PAPER Special Section on VLSI Design and CAD Algorithms

A Sub-mW H.264 Baseline-Profile Motion Estimation Processor Core with a VLSI-Oriented Block Partitioning Strategy and SIMD/Systolic-Array Architecture

Junichi MIYAKOSHI[†a)], **Yuichiro MURACHI**[†], **Tetsuro MATSUNO**[††], ***Student Members***, **Masaki HAMAMOTO**[†], **Takahiro IINUMA**[†], **Tomokazu ISHIHARA**[†], **Hiroshi KAWAGUCHI**[†], ***Nonmembers***, **Masayuki MIYAMA**[††], ***and*** **Masahiko YOSHIMOTO**[†], ***Members***

SUMMARY We propose a sub-mW H.264 baseline-profile motion estimation processor for portable video applications. It features a VLSI-oriented block partitioning strategy and low-power SIMD/systolic-array datapath architecture, where the datapath can be switched between an SIMD and systolic array depending on processing flow. The processor supports all the seven kinds of block modes, and can handle three reference frames for a CIF (352 × 288) 30-fps to QCIF (176 × 144) 15-fps sequences with a quarter-pixel accuracy. It integrates 3.3 million transistors, and occupies 2.8×3.1 mm^2 in a 130-nm CMOS technology. The proposed processor achieves a power of 800 μW in a QCIF 15-fps sequence with one reference picture.

key words: *low power, motion estimation, H.264, SIMD, systolic array*

1. Introduction

H.264/AVC (hereafter, H.264) [1] provides two times higher coding efficiency than the previous standards such as H.263 and MPEG-4, while a computational cost of H.264 reaches a dozen-fold workload of the previous standards. This is because motion estimation in H.264 employs seven kinds of motion compensations whose block sizes range from 16×16 to 4 × 4 pixels with a quarter-pel motion vector accuracy. Therefore, the H.264 algorithm requires complicated hardware, which causes power increase.

Many algorithms have been presented to save a workload and power in H.264. Adaptive search algorithms reduce an average workload by early-termination methods. However, they can not save the worst-case workload [2], [3]. Besides, they have unnecessary idle cycles due to branch processes if implemented in hardware. An algorithm named UMHexagonS (hybrid unsymmetrical-cross multihexagon-grid search) [4] is an eminent ME algorithm reducing an average workload, and thus adopted in the H.264 joint model (JM) [5]. The UMHexagonS is, however, not suitable for hardware implementation because it performs adaptive motion searches that can not carry out a next step until a current step is completed by branch processes. This implies that a pipelined architecture can not be simply implemented. Consequently, the UMHexagonS algorithm causes increase of operating frequency and power in hardware. Another algorithm adopts the sub-sampling full search [6] that is a kind of exhaustive search methods, which can operate without any idle cycles and search the seven kinds of modes at the same time. However, the sub-sampling full search requires a larger workload than the UMHexagonS in the worst case, and needs to access all pixels in a search window, which raises power.

The conventional architecture with the adaptive search implemented is comprised of a parallelized SIMD or systolic-array (SA) datapath [6]–[8]. The SIMD architecture is suitable for a random block-matching because it has a wide bandwidth. On the other hand, the SA architecture is appropriate for a serial block-matching since it has a high reusability of pixels. However, this implies that the SA architecture can not run many data through its datapath, and thus it have to operate at a high frequency. In any case, these conventional configurations cause power increase due to the wide bandwidth or high frequency.

This paper describes a low-power motion estimation processor for the H.264 baseline profile with a VLSI-oriented block partitioning algorithm and SIMD/systolic-array architecture. The worst-case workload of the proposed partitioning algorithm is reduced to 4.5% of that of a full-search method while maintaining high picture quality. This algorithm is explained in Sect. 2. The datapath in the SIMD/systolic-array architecture can be switched between an SIMD and systolic array depending on processing flow, which is mentioned in Sect. 3. Chip implementation and performance comparison are described in Sect. 4. Section 5 summarizes this paper.

2. VLSI-Oriented Algorithm

The simulation conditions in this paper are as follows (Table 1). Modes used on the simulation are Mode 1 (16 × 16), Mode 2 (16 × 8), Mode 3 (8 × 16), Mode 4 (8 × 8), Mode 5 (8 × 4), Mode 6 (4 × 8) and Mode 7 (4 × 4).

Manuscript received March 10, 2006.
Manuscript revised June 13, 2006.
Final manuscript received August 1, 2006.

[†]The authors are with the Graduate School of Science and Technology, Kobe University, Kobe-shi, 657-8501 Japan.
[††]The authors are with the Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa-shi, 920-1192 Japan.
a) E-mail: mjun1@cs28.cs.kobe-u.ac.jp
DOI: 10.1093/ietfec/e89–a.12.3623

Table 1 Simulation conditions.

Profile	Baseline profile
Resolution	CIF (352 × 288 pixels)
Frame rate	30 fps
QP	16, 20, 24, 28, 32, 36, 40
# reference pictures	3
Others	No R-D optimization No rate control
Sequence	Foreman, Bus, Susie, Mobile & Calendar, Flower Garden, Akiyo
# frames	150
Search range	±24 × ±16

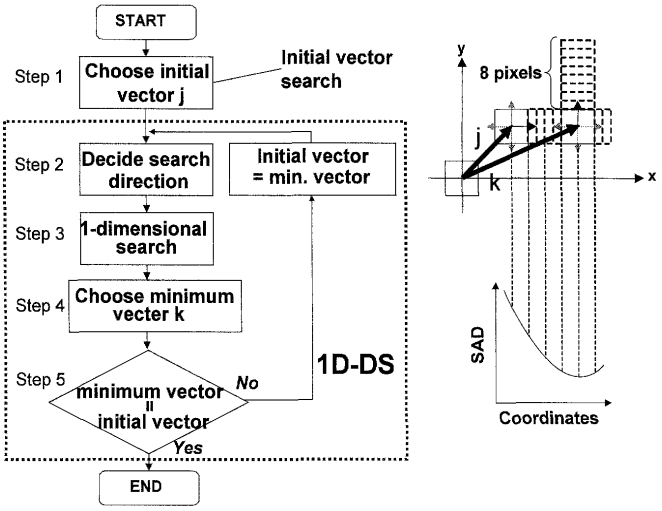


Fig. 1 Flow chart of integer-pel motion search.

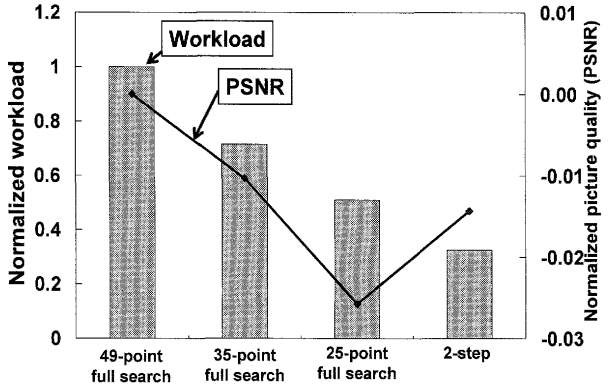


Fig. 2 Normalized worst-case workload and PSNR of sub-pel algorithms.

2.1 Block-Matching Methods in Integer- and Sub-Pel Motion Search

2.1.1 1D-DS for Integer-Pel Motion Search

The one-dimensional diamond search (1D-DS) [7] for an integer-pel motion search is a kind of gradient methods, whose flowchart is shown in Fig. 1. Step 1 is an initial vector search where an initial motion vector is chosen from a predicted vector, 0-vector, a vector of a macroblock (MB) with the same position in a previous frame and four vectors of the neighboring MBs. The initial vector search is to calculate the seven block-matchings located in random coordinate. At Step 2, a search direction is selected out of four points (diamond shape) surrounding the initial vector, according to the minimum sum of absolute difference (SAD). Then, at Step 3, a one-dimensional search is carried out toward the search direction, where eight SADs are computed at eight pixels and the vector that has the smallest SAD is selected as the minimum vector. Next, the minimum vector is compared with the initial vector. If the vectors are not equal, the initial vector is reset to the minimum vector, and the 1D-DS are iterated. Alternatively, if it is equal, the 1D-DS is terminated. The motion vector which indicates the smallest SAD is obtained in this search.

