IEICE TRANS. ELECTRON., VOL.E89–C, NO.12 DECEMBER 2006

PAPER

A Fine-Grain Scalable and Low Memory Cost Variable Block Size Motion Estimation Architecture for H.264/AVC

Zhenyu LIU[†a)], *Nonmember*, **Yang SONG**[††], *Student Member*, **Takeshi IKENAGA**[††], *Member*, *and* **Satoshi GOTO**[††], *Fellow*

SUMMARY One full search variable block size motion estimation (VBSME) architecture with integer pixel accuracy is proposed in this paper. This proposed architecture has following features: (1) Through widening data path from the search area memories, *m* processing element groups (PEG) could be scheduled to work in parallel and fully utilized, where *m* is a factor of sixteen. Each PEG has sixteen processing elements (PE) and just costs 8.5K gates. This feature provides users more flexibility to make tradeoff between the hardware cost and the performance. (2) Based on pipelining and multi-cycle data path techniques, this architecture can work at high clock frequency. (3) The memory partition number is greatly reduced. When sixteen PEGs are adopted, only two memory partitions are required for the search area data storage. Therefore, both the system hardware cost and power consumption can be saved. A 16-PEG design with 48×32 search range has been implemented with TSMC 0.18 μm CMOS technology. In typical work conditions, its maximum clock frequency is 261 MHz. Compared with the previous 2-D architecture [9], about 13.4% hardware cost and 5.7% power consumption can be saved.

key words: *H.264, AVC, variable block size motion estimation, VLSI architecture*

1. Introduction

H.264/AVC is the newest international video coding standard, which is jointly developed by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). Compared with previous standards, H.264/AVC can provide much better peak signal-to-noise ratio (PSNR) and visual quality [1]. This high performance is mainly due to many new techniques adopted by H.264/AVC, such as variable block sizes motion compensation, quarter-sample-accurate motion compensation, multiple reference picture motion compensation, in-the-loop deblocking filtering and so on [2].

In H.264/AVC, motion estimation (ME) is conducted on different blocks sizes including 4×4, 4×8, 8×4, 8×8, 8×16, 16×8 and 16×16, as illustrated in Fig. 1. During ME, all blocks inside one macroblock (MB) are processed and the block mode with the best R-D cost is chosen. This process is named VBSME [3]. Compared with previous fixed block size ME process, VBSME can achieve higher compression ratio and better video quality. However, it puts heavy burden on the ME unit and makes traditional hardware architectures incompatible.

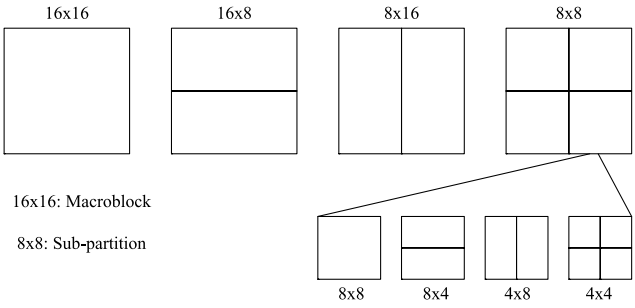


Fig. 1 Variable block sizes in H.264/AVC.

Because of the intensive computation of ME, the hardware accelerator is critical for real-time encoding system. Full search algorithm is widely used because it has following merits: (1) Its performance is superior to other fast algorithms and stable in all applications; (2) Its processing time is predictable and fixed; (3) Its control logic and memory access are simple and regular. Therefore, various architectures have been proposed to realize the full search ME algorithm. For example, a one dimension (1-D) systolic array ME architecture for MPEG-4 is proposed in [4]. Various two dimension (2-D) systolic array ME architectures are presented in [5] and [6]. Reference [7] presents a parallel tree architecture with partial distortion elimination algorithm, which is based on the tree architecture proposed in [8]. By combing sub-trees in horizon, the horizontal adjacent search area can be shared and the shift registers in 1-D and 2-D systolic architectures can be eliminated. However, these architectures can not easily be adopted in H.264/AVC because they can not fully support all the seven block modes in H.264/AVC.

In H.264/AVC reference software, the best matching position for one block is decided by the sum of absolute differences (SAD) and coding cost of motion vector difference (MVD). However, the calculation of MVD needs the exact motion vector (MV) of left, top and top right neighboring blocks. Therefore, the four 8×8 sub-partitions in one MB have to be processed in sequence. This inherent data dependency in the integer ME (IME) algorithm makes the parallel processing of all forty-one blocks within one MB infeasible. A modified IME algorithm is proposed in [9]. In that paper, MVD cost is not taken into account and SAD is the only criterion in IME processing. Because this algorithm avoids

Manuscript received October 6, 2005.
Manuscript revised March 29, 2006.

[†]The author is with Kitakyushu Foundation for the Advancement of Industry Science and Technology, Kitakyushu-shi, 808-0135 Japan.
[††]The authors are with IPS, Waseda University, Kitakyushu-shi, 808-0135 Japan.
a) E-mail: liuzhenyu@kyushu.rise.waseda.ac.jp

DOI: 10.1093/ietele/e89–c.12.1928

the data dependency caused by MVD with negligible PSNR loss, it is more suitable for hardware implementation.

Based on the above algorithm, one efficient 2-D VBSME full-search architecture is proposed [9]. The design has 256 PEs and can achieve a high throughput, which makes it suitable for the real-time high resolution video processing. However, this architecture also has some demerits: (1) The search area data memory is partitioned into sixteen modules in column direction to realize the memory interleaving scheme. (2) In order to fully utilize the 2-D PE array, the search area memory should be further halved in row direction. Totally, there are thirty-two memory modules for the search area data. So many memory modules not only increase the hardware cost but also consume much power. Kim [10] provides another 2-D architecture based on the modified algorithm. Through applying preload registers and search data buffers, only sixteen memory modules are used and the datapath still can be fully utilized. However, in this design some data of search area are stored in the register array in datapath and others are stored in memory modules, which increases not only the datapath hardware cost but also the routing complexity during the back-end design.

