low memory access cycles allows concurrent operation of a half-pel processing unit with no extra cache. Furthermore, the architecture allows various summation schemes for absolute difference values. For that reason, it is applicable to various video coding modes such as the adaptive field/frame mode in MPEG2 and multiple macroblock mode in H.264. When the architecture is introduced to a design of a MPEG2 MP@HL motion estimation processor VLSI, the power consumption of the VLSI is reduced by 45–73% in comparison to cases with conventional architectures for motion estimation.

key words: *motion estimation, MPEG, H.264, block-matching*

1. Introduction

Functions, such as video recording and visual communication, became popular with the portable equipment like a cellular phone. From now on, it will be expected that the service using the higher resolution video spreads more. To guarantee a long time operation with a small battery for power, a low power and a high-resolution video coding VLSI is essential. International video coding standards such as MPEG2, MPEG4 and H.26X employ the block-matching motion estimation technique which occupies a significant portion of whole computation power in the video coding. Also, a high compression efficiency in video coding is required to realize high picture quality with a limited bit-rate for cost-effective archiving and transmission. A detection of an optimal motion vector in the motion estimation is the most critical issue to get a high compression efficiency. So the motion vector estimation greatly affects on the performance of the video codec system. Therefore VLSI motion estimation processor IP have to accomplish a low power feature and a high quality motion vector extraction at the same time, particularly for high-resolution video. This paper describes the systolic array architecture for the motion estimation which has been developed focusing on low power realization for portable video application.

Manuscript received September 3, 2004.
Manuscript revised November 15, 2004.

[†]The authors are with the Faculty of Engineering, Kanazawa University, Kanazawa-shi, 920-8667 Japan.
[††]The author is with the Faculty of Engineering, Kobe University, Kobe-shi, 657-8501 Japan.
a) E-mail: mjun1@cs28.cs.kobe-u.ac.jp

DOI: 10.1093/ietele/e88–c.4.559

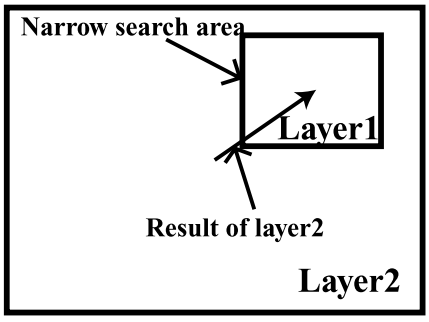

Fig. 1 Hierarchical search.

A hierarchical search method has been utilized frequently as a motion estimation algorithm for high-resolution video [1]–[5]. Figure 1 illustrates the principle of the hierarchical search method. First, a motion vector is searched coarsely in the wide search area named as layer2. The vector to be detected there indicates a start position of the succeeding fine search in the narrow search area named layer1, where the start position is located at the center of layer1. Thus, the computing power can be decreased even for a wide search range. As a motion estimation algorithm in layer1, Ref. [1] used Gradient Descent Search (GDS) method. Studies of Refs. [2] and [5] used the full-search method. The GDS algorithm sacrifices picture quality because the algorithm minimizes the number of block-matching to achieve low power consumption. In contrast, the full-search method provides high picture quality because it is able to obtain the optimal motion vector in the search window. Therefore, an adaptation of the full-search method at layer1 is an optimum solution if it is realized with low power characteristics even for high-resolution video.

A full-search architecture for layer1 must satisfy the following four requirements to realize a hierarchical search algorithm with both low power and flexible features. The first is a reduction of computation cycles. Here, "computation cycles" denotes the sum of working clock cycles that are allocated for SAD calculation and data loading clock cycles. Figure 2 shows a processing timing chart. It illustrates the case for the low-resolution video (a), the case for the high-resolution video with more data load cycles (b), and the case for the high-resolution video with fewer data load cycles (c). Data load cycles in case (a) do not become problematic because a sufficient number of clock cycles are allocated for

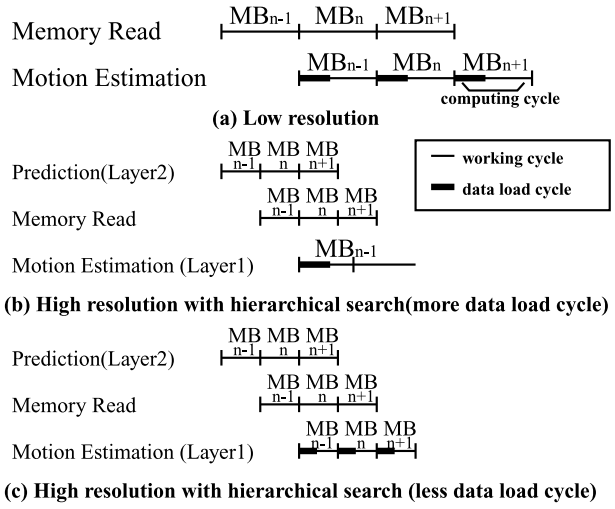

Fig. 2 Macroblock processing timing.

one macroblock. On the contrary, fewer cycles are allocated for one macroblock in high-resolution video. In case (b), although the number of working cycles can be reduced using a hierarchical search, real-time operation is more difficult because of the greater number of data load cycles. Real-time coding is possible if the data load can be performed quickly as in case (c). So VLSI architecture which reduces not only the working cycles, but also the data load cycles is necessary for motion estimation for high-resolution video.

The second requirement is a reduction of memory access cycles. Power consumption of embedded SRAMs for search window data is increasing as memory access cycles increase. Therefore it should be minimized by extensive reuse of accessed pixels. The third one is an adaptability for various video coding modes such as the adaptive field/frame mode in MPEG2 and multiple macroblock mode in H.264. The last one is a concurrent operation with half-pel processing. By doing this, the operating frequency can be reduced allowing less power dissipation.

A full-search architecture for high-resolution video must meet the above requirements. However, the conventional 1D-SA0 [7] meets the second one and the third one. Conventional 1D-SA1 [8] meets only the second one. The conventional 2D-SA [9] meets the first and third one. Finally the conventional tree architecture [9] also meets the first and the third one. None of these conventional architectures support all four features. For that reason, they are not perfect solutions for a low-power motion estimator with high-resolution video.

This paper proposes an efficient systolic architecture to satisfy the above four requirements at the same time, which is able to implement a high quality motion vector estimator IP core with low power characteristics.