In the 1D-DS, the initial vector is properly determined in Step 1. Also the number of search points in Step 2 and the number of iterations in Step 5 are each restricted to enough value to keep video quality [7]. In this case, the number of search points is eight and the number of iterations is two. This makes the number of block-matching small allowing a low worst-case workload.

Although the 1D-DS is a kind of adaptive search, it includes only three branch processes in Steps 2, 4 and 5 as shown Fig. 1. Step 2 has a branch process which decides a search direction after searching all the four points. Steps 4 and 5 also have branch processes after the minimum SAD detections. On the other hand, the UMHexagonS includes seven branch processes (initial vector search, four-point search, un-symmetrical crossing search, full search, six-point search, hexagon search, diamond search) [4]. As a result of the simulation, it is confirmed that the 1D-DS reduces the number of branch processes to 51% compared to the UMHexagonS. By reducing the number of branch processes, idle cycles which occur as a result of a pipeline stall caused by the branch process decrease. In other words, the 1D-DS is suitable for VLSI implementation, since it can reduce the worst-case workload, and operates at lower frequency on hardware by less branch processes.

2.1.2 35-Point Full Search for Quarter-Pel Motion Search

The 35-point full search method is adopted to obtain a motion vector of a quarter-pel accuracy. This method performs a full search in a narrow search area (±0.5 × ±0.75) whose center is a resultant vector in the 1D-DS integer-pel search. The workload for full search is kept low by restricting the search area. Generally, a full search can realize simultaneous matching calculations for all the seven kinds of modes, because it can reuse SAD or SATD of results of one mode. In other words, the seven motion vectors in the seven modes can be obtained by the one-time execution, which denotes that an operating frequency can be lowered in hardware. Furthermore, a full search is easily implemented in hardware due to the simple processing flow. Figure 2 shows normalized workload and PSNR of sub-pel algorithms. The

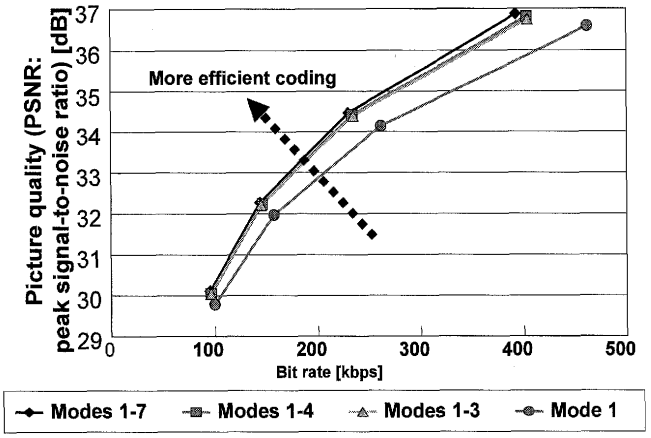


Fig. 3 R-D curves for variable block sizes.

2-step search which has been implemented in [5] is put on Fig. 2 for the comparison. We chose the 35-point full search method as sub-pel algorithm examining trade-off between picture quality and workload.

2.2 Block Partitioning Strategy

It is important to choose the optimum mode and its optimum vector as early as possible for low computation cost. In this subsection, the fast block mode decision algorithms for the integer-pel search method (1D-DS) and sub-pel search method (35-point full search) are proposed.

Figure 3 illustrates rate-distortion (R-D) curves that indicate coding efficiency in several sets of modes, obtained by a simulation with the JM encoder [5]. The R-D curves show that coding efficiency is improved more as the number of modes to be utilized increases. The figure also indicates that the efficiency difference between Mode 1 and other is large, and thus a set of Modes 1–3 is a good design choice.

To the set of Modes 1–3, we apply the 1D-DS method as an integer-pel motion search. If it was applied to all the seven modes, hardware would become complicated and result in large power. To the contrary, the 35-point full search is carried out in a set of Modes 4–7 with a sub-pel accuracy, in order to enhance picture quality. The 35-point full search for smaller blocks (Modes 4–7) can implemented at a low frequency and low power, which is discussed in Sect. 3.

2.2.1 Fast Search for Larger Blocks (FSLB) with Integer-Pel Accuracy

The four rectangles in Fig. 4 indicate block segmentations in Mode 2 and Mode 3. At first, the integer-pel motion search finds four motion vectors (MVs) in the four block using the 1D-DS method (top Block A in Mode 2 (MV_A), bottom Block B in Mode 2 (MV_B), left Block C in Mode 3 (MV_C), and right Block D in Mode 3 (MV_D). Then, the MV in Mode 1 (MV_{Mode1}) should be founded. The fast search for larger blocks (FSLB) is the proposed algorithm for an integer-pel motion search, which supposes that the MV_{Mode1} is inside the quadrilateral pointed by MV_A, MV_B, MV_C, and

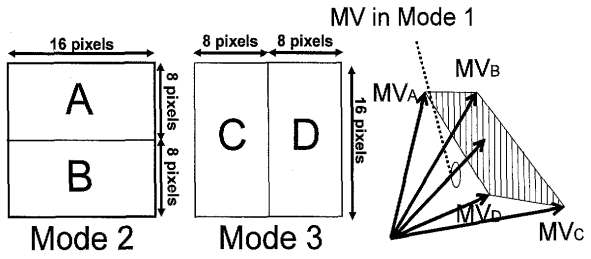


Fig. 4 Optimum MV for mode 1 by FSLB.

MV_D, as depicted in Fig. 4.

In concrete terms, MV_{Mode1} is selected from the following set of eight candidate MVs (CMVs), which means that the integer-pel motion search in Mode 1 is carried out only around the eight points.

$$\begin{cases} CMV_1 = PMV \\ CMV_2 = MV_A \\ CMV_3 = MV_B \\ CMV_4 = MV_C \\ CMV_5 = MV_D \\ CMV_6 = (MV_A + MV_B)/2 \\ CMV_7 = (MV_C + MV_D)/2 \\ CMV_8 = (MV_A + MV_B + MV_C + MV_D)/4 \end{cases}$$

PMV in the first equation indicates a predicted MV. The latter three equations are derivations of the four MVs in Modes 2 and 3. Therefore, MV_{Mode1} sometimes happen to be equal to either of MV_A, MV_B, MV_C, or MV_D, in which case, the sub-pel motion search explained in the following is not carried out around MV_{Mode1}. As a result the worst-case workload can be reduced.

2.2.2 Fast Search for Smaller Blocks (FSSB) with Sub-pel Accuracy

Next, we proceed to the sub-pel motion search with the results of the FSLB. That is, the 35-point full searches are carried out around the five integer-pel MVs in Modes 1–3 (MV_A, MV_B, MV_C, MV_D, MV_{Mode1}) shown in Fig. 4. The salient feature in the proposed algorithm for the sub-pel motion search (FSSB: fast search for smaller blocks) is that the sub-pel motion search even in Modes 4–7 is simultaneously performed during the motion search in Modes 1–3. This implies that there is no additional computation cost to obtain the sub-pel MVs in Modes 4–7 by reusing SATDs in Modes 1–3.

In the FSSB, a sum of absolute transformed difference (SATD) in Modes 1–3 is accumulated in every 4 × 4-pixel block. Each SATD is reused in a motion-vector calculation in Modes 4–7. The procedure in the FSSB to obtain the minimum SATD has five steps as follows.