One 1-D VBSME architecture is proposed in [11]. Compared with 2-D designs, 1-D architecture is more flexible. If the 1-D design is configured with $m \times 16$ PEs, it just requires $m + 1$ memory modules for its search area data storage. For the design in [11], m is one and then only two memory modules are needed. However, because each PE is responsible for one searching candidate, many partial SADs registers, multiplexers for the selection of partial SADs and comparators are required inside one PE. Consequently, the PE hardware efficiency of 1-D VBSME architecture is much lower than that of 2-D designs.

A parallel tree VBSME processor is provided in [12]. This architecture has high clock frequency and provides scalability for users. Its extension grain is one PE Group (PEG), which contains sixteen PEs and accounts for 20.94K gate. The main shortage of this design is that its hardware cost is larger than 2-D ones due to three reasons: (1) Each PEG has its own recent minimum SADs registers and relative update components; (2) Its pipeline arrangement is not optimized. Though this design also applies multi-cycle timing delay approach, this scheme is not fully utilized. For example, its 4×4 and 4×8 SADs generation logic are both in the one-cycle delay domain, which becomes the system bottleneck; (3) Its memory organization is not optimized and the memory IO utilization is only 48%.

In this paper, a fine-grain scalable VBSME architecture for H.264/AVC based on the parallel tree architecture in [12] is presented. This architecture has such features: (1) The PE number in this architecture is scalable and the extension grain is one PEG, which includes sixteen PEs and accounts for 8.5K gates. (2) The 3-stage pipeline structure and multi-cycle path delay scheme are adopted. Original critical paths, which are sensitive to the sub-tree extension, are moved to these multi-cycle constraint domains. Therefore, a very high clock frequency can be obtained. Because the multi-cycle path delay scheme eliminates many waste switches, the system power dissipation is also reduced. (3) Through memory optimization, only two memory modules are required for the search area data when sixteen PEGs are configured. Based on the same process technology (TSMC 0.18 μm 1P6M) and the same throughput requirement (200 MHz, 256PEs), our design could save 13.4% hardware cost and 5.7% power consumption compared with the 2-D structure [9].

The rest of the paper is organized as follows. Our VBSME architecture and its memory organization algorithm are proposed in Sect. 2. The experimental results, the performance analysis and the comparisons with previous works are presented in Sect. 3. Finally, conclusions are given in Sect. 4.

2. Hardware Architecture

2.1 Fine-Grain Scalable Hardware Architecture

ME unit is the most computation intensive part in H.264/AVC encoding. The key to attaining high ME performance lies on the following two aspects: (1) Increasing the system clock frequency, which means fining pipeline or applying more advanced process technology. (2) Widening the data bus, which means that more PEs are scheduled to work in parallel. In practice, the high performance of 2-D VBSME architecture in [9] comes from the wide data path of current MB. Current MB is stored in one 16×16 8-bit register array. In each cycle, all pixels in current MB take part in the SAD computation. In fact, the high performance also can be achieved through widening the data path of search area memories, and this is the key idea of the proposed VBSME architecture in this paper. In order to clarify the data flow of our design, first we label the forty-one blocks in one MB, as shown in Fig. 2.

The basic scalable unit of our architecture is one PEG, which includes sixteen PEs. In theory, if the search width is M pixels, arbitrary m PEGs can be scheduled to work in parallel and fully utilized, as long as m is a factor of M [12]. However, as to be discussed in Sect. 3, because our architecture has obvious advantages when m is not greater than sixteen and M is always a multiple of sixteen, we just focus on $m \in \{1, 2, 4, 8, 16\}$ in this paper. These m PEGs process m horizontal adjacent search positions in parallel. In every

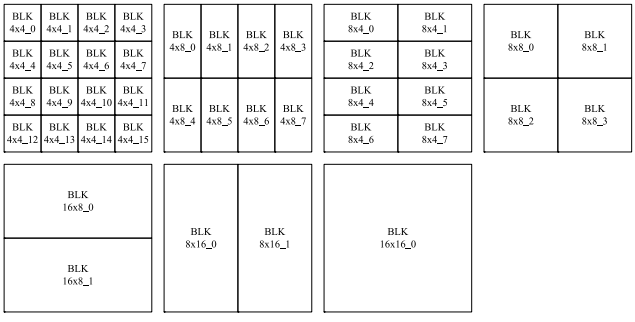


Fig. 2 41 blocks in one MB.

clock cycle, one row (16×1) pixels in current MB and the correspondent row (16×1) pixels in the candidate block are dispatched to one PEG. For instance, if m PEGs are installed, $(15+m)\times 1$ search area pixels and 16×1 current MB pixels are fetched from memories and broadcasted to these PEGs in every clock, as shown in Fig. 3. In this figure, the pixel at the left up corner of search area is denoted as origin and labeled as $R(0,0)$. The coordinate of the search candidate at the left up corner of search area is $(-M/2, -N/2)$. Black line squares in search area represent candidate blocks. Each PEG selects the corresponding 16×1 pixels from the broadcasting $(15+m)\times 1$ pixels as its input data. It takes one PEG sixteen cycles to complete the VBSME in one candidate.

