SIMD-based Datapath with Efficient Operation Structure for Motion Estimation

Yuki FUKAZAWA†, Keita WATANABE†, Yuki MINOURA†, Toshio KONDO† and Takahiro SASAKI†

† Graduate School of Engeneering, Mie University
1577, Kurimamachiya-cho, Tsu-City, Mie, 514-8507 Japan
Email: †{fukazawa, minoura, kondo, sasaki}@arch.info.mie-u.ac.jp

***Abstract*—Even high-performance general purpose processors are not sufficient for speedy motion estimation (ME), though they support some SIMD instructions for ME. In this paper, we propose a SIMD-based general-purpose-oriented datapath with efficient operation structure for ME. The several additional components have been added to a comventional datapath base for ME acceleration. The results of its logic design and chip layout design showed the proposed datapath accelerates motion estimation speed by 3.99-5.06 times on average in the diamond, *SUC* and non-dense patterns compared to previous method with only 2.6% hardware area increase.**

Key words : Motion estimation, SIMD, general purpose oriented, shuffle operation, no-alignment block access.

I. Introduction

The high definition of video image has made considerable in recent years as the test broadcast of Ultra High Definition Television is scheduled for next year in Japan. Under these circumstances, the demand for the compression ratio improvement of video encoding is increasing. In January 2013, the video coding standard H.265/HEVC [1] being twice as efficient as H.264/HVC [2] was recommended by ITU-T for satisfying the high coding efficiency. However, its encoding speed becomes at least several times slower than H.264 by the increase in computation complexity. Improving the motion estimation (ME) is the most important key point to accelerate the encoding speed because ME occupies most of the encoding processing. Although conventional processors such as ARM and x86 processors have enhanced datapaths or instructions for ME to reduce the processing time. For example, PSADBW and MPSADBW instructions utilized for motion estimation have been implemented in x86 processors [3], [4]. However these instructions are less compatible to efficient search patterns, e.g., the expanding diamond pattern [5] that is utilized in H.265 reference software. Though conventional multi-purpose processors with ME accelerating capability has been proposed [6], none of these processors provide satisfactory for recent efficient ME algorithms.

In order to improve this condition, we can design two types of datapaths, one is oriented toward a special-purpose processor for more efficient ME processing, and the other is oriented a general purpose processor with ME acceleration capability. In this paper, we proposed a general-purpose-oriented datapath that adds minimum function to a previous conventional datapath corresponding to SIMD instructions for ME and evaluate its efficiency. In order to accelerate ME processing, we introduced various types of additional function units : a FIFO temporally storing absolute difference (AD) values, a three-stage-structured AD adder, a comparison register extracting the minimum sum of absolute differences (SAD), a short-word-oriented AD adder, a byte-unit address modifier and a barrel shifter.

In order to compare the ME performance of the proposed datapath, we evaluate the number of steps required for SAD operation in diamond pattern, *SUC* pattern and non-dense pattern consisting of scattered search points. In addition, we implemented the proposed datapath and calculated its hardware scale by logic synthesis.

II. Inefficiency of Current SIMD Instructions for ME

Acceleration of the ME processing is a keypoint to obtain high quality of video compression. The current processors such as ARM and x86 architecture support the SIMD instructions for efficient ME operations in recent years. However, these processors cannot always perform efficient ME due to the limitation of the specification instruction. In this section, we introduce about ME and function limitation of a current considerable SIMD instruction for ME.

A. Motion Estimation

ME is the process to determine motion vectors that utilize to reduce the redundant data in a video image. In this process, block matching process between a reference block and a template block is repeated to determine the motion vector. Here, the reference block is a block in a search area in a reference picture and the template block is a coding block in a current picture. As a result, the best reference block with the highest similarity compared with the template block is detected. Most of ME process is composed of large amount of SAD operations that calculates the similarity between blocks in two pictures.