1) Sub-pel ME in Mode 1

In Blocks E, F, G, and H in Fig. 5, an SATD is obtained every 4 × 4-pixel block. The 8 × 8-pixel SATDs in Mode 4 are obtained by accumulating four the 4 × 4-pixel SATDs. In the same way, The SATDs in Mode 4–7 are obtained by reusing

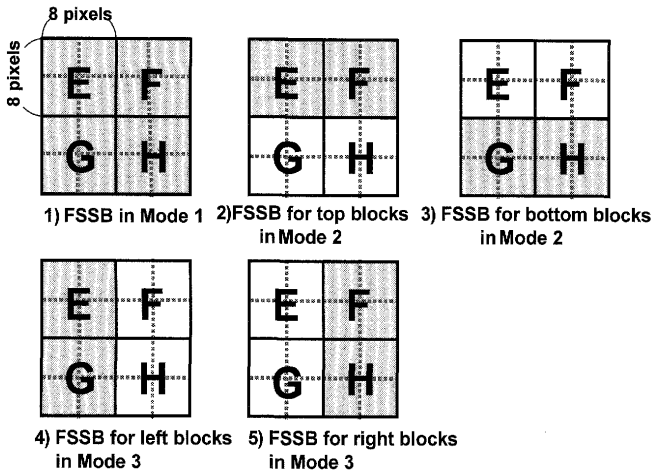


Fig. 5 Sub-pel ME procedure to obtain SATDs in Modes 1–7.

the 4 × 4-pixel SATDs. The hardware structure achieving this feature is described in the next section. Once the accumulation is done, the SATD in Mode 1 is eventually gained. Note that the SATDs in Mode 4–7 in all the steps are updated anytime if a smaller value is found in another CMV calculation.

2) Sub-pel ME for the top blocks in Mode 2 (Block A in Fig. 4)

In Blocks E and F, SATDs in Mode 4–7 are obtained as well. The SATD for the top blocks in Mode 2 is gained once the accumulation in Blocks E and F is done.

3) Sub-pel ME for the bottom blocks in Mode 2 (Block B in Fig. 4)

In the block G and H, calculation and accumulation similar to the previous step are carried out.

4) Sub-pel ME for the left blocks in Mode 3 (Block C in Fig. 4)

In Blocks E and G, SATDs in Mode 4–7 are obtained as well. The SATD for the left blocks in Mode 3 is gained once the accumulation in Blocks E and G is done.

5) Sub-pel ME for the right block in Mode 3 (Block D in Fig. 4)

In the block F and H, calculation and accumulation similar to the previous step are carried out.

Finally, the minimum SATDs in Modes 1–7 are gained as sub-pel MVs. Consequently, picture quality is enhanced, compared to the case that only Modes 1–3 are utilized.

2.3 Flowchart

Again, we explain the flowchart of the proposed block-partitioning algorithm with Fig. 6. The 1D-DS produces the four MVs in Modes 2–3 with an integer-pel accuracy. Then, the eight block-matching for Mode 1 is carried out to obtain MV_{Mode1}. These two steps are included in the FSLB. Next in the FSSB, the 35-point full searches are made using the resultant MVs, which computes sub-pel MVs in all the seven modes.

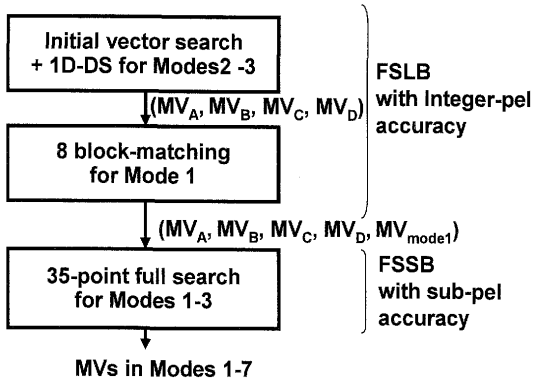


Fig. 6 Flowchart of proposed algorithm.

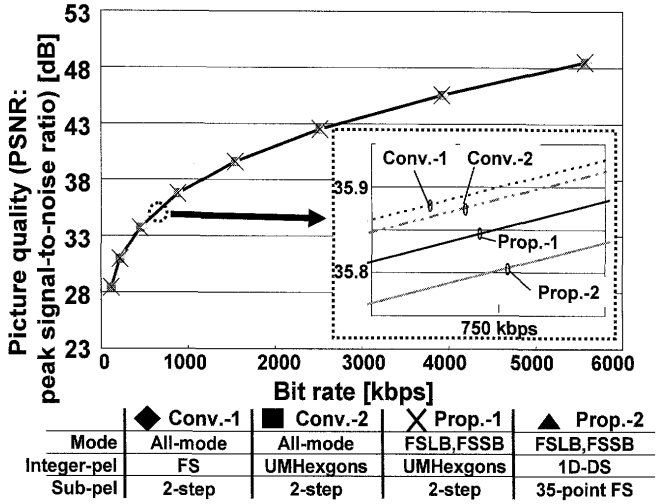


Fig. 7 R-D curves in the conventional and proposed algorithms.

2.4 Simulation Result

We compared the proposed algorithms with the conventional algorithms that have been adopted in the JM. Proposed-1 indicates UMHexagonS and 2-step search algorithm with FSLB and FSSB. Proposed-2 indicates 1D-DS and 35-point search algorithm with FSLB and FSSB.

The R-D curves in Fig. 7 indicate that, in a low-bitrate region, the picture quality degradation in the proposed algorithm is within 0.10 dB compared with Conventional-1. On the other hand in a high-bitrate region, the picture qualities are almost equal.

As shown in Fig. 8, the worst-case workload in Proposed-1 with FSLB and FSSB is 10.5% of that of Conventional-1. Furthermore, by using the 1D-DS method, the worst-case workload of Proposed-2 is lowered to 4.5% of the Conventional-1. Although the workload for the sub-pel search is increased by use of the 35-points full search, the total worst-case workload in the Proposed-2 is decreased due to the reduction of workload for the integer-pel search by use of the 1D-DS, as shown in Fig. 8. Here, these average workloads are calculated from an average of seven sequences with 150 frames each, in Table 1.

The normalized operating frequencies of the conven-

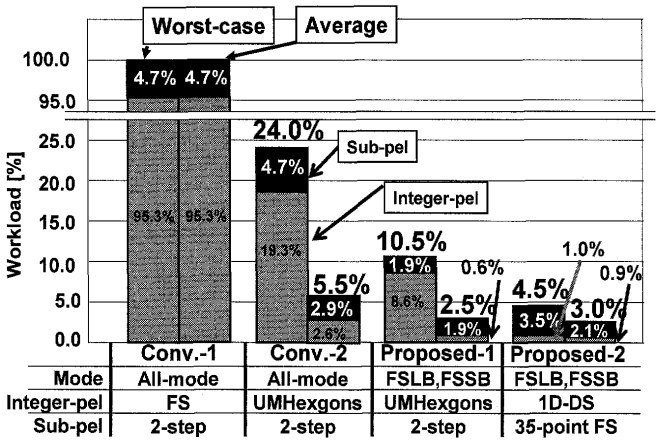


Fig. 8 Workload comparison.

