IPSJ Transactions on System LSI Design Methodology Vol. 2 263–273 (Aug. 2009)

Regular Paper

Macroblock Feature Based Adaptive Propagate Partial SAD Architecture for HDTV Application

Yiqing Huang,[†1] Qin Liu[†1] and Takeshi Ikenaga[†1]

A macroblock (MB) feature based adaptive propagate partial SAD architecture is proposed in this paper. Firstly, by using edge detection operator, the homogeneous MB is detected before motion estimation and three hardware friendly subsampling patterns are adaptively selected for MB with different homogeneity. The proposed architecture uses four different processing elements to realize adaptive subsampling scheme. Secondly, in order to achieve data reuse and power reduction in memory part, the reference pixels in search window are reorganization into two memory groups, which output pixel data interactively for adaptive subsampling. Moreover, a compressor tree based circuit level optimization is included in our design to reduce hardware cost. Synthesized with TSMC 0.18 *um* technology, averagely 10 k gates hardware can be reduced for the whole IME engine based on our optimization. With 481 k gates at 110.5 MHz, an 720-p, 30-fps HDTV integer motion estimation engine is designed. Compared with previous work, our design can achieve 39.8% reduction in power consumption with only 3.44% increase in hardware.

1. Introduction

The H.264/AVC video coding standard can achieve superior coding performance than previous standards such as H.261, MPEG-2, H.263, MPEG-4, etc [14]. The superiority of this standard is due to many new techniques such as variable block size (VBS), multiple reference frame (MRF), In-loop deblocking filter, context aware binary arithmetic coding (CABAC) and so on. For example, in VBS technique, the block matching process of motion estimation (ME) can be executed on inter modes of 16×16, 16×8, 8×16 and 8×8 modes. In case of 8×8 mode, it be can further divided into 8×4, 4×8 and 4×4. So, the ME becomes more accurate, which results in high compression rate. However, the adoption of new techniques also bring about huge computation complexity [7], which makes hardware implementation a tough task.

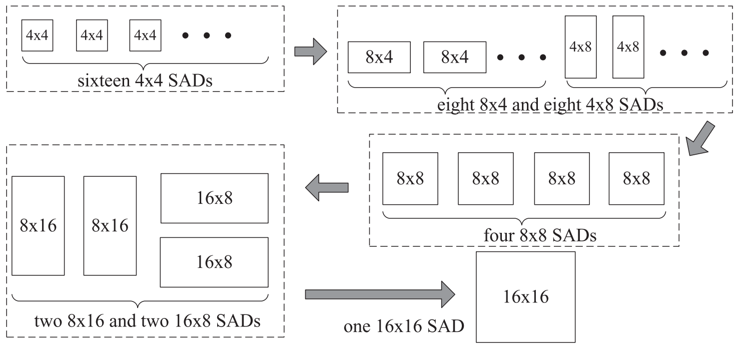


Fig. 1 Bottom-up flow of VBS in hardware implementation.

In Ref. 8), it points out that the huge throughput of ME part makes it a must to divide hardwired ME engine into integer motion estimation (IME) engine and fractional motion estimation (FME) engine. In order to have high data reuse and achieve complexity reduction [8], the implementation of VBS-IME in hardware is based on 'bottom-up' flow, as shown in **Fig. 1**. The sum of absolute difference (SAD) of small 4×4 mode is generated at first. The SADs of modes above 4×4 are all based on the accumulation of sixteen 4×4 SADs. In Ref. 2), it analyzes the existing VBS-IME architectures and proposes two efficient IME engines, namely propagate partial SAD (PPSAD) and SAD Tree architectures, for different applications. For high definition television (HDTV), since the contribution of small inter modes (8×4, 4×8 and 4×4) is very trivial, many designers use mode reduction technique which removes the inter modes below 8×8 to save hardware cost and computation [5),10)]. In Ref. 5), it optimizes the previous PPSAD [2)] architecture and provides a MRPPSAD which can achieve low hardware cost and higher parallelism feature. With improved parallelism feature, the MRPPSAD architecture is competitive to SAD Tree in HDTV application. So far, all the previous works are fixed architectures which do not take the feature of the picture into consideration. In HDTV application, the feature among frames and MBs may vary a lot because of large image size and variation among different scenes. In Ref. 4), it uses Sobel edge detection method to analyze edge infor-

†1 Graduate School of Information, Production and Systems, Waseda University

mation within MB and multiple reference frame motion estimation (MRF-ME) is adaptively controlled. In fact, the edges within MB is a direct reflection of MB's feature. In our paper, we proposes a edge detection based adaptive propagate partial SAD architecture for HDTV application. Since we target HDTV format, the mode reduction technique is also used in our proposed architecture. The Prewitt edge detection operator is used to classify MB types and subsampling patterns are selected adaptively for homogeneous ones. With this flexible architecture and optimized memory organization, the whole IME's complexity and power consumption can be reduced. Moreover, a circuit optimization on 8×8 based processing element array is given out to reduce hardware cost of our design.

