An Efficient Hardware Architecture for Calculating the Minimum SAD with Parallel Execution at the Search Point Level

Tae Sung Kim[1], Hyuk-Jae Lee[2], and Chae Eun Rhee[3]
[1, 2] Department of Electrical Engineering and Computer Science,
Interuniversity Semiconductor Research Center, Seoul National University
1 Kwanak-ro, Kwanak-gu, Seoul, 08826, Korea
[3] Department of Information and Communication Engineering, Inha University
100 Inha-ro, Nam-gu, Incheon, 22212, Korea
E-mail: [1]tskim@capp.snu.ac.kr, [2]hyuk_jae_lee@capp.snu.ac.kr, [3]chae.rhee@inha.ac.kr

Abstract: Integer Motion Estimation (IME) is one of the key components of video coding standard such as high-efficiency video coding (HEVC). As HEVC adopts a highly flexible block partitioning structure from 4x4 to 64x64, performing IME for every block partition demands considerable computational complexity. Although a number of previous works for efficient hardware architectures to accerarate IME have been done, the computational complexity of IME keeps growing as the video resolution increases. In this paper, an efficient hardware architecture for calculating the minimum SAD is proposed. The proposed hardware exploits the parallelism of multiple search points. To improve the efficiency of the proposed hardware, SAD calculation and its comparison steps are performed in a pipelined manner and the workload between the two pipeline stages are finely balanced. The proposed hardware processes 32 search points for an 8x8 block. The gate count is 827 K and the maximum operating clock frequency is 485.44 MHz.

Keywords-- **Video compression, High-efficiency video coding (HEVC), Integer motion estimation (IME), Sum of absolute diffference (SAD), Hardware architecture**

1. Introduction

Motion estimation (ME) is very important to achieve efficient video coding performance. High-Efficiency Video Coding (HEVC) is a video coding standard that has been finalized in 2013 by the joint collaborative team on video coding (JCT-VC) [1]. To improve the coding efficiency offered by H.264/advanced video coding (AVC), HEVC adopts a highly flexible block partitioning structure and fractional sample interpolation with quarter-sample precision for motion compensated inter frame prediction [2]. Therefore, searching accurate motion vectors (MVs) for all block partitions is essential to achieve maximum coding efficiency of HEVC. Integer motion estimation (IME) is the process for finding coarse motion in an integer pixel unit, whereas fractional motion estimation (FME) is for finding motion more precisely up to a 1/4 pixel unit. The block partitioning structure adopted in HEVC includes a coding tree unit (CTU), a coding unit (CU) and a prediction unit (PU). One slice is partitioned into multiple CTUs as the basic processing unit instead of an MB in H.264/AVC. Unlike an MB of which the size is fixed as 16×16 pixels, the size of the CTU is not fixed, varying from 16×16 to 64×64. A CTU can be split, forming a quad-tree structure with a leaf on the tree referred to a CU. For a CU, either the inter- or the intra-prediction mode is determined. A CU consists of multiple PUs. Different predicted blocks are generated for different PUs, which improve the prediction accuracies of the original pixels. Each PU has its unique MV.

The most popular technique for ME is the block-matching algorithm (BMA) [3]. Especially, BMA with a SAD comparison is used for simple implementation of IME. For a hardware-based IME, a sum of absolute difference (SAD) tree is widely used for high throughput [4]. The SAD tree architecture generates SAD costs of different block partitions at a particular search point simultaneously by the combination of small partitions, thereby providing the high degree of parallelism and achieving high speed IME. However, the computational complexity increases as video resolution increases so that the complexity of IME for high-definition videos which has resolution of 4k or 8k is now uncontrollable with only traditional techniques. To overcome the increase in computational complexity, many previous research works attempt to devise various hardware architectures for calculating the minimum SAD [5]-[9]. However, it is not sufficient to cover the increased computational complexity for 4k resolution or larger resolution. Therefore, latest research works attempt to use fast algorithms for a hardware-based IME [10], [11]. However, it is difficult to apply various fast IME algorithms to the IME hardware which processes multiple block partitions in parallel such as a SAD tree because most fast IME algorithms assume the sequential IME processing block by block to obtain additional information. For example, in HEVC test model (HM) reference SW, test zone search (TZS) algorithm observes MVs of neighboring PUs [12], [13].

In this paper, to improve performance of hardware-based IME more effectively, the parallel execution of IME at the search point level is proposed. The proposed hardware architecture supports sequential processing between different block partitions, thereby having a chance of reducing computational complexity of IME by adopting various fast IME algorithms. To achieve high throughput, the minimum SAD is calculated in the pipelined manner with a fine workload balancing between pipeline stages.