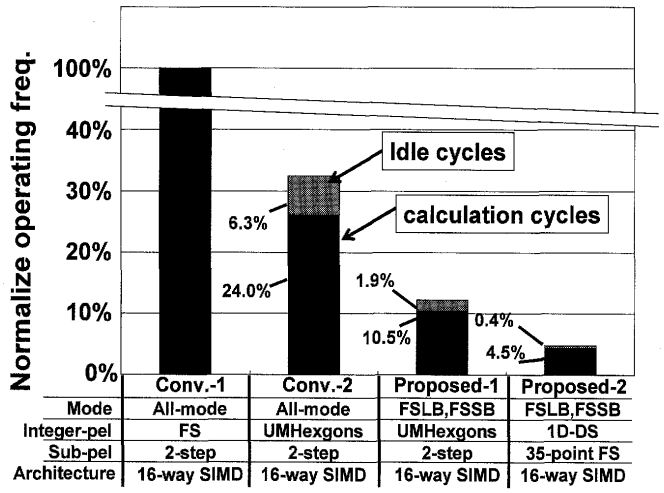


Fig. 9 Comparison of operating frequency assuming conventional 16-way SIMD.

tional and proposed algorithms are shown in Fig. 9 when they are implemented with the SIMD architecture [7]. The normalized frequency in the adaptive algorithm is larger than the worst-case workload by a grayed portion due to idle cycles (compared to Fig. 8). The idle-cycle overheads in the proposed algorithms are 1.9% and 0.4%, respectively, which are much smaller than that of Conventional-2 (6.3%). This is caused by reduction of branch processes as well as workload, as descried in Sect. 2.1.1. Namely, the proposed method reduces idle cycles as well as calculation cycles, so that it lowers the operating frequency allowing low-power processor realization with picture quality maintained.

3. SIMD/Systolic-Array Architecture

The architecture of an H.264 motion estimation processor (ME) to which the proposed algorithm is implemented is illustrated in Fig. 10. An integer-pel motion estimation (IME) processor performs the initial vector search and the 1D-DS method with an integer accuracy. The Sub-pel motion estimation (SME) processor executes the 35-point full search with a quarter-pel accuracy. Two three-port (two read ports + one write port) SRAMs are also implemented as image data caches. One is for reference pictures (SWRAM: search window RAM), and the other is for a current picture (TBRAM: template block RAM). The IME and SME processors share these two SRAMs. By adaptively switching the read ports between the two processors, they can operate in parallel. The proposed processors are connected to an external CPU bus (32 bits) and memory bus (64 bits) through a controller.

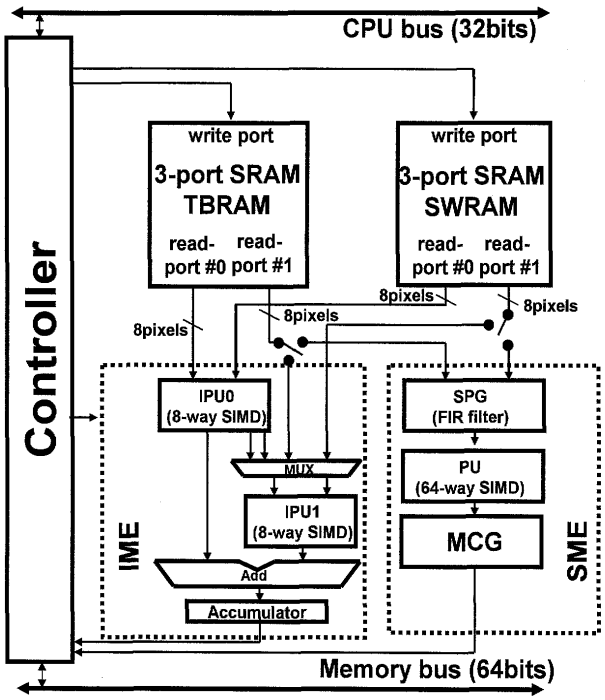


Fig. 10 A block diagram of the proposed ME.

3.1 IME Architecture

3.1.1 Structure of IME Processor

The SIMD architectures [6] have been presented as a parallel datapath as well as parallelized systolic-array (SA) architectures [7], [8] in the conventional motion estimation processors. The conventional SIMD does not need idle cycles to transfer data from an SRAM, since all the processor elements in the SIMD can directly get data from an SRAM. It, however, requires wide internal memory bandwidth, in otherwise, more pixels from an SRAM. The conventional SA has data-transfer-paths between the processor elements, so that the memory bandwidth can be reduced, however, its operating cycles increase. This is because the idle cycle occurs to fill all the processor elements with data. In order to achieve low power characteristics, we developed a novel architecture which can adaptively change datapath configuration to either SIMD or SA according to processing flow of integer-pel search, as shown in Table 2. The proposed architecture named SIMD/SA works as a low operating-cycle SIMD configuration during the initial vector search which is a kind of random block-matching, while it changes its datapath to a little memory-access SA during 1D-DS which is a

Table 2 Datapath configuration of SIMD/SA corresponding to processing flows of integer-pel search.

Processing flow of integer-pel search	Datapath configuration
Initial vector search (including Mode 1 search of FSLB)	SIMD
1D-DS	SA

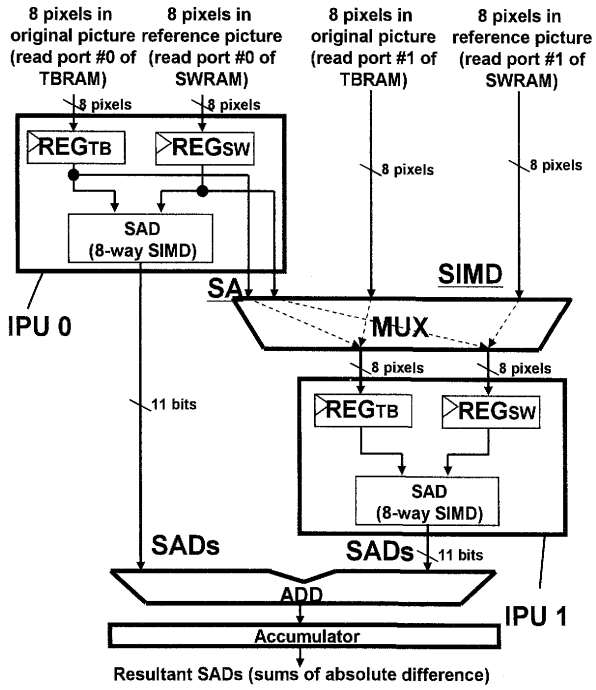


Fig. 11 IME (SIMD/SA) block diagram.

kind of a serial block-matching.

Figure 11 shows the proposed IME architecture consisting of the SIMD/SA. The SIMD/SA is comprised of two IPUs (integer-pel processing units) including an eight-way SIMD module. The MUX in the IME switches datapath connection between the IPU and SRAMs to reconfigure its composition to either the SA or SIMD according to the processing flow in Table 2. Thus, there are two kinds of datapaths as follows.

1) 16-way SIMD datapath
The MUX makes a connection between IPU1 and SRAMs. Both IPU 0 and IPU 1 are connected to the read ports of both TBRAM and SWRAM, and thus both operate in parallel. This structure is utilized for the initial vector search (random block-matching).

2) SA datapath
The MUX makes a connection between IPU0 and IPU1. The IPU 0 forwards TBRAM data and SWRAM data to the IPU 1. The 1D-DS method (serial block-matching) are performed in both IPU 0 and IPU 1 as SA datapath.

