# Analysis and architecture design of variable block-size motion estimation for H.264/AVC — Source link

# Analysis and Architecture Design of Variable Block-Size Motion Estimation for H.264/AVC

Ching-Yeh Chen, Shao-Yi Chien, Yu-Wen Huang, Tung-Chien Chen, Tu-Chih Wang, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—Variable block-size motion estimation (VBSME) has become an important video coding technique, but it increases the difficulty of hardware design. In this paper, we use inter-/intra-level classification and various data flows to analyze the impact of supporting VBSME in different hardware architectures. Furthermore, we propose two hardware architectures that can support traditional fixed block-size motion estimation as well as VBSME with less chip area overhead compared to previous approaches. By broadcasting reference pixel rows and propagating partial sums of absolute differences (SADs), the first design has the fewer reference pixel registers and a shorter critical path. The second design utilizes a two-dimensional distortion array and one adder tree with the reference buffer that can maximize the data reuse between successive searching candidates. The first design is suitable for low resolution or a small search range, and the second design has advantages of supporting a high degree of parallelism and VBSME. Finally, we propose an eight-parallel SAD tree with a shared reference buffer for H.264/AVC integer motion estimation (IME). Its processing ability is eight times of the single SAD tree, but the reference buffer size is only doubled. Moreover, the most critical issue of H.264 IME, which is huge memory bandwidth, is overcome. We are able to save 99.9% off-chip memory bandwidth and 99.22% on-chip memory bandwidth. We demonstrate a 720-p, 30-fps solution at 108 MHz with 330.2k gate count and 208k bits on-chip memory.**

***Index Terms*—Block matching, H.264/AVC, motion estimation (ME), variable block size, very large scale integration (VLSI) architecture.**

## I. Introduction

FOR VIDEO coding systems, motion estimation (ME) can remove most of temporal redundancy, so a high compression ratio can be achieved. Among various ME algorithms, a full-search block matching algorithm (FSBMA) is usually adopted because of its good quality and regular computation. In FSBMA, the current frame is partitioned into many small macroblocks (MBs) of size $N \times N$. For each MB in the current frame (current MB), one reference block that is the most similar to current MB is sought in the searching range of size $[-P, P)$ in the reference frame. The most common used criterion of the similarity is the sum of absolute differences (SAD)

$$\begin{aligned}&\mathrm{SAD}(m,n)\\&\quad=\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{Distortion}(i,j,m,n),\quad -P\le m,n<P \qquad (1)\\&\mathrm{Distortion}(i,j,m,n)\\&\quad=|\mathrm{cur}(i,j)-\mathrm{ref}(i+m,j+n)| \qquad (2)\end{aligned}$$

where $\mathrm{cur}(i,j)$ and $\mathrm{ref}(i+m,j+n)$ are pixel values in the current MB (current pixel) and reference block (reference pixel), respectively, $(m,n)$ is one searching candidate in the search range, $\mathrm{Distortion}(i,j,m,n)$ is the difference between the current pixel and the reference pixel, and $\mathrm{SAD}(m,n)$ is the total distortion of this searching candidate. The row (column) SAD is the summation of $N$ distortions in a row (column). After all searching candidates are examined, the searching candidate that has the smallest SAD is selected as the motion vector of the current MB. Although FSBMA provides the best quality among various ME algorithms, it consumes the largest computation power. In general, the computation complexity of ME varies from 50% to 90% of a typical video coding system. Hence, a hardware accelerator of ME is required.

Variable block-size motion estimation (VBSME) is a new coding technique and provides more accurate predictions compared to traditional fixed block-size motion estimation (FBSME). With FBSME, if an MB consists of two objects with different motion directions, the coding performance of this MB is worse. On the other hand, for the same condition, the MB can be divided into smaller blocks in order to fit the different motion directions with VBSME. Hence, the coding performance is improved. VBSME has been adopted in the latest video coding standards, including H.263 [1], MPEG-4 [2], WMV9.0 [3], and H.264/AVC [4]. For instance, in H.264/AVC, an MB with a variable block size can be divided into seven kinds of blocks including 4 × 4, 4 × 8, 8 × 4, 8 × 8, 8 × 16, 16 × 8, and 16 × 16. Although VBSME can achieve a higher compression ratio, it not only requires huge computation complexity but also increases the difficulty of hardware implementation for ME.

Traditional ME hardware architectures are designed for FBSME, and they can be classified into two categories. One is an inter-level architecture, where each processing element (PE) is responsible for one SAD of a specific searching candidate, as shown in (1), and the other is an intra-level architecture, where each PE is responsible for the distortion of a specific current pixel in the current MB for all searching candidates, as shown

Manuscript received November 1, 2004; revised April 18, 205 and July 7, 2005. This paper was recommended by Associate Editor Y.-K. Chen.

C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, and L.-G. Chen are with the DSP/IC Design Laboratory, Graduate Institute of Electronics Engineering and Department of Electrical Engineering II, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: cychen@cc.ee.ntu.edu.tw; shoayi@cc.ee.ntu.edu.tw; yuwen@cc.ee.ntu.edu.tw; djchen@cc.ee.ntu.edu.tw; lgchen@cc.ee.ntu.edu.tw).

T.-C. Wang is with the DSP/IC Design Laboratory, Graduate Institute of Electronics Engineering and Department of Electrical Engineering II, National Taiwan University, Taipei 10617, Taiwan, R.O.C., and also with Chin Fong Machine Industrial,50445 Chang Hua, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TCSI.2005.858488

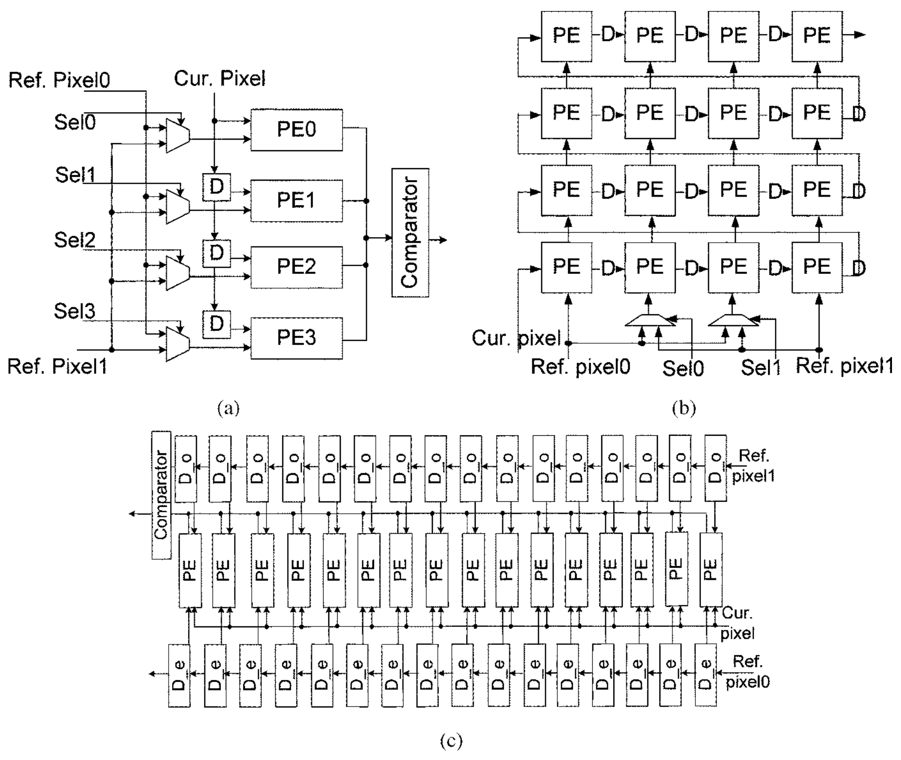

Fig. 1. Hardware architectures of (a) 1DInterYSW, (b) 2DInterYH, and (c) 2DInterLC, where $N = 4$, $P_h = 2$, and $P_v = 2$.

in (2). For example, Yang *et al.* proposed a one-dimensional (1-D) inter-level semisystolic architecture [5], and Komarek and Pirsch proposed a two-dimensional (2-D) intra-level systolic architecture, AB2, in [6].

In this paper, we not only use inter-/intra-level classification and various data flows to analyze the impact of supporting VBSME in different hardware architectures but also propose two hardware architectures, *Propagate Partial SAD* and *SAD Tree*, that can support VBSME as well as FBSME with less chip area overhead compared to previous techniques. After analyzing the impact of supporting VBSME in different hardware architectures, we discuss the hardware design challenges of integer motion estimation in H.264/AVC for D1 size as an example. Because of multiple reference frames and VBSME, integer motion estimation in H.264/AVC not only requires large computation complexity but also needs huge memory bandwidth. Based on the previous analysis, we utilize *SAD Tree* to design a hardware architecture and reduce the required memory bandwidth for H.264/AVC integer motion estimation.

The remainder of this paper is organized as follows. In Section II, six previous hardware architectures of ME are surveyed first, and we propose two hardware architectures for FBSME. Next, we analyze the impact of supporting VBSME in different hardware architectures and directly extend the six previous works and our two architectures to support VBSME. In Section III, we give an example to provide the quantified comparisons of the eight hardware architectures for FBSME and VBSME, respectively. After that, based on our analysis results, we develop a hardware architecture for H.264/AVC integer motion estimation as an example in Section IV. Finally, a conclusion is given in Section V.

## II. Impact of Supporting VBSME in Different Hardware Architectures

In this section, six representative previous works of ME hardware architectures for FBSME are introduced in the beginning. They are the works of Yang *et al.*[5], Yeo and Hu [7], Lai and Chen [8], Komarek and Pirsch [6], Vos and Stegherr [9], and Hsieh and Lin [10]. These six architectures are significant works, and many hardware architectures are proposed based on them. For example, reference [11] is the extension of [5]. Reference [12] is proposed based on [9], and reference [13] combined [10] with multilevel successive elimination algorithm [14], [15]. Reference [16] is the extension of [6]. Besides pure inter-/intra-level architectures, there are other kinds of architectures such as AS2 in [6] and a tree-based architecture in [17], which are hybrids of inter- and intra-level architectures. For the sake of simplicity, we only discuss the pure inter-/intra-level architectures, and the others can be easily extended based on our analysis.

Moreover, we also propose two intra-level hardware architectures and analyze the impact of supporting VBSME in these hardware architectures based on various data flows in this section. The direct extensions of the eight architectures are also proposed. In the following discussion, we assume that the block size is $N \times N$ and that the search range is $[-P_h, P_h)$ and $[-P_v, P_v)$ in the horizontal and vertical directions.