The rest of this paper is organized as follows. The proposed hardware architecture is explained in Section 2. The implementation result and design comparisons are given in Section 3. Section 4 concludes the paper.

2. Proposed Hardware Architecture

To achieve a high degree of parallelism along with fast algorithms, the paper presents efficient hardware architecture of calculating the minimum SAD. One of the significant differences between the conventional SAD tree and the proposed hardware is that the SAD tree processes multiple blocks for a single search point at a time, whereas the proposed hardware processes multiple search points for a particular block at a time. Figure 1 shows the steps of calculating the minimum SAD in the proposed hardware architecture. All CUs in a CTU are processed sequentially one by one. PUs in a CU are processed in parallel as like a SAD tree. M denotes the number of search points examined in a cycle, whereas N denotes width and height of a CU. Therefore, MN^2 represents a pixel capability per cycle in calculating SAD. For the proposed hardware architecture, MN^2 is set to 2,048 where the SADs of 2,048 pixels are calculated in a cycle. To support various block partition sizes of HEVC, N can be 8, 16 or 32. Thus, the corresponding M can be 32, 8 or 2, respectively. For example, if N is 8 then M is 32. In this case, the proposed hardware examines 32 search points for an 8x8 CU in a cycle. For a 64x64 CU, the operation of 32x32 CU is performed in four times iteratively. The proposed hardware is divided into two pipeline stages. In the first step, the absolute difference and its sum for each search point are calculated. In the second step, the SAD values of search points are compared to find the search point which has the smallest SAD value. This is chosen as the best MVs of the current CU. Note that, the conventional SAD tree needs to compare the SAD values between the previous smallest one and the current one, whereas the proposed hardware needs to compare the SAD values among N search points in addition to the previous smallest one as shown in Figure 1.

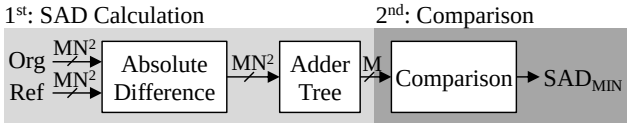

Figure 1. Steps of calculating the minimum SAD in the proposed hardware architecture

Figure 2 shows the maximum path delay of different block partitions. The numbers in light gray and dark gray boxes represent the path delay for SAD calculation and comparison, respectively. The path delay of SAD calculation step is proportional to the size of block partition. It is reasonable because the number of additions increases as the size of block partition increases. On the other hand, the path delay of comparison and the size of block partition are in inverse proportion to each other. As explained in Figure 1, MN^2 is fixed and thus, the number of search points M to be examined and the number of comparison among search points become small for the large size of block partition. Based on this observation, the workload of each block partition is distributed evenly to two pipeline stages. For an 8x8 CU, 8x8 SAD calculations and the comparison to choose 16 MVs among 32 MVs are assigned to the first stage, whereas the comparison for the remaining 16 MVs are done in the second stage. For a 64x64 CU, four 32x32 SAD calculations are performed in the first stage. After that, four 32x32 SAD values are accumulated and the comparison is done in the second stage. For 16x16 and 32x32 CUs, SAD calculation and comparison are assigned to the first and second stages, respectively.

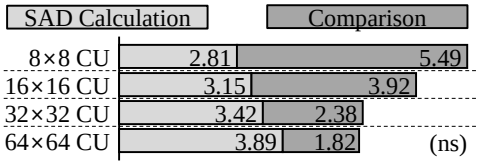

Figure 2. Maximum path delay of different block partitions with parallel execution at the search point level

3. Implementation Result and Design Comparison

3. 1 Hardware implementation

Figures 3 (a) and (b) illustrate the method to calculate the SAD of each size in the adder tree. The solid rectangles illustrate the PUs which are currently calculating SAD values, whereas the dotted rectangles illustrate the SADs which have been already calculated in the lower level adder tree. The dotted rectangles in white, light gray, dark gray and black denote the SADs from the four positions of one level lower adder tree. Figure 3 (a) shows the method for an 8x8 adder tree. Because an 8x8 CU has no AMP, only three SADs for 2Nx2N, 2NxN and Nx2N PUs are calculated using four 4x4 SADs. To use the current SAD for the upper level, all calculated SADs are transmitted to the upper level adder tree, i.e. a 16x16 adder tree. Figure 3 (b) shows the method for 16x16, 32x32, and 64 x 64 adder trees. SADs