3.1.2 1D-DS Implementation with SA

Figure 12 illustrates a template block and reference blocks

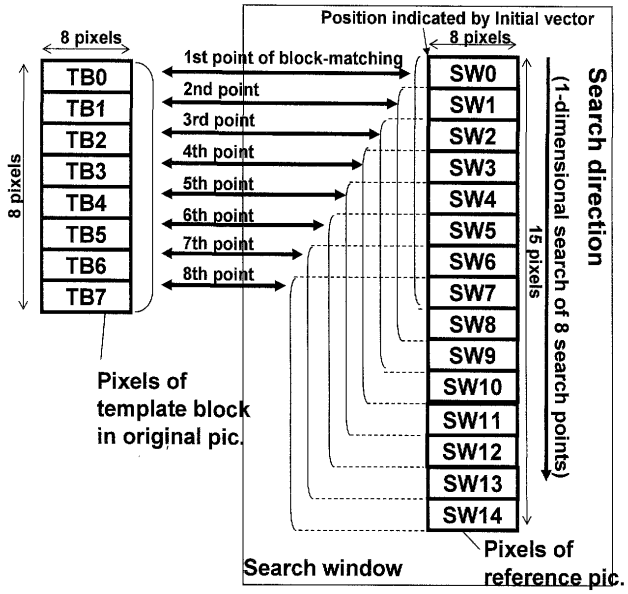


Fig. 12 A template block and reference blocks utilized for the 1D-DS execution.

utilized for the 1D-DS execution. The number of search points is eight as am example. TBN (TB0, TB1, ...) represents eight pixels in the N-th row in the TBRAM (original picture). SWN (SW1, SW2, ..., SW12) indicates eight pixels in the N-th row in the SWRAM (reference picture). The data used to calculate SAD are TB0 to TB7 and SW0 to SW7 in the first block-matching in the one-dimensional search (Step 3 in Fig. 1), and in the second block-matching, TB0-7 and SW1-8 are utilized. In this manner, 1D-DS proceeds toward the search direction.

The 1D-DS computation flow in the SA configuration is illustrated in Fig. 13. TBN (TB0, TB1, ...) are stored in a register, $\mathrm{REG_{TB}}$. SWN (SW1, SW2, ..., SW8) are stored in $\mathrm{REG_{SW}}$ as well. An SAD can be obtained by calculations between $\mathrm{REG_{TB}}$ and $\mathrm{REG_{SW}}$. Note that, as already shown in Fig. 11, data in the IPU 0 are forwarded to the IPU 1 in the SA mode, and thus data in $\mathrm{REG_{SW}}$ (IPU 0) is transferred to $\mathrm{REG_{SW}}$ (IPU 1) in the next cycle.

As shown in Fig. 12, in the first cycle, data TB0 and SW0 are transferred to the IPU0 from TBRAM and SWRAM, respectively. In the second cycle, data TB1 and SW1 are fed into IPU0 from the above SRAMs, and at the same time, data TB0 and SW0 are transferred to the IPU 1 from IPU0. Thus the 1st point of block matching is started and a partial SAD using 2×8 pixels is obtained in the second cycle. In the same manner, the 2nd point of block matching is initiated using data TB1 and SW2 in IPU0 and data TB0 and SW1 in IPU1 in the third cycle. Each block matching is begun one after another. In the 11th cycles, the second partial SAD in the 1st block mathing is calculated and accumulated. Repeating the above-mentioned operation, the 1-dimensional search of 8 search points is accomplished. Reference pixel data which are read from SWRAM can be reduced. Furthermore, current pixel data in TBRAM are read out only once in one dimensional search. There-

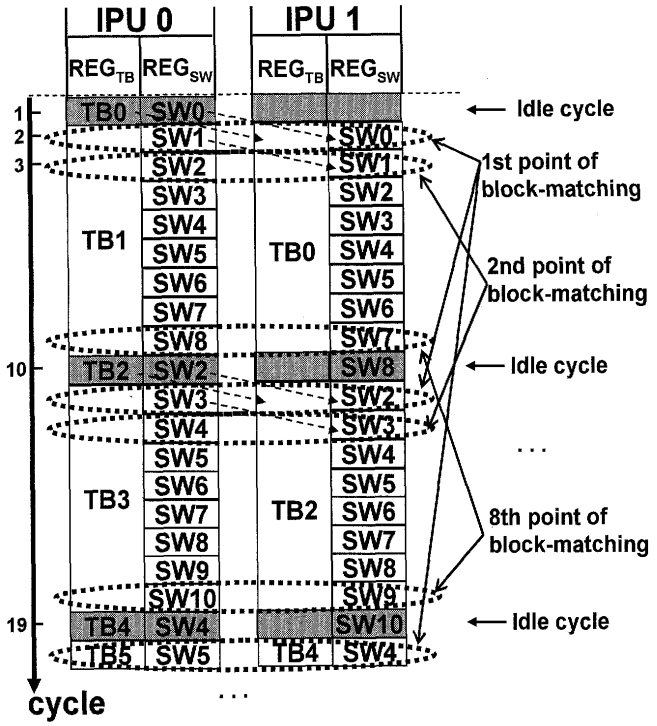


Fig. 13 1D-DS Computing flow in SA mode.

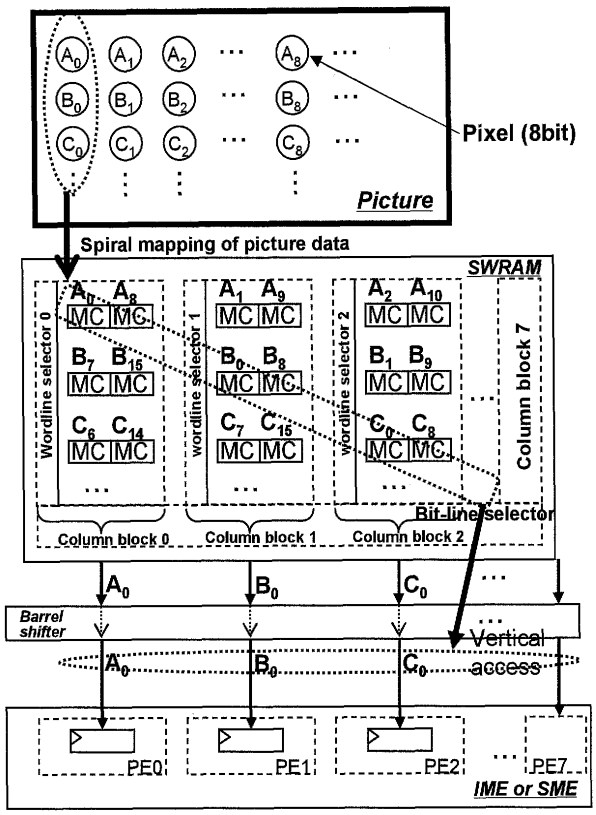


Fig. 14 An access method for port 1 of the SRAM. (accessing A0 to I0 pixels on a vertical line)

fore, proposed SIMD/SA can lower pixel data accessed from SWRAM and TBRAM.

Eight pixel data should be packed and fed into REG_{TB}s and REG_{SW}s properly, depending on a search direction in the 1D-DS process. If the search direction is horizontal, TBN and SWN include pixels in N-th column while they include pixels in N-th row if vertical. Here, RAMs which can read out successive pixel data either on horizontal line or on vertical line at a cycle are required as SWRAM and TBRAM in order to transfer pixels data to the SIMD/SA. The custom SRAM was tailored for search window buffer and template buffer [9]. Figure 14 shows configuration of the RAM and data mapping. Pixels data on vertical line in a picture are spirally mapped to memory plane to avoid the access conflict which is caused by storing vertical pixels data in the same column block, as shown in Fig. 14. Therefore, any successive eight pixels on horizontal line or on vertical line can be read out at a cycle.

3.1.3 Effects of the Adaptive SIMD/SA Switching

A comparison of operating cycles and the number of required pixels among three types of IME architectures is summarized in Fig. 15. The cycle counts to execute the initial vector search with the SIMD/SA architecture are as low as that with the SIMD architecture and 53% of that with the SA architecture. Furthermore, the pixels counts required to perform the 1D-DS with the proposed SIMD/SA are as low as that with the SA, and 40% of that with the SIMD. Consequently, in case of the proposed, the total cycle count is decreased by 25% compared with the SA architecture, and the required pixel count is reduced by 66% compared with the

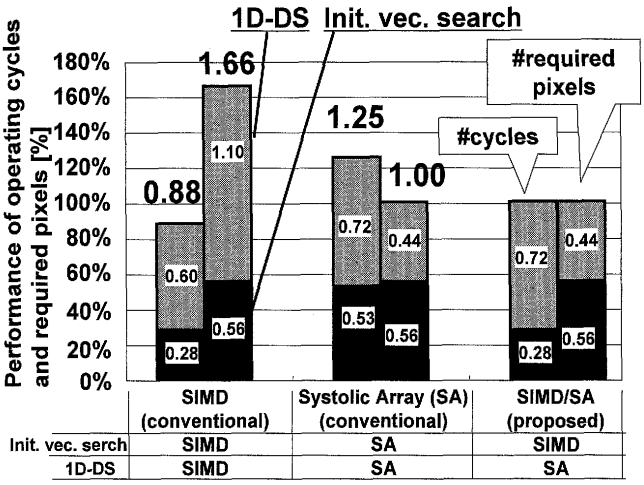


Fig. 15 Performance comparison in operation cycle and the number of required pixels in integer-pel search with FSLB among three kinds of IME architectures.