### A. *Work of Yang et al.*

Yang *et al.* implemented the first VLSI motion estimator in the world, as shown in Fig. 1(a), which is a 1-D inter-level hardware architecture (1DInterYSW). The number of PEs is equal to

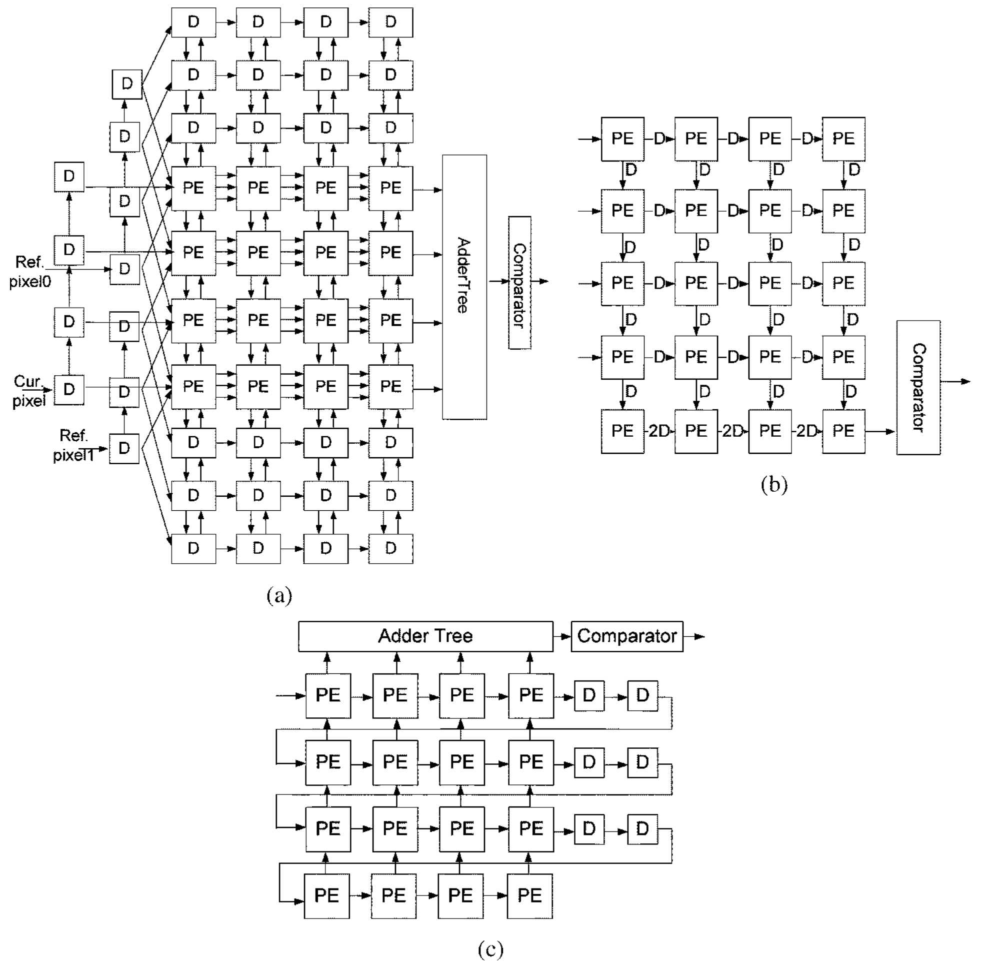

Fig. 2. Hardware architectures of (a) 2DIntraVS, (b) 2DIntraKP, and (c) 2DIntraHL, where $N = 4$, $P_h = 2$, and $P_v = 2$.

the number of searching candidates in the horizontal direction, $2P_h$. Reference pixels are broadcasted into all PEs. By selection signals, the corresponding reference pixel is selected and inputted into each PE. Current pixels are propagated with propagation registers, and the partial SAD is stored in each PE. In each cycle, each PE computes the distortion and accumulates the SAD of a searching candidate. In this architecture, the most important concept is data broadcasting. With broadcasting technique, the memory bit width which is defined as the number of bits for the required reference data in one cycle is reduced significantly, although some global routings are required.

### B. *Work of Yeo and Hu*

Fig. 1(b) shows a 2-D inter-level hardware architecture which is proposed by Yeo and Hu (2DInterYH). 2DInterYH consists of $2P_h \times 2P_v$ PEs and is similar to 1DInterYSW. Reference pixels are broadcasted into PEs, and current pixels are propagated with propagation registers. The partial SADs are stored and accumulated in PEs, respectively. Because of broadcasting reference pixels in both directions, the number of PEs has to match the MB size. Hence, the search range is partitioned into $(2P_h/N) \times (2P_v/N)$ regions, and each region is computed by a set of $N \times N$ PEs. The characteristic of 2DInterYH is broadcasting in two directions at the same time, which can increase the data reuse.

### C. *Work of Lai and Chen*

Lai and Chen also proposed another 1-D PE array that implemented a 2-D inter-level architecture with two data-interlacing reference arrays (2DInterLC). The hardware architecture is shown in Fig. 1(c) and is similar to 2DInterYH except for two aspects. Reference pixels are propagated with propagation registers, and current pixels are broadcast into PEs. The partial SADs are still stored and accumulated in PEs. Besides, 2DInterLC has to load reference pixels into propagation registers before computing SADs. The latency of loading reference pixels can be reduced by partitioning the search range in 2DInterLC. For example, the search range can be partitioned into $(2P_h/N) \times (2P_v/N)$ parts for a shorter latency.

### D. *Work of Vos and Stegherr*

A 2-D intra-level architecture is proposed by Vos and Stegherr (2DIntraVS), as shown in Fig. 2(a), where the number of PEs is equal to the MB size. Each PE corresponds to a current pixel, and current pixels are stored in PEs, respectively. The important concept of 2DIntraVS is the scanning order in searching candidates, which is known as the snake scan. In order to realize this, a great deal of propagation registers are used to store reference pixels, and the data in propagation registers can be shifted in upward, downward, and right directions. These propagation registers and the long latency for loading reference pixels are

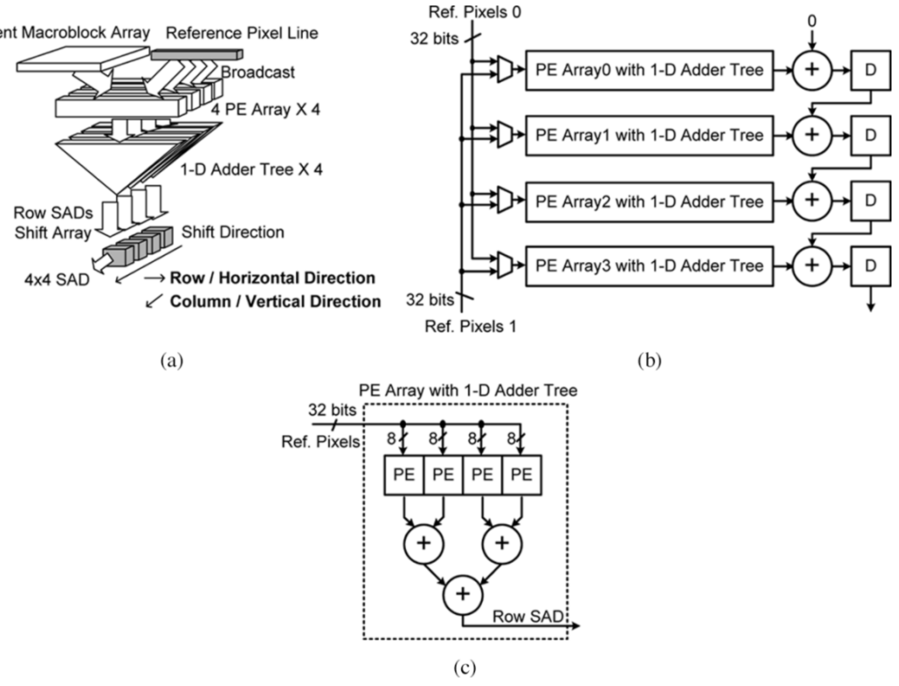

Fig. 3. (a) Concept, (b) hardware architecture, and (c) detailed architecture of a PE array with a 1-D adder tree using *Propagate Partial SAD* architecture, where $N = 4$.

the tradeoffs for the reduction of memory usages. The computation flow is as follows. First, the distortion is computed in each PE, and $N$ partial-row SADs are propagated and accumulated in the horizontal direction. Second, an adder tree is used to accumulate the $N$-row SADs to be SAD. The accumulations of row SADs and SAD are done in one cycle. Hence no partial SAD is required to be stored.

## *E. Work of Komarek and Pirsch*

Komarek and Pirsch contributed a detailed systolic mapping procedure by the dependence graph (DG). By using different DGs, including different scheduling and projections, different systolic hardware architectures can be derived. AB2 (2DIntraKP) is a 2-D intra-level architecture, as shown in Fig. 2(b). Current pixels are stored in corresponding PEs. Reference pixels are propagated PE by PE in the horizontal direction. The $N$ partial-column SADs are propagated and accumulated in the vertical direction first. After the vertical propagation, these $N$-column SADs are propagated in the horizontal direction. In each PE, the distortion of a current pixel in the current MB is computed and added with the partial-column SAD which is propagated in PEs from top to bottom in the vertical direction. In the horizontal propagation, these $N$-column SADs are accumulated one by one by $N$ adders and $2N$ registers.

## *F. Work of Hsieh and Lin*

Hsieh and Lin proposed another 2-D intra-level hardware architecture with a search range buffer (2DIntraHL), as shown in Fig. 2(c). 2DIntraHL consists of $N$ PE arrays in the vertical direction, and each PE array is composed of $N$ PEs in a row. In 2DIntraHL, reference pixels are propagated with propagation registers one by one, which can provide the advantages of serial data input and increasing the data reuse. Current pixels are still stored in PEs. The $N$ partial-column SADs are propagated in the vertical direction from bottom to up. In each computing cycle, each PE array generates $N$ distortions of a searching candidate and accumulates these distortions with $N$ partial-column SADs in the vertical propagation. After accumulation in the vertical direction, $N$-column SADs are accumulated in the top adder tree in one cycle. The longer latency for loading reference pixels and large propagation registers are the penalties for the reduction of memory bandwidth and memory bit width.

## *G. Proposed Propagate Partial SAD*

We propose a 2-D intra-level architecture called the *Propagate Partial SAD* [18]. Fig. 3(a) and (b) shows the concept and hardware architecture of *Propagate Partial SAD*, respectively. The architecture is composed of $N$ PE arrays with a 1-D adder tree in the vertical direction. Current pixels are stored in each PE, and two sets of $N$ continuous reference pixels in a row are broadcasted to $N$ PE arrays at the same time. In each PE array with a 1-D adder tree, $N$ distortions are computed and summed by a 1-D adder tree to generate one-row SAD, as shown in Fig. 3(c). The row SADs are accumulated and propagated with propagation registers in the vertical direction, as shown in the right-hand side of Fig. 3(b).

The detailed data flow of *Propagate Partial SAD* is shown in Fig. 4. The reference data of searching candidates in the even and odd columns are inputted by *Ref. Pixels 0* and *Ref. Pixels 1*, respectively. After initial cycles, the SAD of the first searching candidate in the zeroth column is generated, and the SADs of the other searching candidates are sequentially generated in the following cycles. When computing the last $N-1$ searching candidates in each column, the reference data of searching candidates in the next columns begin to be inputted through another reference input. Then, the hardware utilization is 100% except

R***YX*** is the reference pixel in the searching region, C***YX*** is the current pixel in current block, and (***X***, ***Y***) is the searching candidate, where ***Y*** is the vertical index and ***X*** is the horizontal index

| Cycle | Ref. pixel0 | Ref. pixel1 | PE Array0 | PE Array1 | PE Array2 | PE Array3 | Generated SAD |
|---|---|---|---|---|---|---|---|
| 0 | {R00, R01, R02, R03} | - | 1st RowSAD of (-2, -2) | - | - | - | - |
| 1 | {R10, R11, R12, R13} | - | 1st RowSAD of (-2, -1) | 2nd RowSAD of (-2, -2) | - | - | - |
| 2 | {R20, R21, R22, R23} | - | 1st RowSAD of (-2, 0) | 2nd RowSAD of (-2, -1) | 3rd RowSAD of (-2, -2) | - | - |
| 3 | {R30, R31, R32, R33} | - | 1st RowSAD of (-2, 1) | 2nd RowSAD of (-2, 0) | 3rd RowSAD of (-2, -1) | 4th RowSAD of (-2, -2) | (-2, -2) |
| 4 | {R40, R41, R42, R43} | {R01, R02, R03, R04} | 1st RowSAD of (-1, -2) | 2nd RowSAD of (-2, 1) | 3rd RowSAD of (-2, 0) | 4th RowSAD of (-2, -1) | (-2, -1) |
| 5 | {R50, R51, R52, R53} | {R11, R12, R13, R14} | 1st RowSAD of (-1, -1) | 2nd RowSAD of (-1, -2) | 3rd RowSAD of (-2, 1) | 4th RowSAD of (-2, 0) | (-2, 0) |
| 6 | {R60, R61, R62, R63} | {R21, R22, R23, R24} | 1st RowSAD of (-1, 0) | 2nd RowSAD of (-1, -1) | 3rd RowSAD of (-1, -2) | 4th RowSAD of (-2, 1) | (-2, 1) |
| 7 | - | {R31, R32, R33, R34} | 1st RowSAD of (-1, 1) | 2nd RowSAD of (-1, 0) | 3rd RowSAD of (-1, -1) | 4th RowSAD of (-1, -2) | (-1, -2) |
| 8 | {R02, R03, R04, R05} | {R41, R42, R43, R44} | 1st RowSAD of ( 0, -2) | 2nd RowSAD of (-1, 1) | 3rd RowSAD of (-1, 0) | 4th RowSAD of (-1, -1) | (-1, -1) |
| 9 | {R12, R13, R14, R15} | {R51, R52, R53, R54} | 1st RowSAD of ( 0, -1) | 2nd RowSAD of ( 0, -2) | 3rd RowSAD of (-1, 1) | 4th RowSAD of (-1, 0) | (-1, 0) |
| ⋮ | ⋮ | | | ⋮ | | | |

Fig. 4. Detailed data flow of the proposed *Propagate Partial SAD* architecture, where $N = 4$ and $P_h = P_v = 2$.