To explain this architecture, a 16-PEG design with a search range of 48×32, $(M = 48, N = 32)$ is implemented. In order to simplify the data flow explanation, the pipeline latency is not taken into account. The data flow schedule is shown in Table 1. At cycle zero, one row pixels from search area, denoted as $\{R(0,0)$–$R(30,0)\}$, are fed into sixteen PEGs. $\{R(l,0)$–$R(l+15,0)\}$ pixels are dispatched to the lth PEG, where $l \in [0,15]$. At cycle zero, each PEG computes 4×1 partial SADs in the first row of its candidate block. At cycle one, pixels $\{R(0,1)$–$R(30,1)\}$ are broadcasted, so each PEG gets its second row 4×1 partial SADs. The same procedure continues. At cycle three, each PEG gets SADs of "BLK4 × 4_X"(X:0-3). As mentioned before, data reusing methodology is adopted in this architecture. "BLK4 × 4_0" SAD is combined with "BLK4 × 4_1" SAD to generate the SAD of "BLK8 × 4_0." In the same way, the SAD of "BLK8 × 4_1" is derived from "BLK4 × 4_2" and "BLK4 × 4_3" SADs. To calculate 4×8 SADs, "BLK4 × 4_X"(X:0-3) SADs are stored in temporary registers. At cycle seven, each PEG gets SADs of "BLK4 × 4_X"(X:4-7). Through the similar reusing scheme, the SADs of "BLK8 × 4_2" and "BLK8 × 4_3" can be obtained. Since "BLK4 × 4_X" (X:0-3) SADs have been saved in temporary registers, they are combined with "BLK4 × 4_X" (X:4-7) SADs to compute "BLK4×8_X" (X:0-3) SADs, "BLK8×8_X" (X:0-1) SADs and "BLK16 × 8_0" SAD. At the same cycle, 8×8 block SADs are stored in temporary registers. At cycle fifteen, these saved 8×8 block values are applied to calculate the 8×16 SADs and the 16×16 SAD. The calculation procedure for other blocks can be traced by analogy.

Our proposed design with sixteen PEGs can process sixteen successive search candidates in one row in parallel and the search area data can be shared horizontally. The generated SADs of these PE groups are compared through 16-input comparator trees to get the local minimum SADs and the associated MVs. Because these comparator trees could be shared, there are totally sixteen comparator trees for the forty-one blocks in MB, namely four 4×4 comparator trees, four 4×8 comparator trees, two 8×4 comparator

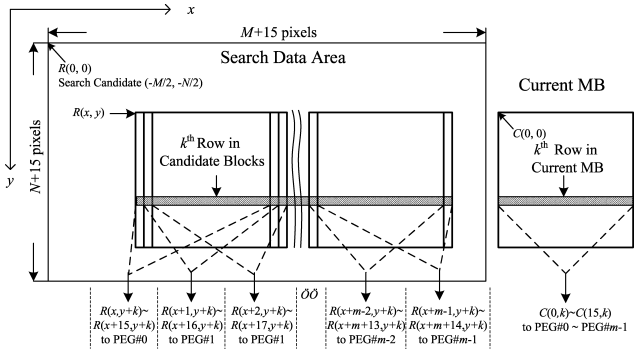


Fig. 3 Data flow of proposed architecture.

Table 1 Data flow schedule.

Cycle	PEG#0	PEG#1	...	PEG#14	PEG#15
0	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i,0)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+1,0)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+14,0)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+15,0)\rvert$
1	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i,1)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+1,1)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+14,1)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+15,1)\rvert$
...	...	...	...	...	...
14	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i,14)\rvert$	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i+1,14)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i+14,14)\rvert$	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i+15,14)\rvert$
15	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i,15)\rvert$	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i+1,15)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i+14,15)\rvert$	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i+15,15)\rvert$
16	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i,1)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+1,1)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+14,1)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+15,1)\rvert$
17	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i,2)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+1,2)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+14,2)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+15,2)\rvert$
...	...	...	...	...	...
496	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i,31)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+1,31)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+14,31)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+15,31)\rvert$
497	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i,32)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+1,32)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+14,32)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+15,32)\rvert$
...	...	...	...	...	...
510	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i,45)\rvert$	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i+1,45)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i+14,45)\rvert$	$\sum_{i=0}^{15} \lvert C(i,14)\text{-}R(i+15,45)\rvert$
511	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i,46)\rvert$	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i+1,46)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i+14,46)\rvert$	$\sum_{i=0}^{15} \lvert C(i,15)\text{-}R(i+15,46)\rvert$
512	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+16,0)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+17,0)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+30,0)\rvert$	$\sum_{i=0}^{15} \lvert C(i,0)\text{-}R(i+31,0)\rvert$
513	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+16,1)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+17,1)\rvert$	...	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+30,1)\rvert$	$\sum_{i=0}^{15} \lvert C(i,1)\text{-}R(i+31,1)\rvert$
...	...	...	...	...	...

$R(x,y)$ is the reference pixel in the search area, $C(x,y)$ is the current pixel in the current MB, where x is the horizontal index and y is the vertical index.

trees, two 8×8 comparator trees, two 8×16 comparator trees, one 16×8 comparator tree and one 16×16 comparator tree. To explain the work flow, four 4×4 comparator trees are used as illustration. In clock cycle three, these comparator trees are used by "BLK4 $\times$ 4_X" (X:0-3). At cycle seven, they are used by "BLK4 $\times$ 4_X" (X:4-7) and so on. The outputs from comparator trees are used to update the corresponding SADs values stored in recent minimum SADs registers. So at cycle fifteen, sixteen PEGs finish sixteen searching candidates in horizon, which are labeled as $\{(-24,-16),(-23,-16),\ldots,(-9,-16)\}$. At cycle sixteen, the search candidates are moved down one pixel in vertical. It takes another 16 cycles to calculate the distortions at candidates $\{(-24,-15),(-23,-15),\ldots,(-9,-15)\}$. If the clock period is denoted as T_{clk}, the architecture with sixteen PEGs can process $1/T_{clk}$ search candidates in one second.

However, the architecture without pipelining technique has following drawbacks: (1) Because of the long critical path delay, its clock frequency is low, so it can not get high performance. (2) Many invalid switches are incurred, therefore much power is wasted. For example, the comparator trees of 4×4 and 8×4 blocks are active in every cycle. Because the real SADs of these blocks are generated once in every four cycles, 75% switches are wasted. In order to increase the clock frequency and reduce the power consumption, the 3-stage pipeline architecture is proposed in this paper.