Section 2 describes an algorithm of block-matching motion estimation. Conventional architectures are analyzed in Sect. 3. Section 4 describes the proposed architecture for high-resolution video. In Sect. 5, the estimated performance of the architecture is presented in comparison with conventional methods. This is followed by application to the VLSI motion estimator IP core for HDTV resolution video in Sect. 6. Section 7 concludes this presentation.

2. Block-Matching Motion Estimation

In block-matching motion estimation, the motion vector indicates the position of minimum distortion between the current macroblock and the reference macroblock. The full search method based on block-matching can always determine the optimal motion vector because the method searches all candidate blocks in the search area. Thereby, a low bit-rate is obtained without sacrificing the picture quality.

In the full-search method, the sum of absolute differences in luminance as a distortion function is defined as

$$D(V_x, V_y) = \sum_{i}^{N} \sum_{j}^{N} |TB_{i,j} - SW_{i+V_x, j+V_y}| \quad (1)$$

where $N \times N$ represents a macroblock size. $TB_{i,j}$ and $SW_{i+V_x, j+V_y}$ represent pixels at position (i, j) of the current frame and at the position $(i + V_x, j + V_y)$ of the reference frame, respectively. The motion vector (MV_x, MV_y) is obtained by the following equation (2).

$$D(MV_x, MV_y) = D_{min}(V_x, V_y), \quad (V_x, V_y) \in V \quad (2)$$

In that equation, V indicates all candidate vectors.

3. Conventional Architecture

Conventional architectures for the full-search method are explained in this section. Tables 1, 2 and 3 shows the number of accessed pixels and computation cycles of these architectures. The column of the initial load cycle shows the number of cycles loading pixels for SAD calculations. In the initial load cycle, pixels are transferred to PEs and side registers (SRs) before SAD calculations. The column of the idle cycle shows the number of cycles for loading and shifting pixels among PEs and SRs at a search range boundary. A sum of the initial load cycle and the idle cycle is equivalent to the data load cycle. SADs are not calculated in the idle cycles. The column of the working cycle shows the number of cycles calculating SADs. The N represents the number of pixels on a side of a macroblock. A search range is $0 < x \leq 2dx$, $0 < y \leq 2dy$. The number of candidate vectors equals $2dx \times 2dy$. $SR_x and SR_y$ indicate the number of SRs in the vertical and horizontal directions, respectively.

The typical one-dimensional systolic array architecture has been presented in [7]. It is named as 1D-SA0 in this paper. A block diagram of the 1D-SA0 is shown in Fig. 3. It consists of $N \times N$ PEs and $N \times SR_y$ SRs. It has a serial communication path connecting neighboring PEs and SRs. A small amount of pixel data is transferred on the serial communication path because all pixels in the search area are basically read only once. In practice, it is necessary to iterate searches for divided areas on the condition of

Table 1 Comparison of architectures for total accessed pixels [/MB].

	#	Total Accessed Pixels	
	PE	TB	SW
1D-SA0 [7]	N^2	N^2	$(2dx+N)\times(2dy+N)+(2dy+N)\times(N-1)\times((2dx/N)-1)$
1D-SA1 [8]	N^2	N^2	$(2dx+N)\times(2dy+N)+(2dy+N)\times(N-1)\times((2dx/N)-1)$
2D-SA [9]	N^2	N^2	$N\times(N+2dx)\times 2dy$
Tree Arch. [9]	N^2	N^2	$N\times(N+2dx)\times 2dy$
Proposed Arch.	N^2	N^2	$(2dx+N)\times(2dy+N)+(2dy+N)\times(N-1)\times((2dx+N)/(N+SRx))-1)$

Table 2 Comparison of arhitectures for computation cycles [/MB].

	#	Computation Cycles		
	PE	Inital load cycle	Idle cycle	Working cycle
1D-SA0 [7]	N^2	$N\times(N+SRy)$	$2dx\times N\times(2dy/SR_y)$	$2dx\times 2dy$
1D-SA1 [8]	N^2	$N\times N$	$2dy\times(N-1)$	$2dx\times 2dy$
2D-SA [9]	N^2	N	$2dy\times(N-1)$	$2dx\times 2dy$
Tree Arch. [9]	N^2	N	$2dy\times(N-1)$	$2dx\times 2dy$
Proposed Arch.	N^2	N	$N\times((2dx+N)/(N+SRx)-1)$	$2dx\times 2dy$

Table 3 Comparison of arhitectures($N=16, dx=dy=8, SR_x=SR_y=16$)[/MB].

	#	Total Accessed Pixels	Computation Cycles			
	PE	SW	Inital load cycle	Idle cycle	Working cycle	Total
1D-SA0 [7]	256	1024	512	256	256	1024
1D-SA1 [8]	256	1024	256	240	256	752
2D-SA [9]	256	8192	16	240	256	512
Tree Arch. [9]	256	8192	16	240	256	512
Proposed Arch.	256	1024	16	0	256	272

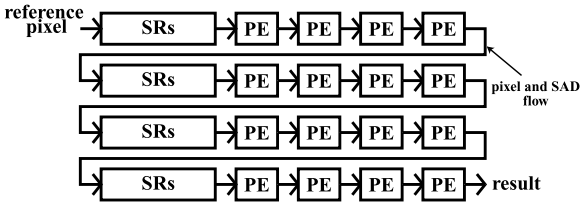

Fig. 3 Block diagram of 1D-SA0.

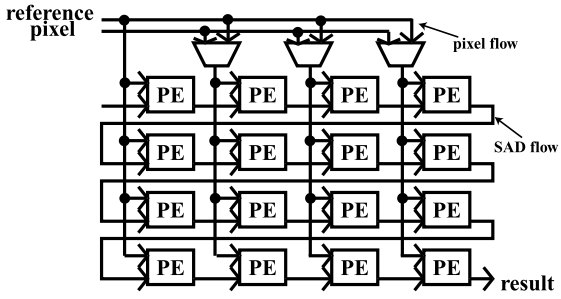

Fig. 4 Block diagram of 1D-SA1.

the wide search area. This situation requires pixels which have been already accessed. The number of these pixels is represented as $(2dy+N)\times(N-1)\times((2dx/N)-1)$. This architecture presents problems of the huge number of initial load cycles and idle cycles. In the initial load cycle, pixels must be transferred to all PEs and SRs to commence SAD calculations. The number of the initial load cycles is the same as the number of PEs and SRs. The number of computation cycles is represented as $(2dx+N)\times(2dy+N)\times(dy/SR_y)$.