The rest of paper is organized as follows. In Section 2, the impact of subsampling and edge detection on image is analyzed. After that, the edge detection based adaptive subsampling algorithm is given out. Section 3 describes the proposed architecture for adaptive subsampling scheme. The memory organization and circuit optimization are discussed in detail. Section 4 shows the experimental results and comparisons with others' works. The paper is concluded in Section 5.

2. Impact of Subsampling and Edge Detection

In the image processing filed, subsampling is a useful technique for both software and hardware. The idea of subsampling algorithm is to use part of candidate pixels to represent the original full pixel image. The are abundant subsampling pattern in software field. However, in hardwired field, comparing with previous standards which do not have VBS feature, the hardware data flow and data reuse problems in VBS based H.264/AVC standard degrade the efficiency of many irregular subsampling or fast motion estimation pattern such as N-queen [13] and UMhexagon patterns [3]. For VBS-IME engine, many designers prefer directly half or quarter subsampling technique, which apply one subsampling pattern during the whole IME process. **Figure 2** is an example of three hardware friendly subsampling patterns and full pixel pattern (pattern 4). Pattern 1 is the quarter subsampling pattern which uses 1 pixel to represent four pixels (surrounding 3 pixels and itself). Pattern 2 and 3 are horizontal and vertical half subsampling respectively. In these two patterns, each pixel represents itself and its horizontal or vertical neighbors. For VBS-IME, the candidate pixels for block matching process are reduced 75% or 50% by adopting pattern 1 to 3. In Ref. 6), it also adopts direct half subsampling in its design. However, the direct subsampling will cause video quality degradation inevitably because of feature loss caused by subsampling pattern. **Table 1** shows the video quality comparison between full pixel pattern and direct quarter subsampling pattern. The full search algorithm which has the best video quality is used as a comparison. We use Bjøntegaard [1] delta PSNR (PSNR_BD) and delta bit rate (BR_BD) to analyze the video quality. It is shown that, compared with full search, about 2.19% bit rate increase is caused by direct subsampling in knightshields_720p case. For crew_720p, the bit rate increase is less severe (0.55%). In case of HDTV1080p format, up to 3.54% and 1.70% BDBR gain for Sunflower_1080p and Pedestrian_1080p respectively. Considering the huge bit rate in HDTV application, such kind of increase is not trivial.

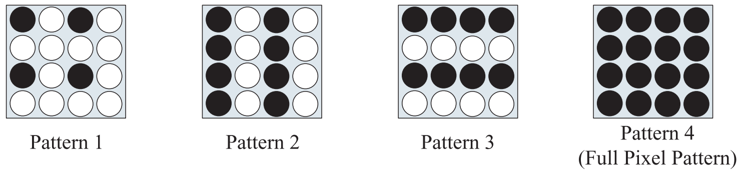


Fig. 2 Subsampling patterns and full pixel pattern.

Table 1 Quality comparison.

	BR_BD		PSNR_BD	
Subsampling	Direct	Adaptive	Direct	Adaptive
Knightshields_720p	2.19%	0.23%	−0.07 dB	−0.01 dB
Crew_720p	0.55%	0.25%	−0.02 dB	−0.00 dB
Sunflower_1080p	3.54%	0.98%	−0.08 dB	−0.02 dB
Pedestrian_1080p	1.70%	0.39%	−0.05 dB	−0.01 dB
etc	no B Slice, CAVLC, GOP is IPPP			

Judging from the real HDTV pictures, there are many MBs in each picture and each MB will have different feature according to the content of image. So, the 'blind subsampling' on all the MBs within the frame will consequently increases the bit rate which is a reflection of matching accuracy. It is obvious that, for homogeneous MB which has small pixel difference, the subsampling will not cause great damage. In case of nonhomogeneous MB, full pixel block matching will ensure accurate ME process. In Ref. 4), it points out that many textures and edges exist in nonhomogeneous MBs and edge detection is useful for MRF-ME process. In Ref. 11), edge operator is used to filter out candidate INTRA modes. In fact, edge detection is also an efficient way to classify MB types.

In our paper, in order to release extra computation and hardware resources incurred by edge detection, we use Prewitt edge operator[15], which uses 6 pixels to generate one gradient value.