The detailed structure of one PEG is shown in Fig. 4. In "Stage1," one row of 16×1 pixels in current MB and one row of 16×1 pixels in the searching candidate are inputted. Four partial 4×1 SADs are calculated and accumulated. In every four cycles, four 4×4 block SADs are generated and propagated to the second stage. In "Stage2," 4×4 block SADs are combined or accumulated to generate 8×4 and 4×8 block SADs. 8×4 block SADs are obtained by combining two horizontal adjacent 4×4 block SADs. 4×8 SADs are derived by accumulating two vertical neighboring 4×4 SADs. These values are outputted and compared through comparator trees to get local minimum SADs among sixteen PEGs. These local minimum SADs are used to update the corresponding recent minimum SADs registers. It should be noticed that the inter stage registers between "Stage1" and "Stage2" are changed once in every four cycles, so the timing constrains of "Stage2" logic is 4-cycle timing delay. In every eight cycles, four 4×8 SADs are latched to "Stage3" to generate 8×8, 16×8, 8×16 and 16×16 SADs. Because 4×8 SADs are changed every eight cycles, the "Stage3" are in 8-cycle clock domain. It should be mentioned that the maximum adder tree level in "Stage3" is three and the maximum word width in this stage is sixteen bits. In contrast, the maximum adder tree level and word width in "Stage2" are one and thirteen bits respectively. These factors make "Stage3" have much longer path delay than "Stage2." However, the timing constraints in "Stage3" are 8-cycle delay ones and thus paths in "Stage3" will not become the system timing bottleneck.

The 3-stage pipeline top level block diagram is illustrated in Fig. 5. Each stage is indicated by the dash line polygon. "Design Stage1" is composed of sixteen PEG "Stage1" modules. "Design Stage2" includes PEG "Stage2" modules, 4×4 and 8×4 comparator trees, 4×4 and 8×4 recent minimum SADs registers and the associated update logic. PEG "Stage3" modules, 4×8, 8×8, 8×16, 16×8 and 16×16 comparator trees, the relative minimum SADs registers and the update logic belong to "Design Stage3." It is obvious that configuring more PEGs will increase the path delay of the comparator trees. However, through the multi-cycle time delay approach, all paths that are sensitive to the PEG number are arranged in loose timing constraint domains. This provides large timing margin for our design. For instance, when the target clock frequency is 200 MHz, the timing constraints for "Design Stage2" and "Design Stage3" are 20 ns and 40 ns, respectively. With 0.18 μm process, these timing constraints provide large timing margin to the 16-PEG design. With loose timing constraints, the synthesizer can choose low hardware cost and low power components to implement the logic in these stages. For instance, in our design, all adders in "Design Stage2" and "Design Stage3" are implemented with ripple-carry adders. The multi-cycle path scheme efficiently reduces the hardware cost and power consumption.

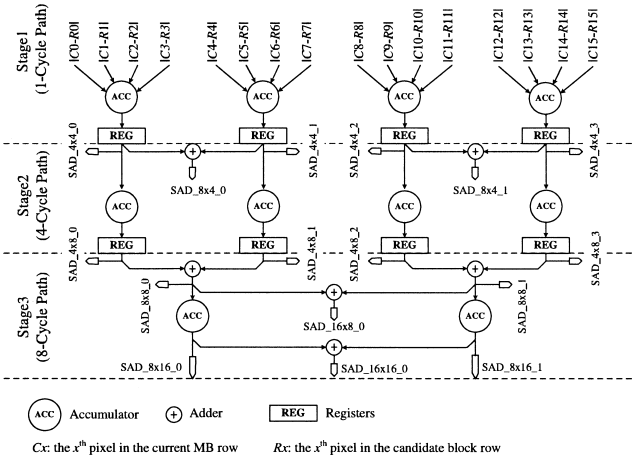


Fig. 4 3-stage pipeline PEG structure.

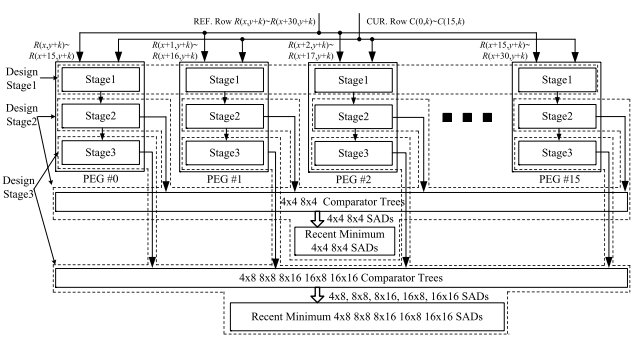


Fig. 5 3-stage pipeline VBSME architecture with 16 PEGs.

2.2 Search Area Memory Organization

The memories for the search area data consume non-trivial hardware cost and power dissipation. For instance, the memory modules in the 2-D design [9] occupy almost 50% die area. In fact, the memory organization is one critical issue for the system design, especially in H.264/AVC which applies multiple reference frames. In this section, one memory mapping algorithm is proposed to reduce the memory partition number in the proposed architecture. Consequently, the system hardware cost and the power consumption are both optimized.

The hardware cost of one memory is mainly decided by three factors: (1) Storage cell array; (2) Address decoder logic; (3) Peripheral logic in the form of sense amplifiers, prechargers and write drivers. The volume of the storage cell array depends on the search area size. Similar to reference [9], level C data [13] reuse scheme is applied in the search area data updating in our design. So, the storage cell array volume of our design is the same as the 2-D design. The hardware cost of cell array can not be saved by our method. However, our memory organization reduces the memory partition number, so the chip area consumed by the address decoder logic and the peripheral logic is saved.