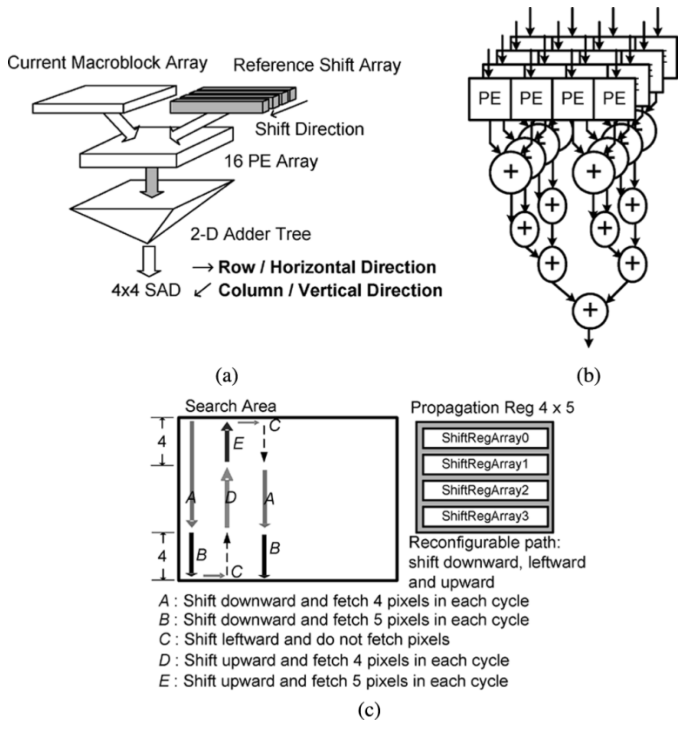
Current Macroblock Array
Reference Shift Array
Shift Direction
16 PE Array
2-D Adder Tree
4x4 SAD
→ Row / Horizontal Direction
↙ Column / Vertical Direction
(a)
PE
+
(b)
Search Area
Propagation Reg 4 x 5
ShiftRegArray0
ShiftRegArray1
ShiftRegArray2
ShiftRegArray3
Reconfigurable path: shift downward, leftward and upward
4
A
B
C
D
E
A : Shift downward and fetch 4 pixels in each cycle
B : Shift downward and fetch 5 pixels in each cycle
C : Shift leftward and do not fetch pixels
D : Shift upward and fetch 4 pixels in each cycle
E : Shift upward and fetch 5 pixels in each cycle
(c)

Fig. 5. (a) Concept, (b) hardware architecture, and (c) scan order and memory access of the proposed *SAD Tree* architecture, where $N = 4$.

the initial latency. In *Propagate Partial SAD*, by broadcasting reference pixel rows and propagating partial-row SADs in the vertical direction, it provides the advantages of fewer reference pixel registers and a shorter critical path.

## H. Proposed SAD Tree

Fig. 5(a) shows the concept of the proposed *SAD Tree* architecture. The proposed *SAD Tree* is a 2-D intra-level architecture and consists of a 2-D PE array and one 2-D adder tree with propagation registers, as shown in Fig. 5(b) and (c). Current pixels are stored in each PE, and reference pixels are stored in propagation registers for data reuse. In each cycle, $N \times N$ current and reference pixels are inputted to PEs. Simultaneously, $N$ continuous reference pixels in a row are inputted into propagation registers to update reference pixels. In propagation registers, reference pixels are propagated in the vertical direction row by row. In *SAD Tree* architecture, all distortions of a searching candidate are generated in the same cycle, and by an adder tree, $N \times N$ distortions are accumulated to derive the SAD in one cycle.

In order to provide a high utilization and data reuse, the snake scan is adopted and reconfigurable data path propagation registers are developed in the proposed *SAD Tree*, as shown in Fig. 5(c), which consists of five basic steps from A to E. The first step, A, fetches $N$ pixels in a row and the shift direction of propagation registers is downward. When calculating the last $N$ candidates in a column, one extra reference pixel is required to be inputted, that is, step B. When finishing the computation of one column, the reference pixels in the propagation registers are shifted left in step C. Because the reference data have already been stored in the propagation registers, the SAD can be directly calculated. The next two steps, D and E, are the same as steps A and B except that the shift direction is upward. After finishing the computation of one column in the search range, we execute step C and then go back to step A. This procedure will iterate until all searching candidates in the search range have been calculated. The detailed data flow is shown in Fig. 6. By snake scan and reconfigurable propagation registers, the data reuse between two successive searching candidates can be maximized, and the hardware utilization is approaching 100%.

## I. Impact of Variable Block-Size Motion Estimation in Hardware Architectures

There are many methods to support VBSME in hardware architectures. For example, we can increase the number of PEs or the operating frequency to do ME for different block sizes, respectively. One of them is to reuse the SADs of the smallest blocks, which are the blocks partitioned with the smallest block size, to derive the SADs of larger blocks. By this method, the overhead of supporting VBSME is only the slight increase of gate count, and the other factors, such as frequency, hardware utilization, memory usage, and so on, are the same as those of FBSME. When this method is adopted, the circuit for the SAD calculation is the only difference between FBSME and VBSME for hardware designs. Hence, the impact of supporting VBSME in hardware architectures is dependent on the different data flows of partial SADs. In inter-level architectures, the data flow of partial SADs is simple, where the partial SADs are stored in each PE. In intra-level architectures, there are two

R***YX*** is the reference pixel in the searching region, C***YX*** is the current pixel in current block, and (***X***, ***Y***) is the searching candidate, where ***Y*** is the vertical index and ***X*** is the horizontal index

| Cycle | Step | Read from Memory | ShiftRegArray0 | ShiftRegArray1 | ShiftRegArray2 | ShiftRegArray3 | Generated SAD |
|---|---|---|---|---|---|---|---|
| 0 | A | {R00, R01, R02, R03, - } | - | - | - | - | - |
| 1 | A | {R10, R11, R12, R13, - } | {R00, R01, R02, R03, - } | - | - | - | - |
| 2 | A | {R20, R21, R22, R23, - } | {R10, R11, R12, R13, - } | {R00, R01, R02, R03, - } | - | - | - |
| 3 | A | {R30, R31, R32, R33, - } | {R20, R21, R22, R23, - } | {R10, R11, R12, R13, - } | {R00, R01, R02, R03, - } | - | (-3, -3) |
| 4 | A | {R40, R41, R42, R43, - } | {R30, R31, R32, R33, - } | {R20, R21, R22, R23, - } | {R10, R11, R12, R13, - } | {R00, R01, R02, R03, - } | (-3, -2) |
| 5 | B | {R50, R51, R52, R53, R54} | {R40, R41, R42, R43, - } | {R30, R31, R32, R33, - } | {R20, R21, R22, R23, - } | {R10, R11, R12, R13, - } | (-3, -1) |
| 6 | B | {R60, R61, R62, R63, R64} | {R50, R51, R52, R53, R54} | {R40, R41, R42, R43, - } | {R30, R31, R32, R33, - } | {R20, R21, R22, R23, - } | (-3, 0) |
| 7 | B | {R70, R71, R72, R73, R74} | {R60, R61, R62, R63, R64} | {R50, R51, R52, R53, R54} | {R40, R41, R42, R43, - } | {R30, R31, R32, R33, - } | (-3, 1) |
| 8 | B | {R80, R81, R82, R83, R84} | {R70, R71, R72, R73, R74} | {R60, R61, R62, R63, R64} | {R50, R51, R52, R53, R54} | {R40, R41, R42, R43, - } | (-3, 2) |
| 9 | C | - | {R80, R81, R82, R83, R84} | {R70, R71, R72, R73, R74} | {R60, R61, R62, R63, R64} | {R50, R51, R52, R53, R54} | (-2, 2) |
| 10 | D | {R41, R42, R43, R44, - } | {R81, R82, R83, R84, - } | {R71, R72, R73, R74, - } | {R61, R62, R63, R64, - } | {R51, R52, R53, R54, - } | (-2, 1) |
| 11 | E | {R31, R32, R33, R34, R35} | {R71, R72, R73, R74, - } | {R61, R62, R63, R64, - } | {R51, R52, R53, R54, - } | {R41, R42, R43, R44, - } | (-2, 0) |
| 12 | E | {R21, R22, R23, R24, R25} | {R61, R62, R63, R64, - } | {R51, R52, R53, R54, - } | {R41, R42, R43, R44, - } | {R31, R32, R33, R34, R35} | (-2, -1) |
| 13 | E | {R11, R12, R13, R14, R15} | {R51, R52, R53, R54, - } | {R41, R42, R43, R44, - } | {R31, R32, R33, R34, R35} | {R21, R22, R23, R24, R25} | (-2, -2) |
| 14 | E | {R01, R02, R03, R04, R05} | {R41, R42, R43, R44, - } | {R31, R32, R33, R34, R35} | {R21, R22, R23, R24, R25} | {R11, R12, R13, R14, R15} | (-2, -3) |
| 15 | C | - | {R31, R32, R33, R34, R35} | {R21, R22, R23, R24, R25} | {R11, R12, R13, R14, R15} | {R01, R02, R03, R04, R05} | (-1, -3) |
| 16 | A | {R42, R43, R44, R45, - } | {R32, R33, R34, R35, - } | {R22, R23, R24, R25, - } | {R12, R13, R14, R15, - } | {R02, R03, R04, R05, - } | (-1, -2) |
| 17 | B | {R52, R53, R54, R55, R56} | {R42, R43, R44, R45, - } | {R32, R33, R34, R35, - } | {R22, R23, R24, R25, - } | {R12, R13, R14, R15, - } | (-1, -1) |
| ⋮ | ⋮ | | | ⋮ | | | ⋮ |

Fig. 6. Detailed data flow of the proposed *SAD Tree* architecture, where $N = 4$ and $P_h = P_v = 3$.

kinds of data flows of partial SADs, i.e., propagating with propagation registers or no partial SADs. In the following, the impact of supporting VBSME with three different data flows is analyzed. We assume that the size of an MB is $N \times N$, and it can be divided into $n \times n$ smallest blocks of size $(N/n) \times (N/n)$.