Equation (1) is used to approximately calculate gradients of each pixel. The $Gx(i,j)$ and $Gy(i,j)$ are gradient of $P(i,j)$ in x (horizontal) direction and y (vertical) direction respectively.

$$\begin{cases} Gx(i,j) & = |P(i-1,j-1)+P(i-1,j)+P(i-1,j+1) \\ & \quad -P(i+1,j-1)-P(i+1,j)-P(i+1,j+1) \\ Gy(i,j) & = |P(i-1,j-1)+P(i,j-1)+P(i+1,j-1) \\ & \quad -P(i-1,j+1)-P(i,j+1)-P(i+1,j+1) \end{cases} \tag{1}$$

$$\begin{cases} all\ \ Gx(i,j) < Thr\_X, & HHMB \\ all\ \ Gy(i,j) < Thr\_Y, & VHMB \\ both\ \ Gx(i,j) < Thr\_X\ \ and\ \ Gy(i,j) < Thr\_Y, & DHMB \\ Otherwise, & NHMB \end{cases} \tag{2}$$

Based on the Prewitt operator, our MB classification method is shown in Eq. (2). An MB is defined as horizonal homogeneous MB (HHMB) if only all its x direction gradients within that MB are smaller than a predefined threshold (Thr_X). Similarly, vertical homogeneous MB (VHMB) is the MB with only all its y direction gradients within the Thr_Y. For deep homogeneous MB (DHMB), all its x and y direction gradients are within threshold. Otherwise, it is a non-homogeneous MB (NHMB). In our paper, we treat motion in x and y direction equally and set Thr_X equal Thr_Y. From exhaustive experiments, the threshold here is empirically defined as 4 times of quantization parameter (QP). The three hardware friendly subsampling patterns in Fig. 2 are selected adaptively for different homogeneous MB. In detail, pattern 1 is applied on HMMB, pattern 2 is for VHMB and pattern 3 is used for DHMB. For NHMB, the original full pixel pattern is used. The video quality improvement by our adaptive subsampling algorithm is also demonstrated in Table 1. It is shown that the bit rate and PSNR are greatly improved than direct subsampling.

In the hardware, the adaptive subsampling is not suitable for existing architectures such as PPSAD[2] and MRPPSAD[5] because of low data reuse. For example, in each clock cycle, the MRPPSAD will load 16 pixel data from memory to processing element (PE) array. If current MB is a DHMB or HHMB, 50% pixels are wasted since they are not needed for SAD calculation in adaptive subsampling algorithm. Moreover, modifications in the memory and architecture levels are also required for the original design in order to have efficient hardware data flow and low power dissipation.

3. Adaptive Propagate Partial SAD Architecture

3.1 System Architecture

Figure 3 is our proposed adaptive subsampling based propagate partial SAD architecture (APPSAD). Since the proposed architecture is target for HDTV application, the mode reduction in Ref. 5) is also adopted to save complexity and hardware cost, which means that inter modes below 8×8 is discarded. Compared with fixed PPSAD architecture in Refs. 2) and 5). Two major optimizations is applied in the architecture level.

Firstly, in the previous PPSAD architecture, 64 PEs are grouped together and used to accumulate one 8×8 SADs. In our architecture, the original fixed structure is modified for adaptive algorithm. **Figure 4** (a) is the intuitive implementation of adaptive algorithm on previous architecture. PEs with black color are activated for all the patterns just like conventional PEs (we call it PE_CONV) while PEs represented with triangle, square or grey circle are pattern dependent ones, which will be activated or deactivated according to different subsampling patterns. Besides, since the number of partial SADs will vary based on different subsampling patterns, some multiplexors are added into the architecture to enable adaptive feature. For example, when vertical subsampling is adopted, all the PEs on even lines of Fig. 4 (a) are bypassed by configuring all the multiplexors. It is obvious that in the intuitive way, many multiplexors are required, which will intensify the complexity in control logic. The hardware size will also be dilated