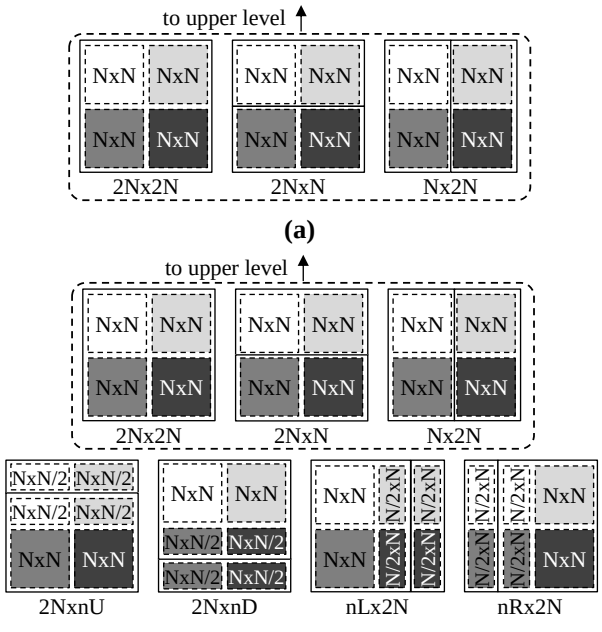

Figure 3. (a) The method of SAD calculation for an 8x8 adder tree (N=4), (b) the method of SAD calculation for 64x64, 32x32, 16x16 adder trees (N=32, 16, 8)

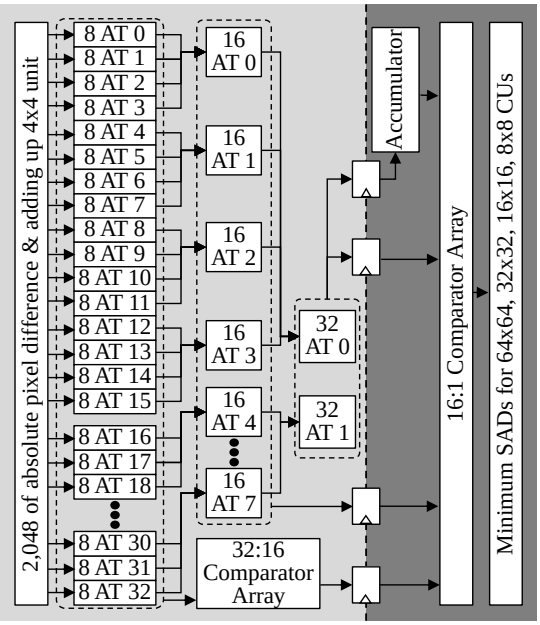

Figure 4. Proposed hardware architecture for calculating minimum SAD

for 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N and nRx2N PU are calculated with NxN, NxN/2 and N/2xN SADs which are from four lower level adder trees. To calculate the SAD of AMP, not only 2Nx2N but also NxN/2 and N/2xN SADs are necessary. For example, to calculate SAD for a 32x24 PU, two 16x16 SADs and two 16x8 SADs are needed.

Figure 4 shows the proposed hardware architecture. The light gray region denotes the first pipeline stage, whereas the dark gray region denotes the second pipeline stage. To get the absolute difference between the original and reference pixels, 2,048 absolute difference units are used. After calculating the absolute differences for all pixels, the differences are added up by a 4x4 unit to supply an 8x8 adder tree (AT) with 4x4 SADs. While the calculated SADs in each level of AT are transferred to the upper level of AT hierarchically, SADs are added up in the current level as shown in the Figures 3 (a) and (b). The proposed hardware has 32 8x8 ATs, 8 16x16 ATs, 2 32x32 ATs. There is an accumulator to add up the results from 32x32 ATs to calculate 64x64 CU's SAD. A 32 to 16 comparator array and a 16 to 1 comparator array are used to get the minimum SAD. Note that, to support the proposed workload balancing which is presented in Section 2, the comparison operations are distributed in two pipeline stages and the accumulation operation for SAD calculation of a 64x64 CU is performed in the second stage.

Table 1 shows the gate count for each component in details. The left column shows the component, whereas the right column shows the corresponding gate count. Each component name matches the hardware modules shown in Figure 4. Over 90% of gates are occupied by absolute difference, adder tree and accumulator components. Two comparator arrays which are used to support the search point-level parallel IME account for just 8.7% in total gate counts.

Table 1. Gate counts of the hardware modules

Component	Gate Count(K)
Absolute difference (×2,048)	380.3
Adder tree & accumulator	375.4
32 to 16 comparator	26.1
16 to 1 comparator	45.6
Total	827.4

3. 2 Design comparison

The proposed hardware architecture for calculating the minimum SAD is implemented in Verilog HDL and synthesized by Synopsys design compiler. The 65 nm CMOS technology with a supply voltage of 1.2V is used.

Table 2 shows the implementation results of two different designs. The second column shows the 2-stage pipeline without workload balancing, whereas the third column shows the design with the proposed workload balance. The maximum operating clock frequency of each design is 359.71 MHz and 485.44 MHz, respectively. The proposed hardware architecture shows 35% higher operating clock frequency than the other design. Even though the proposed hardware architecture uses little bit more hardware resource than that without workload balancing in the third row, the throughput and efficiency are improved as shown in the fourth and fifth rows.