*1) Data Flow I—Storing in PEs:* In inter-level architectures, each PE is responsible for computing the distortion and accumulating the SAD of a searching candidate, as shown in Fig. 7(a). The partial SADs are stored in PEs. When supporting VBSME, the number of partial SADs is increased from one to $n^2$. In order to store these partial SADs, more data buffers are required in each PE, as shown in Fig. 7(b). Besides, there are extra two $n^2$-to-1 and 1-to-$n^2$ multiplexers in each PE for the selection of partial SADs. All PEs of inter-level architectures, including 1DInterYSW, 2DInterYH, and 2DInterLC, should be replaced with that in Fig. 7(b) to support VBSME. The number of bits for the data buffer in each PE is increased from $log_2 N^2 + 8$ to $n^2 \times (log_2(N/n)^2 + 8)$, where $N^2$ and $(N/n)^2$ are the number of pixels in one block, and 8 is the word length of one pixel. For example, if an MB is $16 \times 16$ and can be divided into 16 $4 \times 4$ blocks, the size of the data buffer is increased from 16 b to 16 $\times$ 12 b in one PE.

*2) Data Flow II—Propagating With Propagation Registers:* In intra-level architectures, partial SADs can be accumulated and propagated with propagation registers. Each PE computes the distortion of one corresponding current pixel in current MB. By propagation adders and registers, the partial SAD is accumulated with these distortions. The hardware architecture of *Propagate Partial SAD* is a typical example, as shown in Fig. 3(b), where the partial SADs are propagated in the vertical direction. When supporting VBSME, more propagation registers are required to store partial SADs of the smallest blocks. In each propagating direction, the number of propagation registers are $n$ times of that in the original for the $n$ smallest blocks in the other direction. For example, in Fig. 8(a), when supporting VBSME, because there are four

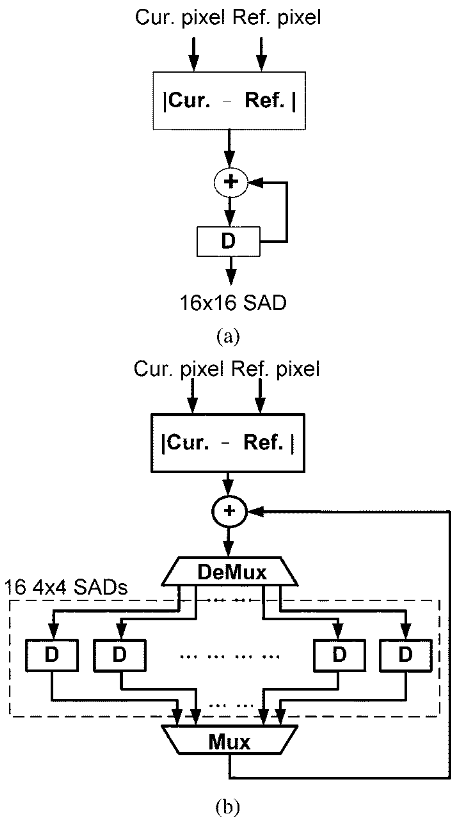

Fig. 7. Hardware architecture of inter-level PE with Data Flow I for (a) FBSME, where $N = 16$, and (b) VBSME, where $N = 16$ and $n = 4$.

smallest blocks in the horizontal direction, we have to propagate four partial SADs of the smallest blocks in the vertical direction at the same time in order to reuse them. Therefore, the propagation registers are duplicated four copies, and the number of propagation registers increases from 16 to 64.

Furthermore, some extra delay registers are required in order to synchronize the timing of the SADs of the smallest blocks, as

TABLE I
PARALLELISM, CYCLES, LATENCY, AND DATA FLOW OF EIGHT HARDWARE ARCHITECTURES

| Name | No. of PEs | Operating Cycles (Cycles/Macroblock) | Latency (Cycles) | Data Flow |
|---|---|---|---|---|
| 1DInterYSW [5] | $2P_h$ | $N^2\times 2P_v+2P_h$ | $N^2$ | Data Flow I |
| 2DInterYH [7] | $2P_h\times 2P_v$ | $2N^2$ | $N^2$ | Data Flow I |
| 2DInterLC [8] | $2P_h\times 2P_v$ | $2N^2$ | $2N^2$ | Data Flow I |
| 2DIntraVS [9] | $N^2$ | $2P_h\times 2P_h+N\times 2P_v$ | $N\times 2P_v$ | Data Flow III |
| 2DIntraKP [6] | $N^2$ | $2P_v\times(N+2P_h)+N$ | $3N$ | Data Flow II |
| 2DIntraHL [10] | $N^2$ | $(2P_v+N-1)\times(2P_h+N-1)$ | $2N+(N-1)\times(2P_h+N-2)$ | Data Flow II |
| *Propagate Partial SAD* | $N^2$ | $2P_h\times 2P_v+N-1$ | $N$ | Data Flow II |
| *SAD Tree* | $N^2$ | $2P_h\times 2P_v+N-1$ | $N$ | Data Flow III |

Note: The analysis is based on a video coding system with macroblock pipelining architecture.

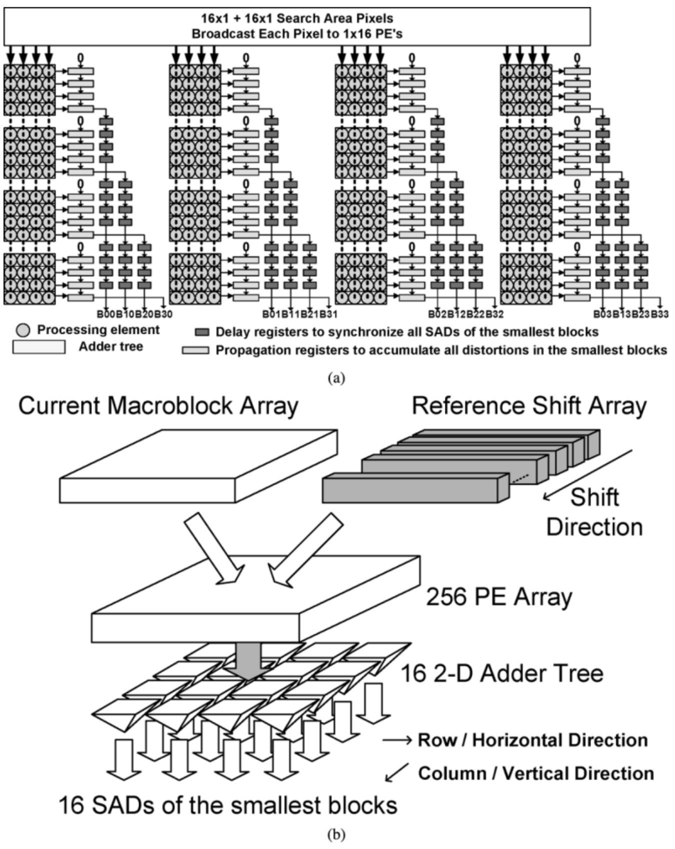
16x1 + 16x1 Search Area Pixels
Broadcast Each Pixel to 1x16 PE's
B00B10B20B30
B01B11B21B31
B02B12B22B32
B03B13B23B33
Processing element
Adder tree
Delay registers to synchronize all SADs of the smallest blocks
Propagation registers to accumulate all distortions in the smallest blocks
(a)
Current Macroblock Array
Reference Shift Array
Shift Direction
256 PE Array
16 2-D Adder Tree
Row / Horizontal Direction
Column / Vertical Direction
16 SADs of the smallest blocks
(b)

Fig. 8. Hardware architecture for VBSME of (a) the proposed *Propagate Partial SAD* architecture with Data Flow II and (b) the proposed *SAD Tree* architecture with Data Flow III, where $N = 16$ and $n = 4$.

shown in Fig. 8(a). In each propagating direction, the number of delay registers is equal to $n\times(n(n-1)/2)\times(N/n)$. That is, in Fig. 8(a), there are four delay register arrays. In each delay register array, the top smallest block requires $(4-1)\times 16/4$ delay registers, the second smallest block requires $(4-2)\times 16/4$ delay registers, the third smallest block requires $(4-3)\times 16/4$ delay registers, and the bottom smallest block does not require delay registers. Totally, there are $4\times(4(4-1)/2)\times(16/4)$ delay registers. In addition to *Propagate Partial SAD*, 2DIntraHL also propagate the partial SADs in the vertical direction, and in 2DIntraKP, the partial SADs are propagated in two directions. In these three architectures, extra propagation and delay registers are required in their propagating directions when VBSME is supported.

*3) Data Flow III—No Partial SADs:* In intra-level architectures, it is possible that no partial SADs are required to be stored, such as *SAD Tree*. Each PE computes the distortion of one current pixel for a searching candidate, and the total SAD is accumulated by an adder tree in one cycle, as shown in Fig. 5(a). Because there is no partial SAD in this architecture, there is no registers overhead to store partial SADs when supporting VBSME. The adder tree is the one to be reorganized to support VBSME, as shown in Fig. 8(b), that is, we partition the 2-D adder tree in order to get the SADs of the smallest blocks first, and then based on these SADs, to derive the SADs of large blocks. Although there is no additional register overhead, the adder tree additions required to support VBSME do require additional area.

## III. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we discuss the performances of the six previous works and our two proposed hardware architectures, *Propagate Partial SAD* and *SAD Tree*, in a video coding system with MB pipelining architectures, where several MBs are processed at the same time but in different functional modules, such as ME and reconstruction. First of all, we summarized the characteristics of eight ME hardware architectures in Tables I and II. In Table I, the number of PEs, required cycles, and latency are discussed to show the degree of parallelism and utilization, and the data flow of partial SADs are listed to categorize the impact of supporting VBSME in each hardware. In Table II, the current buffer, reference buffer, and memory bit width are used to evaluate the tradeoff between data buffer and memory usage. Note that because we reuse the SADs of the smallest blocks to derive the SADs of larger blocks, the impact of supporting VBSME in hardware architectures is only the increase of chip area. The other factors are the same for FBSME and VBSME.

Besides the theoretical analysis in Table I and II, an example is also given to provide a practical comparison in Table III . The specifications of ME are as follows. The MB size is 16 × 16,

TABLE II
DATA BUFFER AND MEMORY BIT WIDTH OF EIGHT HARDWARE ARCHITECTURES

| Name | Current Buffer (Pixels) | Reference Buffer (Pixels) | Memory Bitwidth (Bits/Cycle) |
|---|---|---|---|
| 1DInterYSW [5] | $2P_h - 1$ | - | $(2P_h/N+2)\times 8$ |
| 2DInterYH [7] | $N^2 - 1$ | - | $(2(2P_v/N)\times(2P_h/N)+1)\times 8$ |
| 2DInterLC [8] | - | $2P_h\times 2P_v$ | $(2(2P_v/N)\times(2P_h/N)+1)\times 8$ |
| 2DIntraVS [9] | $N^2$ | $N\times(4P_v+N-2)$ | $\{1,(2)^*\}\times 8$ |
| 2DIntraKP [6] | $N^2$ | $N\times(N-1)$ | $N\times 8$ |
| 2DIntraHL [10] | $N^2$ | $N^2+(N-1)\times(2P_h-2)$ | $1\times 8$ |
| *Propagate Partial SAD* | $N^2$ | - | $\{N,\ (2N)^*\}\times 8$ |
| *SAD Tree* | $N^2$ | $N\times(N+1)$ | $\{N,(N+1)^*\}\times 8$ |

Note: $(.)^*$ is the worst case.