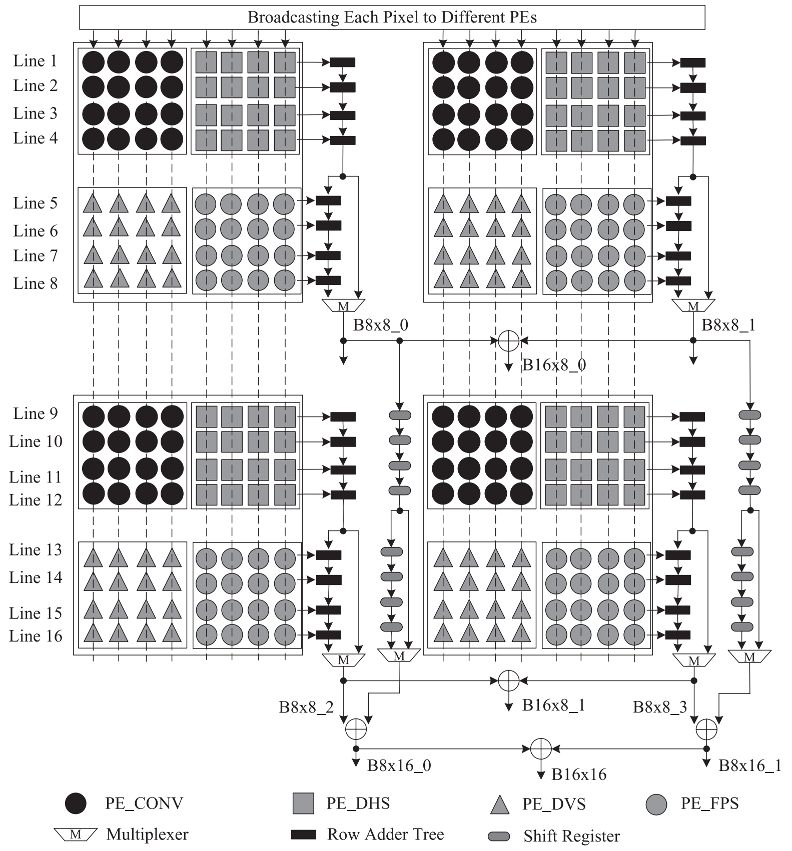


Fig. 3 Adaptive propagate partial SAD architecture.

consequently due to this operation. In our design, as shown in Fig. 3, we group all the PEs according to their types. For example, all the PE_CONV within one 8×8 block are grouped together and we use only one multiplexor to realize control logic of adaptive algorithm. Thus, 75% number of multiplexors are removed.

Secondly, besides conventional PE (PE_CONV), extra three different PEs exist in our design, namely PE_DHS, PE_DVS and PE_FPS, which is represented with square, triangle and grey circle in Fig. 4 (a), respectively. Each of these elements

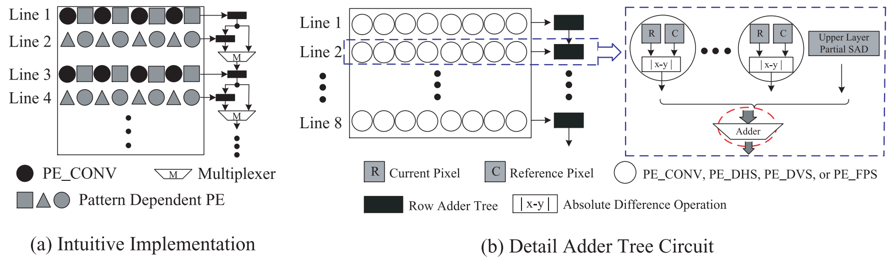


Fig. 4 8x8 PE array in PPSAD architecture.

are activated or deactivated at different subsampling patterns. In detail, PE_DHS is disabled in horizontal subsampling case and PE_DVS is deactivated in vertical subsampling case. As for PE_FPS, it is only enabled for full pixel block matching situation (NHMB case). For PE_CONV, their status are always 'ON' no matter what sampling pattern the system selects. In the detail design, we simply send one disable signal to the PE to be deactivated and the output result of such PE will turn to constant zero. So, under different subsampling patterns, the architecture can enable corresponding PEs for the block matching process. Many absolute difference calculations are saved and power dissipation is reduced in the architecture level consequently.

3.2 Memory Organization

In the memory level, the original memory structure also needs to be modified to improve data reuse. In Ref. 9), it uses a memory overlapping algorithm to fully utilize pixel data loaded from memory. In our design, we divide memory pixels into four types, namely even-row, odd-row, even-column, odd-column, as shown in the left part of **Fig. 5**. All the square pixels in Fig. 5 are odd-row-odd-column pixel (P_{oo}); the triangle represents even-row-even-column pixels (P_{ee}); for circle and diamond symbols, they are odd-row-even-column pixel (P_{oe}) and even-row-odd-column (P_{eo}) respectively.