The memory power consumption is mainly affected by two issues. The first one is the IO bandwidth, which decides the complexity and the power dissipation of the peripheral circuits. With Artisan Memory Compiler for TSMC 0.18 μm technology, we implement three kinds of single port memory with the same storage volume but different IO width. Their power statistics at 100 MHz are illustrated in Table 2. It is clearly illustrated that the power consumption almost increases in direct proportion to the IO bandwidth. One aim of the memory organization is to get the high IO utilization, so the power consumed by these IO circuits can be fully utilized. The second important factor is the address decoder logic. When the memory is partitioned, more power is consumed by the address decoder logic. More memory partitions also increase the die area cost. To demonstrate this point, we calculate the power and the area cost of a 4 KB memory as it is realized with one 4 KB (256 W × 128 b) memory, two 2 KB (256 W × 64 b) memories, four 1 KB (256 W × 32 b) memories and eight 0.5 KB (256 W × 16 b) memories. The results for the four implementations running at 100 MHz are shown in Table 3. Dividing the memory in column to eight smaller modules increases the power by 68.2% and the area by 60.6%. In fact, after the memory is partitioned, the interconnect congestion near the memory modules incurs more cost during the back-end design since input address and data busses are fanned out to multiple modules.

Therefore, fewer memory partitions and high IO utilization are two important issues for obtaining the reduction of the memory area and power consumption. The main problem of the design provided in reference [12] is that its memory IO utilization is very low. Its search area memory is composed of three partitions and each partition has 128-bit output. So its memory IO bandwidth is 384 bits. In each clock, 184 bits are selected from these outputs and processed. Namely, its IO utilization is just 48%. This causes much hardware cost and power dissipation waste. The 2-D architecture [9] has these two drawbacks. First, its search area data are stored in thirty-two memory partitions. Second, if the search height is N, the utilization of its upper sixteen memory partitions is only $16/(N+16)$.

Table 2 Memory power versus IO bandwidth.

Configuration	128 W × 32 b	64 W × 64 b	32 W × 128 b
Power(mw)	15.0	24.9	46.1

Table 3 Total area and active power for 4 KB memory.

Partitions	1	2	4	8
Area(mm^2)	0.317	0.345	0.400	0.509
Power(mw)	47.119	50.724	60.768	79.272

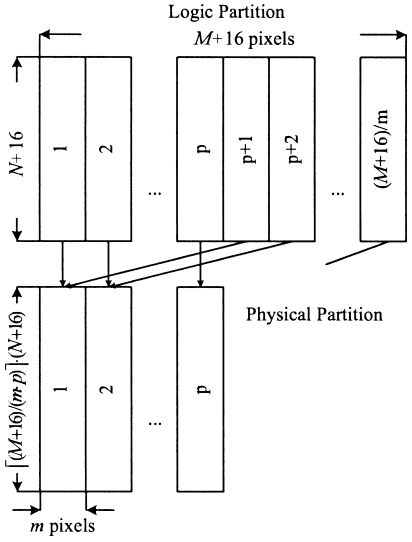
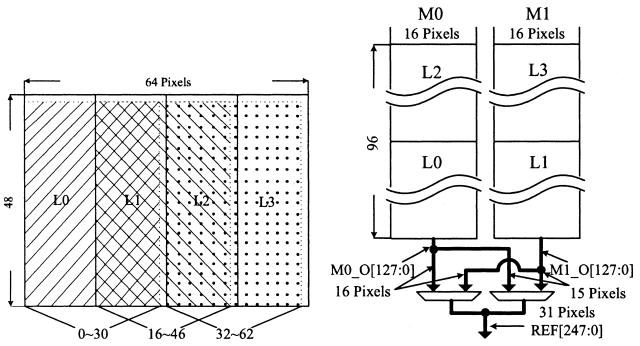


Fig. 6 Memory mapping algorithm.

For the m-PEG configuration, where $m \in \{1, 2, 4, 8, 16\}$, we use the following algorithm to organize search area memories. The search area is $(M+15)$-pixel wide and $(N+15)$-pixel high. In order to make physical implementation convenient, one pixel is extended in both vertical and horizontal directions, so the search area size is $(M+16)\times(N+16)$ pixels. The search area is divided into $(M+16)/m$ logic partitions and each logic partition is m-pixel wide, as illustrated in Fig. 6. There exists $p = \lceil (15+m)/m \rceil$ physical partitions and each partition is also m-pixel wide. The lth logical partition is mapped to the (l mod p)th physical partition and its begin address is $\lfloor l/p \rfloor \times (N+16)$. The depth of each physical memory partition is $\lceil (M+16)/(m \times p) \rceil \times (N+16)$.

For example, in our design, the search width is 48 and the search height is 32. So, the search area is 63×47 pixels. One pixel is extended in both vertical and horizontal directions. Thus, the memory capacity for the search area is 64×48 pixels. When sixteen PEGs are configured, the mem-



(a) Logic memory partitions. (b) Physical memory partitions.

Fig. 7 Proposed memory organization.

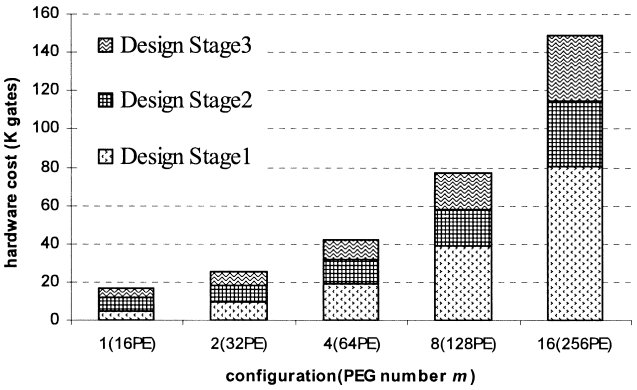


Fig. 8 Hardware cost versus PE group number.

ory is divided into four logic partitions, which are labeled as "L0," "L1," "L2" and "L3," as illustrated in Fig. 7(a). Each solid line rectangle represents one 16-pixel wide and 48 high logic partition. The ME processing includes three stages. In the first stage, the area covered by slash pattern is searched, so "L0" and "L1" are active. In the second stage, the sub search area is moved by 16-pixel in horizon. The rectangle with backslash pattern includes these searching candidates. "L1" and "L2" are active in this stage. In the last phase, "L2" and "L3" are used and the rectangle filled with dot represents this sub search area. The intuitive approach is implementing these logic partitions with four memory modules. Each module is 48 W × 128 b. However this method causes low memory IO utilization, which is only 48%. When the search width is increased, the memory utilization becomes evenv worse.