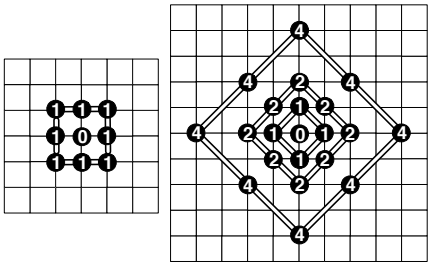

Fig. 1. Search pattern

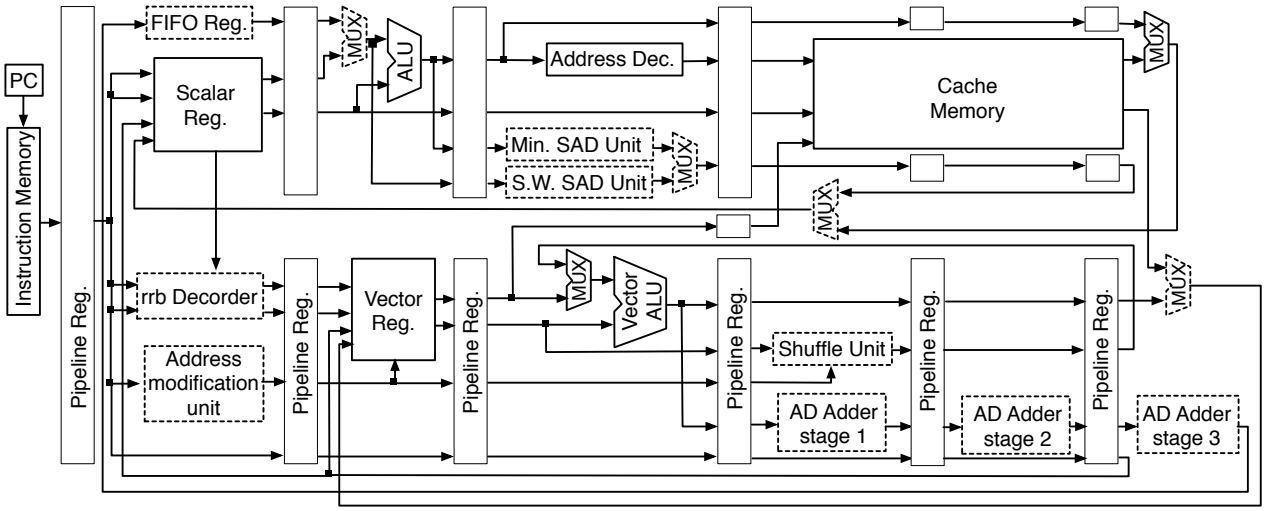

Fig. 2. Architecture of the proposed datapath

ME algorithms are divided into two classes, i.e., hardware-oriented and software-oriented. The former is mainly based on hierarchical raster search for dense square area. The raster search repeats the block matching in raster order in the search area to detect a block with the minimum SAD value. It can provide easy implementation on parallel processing, though it reaches high computational complexity. On the other hand, the latter reduces computational complexity by utilizing tracking search. There are some well-known tracking search patterns, such as UM hexagon pattern [7], EPZS [8]. Figure 1 shows two unit search patterns, a square pattern and a diamond pattern, for tracking search [5], [9], [10]. Tracking search turns the unit search pattern, i.e., the unit center to the minimum SAD value point in the previous unit search until the SAD value of the search center corresponds to the minimum value in the search area. The advantage of the tracking search can increase the chance to find the best prediction block and considerably reduce the computational complexity.

B. Current SIMD Instructions for ME

Nowadays, many general purpose processors supports SIMD instruction set containing several instructions for ME. For example, ARM processor has a 128-bit wide vector register and a SAD operation module for SIMD instruction set and it provides VABA instruction accumulating AD results for SAD operation [11].