The modified one-dimensional systolic array architecture was proposed in [8]. It was named TMSB with the search-area division method in [8]. In this paper, it is 1D-SA1. Figure 4 shows a block diagram of 1D-SA1. The number of initial load cycles is equal to the number of PEs. It reduces the idle cycles by optimizing data flow. It can execute full-search motion estimation with lower computation cycles compared to those of the 1D-SA0. The number of computation cycles is represented as $N^2+(2dx+N-1)\times 2dy$. On the condition that the horizontal size of the search area is more than N pixels, iterated searches for each divided area require pixels that have already been accessed. The 1D-SA1 requires the same number of pixels as 1D-SA0. It is notable that the 1D-SA1 cannot address adaptive field/frame mode and multi template block mode well because of the serially connected adders.

Two-dimensional systolic array architecture was proposed in [9]. That architecture is named 2D-SA herein. Figure 5 displays a block diagram of 2D-SA. It consists of $N\times N$ PEs. It has a mesh-form PE array and realizes fast computation by loading many pixels to the PE array at one time. The 2D-SA has less idle cycles. Nevertheless, it must access many pixels because it cannot reuse pixels that have already been accessed.

The tree architecture has been proposed in [9]. Figure 6 shows a block diagram of the tree architecture. It consists of N^2-way PEs. The number of computation cycles is same as that of 2D-SA. The tree architecture requires the same number of pixels as 2D-SA.

A processor based on one of these architectures cannot

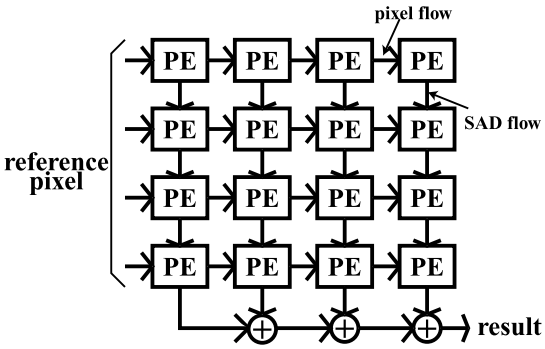

Fig. 5 Block diagram of 2D-SA.

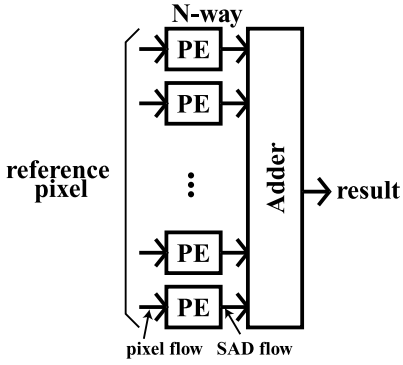

Fig. 6 Block diagram of tree architecture.

operate along with the Half-Pixel Processing Unit (HPPU) simultaneously because the processor or HPPU occupies an image data cache. This engenders an increase in the total computation cycles or the number of caches. Therefore, these conventional architectures are unsuitable for a motion estimator that addresses high-resolution video because they do not satisfy one or more conditions, as mentioned in the previous section.

4. Proposed Systolic Array Architecture

The proposed systolic array architecture is illustrated in Fig. 7. The TB represents a template block buffer. The SW represents a search window buffer.

4.1 Processing Elements and the Shift Registers Array

The architecture consists of the $N \times N$ PEs (PE array), the $N \times N$ SRs (SR array), and an adder tree. Assuming that N is 16, the number of PEs and SRs is 256 each. Sixteen pixels (128 bits) in the search window and 16 pixels (128 bits) in the template block are loaded into the PE array from the SW and TB, respectively. The SR array loads 16 pixels from the SW. Those input ports are shown in the upper side of Fig. 7. The PEs and SRs in one row are connected as a ring-buffer. The PE-SR array can shift pixels to the right side or left side.

Figure 8 shows the PE block diagram. The PE is a circuit to calculate an absolute difference between pixels of the

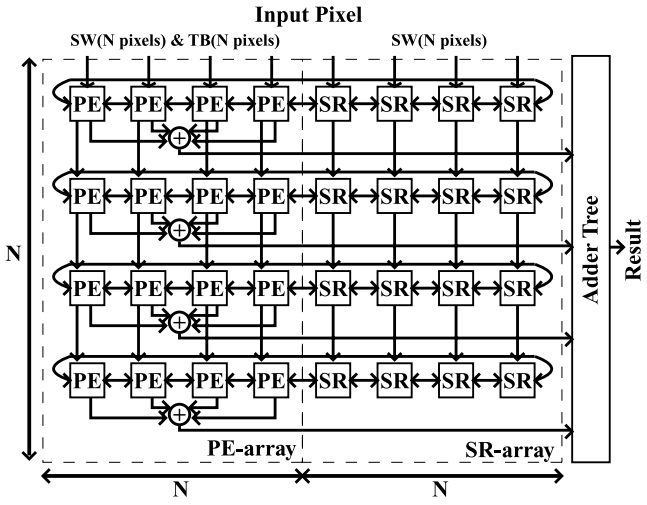

Fig. 7 Block diagram of the systolic array architecture.

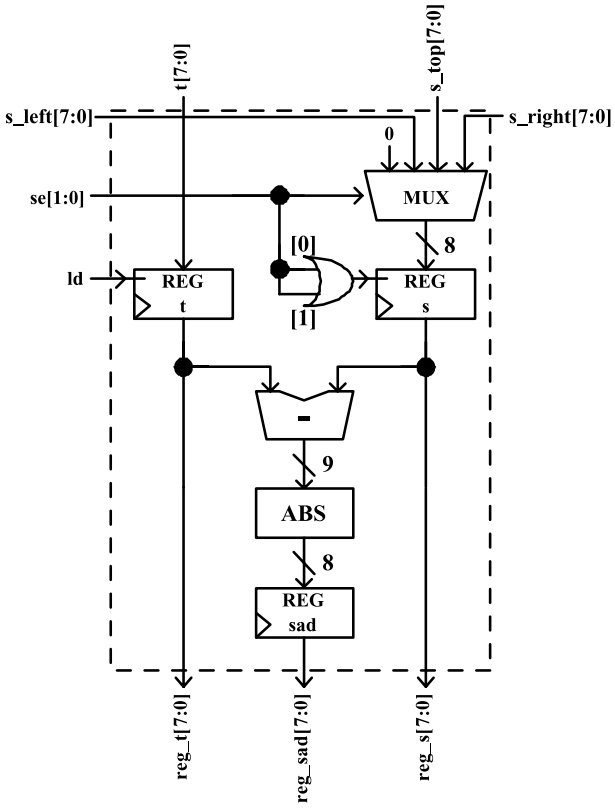

Fig. 8 Block diagram of the processing element.