TABLE III
COMPARISON OF EIGHT HARDWARE ARCHITECTURES FOR FBSME AND VBSME IN SIX CRITERIA

| Name | Area(FBS) (Kgates) | Area(VBS) (Kgates) | Freq. (MHz) | Bitwidth (Bits/Cycle) | Bandwidth (Kbits per MB) | Latency (Cycles) | Util. (%) |
|---|---|---|---|---|---|---|---|
| 1DInterYSW [5] | 61.9 | 359.6 | 222.7 | 80 | 1,290 | 256 | 99.2 |
| 2DInterYH [7] | 2,907.0 | 20,422.0 | 6.9 | 520 | 260 | 256 | 50.0 |
| 2DInterLC [8] | 4,055.0 | 21,647.0 | 6.9 | 520 | 260 | 512 | 50.0 |
| 2DIntraVS [9] | 301.3 | 318.7 | 127.7 | 16 | 90 | 1024 | 88.9 |
| 2DIntraKP [6] | 108.8 | 159.1 | 123.7 | 128 | 1,146 | 48 | 89.3 |
| 2DIntraHL [10] | 231.9 | 254.6 | 152.5 | 8 | 90 | 2162 | 72.5 |
| *Propagate Partial SAD* | 66.6 | 81.5 | 110.8 | 256 | 1,259 | 16 | 99.8 |
| *SAD Tree* | 88.4 | 88.6 | 110.8 | 136 | 1,044 | 16 | 99.8 |

Note: The specification is D1 size, 10 fps, and the search range is [-64, 64) in a video coding system with macroblock pipelining architecture.

and the search range is $P_h = 64$ and $P_v = 32$. The frame size is D1 size, $720 \times 480$. When VBSME is supported, a MB can be partitioned at most to 16 $4 \times 4$ blocks. We use Verilog-HDL and SYNOPSYS Design Compiler with ARTISAN UMC 0.18-$\mu$m cell library to implement each hardware architecture. Because the timing of the critical path in some architectures is too long, which means the maximum operating frequency is limited without modifying the architecture, the frame rate is set as only 10 fps. The discussions of these experimental results are shown below.

## *A. Area and Required Frequency*

The required frequency is dominated by the degree of parallelism in a hardware architecture. The smaller the degree of parallelism is, the higher the required frequency is. In Table I and III, because the degree of parallelism in 1DInterYSW is the smallest, the required frequency is the highest. On the contrary, because 2DInterYH and 2DInterLC have the largest degrees of parallelism among eight hardware architectures, their required frequencies are the smallest.

There are two columns of chip area in Table III. One is for FBSME and the other is for VBSME. The area consists of PE array, Current Buffer and Reference Buffer. Therefore, for FBSME, the area of 1DInterYSW is the smallest. The area of 2DInterLC is larger than that of 2DInterYH because of the huge reference buffer, as shown in Table II. For the same reason, the area of 2DIntraVS is also larger than that of proposed *SAD*

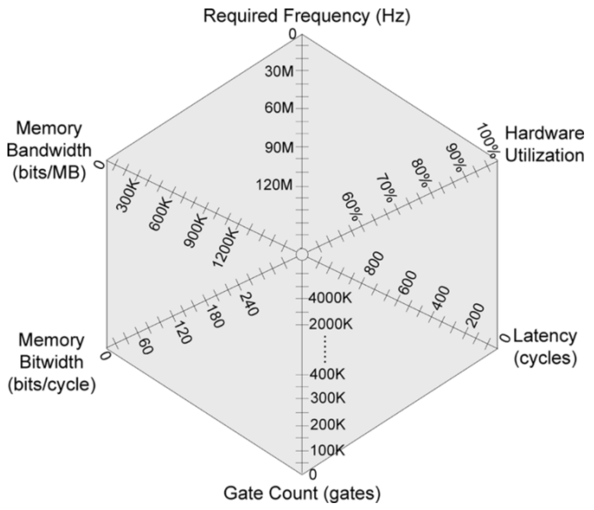

Fig. 9. Hexagonal plot for the comparison of motion estimation hardware architecture.

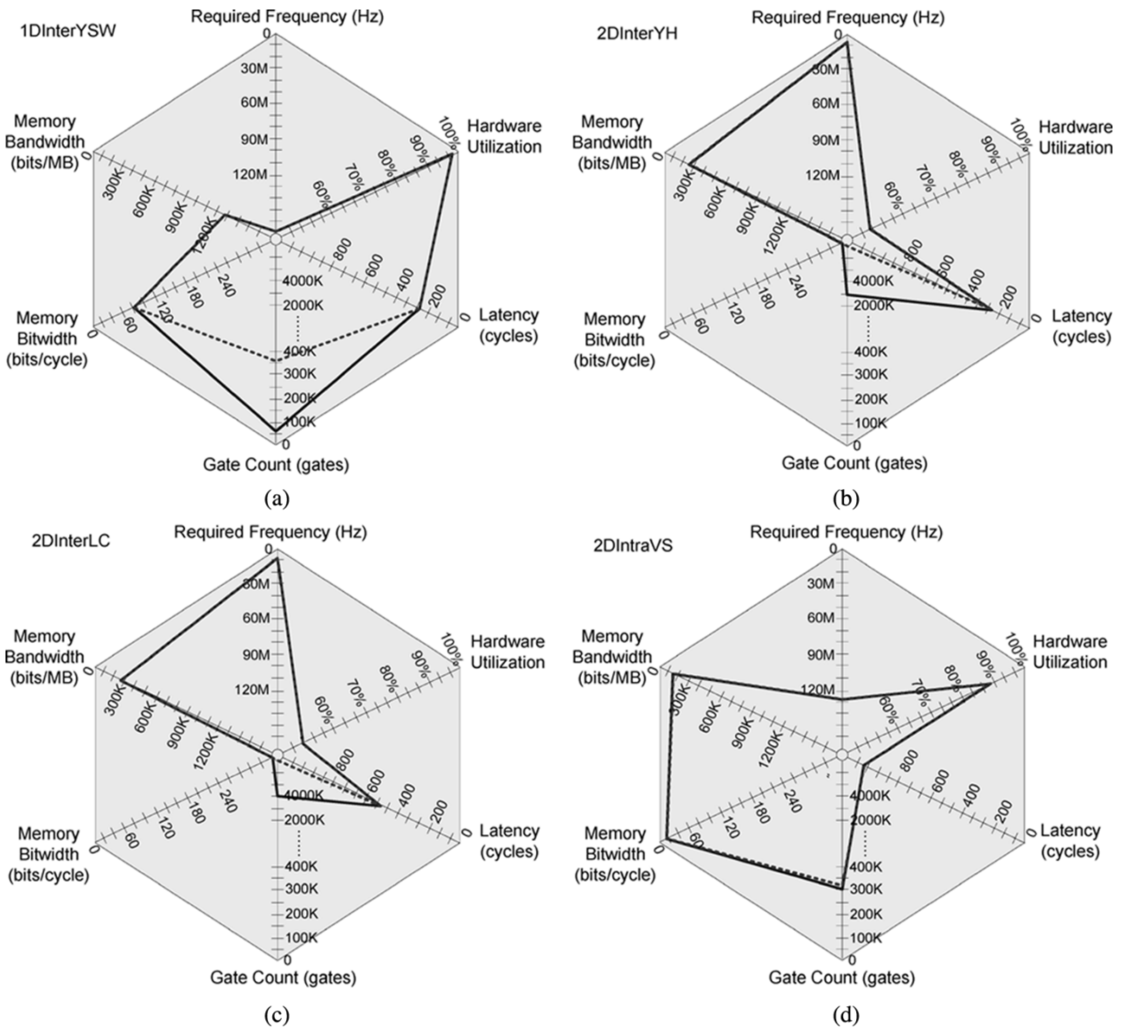

Fig. 10. Hexagonal plots of eight hardware architectures for fixed block-size motion estimation and variable block-size motion estimation. (a) 1DInterYSW. (b) 2DInterYH. (c) 2DInterLC. (d) 2DIntraVS.

*Tree*. The impact of supporting VBSME is apparently observed in the other column of area, the area for VBSME, in Table III. Among these eight hardware architectures, all inter-level architectures with Data Flow I increase gate count dramatically. The chip area is five times that in FBSME at least. In intra-level architectures with Data Flow II, the increase in gate count is much smaller, and the ratio increases from 9.8% to 46.3%. If intra-level architectures have Data Flow III, the area overheads are 0.2% and 5.8% for the proposed *SAD Tree* and 2DIntraVS, respectively.

Due to the characteristics of inter-level architectures, the chip area overhead of inter-level architectures for VBSME is large. In three inter-level architectures, the overhead of 2DInterLC is the smallest because of a lot of propagation registers in 2DInterLC compared to other architectures. A similar condition occurs in the comparison of 2DIntraHL and *Propagate Partial SAD*, so the chip area overhead of supporting VBSME in 2DIntraHL is smaller than that in *Propagate Partial SAD*. 2DIntraKP has the largest chip area overhead in three intra-level architectures with Data Flow II, because the partial SADs are propagated in two directions. In 2DIntraVS, although there is no partial SAD to be stored, the partial SADs are accumulated by propagation. Then there is a long critical path in 2DIntraVS. In *SAD Tree*, SAD is directly summed by a 2-D adder tree, so the chip area overhead of 2DIntraVS is larger than that of *SAD Tree*. Moreover, the chip area overhead of *SAD Tree* is also the smallest in the eight hardware architectures.

## B. Latency

The latency is defined as the number of start-up cycles that a hardware takes to generate the first SAD. The latency is more important for a video coding system than single ME module, because the latency affects the effect of parallel computation. In a video coding system, we usually use a large degree of parallelism to achieve real-time computation. However, if a module has a long latency and it cannot be shortened by parallel architectures, then the effect of parallel computation is reduced, that is, a shorter latency is better for video coding systems. There are two factors to affect the latency. One is the type of a hardware architecture. In inter-level architectures, the latency is at least $N \times N$, as shown in Tables I and III. Conversely, there is no constraint in intra-level architectures. The other factor to affect the latency is the memory bit width and reference buffer, as shown in Table II. If there is a large reference buffer but fewer