X86 processor provides the SIMD extention instructions called MMX, SSE and AVX (Advanced Vector Extensions) [3], [4]. These contains the PASADBW and the MPSADBW instruction for ME acceleration. The former performs SAD operation between a current and a reference unit picture consisting of 16 successive pixels in the horizontal direction. The latter performs SAD operation between a current unit picture and eight reference unit pictures at the successive search points. In this latter case, each unit picture consists of 4 sucessive pixels in the horizontal direction. These instructions utilize 128-bit wide XMM registers and 256-bit wide YMM registers, respectively, so that they make possible to execute parallel matching between a current and eight reference blocks.

However, this parallel execution is restricted to successive eight search points in the horizontal directions. Thus, it provides lower compatibility for software-oriented ME mainly repeating the diamond or square searches. Therefore, almost all commercial general-purpose processors cannot efficiently execute ME.

III. SIMD Datapath with Efficient Operation Structure for ME

Figure 2 shows the architecture of the proposed SIMD based datapath. It performs 8×4 size block matching between a current coding block and a reference candidate block in a ±4 small search range efficiently. Note that, the 8×4 size block is the smallest block composing any block used in H.265 motion estimation. In this figure, the components are shown by solid lines are the conventional equipments and those shown by dotted lines are extensions in this design. The following are the key features or functions of our proposed datapath.

The datapath is a 2-way superscalar processor which can execute a scalar (64-bit) and a SIMD (256-bit) instruction simultaneously. The datapath consists of 7-stage pipeline stages in order to employ the cache memory with both line and tile unit accessibility [12], of which latency is 3 cycles, to improve the data accessibility. The cache memory can provide efficient access to data having two dimensional spatial locality though it increases latency by one cycle.

Both of the reference picture and the current picture are loaded from the cache memory by an 8×4-byte-size unit tile to the 256-bit-width Vector Register. However, if the conventional Vector Register were used, the Vector ALU could not be fed with an 8×4-byte size reference block at an arbitrary position in the search range due to uniform address distribution to each 32-byte unit register composing a Vector Register. Therefore, we introduce a Vector Register with a byte-unit address modification unit and a shuffle unit for unskew of skewed block data in the output side so that Vector ALU can obtain a unalignment block in the vertical direction from the Vector Register.

To reduce the number of loading data from the vector register, the datapath has a byte-unit address modification

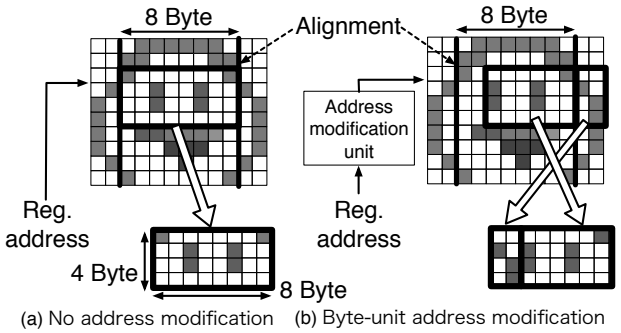

Fig. 3. Address modification

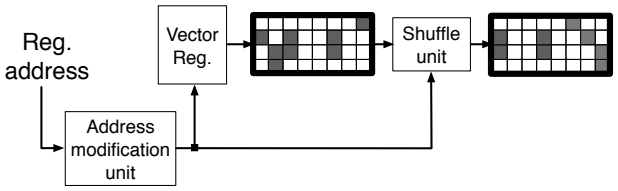

Fig. 4. Shuffle operation

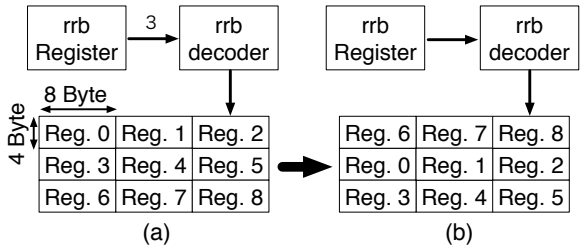

Fig. 5. Register-rename-base operation

unit. Figure 3 shows the byte-unit register address modification function. There is a 8-byte alignment restriction in the register, so that a datapath can only load the image data according to the alignment restriction as shown in Figure 3(a). On the other hand, the proposed datapath has a byte-unit address modification function, so that it can load the block data from an arbitrary point as shown in Figure 3(b).