SIMD. Therefore the adaptive switching between the SIMD and the SA can reduce both operating cycle counts and required pixel counts. It lowers the power consumption for both logic and memory circuits.

3.2 SME Architecture

As illustrated in Fig. 10, the SME consists of an SPG (subpel generator), PU (processor unit), and MCG (motion cost generator). The SPG has an FIR-filter which generates pix-

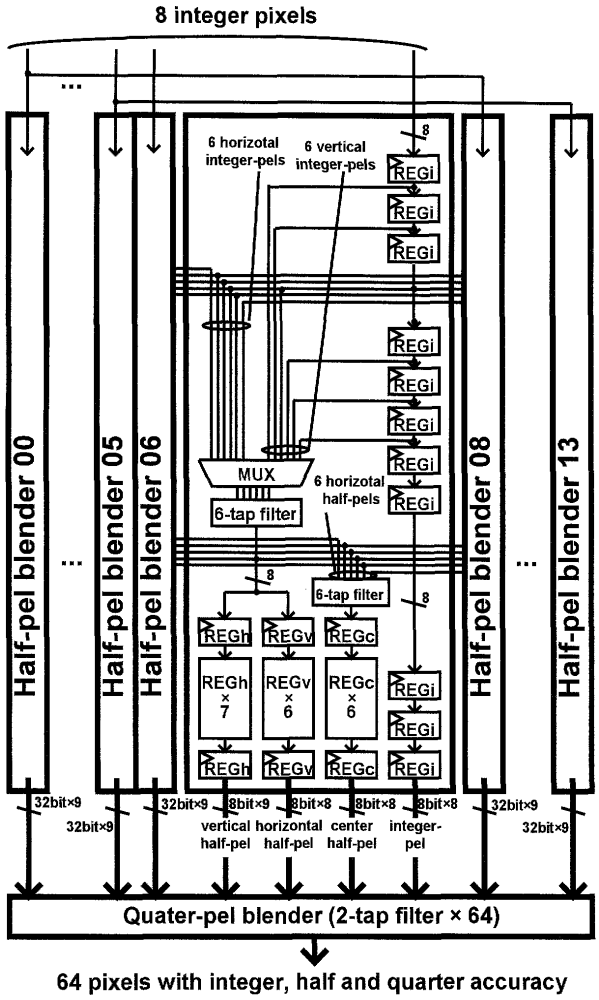


Fig. 16 Block diagram of SPG in SME.

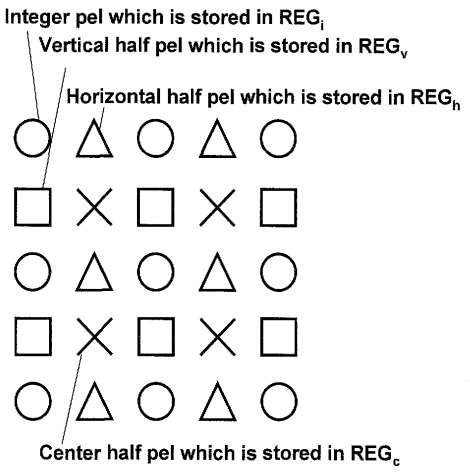


Fig. 17 Integer and half pixel data.

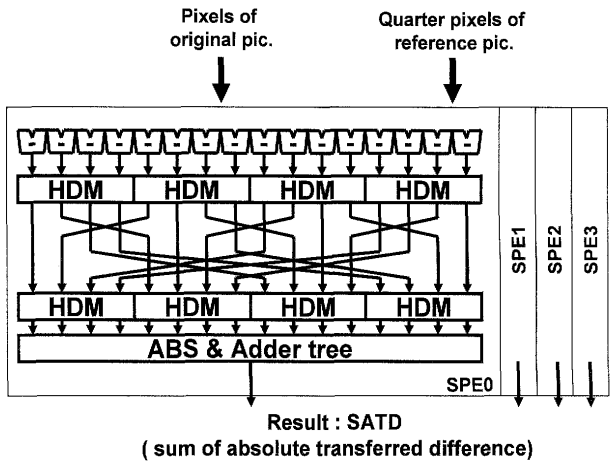


Fig. 18 PU datapath of SME.

els with a half- and quarter-pel accuracy in a reference picture. The PU is comprised of a 64-way SIMD which calculates SATDs. The motion cost that indicates a criterion to obtain the optimum motion vector is generated by the MCG module.

Figure 16 illustrates a block diagram of the SPG which consists of 14 half-pel blenders and a quarter-pel blender. The half-pel blender contains six-tap filters to generate half pixels. Figure 17 shows generated integer- and half-pel data. Vertical half pixel is calculated from six integer pixels in REG_is on vertical direction, and it is stored in REG_v. Horizontal half pixel is also computed from six interger pixels in REG_is on horizontal direction, and it is stored in REG_h. Center half pixel is generated from six horizontal half pixels in REG_c. These all registers, REG_i, REG_v, REG_h and REG_c, transfer the integer and half pixel data to the quarter-pel blender. The quarter-pel blender generates quarter pixels, and supplies 64 pixels with integer-, half- and quarter-pel accuracy to the PU.

Figure 18 shows a block diagram of the PU which performs four 4×4-SATDs computations in a cycle. The SPE (sub-pel processor element) handles a 4×4-SATDs computations. The HDM represents a one-dimensional Hadamard transformer. The cross-wiring between two HDM stages makes two-dimensional Hadamard transforms.

3.3 Concurrent Operation of IME and SME

IME processor operates concurrently with SME processor by a macroblock pipeline. When IME processes N-th macroblock, SME handles $(N-1)$-th macroblock. Figures 19(a) and (b) shows an SRAM read-port allocation to the IME and SME circuits in the 16 way-SIMD mode and the SA mode, respectively. Also Fig. 20 compares timing schedules of the entire motion estimation process assuming three types of IME architecture. Suppose that IME has only SIMD mode. In that case, the number of operation cycle to complete the process is 1177 cycles that is largest, as shown in Fig. 20(a). This is because it uses both the read ports #0 and #1, so that the SME can not obtain pixels to execute the sub-pel search from the SRAMs. As a result, the SME can not operate until the IME processing finishes, and these two processors have to be operated sequentially. On the other hand, suppose that IME has only SA mode. In that case, the number of operation cycle to complete the process is 1098 cycles, as shown in Fig. 20(b). This is because the parallel operation of the IME and the SME processor is possible, allowing decrease of the total operation cycle count. However, the integer-pel