template block and the reference block. The t and s represent pixels of the template block and the reference block, respectively. The input s can be loaded from the right, left, or upper PE. The input t is loaded from the upper PE. The subtracter calculates the difference between t and s. The ABS indicates a module to calculate an absolute value. Figure 9 shows a block diagram of the SR. SR reads a reference pixel from the left, the right or the top SR. The data in the search range are shifted to the right side or the left side, or from

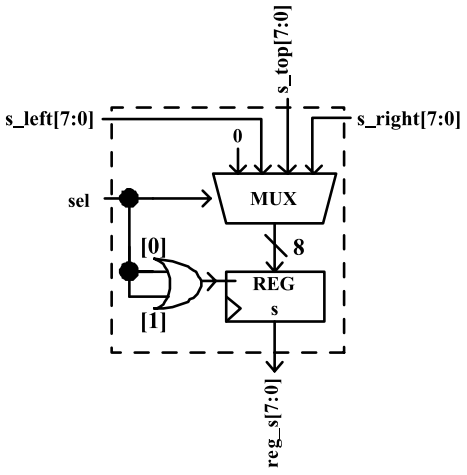

Fig. 9 Block diagram of the shift register.

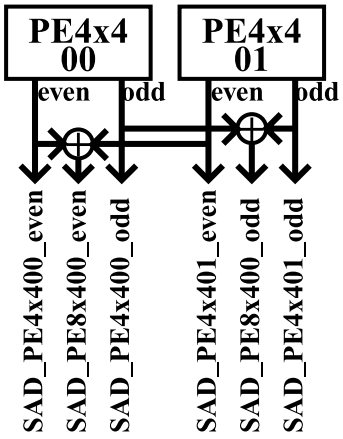

Fig. 11 Block diagram of adder tree.

the upper side to the lower side. Therefore, SR consists of a register to hold a pixel, and a multiplexer to select a pixel. SR has the same configuration as the pixel transmission part of PE.

4.2 Data Flow

Figure 10 illustrates the data flow of the proposed architecture. Four phases execute the full search. The first is an initialize phase (Init-phase). Pixels of the template block and the search window are loaded into all PEs and SRs in this phase. The second one is a calculation phase (Calc-phase). The SADs are calculated by the PE array in this phase. The pixels that were loaded into PEs and SRs in the previous phase are shifted to the right side or left side. The third one is an input phase (Input-phase). In this phase, the subsequent pixels are transferred from TB and SW to a PE array and SR array. The full-search can be executed by iterating the Calc-phase and the Input-phase in turn after the Init-phase. The Calc-phase and the Input-phase are iterated until reaching a boundary of the search area. The subsequent pixels are loaded by vertical shift operations at that time (Idle-phase).

Tables 1 and 2 show the number of total accessed pixels and computation cycles of the proposed architecture. As stated above, the number of the total accessed pixels of the search window is represented as $(2dx + N) \times (2dx + N)$. When the search area is expanded, reloading pixels that have already been accessed becomes a requirement. The number of these pixels is represented as $(2dy+N)\times(N-1)\times((2dx+N)/(N+SR_x)-1)$. It is less than those of 1D-SA0 and 1D-SA1. The number of cycles in the Init-phase, the Calc-phase and the Input-phase are N, one, and one, respectively. The Idle-phase needs $N\times((2dx+N)/(N+SR_x)-1)$ cycles. The Calc-phase and Input-phase are repeated $2dy$ times. Therefore, the computation cycles become $(N+N\times((2dx+N)/(N+SR_x)-1)+2dx\times 2dy)$. The architecture can operate with few computation cycles and access pixels.

4.3 Adder Tree

Figure 11 illustrates part of a block diagram of the adder tree. The SADs of even rows and odd rows are obtained for each 4×4 PEs. The SAD for various template block modes, such as 4×4, 8×8, and 8×16, can be calculated by the adder tree. Furthermore, the adder tree can calculate the frame SAD by adding the even and odd field SADs. Because of these useful functions, the adder tree is applicable to various kinds of video coding.

4.4 Data Transferal

The pixels of the search window are transferred from SW to the PE array and the SR array. A cross path that swaps high 128 bits and low 128 bits is inserted between the SW and the systolic architecture. In the Init-phase, the pixels of the search window and the template block are loaded into the all PEs and SRs during N cycles. In the Input-phase and Idle-phase, pixels of the search window are loaded into the PE array. Reloading pixels of the template block is not necessary because they are held in the PE array in phases.

Moreover, using the SW constructed by 2-port SRAMs, the cross path is not needed. The pixels can be swapped by controlling the SRAM addresses.

4.5 Memory Access Timing

After the Init-phase, the systolic array accesses SW per the Input-phase. During the Calc-phase, in which the SW is not accessed by the array, other modules such as an HPPU can access the SW. The systolic array and the other modules operate simultaneously, thereby raising computing efficiency.

The HPPU executes block matching of neighbor 3×3 macroblocks. Assuming an HPPU constructed by N-way SIMD, the computation cycles are $N\times3\times3$. Using them, the proposed architecture reduces the total computation cycles by $N\times3\times3$.