Based on the proposed mapping algorithm, just two memory modules are required for the search area data storage, as shown in Fig. 7(b). Each memory module is 16-pixel wide and 96 high. "L2" and "L0" are stacked up and mapped to "M0." "L3" and "L1" are mapped to "M1" in the same way. The output pixels dispatched to PEGs come from the outputs of "M0" and "M1." In the first search stage, the read pointers of "M0" and "M1" both begin from row zero. The sixteen most significant pixels (MSP) of "REF" come from "M0_O" and the rest fifteen least significant pixels (LSP) come from "M1_O[127:8]." In the second search stage, the read pointer of "M0" starts from row forty-eight, which is the start point of "L2" logic partition. The positions of "M0_O" and "M1_O" in "REF" are exchanged. Namely, The sixteen MSPs of "REF" come from "M1_O" and the rest fifteen LSPs come from "M0_O[127:8]." In the third stage, both read pointers are initialed to row forty-eight and the format of "REF" is the same as the first stage.

Based on the proposed memory mapping algorithm, the IO utilization can reach 96.9% and just two memory modules are required. Compared with the intuitive memory architecture, 41% hardware cost saving is obtained.

3. Experimental Results and Performance Analysis

As mentioned before, our VBSME architecture provides the advantages of high clock speed, fine configuration grain and fewer memory partitions. In this section, we will provide some experimental results to demonstrate these features.

The fine-grain scalability is an important advantage of our architecture. In order to verify the scalability of this architecture, five designs with m PEGs, where $m \in \{1, 2, 4, 8, 16\}$, are implemented. These designs are synthesized with Synopsys Design Compiler based on TSMC 0.18 μm 1P6M cell library. The timing target is 200 MHz in worst operating conditions (1.62 V, 125°C). The corresponding hardware cost statistics are shown in Fig. 8. The average extension grain of our architecture is 8.5K/PEG, which is just 40.7% of the extension grain of the architecture proposed in [12].

High clock speed is another important feature of our VBSME architecture. In high-end designs, the ME engine always works as one coprocessor, which cooperates with the high clock speed on-chip processor, such as ARM922T. Under 0.18 μm technology, ARM922T can work at 200 MHz [14]. The ME coprocessor is preferred to work at the same clock speed as the on-chip processor because of these reasons: (1) The throughput of ME coprocessor is in direct ratio to its clock speed. (2) If the ME coprocessor works at a different clock speed with the on-chip processor, the overhead of the data transfer between these two clock domains will be increased. (3) The multi-clock design incurs the complexity of back-end design. For example, during the clock synthesis stage, the latter synthesized clock will worsen the clock skew of previous ones. (4) Moreover, if the ME coprocessor and the on-chip processor work at two different asynchronous clock domains, special synchronizer circuits are required to resolve the metastability problem [15], which is a synchronization failure that occurs when a signal generated in one clock domain is sampled too close to the rising edge of another clock signal. These synchronizer circuits not only degrade the system performance but also increase the complexity of implementation and verification [16].

In order to evaluate the clock speed of our designs, we implement the back-end design of the 16-PEG architecture with 48 × 32 search range. Two 96 W × 128 b memories for the search area data storage cost 44.3K gates.

One 16 W × 128 b memory which is used to store the current MB costs about 16.6K gates. Other modules, which mainly include PEGs, comparator trees, minimum SADs and MVs registers and control logic, costs about 151.8K gates. In the floorplan of this design, in order to reduce the critical path delay, the source memory modules for "Design Stage1" is arranged at its bottom and top side. After placed and routed with SYNOPSYS Astro, the core area is 1.717 mm × 1.713 mm. The design layout is shown in Fig. 9. In typical operating conditions (1.8 V, 25°C), its maximum clock frequency is 261 MHz.

Because we adopt the multi-cycle path technique, when m PEGs are configured, the critical path delay is not increased and its work clock frequency keeps constant. Consequently the system throughput is increased in direct proportion to the PEG number. According to the system computation requirement, the number of PEG can be flexibly configured to satisfy different throughput requirements. It takes one m-PEG design $16/m$ cycles to fulfill the VBSME operation at one search candidate. If the clock period is T_{clk}, the corresponding time is $(16 \times T_{clk})/m$. For the real-time application, the required PEG number m is decided by the search candidate number S and the utilization ratio U, which can be expressed as Eq. (1). S is decided by the reference frame number (F_n), the frame rate (F), the frame size ($W \times H$) and the search range ($M \times N$), which is illustrated in Eq. (2). In order to save chip area, we do not use ping-pang mode or dual-port memories. Therefore, extra cycles are needed to load the current MB and search area data and thus the utilization ratio U is less than 100%.