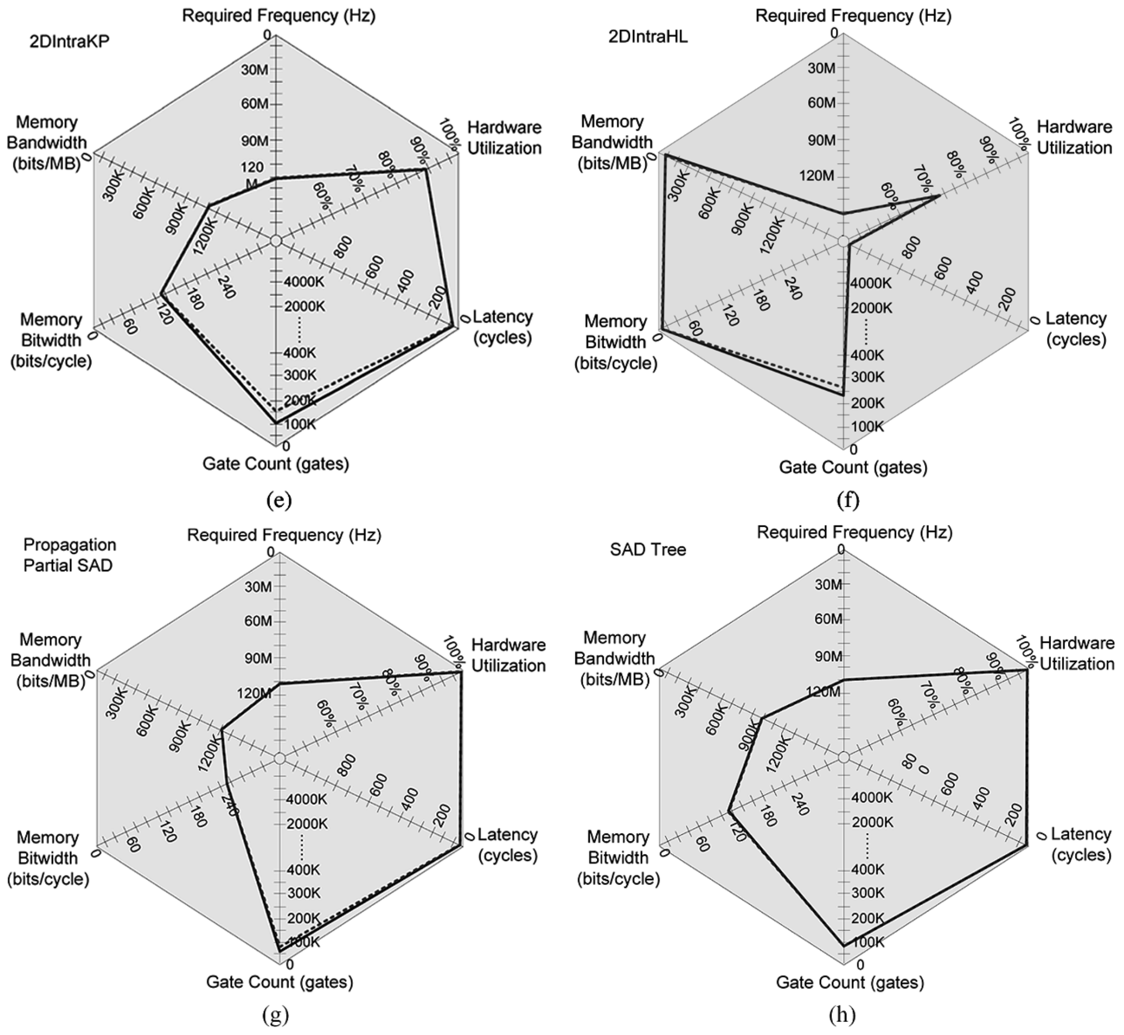

Fig. 10. *(Continued.)* Hexagonal plots of eight hardware architectures for fixed block-size motion estimation and variable block-size motion estimation. (e) 2DIntraKP (f) 2DIntraHL. (g) *Propagate Partial SAD*. (h) *SAD Tree*.

memory bit width, for example, 2DIntraVS or 2DIntraHL, the architecture takes more initial cycles to load reference data into the reference buffer. Compared to these hardware architectures, the other intra-level architectures, such as proposed *Propagate Partial SAD* and *SAD Tree*, have shorter latencies.

## C. Utilization

In general, inter-level architectures can continuously compute MB by MB, so the initial cycles can be neglected and the utilization will be 100%. For instance, if the frame-level pipeline [19] is adopted, the hardware utilization of ME module will be 100%. However, this feature is not necessary for a video coding system with MB pipelining architectures, because MB pipelining interrupts the computation of motion estimation. Therefore, we defined the utilization as *Computing cycles/Operating cycles for a MB*. Then, the utilization is dominated by the operating cycles. The operating cycles include three parts, latency, computing cycles, and bubble cycles. Computation cycles are the number of cycles when we can get one SAD at least. That is, if the utilization is 100%, we can get one SAD in each cycle at least. Fewer operating cycles will let the penalty of the latency be apparent. The more bubble cycles are, the lower the utilization is. 2DInterYH, and 2DInterLC are two examples which have low utilizations because of their fewer operating cycles, as shown in Tables I and III. In our proposed two hardware architectures, there are shorter latencies and no bubble cycles, so their utilizations can achieve 99.8%.

## D. Memory Usage

Memory usage consists of two parts, i.e., memory bit width and memory bandwidth. Memory bit width is defined as the number of bits which a hardware has to access from memory in each cycle, and memory bandwidth is redefined as the number of bits which a hardware has to access from memory for an MB. Memory bandwidth affects the loading of a system bus without on-chip memory or the power of on-chip memory, and memory bit width is the key to the data arrangement of on-chip memories. Memory bit width and bandwidth are affected by the data reuse scheme and operating cycles. From Table II, because 2DIntraHL and 2DIntraVS have larger reference buffers to reuse reference pixels, the required memory bit widths and bandwidths are fewer. In 2DInterYH and 2DInterLC, because of their high degrees of parallelism as shown in Table I, the large memory bit widths are required. However, the memory bandwidths are much fewer because of their fewer operating cycles. The data reuse schemes in 2DIntraKP and the proposed two hardware architectures are similar, and the differences of these three architectures are the different data reuse schemes when changing columns.

## E. Hexagonal Plot

Fig. 9 is a hexagonal plot. It is used to visualize the characteristics of ME hardware architectures. The six design criteria in Table III are shown in the hexagonal plot, in which the closer the point is to the center, the worse the performance is. Therefore, the advantages and disadvantages of ME hardware architectures can be observed easily in a hexagonal plot. Note that, in various video coding systems or hardware system platforms, the weighting of each axis will be very different, and we can use these hexagonal plots to select the optimal architecture based on different constraints for the system integration.

In Fig. 10, there are two lines in one hexagonal plot. One is the solid line for FBSME, and the other is the dotted line for VBSME. From Fig. 10, we can easily observe the characteristics of each hardware and see that the similar architectures has similar hexagonal plots and characteristics. 1DInterYSW provides low hardware cost, high utilization, smaller memory bit width, and short latency. The hexagonal plots of 2DInterYH and 2DInterLC are similar because they are 2-D inter-level architectures with large propagation registers and provide lower required frequencies and lower memory bandwidths. In addition, 2DIntraHL and 2DIntraVS are 2-D intra-level architectures with large propagation registers so they have good performances in the memory usages, and similar hexagonal plots. The hexagonal plots of 2DIntraKP, the proposed *SAD Tree*, and *Propagate Partial SAD* are very alike. However, because the worst case is shown in the dimension of memory bit width, the memory bit width of *Propagate Partial SAD* looks worse than others, and, in the normal case, the performance of *Propagate Partial SAD* is much better than Fig. 10(g).

For VBSME, the changes of hexagonal plots are also dependent on the data flow of partial SADs, as shown in Fig. 10. The axis of area in three inter-level architectures are changed largely in Fig. 10(a)–(c). In intra-level architectures, the changes in the axis of area are not apparent. *SAD Tree* almost has the same performance because its chip area overhead for supporting VBSME is only 0.2%.

## IV. Hardware Architecture of H.264 Integer Motion Estimation (IME)

In this section, based on the above analysis, we propose an ME hardware for H.264/AVC IME as an example. Our specification is that two frame sizes are supported in our specification. One is the D1 Format with four reference frames, 30 fps. In the previous frame, the search range is [−64, 64) and [−32, 32) in the horizontal and vertical directions. In the rest frames, the search range is [−32, 32) and [−16, 16) in the horizontal and vertical directions. The other is 720 p with one reference frame, 30 fps. The search range is the same as that of the previous frame in D1 Format.

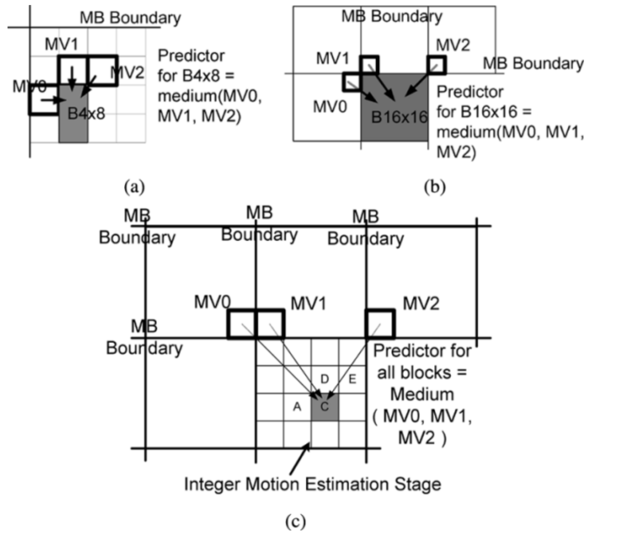
MB Boundary
MV1
MV0
MV2
B4x8
Predictor for B4x8 = medium(MV0, MV1, MV2)
(a)
MB Boundary
MV1
MV2
MB Boundary
MV0
B16x16
Predictor for B16x16 = medium(MV0, MV1, MV2)
(b)
MB Boundary
MB Boundary
MB Boundary
MB Boundary
MV0
MV1
MV2
D
E
A
C
Predictor for all blocks = Medium ( MV0, MV1, MV2 )
Integer Motion Estimation Stage
(c)

Fig. 11. Motion vector predictor for (a) the 4 × 8 block, (b) the 16 × 16 block, and (c) the modified motion vector predictor for all blocks.

TABLE IV
Number of Registers in $M$-Parallel *Propagate Partial SAD* and $M$-Parallel *SAD Tree*

| The Number of PEs | *Propagate Partial SAD* (bits) | *SAD Tree* (bits) |
|---|---|---|
| 1×256 PEs | 3,920 | 4,224 |
| 2×256 PEs | 5,792 | 4,480 |
| 4×256 PEs | 9,536 | 4,992 |
| 8×256 PEs | 17,024 | 6,016 |

In our specification, the computation complexity of H.264 is 2.4 tera instructions per second and 3.8 tera bytes per second in D1 Format and dominated by IME, which is estimated by instruction profiling of reference software, JM7.3 [20] and the simulation environment is P4-1.8 GHz and 1 GB memory with Redhat Linux 6.2. In general, the ultra large computation complexity can be solved by the parallel computation, but the huge external memory bandwidth cannot. Therefore, the huge memory bandwidth is a difficult challenge for hardware design. In addition, there are two problems. First, because of VBSME and Lagrangian mode decision, the data dependency of the motion vector predictor prohibits the parallel computation between the smaller blocks in an MB. Second, when a high processing ability is necessary, the hardware cost of ME hardware architectures with high degrees of parallelism must also be discussed. In the following subsections, we propose a hardware architecture for H.264/AVC IME to reduce memory bandwidth and solve the problems mentioned above.

## A. Hardware Architecture Design

*1) Modified Algorithm:* First, we divide the computation of ME into two parts, integer-pixel ME and fractional-pixel ME (FME), and propose two individual hardware accelerators for IME and FME [21], respectively. This is because the utilization of hardware accelerators can be significantly improved by this

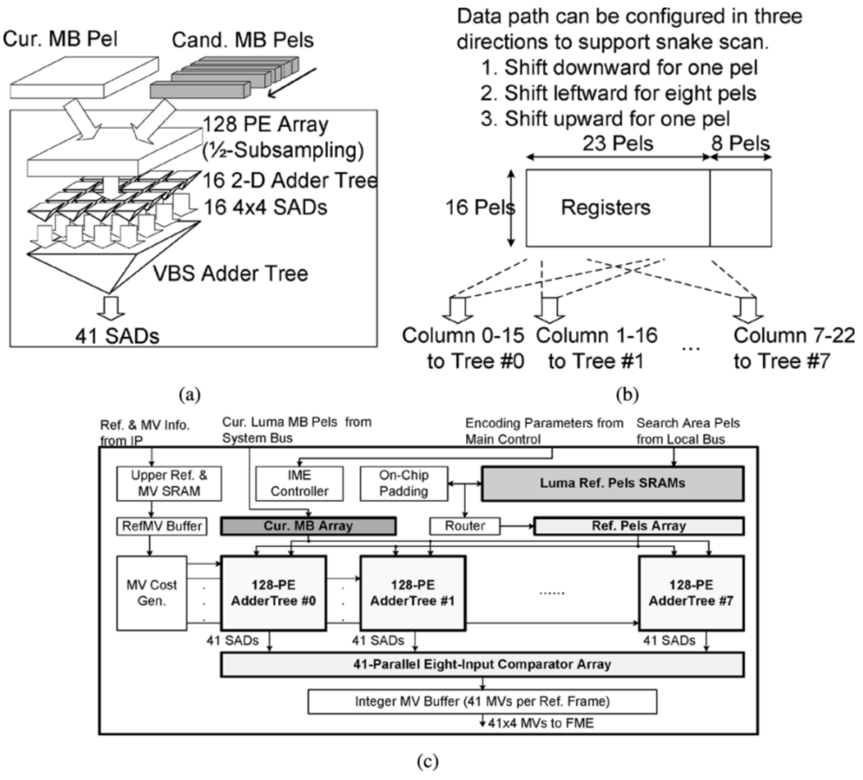

Fig. 12. Hardware architecture of H.264 integer motion estimation. (a) Architecture of SAD Tree with subsample and truncation and VBS Adder Tree. (b) Share reference buffer with reconfigurable data paths and data sharing. (c) Overview of H.264 integer motion estimation.

way, and in this paper, we only focus on the part of IME. Second, in the original Lagrangian mode decision, the MV predictor of a block is the medium MV among the MVs of top, top-right, left neighboring $4 \times 4$ blocks, as shown in Fig. 11(a) and (b), but, in the parallel computation of hardware architectures, the coding modes of the neighboring $4 \times 4$ blocks can not be decided in parallel, especially when the block size is $4 \times 4$. Hence, the parallel computation conflicts with the Lagrangian mode decision in VBSME. In order to solve this conflict, we modify the MV prediction of Lagrangian mode decision. The median of the MVs of the top-right, top, top-left blocks is used instead of the original. The modified MV prediction is shown in Fig. 11(c). By this modification, not only does the parallel computation in an MB become feasible, but the quality is also maintained [22].