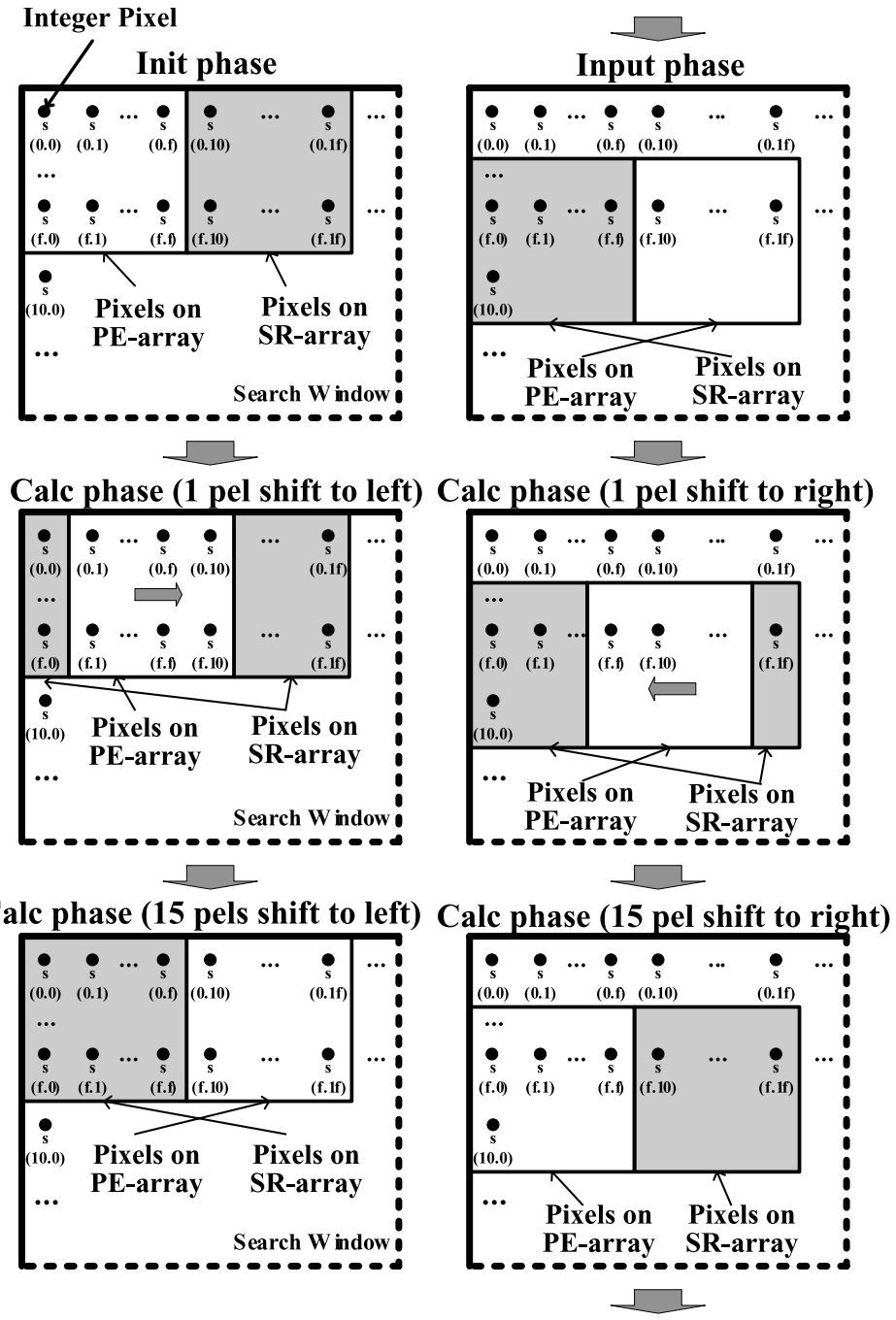

Fig. 10 Data flow of proposed systolic architecture.

5. Area and Power Estimation

Assuming $N = 16, dx = dy = 8$ and $\#PE = 256$, performance of the proposed architecture is estimated and compared with that of the conventional techniques.

Table 4 Module configuration of each architecture.

Name	PE,SR	AT	RAM(SW)	RAM(TB)
1D-SA0	no mux.	global	8 bit,1024 word,1 port	8 bit,256 word
1D-SA1	no mux.	local	8 bit,1024 word,2 port	8 bit,256 word
2D-SA	no mux.	local	128 bit,64 word,1 port	128 bit,16 word
Tree	no mux.	global	128 bit,64 word,1 port	128 bit,16 word
Proposed	3 inputs mux	global	128 bit,64 word,2 port	128 bit,16 word

5.1 Area Estimation

An area ratio is estimated in order to compare the circuit scale of each architecture. Table 4 shows the configuration of each module of each architecture. Regarding PE and SR, the number of input ports for a reference pixel and multi-

Table 5 Area and energy (E) ratio of each module.

Module	configuration	Area	E
PE	no mux.	0.89	0.89
	3 inputs mux.	1.00	1.00
SR	no mux.	0.58	0.58
	3 inputs mux.	1.00	1.00
AT	local	1.60	1.60
	global	1.00	1.00
RAM (SW)	8 bit,1024 word,1 port	0.22	0.13
	8 bit,1024 word,2 port	0.49	0.15
	128 bit,64 word,1 port	0.44	0.70
	128 bit,64 word,2 port	1.00	1.00
RAM (TB)	8 bit,256 word,1 port	0.27	0.17
	128 bit,16 word,1 port	1.00	1.00

Table 6 Area and energy (E) ratio of the proposed architecture based on actual design data.

	PE	SR	AT	CTRL	TB	SW	Total
Area	0.42	0.11	0.09	0.04	0.09	0.26	1.00
E_{work}	0.39	0.10	0.08	0.04	0.06	0.34	1.00
E_{idle}	0.10	0.10	0.00	0.04	0.06	0.34	0.63

Table 7 Area comparison.

Name	PE	SR	AT	CTRL	TB	SW	Total
1D-SA0	0.38	0.06	0.09	0.04	0.02	0.06	0.65
1D-SA1	0.38	0.00	0.14	0.04	0.02	0.13	0.71
2D-SA	0.38	0.00	0.14	0.04	0.09	0.11	0.76
Tree	0.38	0.00	0.09	0.04	0.09	0.11	0.70
Proposed	0.42	0.11	0.09	0.04	0.09	0.26	1.00

plexer configuration differ for respective architectures. The AT part has two kinds: a local AT that calculates a partial sum in a PE, and a global AT that calculates it externally. The gate count of local AT is larger because registers (FF) that store a partial sum are needed in each PE. As for RAM, the number of bits and the number of ports are different for each architecture. Table 5 indicates the configuration and area ratio of each module. The technology used for estimation was a 0.18 μm six-metal layer CMOS process. Regarding the logic part, its area was estimated by logic synthesis. The RAM area was estimated with a RAM generator. The SW has 8 kbit capacity and the TB has 2 Kbit capacity. Table 6 summarizes the area ratio of each module which is obtained from the actual design data of the motion estimation processor based on the proposed architecture in the technology. Table 7 shows the area ratios of respective parts of each architecture as calculated from the results in Tables 4, 5 and 6.

5.2 Power Estimation

Figure 12 shows an estimation of normalized computation cycles. It shows that estimations for the proposed method are lower than those for the conventional ones. Figure 13 indicates the quantity of normalized total accessed pixels. It shows that the proposed architecture requires a minimum number of pixels, which results in a minimum number of memory cycles.