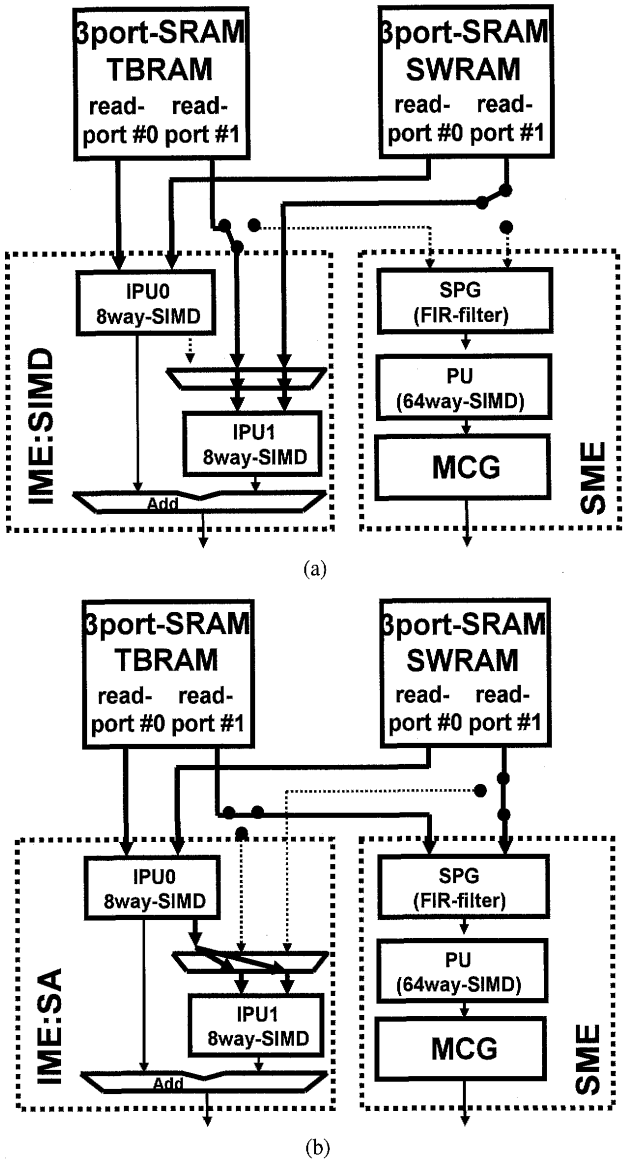


Fig. 19 (a) Datapath configuration in 16way-SIMD mode. (b) Datapath configuration in SA mode.

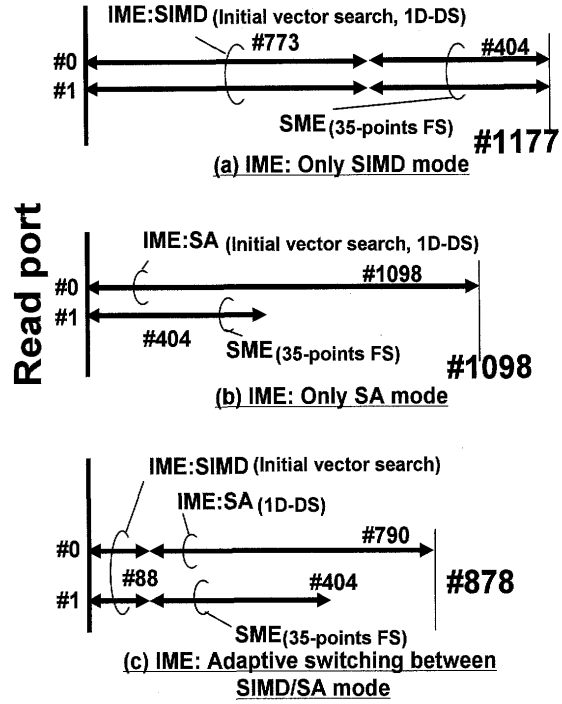


Fig. 20 Comparison of timing schedule for the entire motion estimation process among three kinds of IME architecture.

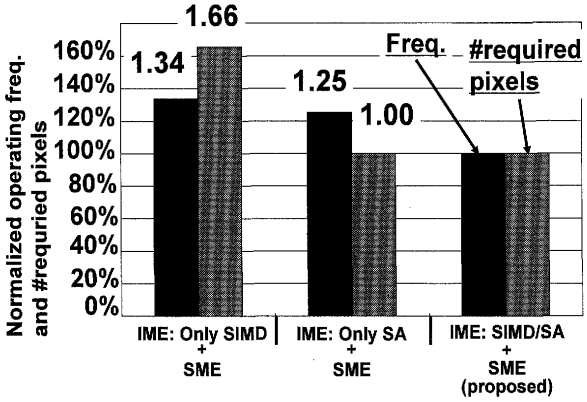


Fig. 21 Operating frequency and required pixel counts for the entire motion estimation process.

search itself requires larger operating cycle because of idle cycles caused by the SA configuration.

Figure 20(c) shows timing schedules in case of the proposed SIMD/SA configuration which demonstrates the minimum cycle counts, 878 cycles. As illustrated in Fig. 19(a), in the 16-way SIMD mode, the IME occupies both the read ports #0 and #1 to perform a parallel processing during the initial vector search. Furthermore, when the processor configure SA mode, the IME and SME can operate in parallel to execute the 1D-DS and 35-point full search. The concurrent operation by the proposed SIMD/SA architecture attains the minimum power since the required cycle is less, and thus an operating frequency can be lowered. The operating frequencies in three types of configuration are shown in Fig. 21. It is seen that 25% to 34% reduction of the operating frequency in the proposed is achieved compared with the SIMD and the SA configuration.

4. Chip Implementation Design

Figure 22 shows a chip layout of the motion estimation processor core in a 130-nm CMOS technology. It integrates 3.3 M transistors, and occupies in $2.8 \times 3.1\,\mathrm{mm}^2$. The ME core can operate at 54 MHz when a nominal supply voltage is 1.0 V. The chip specification is shown in Table 3.

The power estimated with a post-layout simulation is shown in Fig. 23. The proposed ME core consumes $800\,\mu\mathrm{W}$ in a QCIF (352×288), 15-fps sequence with one reference picture. On the other hand, the conventional one which adopts a full search method using sub-sampling [6] dissipates 3.9 mW, which means that a power reduction of 20% is achieved. The conventional power is estimated by scaling a process technology, a resolution and the number of reference pictures.

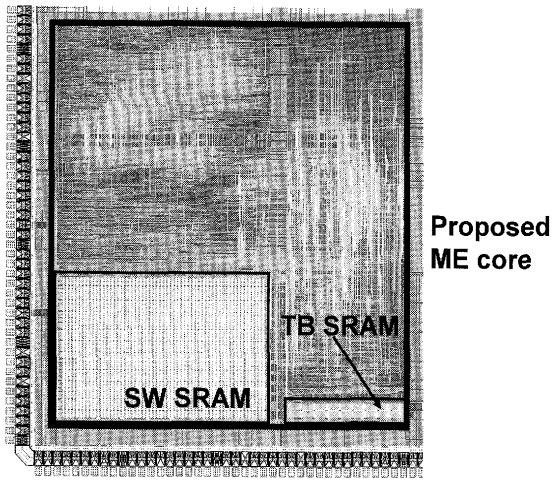


Fig. 22 Layout of H.264 motion estimation processor core.

Table 3 Specification of the core.

Technology	130-nm
Supply voltage	1.0 V
Max freq.	60MHz
Search range	Max ±32 × ±32
# ref. pictures	Max 3
Memory size	SWRAM : 160 Kbit TBRAM : 7 Kbit
Power	- 800 μW at QCIF 15-fps, 1 ref. frame, 1MHz - 18 mW at CIF 30-fps, 3 ref. frames, 54MHz

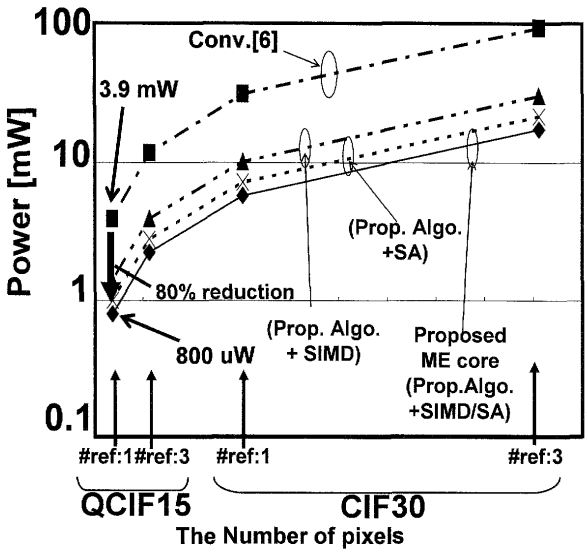


Fig. 23 Power consumption comparison.