However, the address modification function affects the loaded block data from the register. As a result, the loaded block data from the vector register has been skewed according to a register address as shown in Figure 3(b). In order to reorder the skewed block data, the datapath has a shuffle function. The shuffle unit performs as a barrel shifter, so that the skewed block data has been transformed into the expected order as shown in Figure 4.

For the reusability of data, the datapath has an rrb (register-rename-base) function [13]. It performs register rotation by using rrb register (Scalar Register 0) with an rrb Decoder. Figure 5(a) shows the rrb operation when the rrb register holds the value 3. The rrb decoder decodes the rrb value and then, the vector register numbers change according to the rrb value without loading another data as shown in Figure 5(b). The rrb function can reduce the number of loading data because no additional cache memory access is needed.

In the Vector ALU, 32 ADs between two input 32 byte-wide vectors are calculated simultaneously. In these scalar and vector pipeline stages, calculations of SADs between

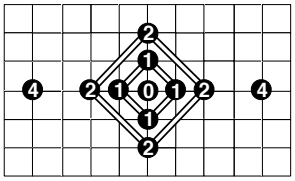

Fig. 6. *SUC* pattern

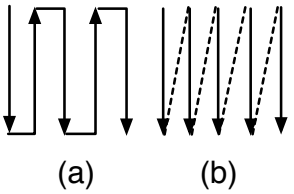

Fig. 7. Scan Order

the 8×4 size blocks and the minimum SAD are executed in parallel for ME. The sum of these ADs are calculated in a 3-stage structured AD adder and it is fed to a FIFO (First-in First-out) register. The sum values in the FIFO are accumulated to the SAD value according to the block size by the ALU. The SAD value at each search point is compared with the value held in the Minimum SAD Unit (Min. SAD Unit). If it is smaller than the held value, it is stored in the Minimum SAD Unit as the minimum value detected so far. On the other hand, the AD values are also fed into S.W. (Short Word) SAD Unit for the short-word-oriented SAD calculation.

IV. Evaluation

In order to compare the performance for ME with that of proposed SIMD-based datapath, we evaluate the number of search processing steps of the reference picture for three types of search pattern. Figure 6 shows a novel search pattern consisting of 11 search points, which we call Small-Unsymmetric Cross (*SUC*) pattern [14]. It is based on the diamond pattern as shown in Figure 1(b) and focused on horizontal search. This is because the detected motion is inclined toward the horizontal direction. Furthermore, it can reduce the amount of loading the reference picture data during ME because it reduces the number of vertical search points. In this evaluation, we employ the small diamond pattern, the *SUC* pattern and non-dense pattern that consists of search points corresponding to reference blocks each of which has no overlapped area to each other.

The proposed datapath and the conventional datapath executed BB_SAD instruction and the MPSADBW instruction to calculate SAD value, respectively. We set eleven types of block size that are utilized in H.265 from 8×4 (4×8) until 64×64. The proposed datapath and conventional one employed snake scan order [15] as shown in Figure 7(a) and conventional column order as shown in Figure 7(b), respectively. Snake scan order is able to achieve high data reusability because it can also reutilize the data for the next AD operation by the rrb function when the scan direction is turned at the end side.

Table I shows the total number of steps and average steps per a pixel for each block size and search pattern. From this table, we can see that the average number of steps for ME of the proposed datapath is 3.99, 5.03 and 4.56 times smaller than conventional one for each pattern, respectively. Especially, the proposed *SUC* pattern has the smaller number of vertical search points than the small diamond pattern As a result, the proposed datapath could reduce the amount of loading the reference image data, so that the *SUC* pattern could improve the performance for ME compared with other patterns. Furthermore, the block size becomes larger, the average decrease rate of