We next derive the formula that describes power consumption. The power consumption value comprises a logic part (P_{logic}) and a RAM part (P_{RAM}). The power consumption of a logic part can be divided into that of a PE-array (P_{PE}), an SR-array (P_{SR}), the AT (P_{AT}), and the CTRL (P_{CTRL}). Moreover, the RAM part is devided into P_{SW} and P_{TB}. Consequently, the power consumption (P_{total}) can be described as

$$P_{total} = P_{logic} + P_{RAM} \quad (3)$$

$$= (P_{PE} + P_{SR} + P_{AT} + P_{CTRL}) + (P_{TB} + P_{SW}) \quad (4)$$

$$= f \times \{\alpha_{work}(E_{PEw} + E_{SRw} + E_{ATw} + E_{CTRLw}) + \alpha_{idle}(E_{PEi} + E_{SRi} + E_{CTRLi}) + \alpha_{SW}E_{SW} + \alpha_{TB}E_{TB}\} \quad (5)$$

where α_{work} represents the activity of the working cycle of a logic part and α_{idle} shows that of the idle cycle of the logic part. The idle cycle includes an initial load cycle. Because operations of PE, SR, and AT differ, energy consumption also differs. E_{PEw}, E_{SRw}, E_{ATw}, and E_{CTRLw} indicate the energy consumption of the working cycle of a PE-array, an SR-array, the AT, and the control part, respectively. On the other hand, E_{PEi}, E_{SRi}, and E_{CTRLi} show the energy consumption of the idle cycle of the PE-array, SR-array, and the control part, respectively. α_{SW} and α_{TB} represent the activity of each RAM. E_{SW} and E_{TB} show the energy consumption

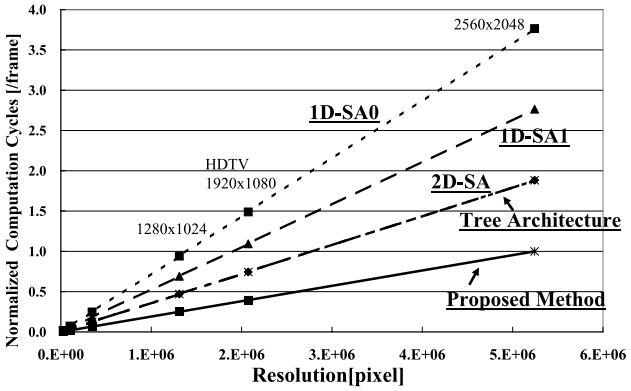

Fig. 12 Computation cycle estimation.

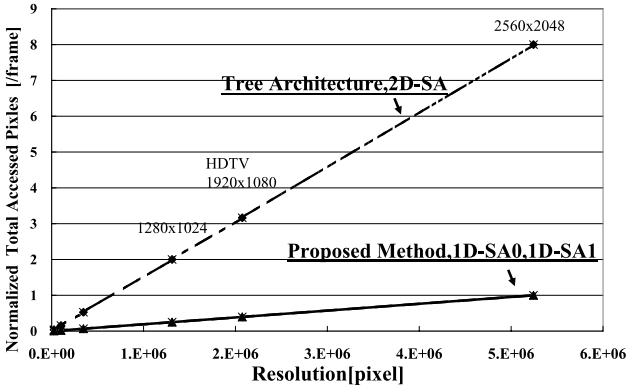

Fig. 13 Total accessed pixels estimation.

of each RAM.

It is assumed that the E ratio of each module in logic portion is equal to the ratio of gate count (which is equal to the area ratio here,) as shown in Table 5. The E ratio computed with a RAM generator is shown in Table 5. In addition, E ratio of each module of the proposed architecture, as calculated by the circuit simulation, is shown in Table 6. Table 8 shows the obtained E ratio of respective parts in respective architectures. Here, for architectures for which SR are unnecessary (i.e., 1D-SA0, 2D-SA, Tree), $E_{SRw} = E_{SRi} = 0$. E_{CTRL} of a working cycle and of an idle cycle are assumed as equal.

Power consumption of circuits based on respective architectures are calculated using Eq. (5). For f, we used the value of computation cycle in Fig. 12 that is computed theoretically. α_{work} and α_{idle} are obtained from Table 2. For α_{SW}, we used the multiplication result for the number of total accessed pixels in Fig. 13 and the number of output bits of RAM of each architecture divided by the number of output bits of the proposed architecture. Because the numbers of read-out pixels of RAM for all architectures are equal, the reciprocal of the above-mentioned ratio was utilized as α_{TB}. E ratio is shown in Table 8. The power consumption estimated from Eq. (5) is shown in Fig. 14.

Therefore, the area overhead of the proposed architecture is 28–35%. Power consumption reduction by the proposed method is about 87% for 2D-SA, about 68% for 1D-SA1, 87% for Tree, and 86% for 1D-SA0. Low power consumption is crucial for chips that are used in mobile HDTV video equipments; the proposed method can meet that demand.

Table 8 Energy E ratio of each module.

Name	E_{PEw}	E_{SRw}	E_{ATw}	E_{PEi}	E_{SRi}	E_{CTRL}	E_{TB}	E_{SW}
1D-SA0	0.89	0.15	0.21	0.23	0.15	0.10	0.03	0.12
1D-SA1	0.89	0.00	0.33	0.23	0.00	0.10	0.03	0.13
2D-SA	0.89	0.00	0.33	0.23	0.00	0.10	0.15	0.61
Tree	0.89	0.00	0.21	0.23	0.00	0.10	0.15	0.61
Proposed	1.00	0.26	0.21	0.23	0.26	0.10	0.15	0.87

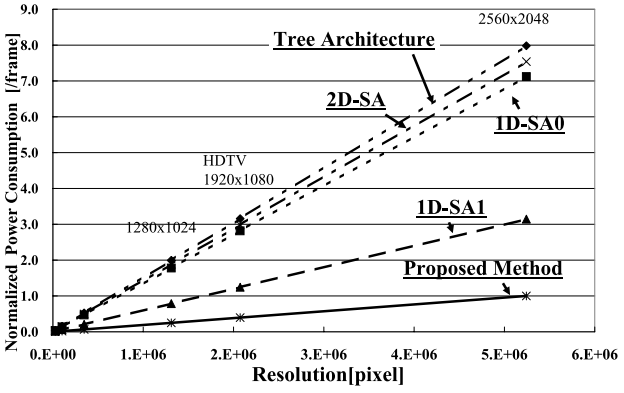

Fig. 14 Power consumption estimation.