Area overhead of the proposed SIMD/SA is 5% of IME, and 0.2% of the whole.

5. Conclusions

This paper proposed an 800-μW H.264 baseline profile motion estimation processor core. The processor supports all the seven kinds of block modes, and can handle three reference frames for a CIF (352×288) 30-fps to QCIF (176×144) 15-fps sequences with a quarter-pixel accuracy.

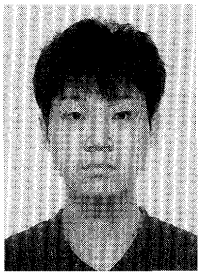
The processor features the adaptive block-matching algorithm and novel block partitioning strategy to reduce both the worst-case workload and operating frequency. By applying the SA architecture to the 1D-DS algorithm, pixels read from SRAMs can be lowered. Furthermore the SIMD/SA architecture enables a parallel operation of the IME and SME processors so that the operating frequency decreases. It is expected to be applicable to an SoC with H.264 codec function for energy-aware portable video terminals.

Acknowledgments

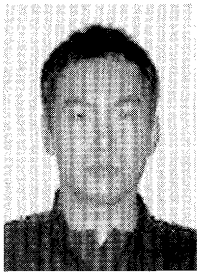
This work was supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Mentor Graphics and Synopsys, Inc.

References

[1] ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, 2003.
[2] W.I. Choi, B. Jeon, and J. Jeong, "Fast motion estimation with modified diamond search for variable motion block size," IEEE International Conference on Image Processing (ICIP), vol.3, pp.371–374, Sept. 2003.
[3] L. Yang, K. Yu, J. Li, and S. Li, "An effective variable block-size early termination algorithm for H.264 video coding," IEEE Trans. Circuits Syst. Video Technol., vol.15, no.6, pp.784–788, June 2005.
[4] ISO/IEC | ITU-T VCEG, Fast Integer Pel and Fractional Pel Motion Estimation for JVT, JVT-F017, 2002
[5] JM 8.5, http://iphome.hhi.de/suehring/tml/
[6] Y.-W. Huang, T.-C. Chen, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, C.-S. Chen, C.-F. Shen, S.-Y. Ma, T.-C. Wang, B.-Y. Hsieh, H.-C. Fang, and L.-G. Chen, "A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications," IEEE International Solid-State Circuit Conference, pp.128–129, Jan. 2005.
[7] Y. Murachi, K. Hamano, T. Matsuno, J. Miyakoshi, M. Miyama, and M. Yoshimoto, "A 95 mW MPEG2 MP@HL motion estimation processor core or portable high-resolution video application," IEICE Trans. Fundamentals, vol.E88-A, no.12, pp.3492–3499, Dec. 2005.
[8] J. Miyakoshi, Y. Murachi, K. Hamano, T. Matsuno, M. Miyama, and M. Yoshimoto, "A low-power systolic array architecture for block-matching motion estimation," IEICE Trans. Electron., vol.E88-C, no.4, pp.559–569, April 2005.
[9] J. Miyakoshi, Y. Murachi, M. Hamomoto, T. Iinuma, T. Ishihara, H. Kawaguchi, and M. Yoshimoto, "A power- and area-efficient SRAM core architecture for super-parallel video processing," 14th IFIP International Conference on Very Large Scale Integration VLSI-SoC, pp.192–197, Oct. 2006.

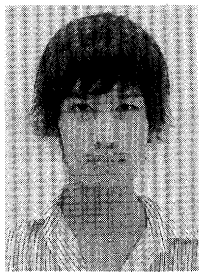


Junichi Miyakoshi received the B.E. degree in electrical and information engineering in 2002 and the M.E. degree in electronics and computer science from Kanazawa University, Ishikawa, Japan, in 2004. He is currently a Ph.D. candidate at Kobe University, Kobe, Japan. His current research interests include high-performance and low-power multimedia VLSI designs. He is a student member of IEEE.

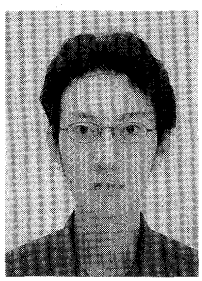


Yuichiro Murachi was born on November 1, 1980. He received a B.S. degree from Kanazawa University in 2003. He received an M.E. degree from Kanazawa University, Ishikawa, Japan, in 2005. He is currently enrolled in the Doctoral course in Kobe University. His research interests are VLSI systems and implementation of multimedia communication systems.

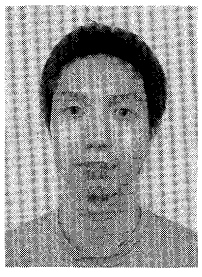
Tetsuro Matsuno received a B.E. degree from Kanazawa University in 2004. He received an M.S. degree in Electrical and Computer Engineering from Kanazawa University in 2006. He is currently a doctoral course student of Kobe University. His present research focus is design techniques of mixed-signal LSI.



Masaki Hamamoto was born on June 1, 1982. He received a B.E. degree from Kanazawa University, Ishikawa, Japan in 2005. He is currently on the master course at Kobe University. His research interests include low-power VLSI algorithms and architectures, and multimedia signal processing.

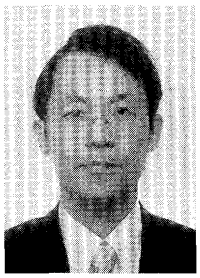


Takahiro Iinuma received a B.E. degree in Computer and Systems Engineering from Kobe University, Hyogo, Japan in 2006. He is currently on the master course at Kobe University. Since 2005, he has been involved in the research and development of low-power multimedia VLSI.

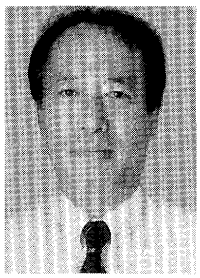


Tomokazu Ishihara was born on December 28, 1983. He received a B.E. degree from Kobe University, Hyogo, Japan in 2006. He is currently on the master course at Kobe University. His research interests include low-power VLSI algorithms and architectures, and multimedia signal processing.

Hiroshi Kawaguchi was born in Kobe, Japan, in 1968. He received the B.E. and M.E. degrees in electronic engineering from Chiba University, Japan, in 1991 and 1993, respectively. He joined Konami Corporation, Japan, in 1993, in which he developed arcade entertainment systems. He moved to the Institute of Industrial Science, the University of Tokyo, Japan, in 1996 as a technical associate, and was appointed to be a research associate in 2003. He moved to the Department of Computer and Systems Engineering, Kobe University, Japan, in 2005, as a research associate. He is also a collaborative researcher of the Institute of Industrial Science, the University of Tokyo. He is a recipient of the IEEE ISSCC 2004 Takuo Sugano Award for Outstanding Paper. He has served as a program committee member for IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips). He is a guest associate editor of IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. His research interests include low-voltage VLSI designs, low-power hardware systems, wireless circuits, and organic-transistor circuits. He is a member of IEEE and ACM.



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 received the Ph.D. degree in electrical engineering and computer science from Kanazawa University in 2004. 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 the B.S. degree in electronic engineering from Nagoya Institute of Technology, Nagoya, Japan, in 1975, and the 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 Corp., Itami, Japan, in April 1977. From 1978 to 1983 he was engaged in the design of NMOS and CMOS static RAM including a 64 K full CMOS RAM with the world's 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 at 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 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.