Fig. 9 16-PEG design layout.

$$\frac{1/T_{clk} \times U}{16/m} \geq S \tag{1}$$

$$S = F \times F_n \times W \times H \times M \times N/256 \tag{2}$$

For example, in order to process QCIF (176×144) with four reference frames, 30 fps frame rate and 32 × 32 search range, if it is assumed that these four reference frames share the same single-port memory modules, the input IO bandwidth is 128-bit at 261 MHz and there is no stall during the memory refilling procedure. In each second, 721440 extra cycles are required for the data loading. The clock period T_{clk} of our design is 3.83 ns, so the maximum utilization ratio is expressed as Eq. (3). The minimum required PEG number is illustrated in Eq. (4). Therefore only one PEG can realize the real-time VBSME processing. One resolution to save the data transmission overhead is segmenting the memories to make each reference have its dedicated memories. In this way, the data loading can be processed in parallel with the VBSME. Of course, the overhead of this method is the increase in chip area. Without special explanations, it is assumed that the shared memory scheme is used in the following discussions. The numbers of PEG versus some typical applications are listed in Table 4.

$$U = \frac{261,000,000 - 721,440}{261,000,000} = 99.7\% \tag{3}$$

$$m \geq \frac{30 \times 4 \times 176 \times 144 \times 32 \times 32}{256} \cdot \frac{16}{260,278,560} = 0.748 \tag{4}$$

The performance comparisons between the proposed 16-PEG architecture and previous designs are listed in Table 5. Because these designs are implemented un-

Table 4 PEG number versus video application.

Frame Size	Search Range	Frame Rate	Reference Number	PEG Number
QCIF (176 × 144)	32 × 32	30 fps	4	1
CIF (352 × 288)	32 × 32	30 fps	4	4
525HHR (352 × 480)	32 × 32	30 fps	4	8
525SD (720 × 480)	48 × 32	30 fps	4	16

Table 5 Comparisons with previous designs.

	1-D Design [11]	2-D Design [9]	2-D Design [10]	Parallel-Tree [12]	This Work
PE Number	16	256	256	128	256
Search Range	16×16	48×32	64×64 32×32	32×32	48×32
Process Technology	TSMC 0.13 μm	TSMC 0.35 μm	TSMC 0.18 μm	TSMC 0.18 μm	TSMC 0.18 μm
Working Conditions	– post-synthesis	– post-layout	–	Typical post-layout	Typical post-layout
Max Frequency	294 MHz	66.7 MHz	100 MHz	228 MHz	261 MHz
Gate Count (SRAM excluded)	61K	105.6K	154K	180.6K	151.8K
Search Area SRAM Modules	2	32	16	3	2
PHR	77.1	161.7	166.2	161.6	440.2

der different processes and timing constraints, for efficient comparisons, a new definition, "performance hardware ratio"(*PHR*), is given in this paper. *PHR* can be expressed as Eq. (5). If the utilization is the same, the product of the processor element number and the clock frequency presents the performance capability. Higher *PHR* score represents higher hardware efficiency. Under the same process technology, *PHR* can accurately illustrate the hardware efficiency in datapath logic of a design.

$$PHR = (PE_n \times F_{clk})/G \quad (5)$$

Where PE_n is the number of PE; F_{clk} is the clock frequency and its unit is " MHz"; G is the datapath gate account and its unit is "K gates."

Among the designs listed in Table 5, all but the 2-D design in [9] apply the same or more advanced technology. Thus, *PHR* is an efficient criterion for the performance comparison. It is clearly illustrated that the provided architecture has much higher *PHR* score than others. This is mainly contributed by pipelining and multi-cycle path schemes.

We can see that 1-D design has the lowest *PHR*. Because each PE is responsible for a specific searching candidate, in each PE, the data buffers for partial SADs storage, the multiplexers for the selection of these partial SADs and the comparators for the minimum SADs update significantly degrade the hardware efficiency.

The data flow of our design is similar to the parallel-tree design in [12]. However, through the provided multi-cycle path and comparator tree approaches, the dedicated minimum SADs registers in each PEG are eliminated and the clock speed is enhanced by 14.5%. Consequently, the *PHR* of our design is 2.7 times as high as one of parallel-tree architecture. Moreover, our memory organization method can obtain more hardware and power saving.

Compared with the 2-D VBSME architecture [9], our architecture can provide more flexibility to users. For the application which needs no more than 256 PEs, our architecture can scale down its hardware according to the performance requirement. Fewer memory partitions is another important advantage of our design. However, because the 2-D design in reference [9] is implemented under different timing constraints and process technology, it is difficult to make fair comparisons. Based on the 2-D VBSME structure in [9], one design is implemented with TSMC 0.18 μm 1P6M technology. The timing constraint for synthesis is 200 MHz in worst operating conditions (1.62 V, 125°C). The post synthesis comparisons between these two designs are listed in Table 6. It can be seen that, because fewer memory modules are adopted in our proposed architecture, 13.4% hardware cost can be saved. We use SYNOPSYS Power Compiler to do the power simulation under the same test sequence and work conditions. 5.7% power consumption can be saved, which is also contributed by fewer memory partitions. For those multiple reference frames applications, because more memory modules are needed, our design provides better performance than the 2-D counterpart. However, the datapath of the 2-D architecture has 19% higher PHR than our design owning to its higher datapath efficiency. For applications, such as HD-720p, which need more than 256 PEs, the 2-D architecture is a better choice.

Table 6 Comparisons with 2-D architecture.

		2-D Design	This Work
Hardware Cost (gates)	CUR. SRAM	15.7K	16.6K
	REF. SRAM	102.3K	44.3K
	Datapath	127.7K	151.8K
	Total	245.7K	212.7K
Power Consumption(mw)		514	484

4. Conclusion

A hardware architecture for VBSME in H.264/AVC is proposed in this paper. By widening the data path from the search area memories, m PEGs are scheduled to process in parallel, where $m \in \{1, 2, 4, 8, 16\}$. The proposed architecture can successfully eliminate the hardware and power consumption caused by shifting registers which are required in systolic array architectures. Five designs with m PEGs, $m \in \{1, 2, 4, 8, 16\}$, are implemented with TSMC 0.18 μm 1P6M cell library. The average extension grain is 8.5K/PEG. This fine-grain scalability provides high flexibility to users to make tradeoff between the hardware cost and the performance. With pipelining and multi-cycle path delay schemes, this architecture can work at high clock frequency with low hardware cost. Based on the proposed memory mapping algorithm, the memory partitions for the search area can be effectively reduced. As results, the system hardware cost and power consumption are both saved.