TABLE I. TOTAL NUMBER OF STEPS AND AVERAGE STEPS

	Total number of steps (Average steps per pixel)					
	Diamond pattern		*SUC* pattern		Non-dense pattern	
	Previous	Proposed	Previous	Proposed	Previous	Proposed
8×4	51 (1.594)	22 (0.688)	99 (3.094)	34 (1.063)	27 (0.844)	10 (0.313)
8×8	101 (1.578)	32 (0.500)	197 (3.078)	50 (0.781)	53 (0.828)	15 (0.234)
8×16	201 (1.570)	54 (0.422)	393 (3.070)	84 (0.656)	105 (0.820)	25 (0.195)
16×8	201 (1.570)	53 (0.414)	393 (3.070)	83 (0.648)	105 (0.820)	23 (0.180)
16×16	401 (1.566)	97 (0.379)	785 (3.066)	151 (0.590)	209 (0.816)	45 (0.176)
16×32	801 (1.564)	185 (0.361)	1,569 (3.064)	287 (0,561)	417 (0.814)	85 (0.166)
32×16	801 (1.564)	183 (0.357)	1,569 (3.064)	285 (0.557)	417 (0.814)	81 (0.158)
32×32	1,601 (1.563)	359 (0.351)	3,137 (3.063)	557 (0.540)	833 (0.813)	165 (0.161)
32×64	3,201 (1.563)	711 (0.347)	6,273 (3.063)	1,101 (0.538)	1,665 (0.813)	325 (0.158)
64×32	3,201 (1.563)	707 (0.345)	6,273 (3.063)	1,097 (0.536)	1,665 (0.813)	325 (0.158)
64×64	6,401 (1.563)	1,411 (0.344)	12,545 (3.063)	2,137 (0.522)	3,329 (0.813)	645 (0.157)

TABLE II. HARDWARE SCALE

	#NAND gate
rrb Decoder	237
3-Stage structured AD adder	2,167
Shuffle Unit	3,986
FIFO Register	1,944
Min. SAD Unit	873
S.W. SAD Unit	468
Additional Hardware	9,677
Total	365,217

TABLE III. NUMBER OF TRANSFER DATA OF REFERENCE PICTURE (BYTE)

	Proposed	Previous
8×4	192	256
8×8	352	384
8×16	544	640
16×8	512	768
16×16	896	1,280
16×32	1,664	2,304
32×16	1,600	2,560
32×32	3,136	4,608
32×64	6,208	8,704
64×32	6,080	9,216
64×64	12,224	17,408

search processing steps of the proposed datapath becomes smaller. Table III shows the total amount of transfer data of the reference picture. From this table, we can see that the amount of transfer data of the proposed datapath is smaller than conventional one. This is because the proposed datapath has rrb function. In other words, the proposed datapath can improve the reusability of data and reduce the amount of the loading image data from the cache memory.

In order to calculate the area increase of the proposed datapath, we implemented a chip design in system verilog. The datapath logic is synthesized by Design Compiler (Synopsys, Inc.) to ROHM 0.18µm process of VDEC (VLSI Design and Education Center) under the delay constraint of 6ns. Note that, we implemented the part of FFs such as tag-memories, valid-memories and data-memories in the cache memory with ROHM SRAM memory macros. Table II shows the hardware increase of the proposed datapath. This table shows the area sizes for each additional module and whole the datapath, which are converted into 2-input NAND gates. We can see that the area increase of added 6 modules are only 2.6% of the whole of datapath. From these implementation results, the total silicon area is fitted into 2.5*mm*×2.5*mm* as shown in Figure 8. Thus, the proposed datapath achieved considerable performance improvement in spite of a little area increase compared with the conventional datapath.