*2) Hardware Architecture With $M$-Parallelism:* Based on the analysis in Sections II and III, our proposed two hardware architectures have better performances for VBSME and provide a small chip area and high utilization. We select them to further discuss the impact of parallelism, because a high degree of parallelism is required for our specification. In our specification, we require eight sets of *Propagate Partial SAD* or *SAD Tree* to achieve the realtime computation. Therefore, eight sets of *Propagate Partial SAD* and *SAD Tree*, which can process eight successive candidates in a row at the same time, are combined as *Eight-Parallel Propagate Partial SAD* and *Eight-Parallel SAD Tree*, respectively.

When $M$-parallel architectures are required, the propagation registers for partial SADs cannot be shared, and they should be duplicated. However, if the registers are used to store reference pixels, the registers can be shared in different PE arrays, and only a few extra registers are required. For example, in *Eight-Parallel Propagate Partial SAD*, eight sets of propagation registers for partial SADs are required, but, in *Eight-Parallel SAD Tree*, the number of registers for storing reference pixels is only increased to 16 pixels in one row. Then, the increasing ratio is only $(16-1+16)/16$, not 8. Table IV shows the number of required registers in these two architectures with different degrees of parallelism. The increasing ratio of *M-parallel Propagate Partial SAD* is much larger than that of *M-parallel SAD Tree*. Hence, *SAD Tree* is much more suitable for high resolution or large computation complexity. Although *Propagate Partial SAD* has no advantages of a high degree of parallelism, for low resolution or small computation complexity, *Propagate Partial SAD* still has a similar performance and shorter critical path compared to *SAD Tree*.

*3) Applied Techniques for Hardware Cost Reduction:* Due to a high degree of parallelism, the required chip area is huge even if the proposed *SAD Tree* has the smallest chip area. We adopt several techniques to reduce the hardware cost and memory bandwidth. The first is the subsample [23]. The pattern of the subsample is interleaved and is similar to the pattern of chess. The second method is pixel truncation [24]. We only preserve 5-b precision for all current and reference pixels. Fig. 12(a) shows the modified *SAD Tree* with subsample and pixel truncation and a variable block-size adder tree (VBS Adder Tree), which is used to reuse the SADs of the smallest blocks to derive the SADs of larger blocks. The shared reference buffer of *Eight-Parallel SAD Tree* not only can provide the required reference pixels for eight sets of SAD Tree, as shown

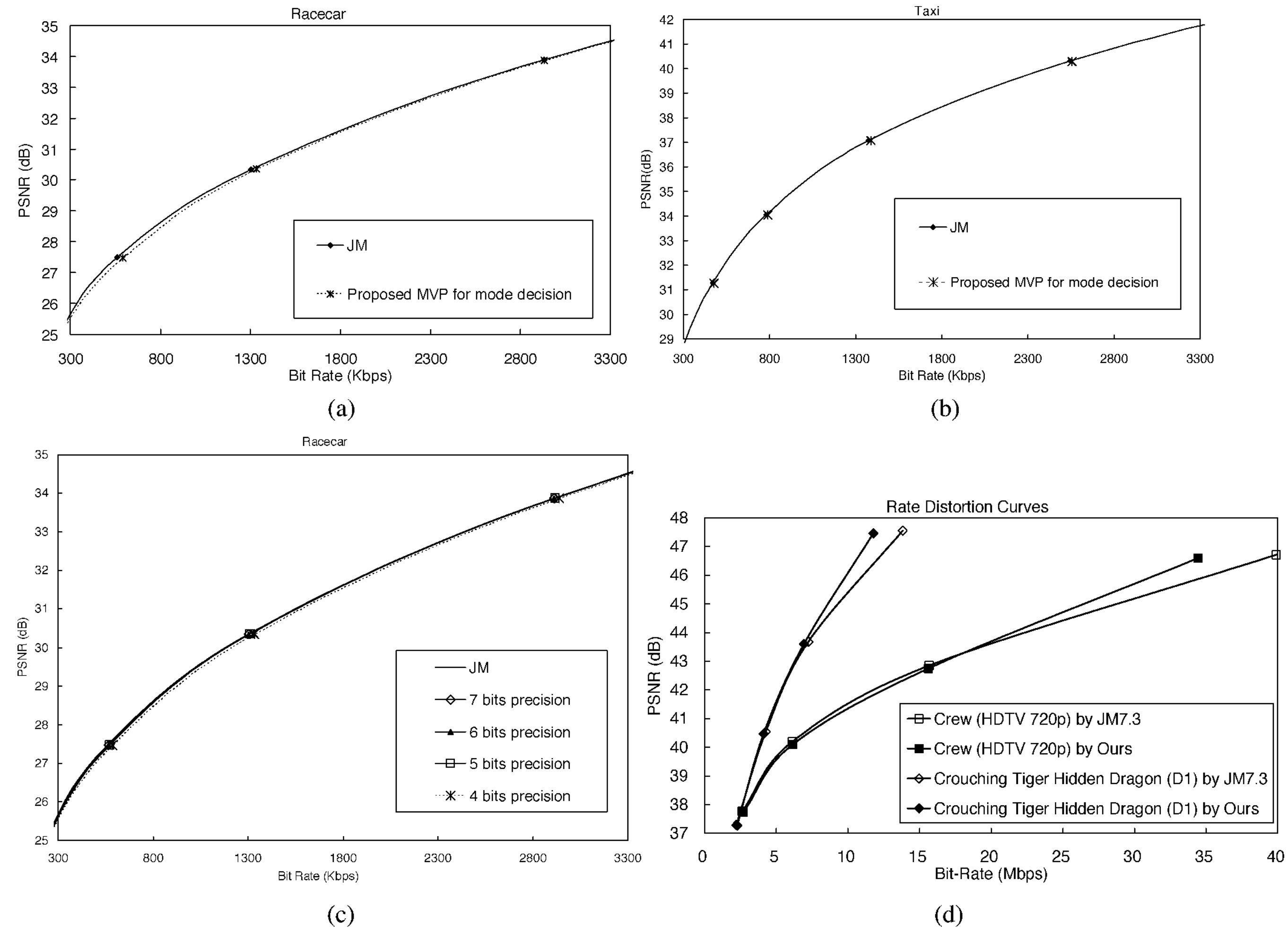

Fig. 13. Comparison of RD curves between JM7.3 and our proposed encoder. (a) The comparison of modified motion vector prediction in Racecar. (b) The comparison of modified motion vector prediction in Taxi. (c) The comparison of pixel truncation in Racecar. (d) The comparison of our proposed encoder.

in Fig. 12(b), but also save the on-chip memory bandwidth. A moving window is also adopted to save the computation complexity and reduce on-chip memory bandwidth. The search range is partitioned into four regions, i.e., [−64,15], [−48,31], [−32,47], and [−16,63] in the horizontal direction. According to an MV predictor, one of four regions is selected and the searching candidates in this region are computed. Besides, Level C Search Area Reuse [25], [26] and on-chip frame boundary padding are applied to reduce the off-chip memory bandwidth.

*4) Hardware Architecture of IME:* Fig. 12(c) shows the whole architecture of H.264 IME. *IME Controller* controls the actions of all submodules. *On-Chip Padding* is responsible for the extension of a frame boundary. *Upper Ref. & MV SRAM* and *RefMV Buffer* are used to store MVs of the top MB row to generate an MV predictor. MV cost in Lagrangian mode decision is calculated in *MV Cost Gen.*. *Router* is used to re-order the order of the data which are read from *Luma Ref. Pels SRAMs*. *Ref. Pels Array* is the data buffer for storing reference pixels to reuse reference pixels and reduce the memory access. *41-Parallel Eight-Input Comparator Array* is responsible for finding optimal 41 MVs, which have the smallest cost for different block sizes.

## *B. Experimental and Implementation Results*

*1) Experimental and Implementation Results:* Fig. 13(a) and (b) shows the comparisons of RD curves between JM7.3 and our proposed motion vector prediction. We have tested many sequences from QCIF to HDTV, and two of them are Racecar (720 $\times$ 288, 30 fps) and Taxi (672 $\times$ 288, 30 fps). At high bit rates (larger than 1 Mb/s), the quality loss is near zero, and, at a low bit rate, the quality is degraded 0.1 dB. The results of pixel truncation are shown in Fig. 13(c). Based on the simulation results, the degradation of 5-b precision is little, but that of 4-b precision is from 0.1 to 0.2 dB. Finally, the RD curve of our encoder chip is shown in Fig. 13(d). The test sequences are Crouching Tiger Hidden Dragon and Crew. The former is D1 Format, and the latter is HDTV Format. The coding performance of our encoder chip is competitive with that of JM7.3. Moreover, because we also refine the Lagrangian multiplier, our performance is better than that of the original at a very high bit rate.

In the hardware implementation, our specification is stated in the beginning of Section IV. In SDTV, the block size can be from 16 $\times$ 16 to 4 $\times$ 4. In HDTV, although our proposed IME architecture can support all kinds of block sizes, we only support the block modes which block sizes are larger than or equal

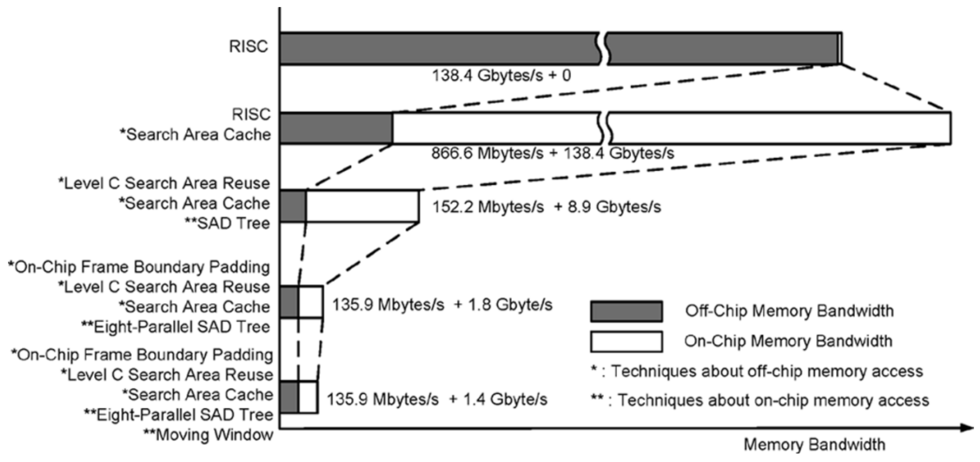

Fig. 14. Memory reduction of H.264 IME.

to 8 $\times$ 8 due to the limit of FME [22]. Verilog-HDL and SYNOPSYS Design Compiler with ARTISAN UMC 0.18-$\mu$m cell library are used to design the hardware. Because *SAD Tree* can support VBSME with less overhead and many techniques for the reduction of hardware cost are applied, the total gate count of H.264/AVC IME is only 330.2 K gates, and the operating frequency is 81 MHz at SDTV or 108 MHz at HDTV. The on-chip memory size is 208 kb.