6. Application to MPEG2 MP@HL Motion Estimation Processor Core (MEH)

6.1 Overview of MEH

The proposed architecture was applied to design of a HDTV MP@HL motion estimation processor core (MEH). Specifications of the MEH are shown in Table 9. The technology used was a 0.18 μm 6-metal layer CMOS process. Figure 15 shows a block diagram of MEH. The plot image is shown in Fig. 16. MEH uses a hierarchical search algorithm. It performs a one-dimensional diamond search (1D-DS) [11] on layer2 with MEH2 and a full-search on layer1 with MEH1. In addition, a half-pel search is executed using MEHH.

MEH2 consists of an 89 kbit image data cache and a 16-way SIMD circuit. Using MEH2, a low-computing-power algorithm that is based on the gradient method is adopted to search for a wide area roughly. Therefore, the computing power of MEH2 is as low as about 5% of that of MEH1. The power consumption becomes similarly small. Nevertheless, MEH2 requires a large capacity of RAM to perform a wide search. For that reason, MEH2 occupies a large area.

On the other hand, MEH1 searches a narrow range in detail. The proposed architecture is used for a systolic array

Table 9 MEH specification.

Tech.	0.18 μm 6-layer metal
Core Size	3.1 mm × 3.1 mm
Supply	1.0 V
Freq.	108 MHz
Transistor#	2.25 M
Power consumption	95 mW

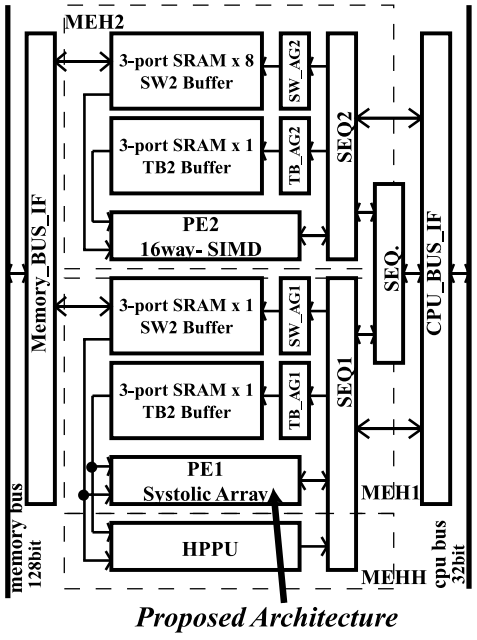

Fig. 15 Block diagram of MEH.

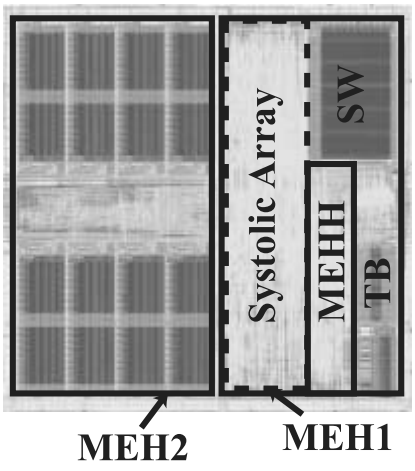

Fig. 16 MEH plot image.

processor unit (PE1) of MEH1. In MEH1, a 3-port SRAM (1write/2read) used as an image cache is well suitable for the proposed architecture. By this configuration MEH1 does not require complicated wiring between a processor unit and a memory. Furthermore, the proposed architecture can perform both field and frame mode search.

A memory bandwidth is 13 Gbps assuming that the memory bus operates at 108 MHz with 128 bit width. The total amount of transmission of MEH1 is 3.5 Gbps; that of MEH2 is 1.5 Gbps. In addition, the total amount of transmission of other modules (DCT, IDCT, etc) is 3.3 Gbps. Therefore, the bandwidth of memory I/F of MEH is fully secured.

6.2 Performance Estimation

Figure 17 shows the area for each architecture. This graph shows respective areas: MEH2, MEH1, MEHH, and Other (interface, clock tree). The area of MEH1 with each architecture is obtained from Table 7. The areas of modules other than MEH1 (i.e., MEH2, MEHH, and Other) are fixed. The area overhead values of the proposed architectures are 10% for Tree architecture, 12% for 1D-SA0, 10% for 1D-SA1, and 8% for 2D-SA. A large SW memory in MEH2 mitigates the area overhead of proposed architecture for MEH1.

Figure 18 depicts the power consumption of MEH with respective architectures. "Other" indicates power of the interface and clock tree. Power consumption of the proposed architecture is estimated by a circuit simulator; those of the conventional architecture are evaluated according to Fig. 14. MEH with the proposed architecture achieves about 95 mW. Power consumption is reduced by 73% in comparison to that of the Tree architecture, by 70% compared to 1D-SA0, by 72% compared to 2D-SA, and by 45% compared to 1D-SA1. Therefore, MEH with the proposed architecture reduces power consumption dramatically despite the 8–12% area overhead.

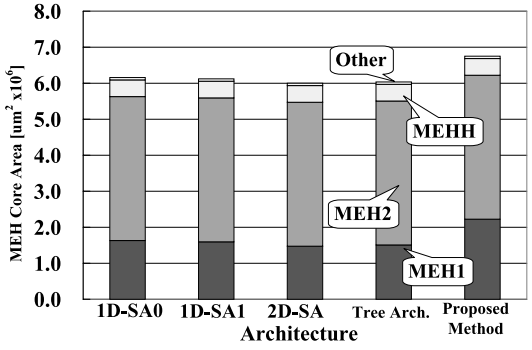

Fig. 17 Core area of MEH.

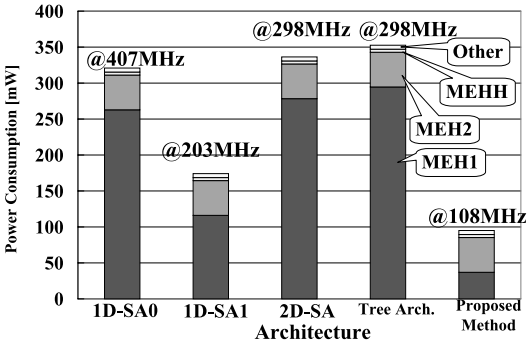

Fig. 18 Power consumption of MEH.

7. Conclusion

This paper presents a novel low-power systolic array architecture for blockmatching motion estimation for a portable high-resolution video application. It offers the following features:

- It can execute full-search algorithm efficiently.
- The computation cycles and memory access cycles decrease through the use of a ring-connected PE array.
- Field/frame and various template block modes can be addressed using the adder tree.
- A half-pixel processing unit can operate in parallel using the low memory access capability, with no extra cache.