We implemented the back-end design of the 16-PEG architecture with 48 × 32 search range to estimate its performance. The core area of the design is 1.717 mm × 1.713 mm. In typical work conditions, its maximum clock frequency is 261 MHz. Compared to the 2-D architecture with the same search range, 13.4% hardware cost and 5.7% power dissipation can be saved. The peak performance of the 16-PEG architecture is the real-time processing of 525SD resolution frame with 48×32 search range and four reference frames at 30 fps.

Acknowledgments

This work was supported by fund from the Japanese Ministry of ECSST via Kitakyushu knowledge-based cluster project.

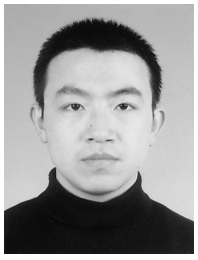
References

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity," IEEE Circuits Syst. Mag., vol.4, no.1, pp.7–28, First Quarter 2004.

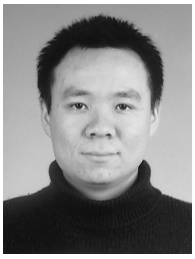
[2] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.7, pp.560–576, July 2003.

[3] T. Wiegand, G. Sullivan, and A. Luthra, Draft ITU-T recommendation and final draft international standard of joint video specification

(ITU-T Rec. H.264—ISO/IEC 14496-10 AVC), 2003.
[4] P.M. Kuhn, "Fast MPEG-4 motion estimation: Processor based and flexible VLSI implementations," J. VLSI Signal Processing, vol.23, no.1, pp.67–92, Oct. 1999.
[5] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," IEEE Trans. Circuits Syst., vol.36, no.10, pp.1301–1308, Oct. 1989.
[6] C.H. Chou and Y.C. Chen, "A VLSI architecture for real-time and flexible image template matching," IEEE Trans. Circuits Syst., vol.36, no.10, pp.1336–1342, Oct. 1989.
[7] S.S. Lin, P.C. Tseng, and L.G. Chen, "Low-power parallel tree architecture for full search block-matching motion estimation," ISCAS'04. Proc. 2004 International Symposium on Circuits and Systems, pp.313–316, May 2004.
[8] Y.S. Jehng, L.G. Chen, and T.D. Chiueh, "An efficient and simple VLSI tree architecture for motion estimation algorithms," IEEE Trans. Signal Process., vol.41, no.2, pp.889–900, Feb. 1993.
[9] Y.W. Huang, T.C. Wang, B.Y. Hsieh, and L.G. Chen, "Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264," ISCAS'03. Proc. 2003 International Symposium on Circuits and Systems, pp.796–799, May 2003.
[10] M. Kim, I. Hwang, and S.I. Chae, "A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264," Asia and South Pacific Design Automation Conference, 2005. Proc. ASP-DAC 2005, pp.631–634, Jan. 2005.
[11] S.Y. Yap and J.V. McCanny, "A VLSI architecture for variable block size video motion estimation," IEEE Trans. Circuits Syst. II, Express Briefs, vol.51, no.7, pp.384–389, July 2004.
[12] Y. Song, Z.Y. Liu, S. Goto, and T. Ikenaga, "Scalable VLSI architecture for variable block size integer motion estimation in H.264/AVC," IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.979–988, April 2006.
[13] J.C. Tuan, T.S. Chang, and C.W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," IEEE Trans. Circuits Syst. Video Technol., vol.12, no.1, pp.61–72, Jan. 2002.
[14] ARM Corp., ARM922T™ with AHB system-on-chip platform OS processor product overview, 2001.
[15] W.J. Dally and J.W. Poulton, Digital Systems Engineering, pp.462–470, Cambridge University Press, 1998.
[16] C.E. Cummings, "Synthesis and scripting techniques for designing multi-asynchronous clock designs," SNUG, 2001.



Zhenyu Liu received his B.E., M.E. and Ph.D. degrees in electronics engineering from Beijing Institute of Technology in 1996, 1999 and 2002, respectively. His doctor research focused on real time signal processing and relative ASIC design. From 2002 to 2004, he worked as post doctor in Tsinghua University of China, where his research mainly concentrated on embedded CPU architecture. Currently he is a researcher in Kitakyushu Foundation for the Advancement of Industry Science and Technology. His research interests include real time H.264 encoding algorithms and associated VLSI architecture.



Yang Song received the B.E. degree in Computer Science from Xi'an Jiaotong University, China in 2001 and M.E. degree in Computer Science from Tsinghua University, China in 2004. He is currently a Ph.D. candidate in Graduate School of Information, Production and Systems, Waseda University, Japan. His research interest includes video coding and associated very large scale integration (VLSI) architecture.

Takeshi Ikenaga received his B.E. and M.E. degrees in electrical engineering and the Ph.D. degree in information & computer science from Waseda University, Tokyo, Japan, in 1988, 1990, and 2002, respectively. He joined LSI Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 1990, where he has been undertaking research on the design and test methodologies for highperformance ASICs, a real-time MPEG2 encoder chip set, and a highly parallel LSI & system design for image-understanding processing. He is presently an associate professor in the system LSI field of the Graduate School of Information, Production and Systems, Waseda University. His current interests are application SoCs for image, security and network processing. Dr. Ikenaga is a member of the IPSJ and the IEEE. He received the IEICE Research Encouragement Award in 1992.

Satoshi Goto was born on January 3rd, 1945 in Hiroshima, Japan. He received the B.E. degree and the M.E. degree in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He also received the Dr. of Engineering from the same university in 1981. He is IEEE fellow, Member of Academy Engineering Society of Japan and professor of Waseda University. His research interests include LSI System and Multimedia System.