*2) Memory Bandwidth:* Fig. 14 shows the required off-chip and on-chip memory bandwidth of four reference frames in our D1 specification. In Fig. 14, five data reuse schemes are discussed, and the related data are theoretical value, not the simulated results. The first data reuse scheme is the simplest one. There is only one RISC in the hardware without any cache. Because of no on-chip memories as the search area cache, reference pixels are inputted directly from off-chip memories and no on-chip memory bandwidth is required. The second scheme is one RISC with Search Area Cache. By use of the on-chip memory, 866.6 Mbytes/s of the off-chip memory bandwidth is required, but the on-chip memory bandwidth is increased to 138.4 Gbytes/s. However, this tradeoff is worth it because the access of off-chip memories takes much more power and cycles than those of on-chip memories, in general. The third data reuse scheme is the second one with Level C Search Area Reuse, and then the off-chip memory bandwidth can be reduced again. Thus, only 152.2 Mbytes/s of the off-chip memory bandwidth is necessary. On the other hand, because of the data reuse in *SAD Tree*, the on-chip memory bandwidth is only 8.9 Gbyte/s. In the fourth data reuse scheme, on-chip frame boundary padding is adopted, so the off-chip memory bandwidth is reduced again. As for on-chip memory bandwidth, by *Eight-Parallel SAD Tree*, 1.8 Gbytes/s of the on-chip memory bandwidth is required. Finally, because the moving widow is applied in the fifth data reuse scheme, the on-chip memory bandwidth is reduced to 1.4 Gbytes/s.

In summary, we saved 99.9% off-chip memory bandwidth compared to the data reuse scheme of RISC. Compared to the general data reuse scheme, Level C Search Area Reuse, the off-chip memory bandwidth is reduced to 89.6%. Furthermore, 99.22% on-chip memory bandwidth is saved by our proposed *Eight-Parallel SAD Tree* and moving window.

## V. Conclusion

In this paper, we not only propose two hardware architectures but also analyze the impact of supporting VBSME in hardware architectures. Base on our analysis, the impact of supporting VBSME is dependent on the data flow of partial SADs in a hardware architecture. In general cases, inter-level architectures have large penalties when supporting VBSME, because the number of registers and hardware circuits for calculating partial SADs in VBSME are increased largely. In intra-level architectures, there are two cases. If the partial SADs are propagated and accumulated, extra propagation and delay registers are required. If there is no partial SAD in intra-level architectures, the chip area overhead of supporting VBSME is less than others. Moreover, we also utilize a hexagonal plot to show the characteristics of a hardware architecture. By the hexagonal plots, the advantages and disadvantages of each hardware architecture are shown apparently. Therefore, based on different system constraints, we can easily select the optimal architecture by use of the hexagonal plots.

Our proposed hardware architectures, *Propagate Partial SAD* and *SAD Tree*, can support FBSME as well as VBSME. *Propagate Partial SAD* has the advantages of fewer reference pixel registers and a shorter critical path by broadcasting reference pixel row and propagating partial SADs. *SAD Tree* utilizes a 2-D PE array with one adder tree and the reconfigurable reference buffer, which can maximize the data reuse between successive searching candidates. Our proposed two hardware architectures can provide low cost, high utilization and less area overhead when supporting VBSME compared to six previous approaches. Moreover, *Propagate Partial SAD* is suitable for the low resolution or small search range, and *SAD Tree* has advantages when a high degree of parallelism is required.

In the last part of this paper, a hardware architecture for H.264/AVC IME was also proposed. A modified algorithm is proposed to solve the data dependency of motion vector prediction in Lagrange mode decision. Pixel truncation, subsample, and moving window are applied to reduce the hardware cost. By the proposed *Eight-Parallel SAD Tree*, on-chip memory with Level C Search Area Reuse, and On-Chip Frame Boundary Padding, the most critical issue of H.264/AVC IME, which is large memory bandwidth, is solved. The 99.9% off-chip memory bandwidth and 99.22% on-chip memory bandwidth are significantly reduced. We use Verilog-HDL and SYNOPSYS Design Compiler with ARTISAN UMC 0.18-$\mu$m cell library to implement the hardware. The total gate count of this design is 330.2 K, and the on-chip memory size is 208 kb. The design can achieve full frame rate for D1 Format with four reference frames at 81 MHz and for a 720-p stream with one reference frame at 108 MHz.

## References

[1] *Video Coding for Low Bit Rate Communication*, Feb. 1998.

[2] *Information Technology-Coding of Audio-Visual Objects—Part 2: Visual*, ISO/IEC 14 496-2, 1999.

[3] S. Srinivasan, J. Hsu, T. Holcomb, K. Mukerjee, S. L. Regunathan, B. Lin, J. Liang, M.-C. Lee, and J. Ribas-Corbera, "Windows media video 9: overview and application," *Signal Process.: Image Commun.*, vol. 19, pp. 851–875, Sep. 2004.

[4] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, May 2003.

[5] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1317–1325, Oct. 1989.

[6] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1301–1308, Oct. 1989.

[7] H. Yeo and Y. H. Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, pp. 407–416, Oct. 1995.

[8] Y. K. Lai and L. G. Chen, "A data-interlacing architecture with two-dimensional data-reuse for full-search block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 2, pp. 124–127, Apr. 1998.

[9] L. De Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1309–1316, Oct. 1989.

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

[11] J. F. Shen, T. C. Wang, and L. G. Chen, "A novel low-power full search block-matching motion estimation design for H.263+," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 7, pp. 890–897, Jul. 2001.

[12] N. Roma and L. Sousa, "Efficient and configurable full-search block-matching processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1160–1167, Dec. 2002.

[13] V. L. Do and K. Y. Yun, "A low-power VLSI architecture for full-search block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 393–398, Aug. 1998.

[14] X. Q. Gao, C. J. Duanmu, and C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 501–504, Mar. 2000.

[15] M. Brünig and W. Niehsen, "Fast full-search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 241–247, Feb. 2001.

[16] S. F. Chang, J. H. Hwang, and C. W. Jen, "Scalable array architecture design for full search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 4, pp. 332–343, Aug. 1995.

[17] H. M. Jong, L. G. Chen, and T. D. Chiueh, "Parallel architectures for 3-step hierarchical search block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 407–416, Aug. 1994.

[18] 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," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2003, pp. 796–799.

[19] S. Kittitornkun and Y. H. Hu, "Frame-level pipelined motion estimation array processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 248–251, Feb. 2001.

[20] (2003, Aug.) Joint Video Team Reference Software JM7.3. [Online]http://bs.hhi.de/suehring/tml/download/

[21] T.-C. Chen, Y.-W. Huang, and L.-G. Chen, "Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2004, pp. 273–276.

[22] 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 1.3 tops H.264/AVC signle-chip encoder for HDTV applications," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2005, pp. 128–129.

[23] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf.*, New Orleans, LA, 1981, pp. G5.3.1–G5.3.5.

[24] Z. He and M.-I. Liou, "Reducing hardware complexity of motion estimation algorithms using truncated pixels," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1997, pp. 2809–2812.

[25] 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.

[26] M.-Y. Hsu, "Scalable module-based architecture for MPEG-4 BMA motion estimation," master's thesis, Dept. Elect. Eng., National Taiwan Univ., Taipei, Jun. 2000.

**Ching-Yeh Chen** was born in Taipei, Taiwan, R.O.C., in 1980. He received the B.S. degree in electrical engineering, National Taiwan University, Taipei, Taiwan, R.O.C., in 2002, where he is currently working toward the Ph.D. degree.

His research interests include intelligent video signal processing, global/local motion estimation, scalable video coding, and associated VLSI architectures.

**Shao-Yi Chien** was born in Taipei, Taiwan, R.O.C., in 1977. He received the B.S. and Ph.D. degrees in electrical engineering, National Taiwan University (NTU), Taipei, in 1999 and 2003, respectively.

From 2003 to 2004, he was a Member of the Research Staff with Quanta Research Institute, Tao Yuan Shien, Taiwan. In 2004, he joined the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, NTU, as an Assistant Professor. His research interests include video segmentation algorithm, intelligent video coding technology, image processing, computer graphics, and associated VLSI architectures.

**Yu-Wen Huang** was born in Kaohsiung, Taiwan, R.O.C., in 1978. He received the B.S. degree in electrical engineering and the Ph.D. degree from National Taiwan University (NTU), Taipei, Taiwan, R.O.C., in 2000 and 2004, respectively.

He joined MediaTek, Inc., Hsinchu, Taiwan, in 2004, where he develops integrated circuits related to video coding systems. His research interests include video segmentation, moving object detection and tracking, intelligent video coding technology, motion estimation, face detection and recognition, H.264/AVC video coding, and associated VLSI architectures.

**Tung-Chien Chen** was born in Taipei, Taiwan, R.O.C., in 1979. He received the B.S. degree in electrical engineering and the M.S. degree in electronic engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2002 and 2004, respectively, where he is working toward the Ph.D. degree in electronics engineering.

His major research interests include motion estimation, algorithm and architecture design of MPEG-4 and H.264/AVC video coding, and low-power video coding architectures.

**Tu-Chih Wang** was born in Taipei, Taiwan, R.O.C., in 1975. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Nation Taiwan University, Taipei, Taiwan, R.O.C., in 1997, 1999, and 2003, respectively.

His main research interests include video coding technology, DSP architecture, and media processor architecture. He is currently with Chin Fong Machine Industrial, Chang Hua, Taiwan, R.O.C.

**Liang-Gee Chen** (S'84–M'86–SM'94–F'01) was born in Yun-Lin, Taiwan, R.O.C., in 1956. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1979, 1981, and 1986, respectively.

He was an Instructor (1981–1986) and an Associate Professor (1986–1988) with the Department of Electrical Engineering, National Cheng Kung University. In the military service during 1987 and 1988, he was an Associate Professor with the Institute of Resource Management, Defense Management College. In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, From 1993 to 1994, he was a Visiting Consultant with the DSP Research Department, AT&T Bell Laboratories, Murray Hill, NJ. In 1997, he was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor with National Taiwan University. Since 2004, he has also been the Executive Vice President and the General Director of Electronics Research and Service Organization (ERSO) with the Industrial Technology Research Institute (ITRI). His current research interests are DSP architecture design, video processor design, and video coding system. He was an Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* from 1999 until recently, and he served as the Guest Editor of *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* in November 2001.

Dr. Chen is a member of Phi Tan Phi. He was the general chairman of the 7th VLSI Design CAD Symposium. He is also the general chairman of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since June 1996 and an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS from since January 1999. He is also the Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS. Since 2002, he has also been an Associate Editor for PROCEEDINGS OF THE IEEE. He was the recipient of the Best Paper Award from the ROC Computer Society in 1990 and 1994 and Long-Term (Acer) Paper Awards each year from 1991 to 1999. He was the recipient of the Best Paper Award of the 1992 Asia-Pacific Conference on Circuits and Systems in VLSI design track, the Annual Paper Award of the Chinese Engineer Society in 1993, and the Outstanding Research Award from the NSC and the Dragon Excellence Award for Acer, both in 1996. He was elected an IEEE Circuits and Systems Distinguished Lecturer from 2001 to 2002.