Therefore, the proposed architecture is applicable to high-resolution video motion estimation. When it is implemented in the motion estimator for MPEG2 MP@HL, the overhead of area is 8–12%, and the power consumption is reduced by 73% from the tree architecture, by 70% from the 1D-SA0, by 72% from the 2D-SA, and by 45% from the 1D-SA1.The estimated power consumption of the processor is 95 mW.

Acknowledgments

The authors would like to thank the Semiconductor Technology Academic Research Center (STARC) for allowing

us the opportunity to develop the LSI. The VLSI chip in this study was fabricated in the chip fabrication program of the VLSI Design and Education Center (VDEC) and MOSIS.

References

[1] M. Miyama, O. Tooyama, N. Takamatsu, T. Kodake, K. Nakamura, A. Kato, J. Miyakoshi, K. Imamura, H. Hashimoto, S. Komatsu, M. Yagi, M. Morimoto, K. Taki, and M. Yoshimoto, "An ultra low power motion estimation processor for MPEG2 HDTV resolution video," IEICE Trans. Electron., vol.E86-C, no.4, pp.561–569, April 2003.

[2] A. Harada, S. Hattori, T. Kasezawa, H. Sato, T. Matsumura, S. Kumaki, K. Ishihara, H. Segawa, A. Hanami, Y. Matsuura, K. Asano, T. Yoshida, M. Yoshimoto, and T. Murakami, "An architectural study of an MPEG-2 422P@HL encoder chip set," IEICE Trans. Fundamentals, vol.E83-A, no.8, pp.1614–1623, Aug. 2000.

[3] M. Ikeda, T. Kondo, K. Nitta, K. Suguri, T. Yoshitome, T. Minami, H. Iwasaki, K. Ochiai, J. Naganuma, M. Endo, Y. Tashiro, H. Watanabe, N. Kobayashi, T. Okubo, T. Ogura, and R. Kasai, "SuperENC: MPEG-2 video encoder chip," IEEE Micro, vol.19, no.4, pp.56–65, July–Aug. 1999.

[4] J.-M. Rovati, F.S. Pau, D. Piccinelli, E. Pezzoni, and L. Bard, "An innovative, high quality and search window independent motion estimation algorithm and architecture for MPEG-2 encoding," IEEE Trans. Consum. Electron., vol.46, no.3, pp.697–705, Aug. 2000.

[5] T. Onoye, G. Fujita, M. Takatsu, I. Shirakawa, and N. Yamai, "Single chip implementation of motion estimator dedicated to MPEG2 MP@HL," IEICE Trans. Fundamentals, vol.E79-A, no.8, pp.1210–1216, Aug. 1996.

[6] P. Kuhn, Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwer Academic Publishers, 1999.

[7] S. Uramoto, A. Takabatake, M. Suzuki, H. Sakurai, and M. Yoshimoto, "A half-pel precision motion estimation processor for NTSC-resolution video," IEICE Trans. Electron., vol.E77-C, no.12, pp.1930–1936, Dec. 1994.

[8] T. Minami, T. Kondo, K. Suguri, and R. Kasai, "A proposal of a one-dimensional array architecture for the full-search block matching algorithm," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J78-D-I, no.12, pp.913–925, Dec. 1995.

[9] Y. Jehng and L. Chen, "An efficient and simple VLSI tree architecture for motion estimation algorithms," IEEE Trans. Circuits Syst. Video Technol., vol.41, no.2, pp.889–900, Feb. 1993.

[10] V.L. Do and K.Y. Yun, "A low-power VLSI architecture for full-search block-matching motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.8, no.4, pp.613–619, Aug. 1998.

[11] Y. Murachi, K. Hamano, J. Miyakoshi, M. Miyama, and M. Yoshimoto, "A low power MPEG2 MP@HL motion estimation processor for mobile video systems," IEICE Technical Report, ICD2004-5, May 2004.

Junichi Miyakoshi was born in 1980. He received a B.S. degree from Kanazawa University in 2002. He received an M.S. degree from Kanazawa University in 2004. He is currently enrolled in the doctoral course there. His research focus is low power VLSI techniques for image processing.

Yuichiro Murachi was born on November 1, 1980. He received a B.S. degree in electrical and information engineering from Kanazawa University in 2003. He is currently a master's course student at Kanazawa University. His research interests are VLSI systems and implementation of multimedia communication systems.

Koji Hamano was born on November 5, 1981. He received a B.E. degree in electrical and electronic engineering from Kanazawa University in 2004. He is currently a master's course student at Kanazawa University. His research interests are low power VLSI systems.

Tetsuro Matsuno received a B.E. degree in electrical and electronic engineering from Kanazawa University in 2004. He is currently a master's student at Kanazawa University. His interests include low power VLSI systems.

Masayuki Miyama was born on March 26, 1966. He received a B.S. degree in computer science from the University of Tsukuba in 1988. He Joined PFU Ltd. in 1988. He received an M.S. degree in computer science from the Japan Advanced Institute of Science and Technology in 1995. He joined Innotech Co. in 1996. He is a research assistant at the Department of Electrical and Electronic Engineering at Kanazawa University. His present research focus is low power design techniques for multimedia VLSI.

Masahiko Yoshimoto received a B.S. degree in electronic engineering from Nagoya Institute of Technology, Nagoya, Japan, in 1975, and an M.S. degree in electronic engineering from Nagoya University, Nagoya, Japan, in 1977. He received a Ph.D. degree in Electrical Engineering from Nagoya University, Nagoya, Japan in 1998. He joined the LSI Laboratory, Mitsubishi Electric Corporation, Itami, Japan, in April 1977. From 1978 to 1983 he was engaged in the design of NMOS and CMOS static RAM including a 64K full CMOS RAM with the world-first divided-word-line structure. From 1984 he was involved in research and development of multimedia ULSI systems for digital broadcasting and digital communication systems based on MPEG2 and MPEG4 Codec LSI core technology. Since 2000, he has been a professor of the Dept. of Electrical and Electronic Systems Engineering at Kanazawa University, Japan. Since 2004, he has been a professor of the Dept. of Computer and Systems Engineering in Kobe University, Japan. His current activity is focused on research and development of multimedia and ubiquitous media VLSI systems including an ultra low power image compression processor and a low power wireless interface circuit. He holds 70 registered patents. He has served on the program committee of the IEEE International Solid State Circuit Conference from 1991 to 1993. In addition, he has served as a Guest Editor for special issues on Low-Power System LSI, IP, and Related Technologies of IEICE Transactions in 2004. He received the R&D100 awards from R&D magazine for development of the DISP and development of a realtime MPEG2 video encoder chipset in 1990 and 1996, respectively.