Table 2. Implementation result and design comparison between the hardware architecture without and with workload balancing

	without workload balancing	with workload balancing
Maximum frequency (MHz)	359.71	485.44
Area (K Gates)	803.2	827.4
Throughput (search points per second)	1.15×10^9	1.55×10^9
Throughput per gate	1431.77	1873.34

4. Conclusions

The main contribution of this paper is that it raises an issue about the limitation of a traditional hardware-based IME architecture and proposes the new hardware architecture exploiting parallelism in multiple search points. The proposed hardware can adopt fast IME algorithms that are based on the sequential processing order among different block partitions, and therefore it is beneficial in both ways: parallelism and fast algorithm. For future research, there exist additional opportunities to improve the performance if the fast IME algorithm is adopted for the proposed hardware architecture.

Acknowledgement

This work was supported by the Ministry of Science, ICT and Future Planning, Korea, through the Information Technology Research Center under Grant IITP-2016-H8501-16-1005 supervised by the Institute for Information and Communications Technology Promotion, and by the Basic Science Research Program through the National Research Foundation of Korea within the Ministry of Science, ICT and Future Planning under Grant NRF-2015R1C1A1A02037625.

References

[1] G. J. Sullivan, J. Ohm , W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp.1649-1668, Dec 2012.

[2] I. K. Kim, J. Min, T. Lee, W.-J. Han, and J. Parkand, "Block Partitioning Structure in the HEVC Standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp.1697-1706, Dec 2012.

[3] C.-H. Hsieh and T.-P. Lin, "VLSI architecture for block-matching motion estimation algorithm," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 2, no. 2, pp.169-175, Jun 1992.

[4] C.-Y. Chen, S.-Y. Chien, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and Architecture Design of Variable Block-Size Motion Estimation for H.264/AVC," *IEEE Trans. on Circuits and Systems*, vol. 53, no. 3, pp.578-593, Mar 2006.

[5] M. E. Sinagil, V. Sze, Minhua Zhou, and A. P. Chandrakasan, "Cost and Coding Efficient Motion Estimation Design Considerations for High Efficiency Video Coding (HEVC) Standard," *IEEE Journal. Selected Topics in Signal Processing*, vol. 7, no. 6, pp.1017-1028, Jul 2013.

[6] S.-Y. Jou, S.-J. Chang, and T.-S. Chang, "Fast Motion Estimation Algorithm and Design for Real Time QFHD High Efficiency Video Coding," *IEEE Journal. Selected Topics in Signal Processing*, vol. 25, no. 9, pp.1533-1544, Jul 2013.

[7] JCT-VC, (2014, Jan.). High Efficiency Video Coding Reference Software. High Efficiency Video Coding Test Model 13 (HM13). [Online]. Available: http://hevc.hhi.fraunhofer.de/

[8] N. Purnachand, L. N. Alves, and A. Navarro, "Improvements to TZ search motion estimation algorithm for multiview video coding," in *Proc. IWSSIP*., Vienna, Austria, 2012, pp. 388 - 391.

[9] Z. Liu, S. Goto, and T. Ikenaga, "Optimization of Propagate Partial SAD and SAD Tree Motion Estimation Hardwired Engine for H.264," in *Proc. ICCD*., Lake Tahoe, CA, USA, 2010, pp. 328 - 333.

[10] C. Diniz, G. Corrêa, A. Susin, and S. Bampi, "Comparative Analysis of Parallel SAD Calculation Hardware Architectures for H.264/AVC Video Coding," in *Proc. LASCAS*., Foz do Iguacu, Brazil, 2010, pp. 113 - 116.

[11] N. C. Vayalil, A. Safari, and Y. Kong, "ASIC Design in Residue Number System for Calculating Minimum Sum of Absolute Differences," in *Proc. ICCES*., Cairo, Egypt, 2015, pp. 129 - 132.

[12] P. Nalluri, L.N. Alves, and A. Navarro, "A Novel SAD Architecture for Variable Block Size Motion Estimation in HEVC Video Coding," in *Proc. International Symposium on SoC*., Tampere, Finland, 2013, pp. 129 - 132.

[13] A. Medhat, A. Shalaby, M. S. Sayed, M. Elsabrouty, and F. Mehdipour, "A highly parallel SAD architecture for motion estimation in HEVC encoder," in *Proc. APCCAS*., Ishigaki, Japan, 2014, pp. 280 - 283.

[14] Recommendation ITU-T H.265, MPEG H -- Part 2: High efficiency video coding, ISO/IEC 23008-2, 2013.