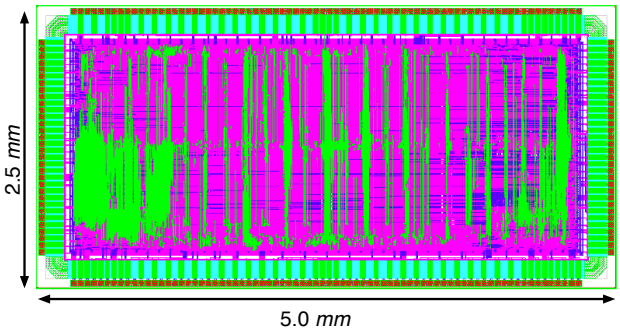

Fig. 8. Chip layout

V. CONCLUSION

In this paper, we proposed a SIMD-based Datapath with efficient operation structure for ME. The proposed datapath possesses some additional components to perform ME efficiently, so that it can perform SAD operation of ME on average 3.99-5.06 faster on average than conventional datapath for various search patterns. We evaluated the area size of proposed datapath and showed the area increase is only 2.6% in comparison with the conventional datapath. Since the acceleration technologies adopted to the proposed datapath is easily applicable to another SIMD based datapath, it can contribute to improving video coding capability for a number of general purpose processors.

ACKNOWLEDGMENT

This work is supported by Grant-in-Aid for Scientific Research C(2) Grant Number 24500059 and VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

REFERENCES

[1] ITU-T Recommendation H.265, "Advanced video coding for generic audiovisual services," April 2013.

[2] ITU-R Recommendation H.264, "Advanced video coding for generic audiovisual services," May 2003.

[3] Intel, Corp. (April, 2010). Intel Advanced Vector Extentions Programming Reference [Online]: http://www.intel.co.jp/

[4] Intel, Corp. (June, 2011). Intel Advanced Vector Extensions Programming Reference [Online]: http://www.intel.co.jp/

[5] T. Xiu-Li, D. Sheng-Kui and C. Can-Hui, "An analysis of TZ search algorithm in JMVC," IEEE Proc. ICGCS., pp.516–520, June 2010.

[6] S. D. Kim and M. H. Sunwoo, "MESIP: A Configurable and Data Reusable Motion Estimation Specific Instruction-Set Processor," IEEE Trans. on Circuit Systems., pp.1767–1780, Oct. 2013.

[7] Z. Chen, J. Xu, Y. He and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H. 264/AVC," Journal of Visual Communication and Image Representation 17.2, pp.264–290, April. 2006.

[8] A. M. Tourapis, “Enhanced predictive zonal search for single and multiple frame motion estimation,” IEEE Proc. VSIP, pp.1069–1079, Jan. 2002.

[9] O. Ndili and T. Ogunfunmi, “Hardware-oriented modified diamond search for motion estimation in H.264/AVC,” IEEE Proc. ICIP., pp.749–752, Sep. 2010.

[10] Z. Chen, P. Zhou and Y. He, “Fast integer PEL and fractional PEL motion estimation for JVT,” JVT-F017, pp.5–13, Dec. 2002.

[11] ARM Software development tools RealView Development Suite Version 4.0 RealView Compilation Tools Assembler Guide [Online]: http://infocenter.arm.com/help/index.jsp

[12] B. Wang, Y. Fukazawa, T. Kondo and T. Sasaki, “A Cache Memory with Line and Tile Data Accessibility [In Japanese]”, IEICE Technical Report (Integrated Circuits and Devices), Vol.114, No.232, ICD2014-60, pp. 39–44, Oct. 2014.

[13] Intel, Corp. (May, 1999). IA-64 Applicatioon Developers Architecture Guide [Online]: http://www.intel.co.jp/

[14] Y. Minoura, T. Kondo, Y. Fukazawa and T. Sasaki, “Highly Efficient Motion Estimation Algorithm Frequently Using SIMD Instructions for Small Search Range [In Japanese]”, IEICE Technical Report (Image Engineering), Vol.115, No.96, IE2015-38, pp. 35–40, June 2015.

[15] X. Wen, O. C. Au, J. Xu, L. Fang, R. Cha and J. Li, “ An analysis of TZ search algorithm in JMVC,” IEEE Trans. on Circuit Syst. Video Technol., Vol.21, No.2, pp.206–219, Feb. 2011.