In the second step, all pixels are grouped together according to their types. Since there four patterns (including full pixel pattern) in our design, two memory groups are needed to store them. For instance, in case of NHMB and VHMB, the required pixel number (16 pixels per clock) for APPSAD architecture is two times of HHMB and DHMB cases (8 pixels per clock). So, as shown in Fig. 5, two

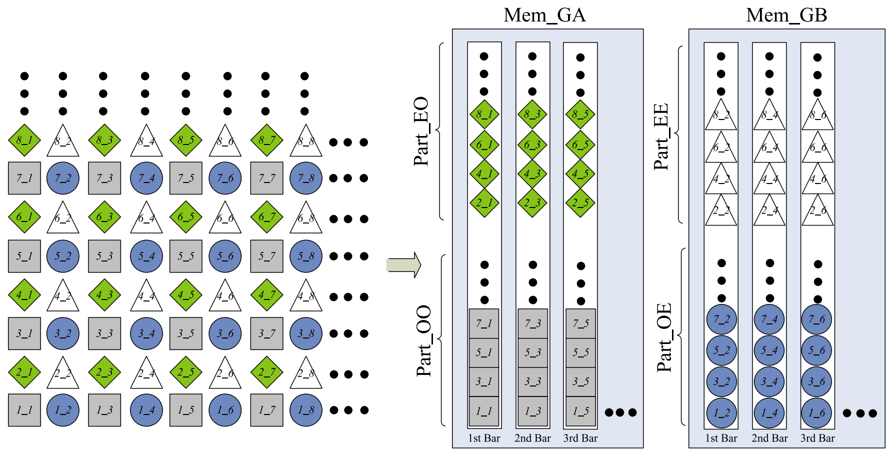


Fig. 5 Pixel classification and memory organization.

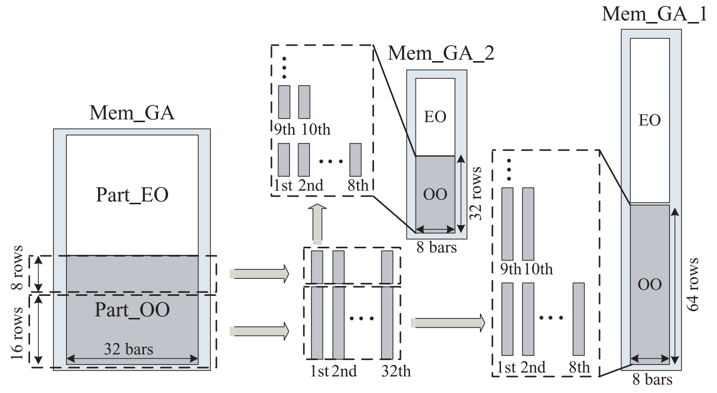


Fig. 6 Memory separation and overlapping.

memory groups, namely Mem_GA and Mem_GB, are used in our design. Each group contains several one-pixel width memory bars.

All the P_{oo} (Part_OO) and P_{eo} (Part_EO) are stored in Mem_GA while the other two type pixels (Part_OE and Part_EE) are stored in Mem_GB. To improve the IO bandwidth utilization and erase bubble clock cycles of PPSAD based architecture [2)], we further separate each group into 2 sub-group, namely Mem_GA_1, Mem_GA_2, Mem_GB_1 and Mem_GB_2, and apply memory mapping algorithm [9)]. **Figure 6** gives out an example. We assume that search range size is 48 in width (W=48) and 32 (H=32) in height. Last fifteen rows and columns are added for block matching on the boundary parts. One row and column is added for hardware implementation. So, the search window size is (W+16)×(H+16). Based on our pixel classification, the size of each part in Fig. 5, for example Part_OO, is 32×24. As shown in Fig. 6, we separate the last 8 rows of each part and apply memory overlapping algorithm [9)] on both Mem_GA_1 and Mem_GA_2. So, the clock bubble in PPSAD based architecture is removed. Each memory group contains eight memory bars, which makes 100% IO bandwidth utilization for different subsampling patterns. In our paper, we only focus on the IME's on-chip memory and do not deal with pixel organization of off-chip frame memory. The original Level C or Level D [12)] off-chip to on-chip data reuse scheme and their corresponding scan order still can be used for the whole encoder system. So, the required off-chip to on-chip memory bandwidth is the same with Level C or Level D scheme. The proposed pixel classification can be done in the encoder's system level, which is not ascribed to the IME engine's job.

Thirdly, the data flow of our architecture is different from previous design. **Figure 7** is our memory data loading flow for four types of patterns. To simplify the explanation, Mem_GA_1 and Mem_GA_2, Mem_GB_1 and Mem_GB_1 are merged together in our description.

As shown in Fig. 7, there are two stages in HHMB case. In the 1st Stage, the Part_Sel signal chooses data from Mem_GA, which means that only Part_OO and Part_EO are the candidate Parts. The pixel data are loaded interactively from these two parts and Mem_GB is set to idle state, which saves power of Mem_GB part. Based on our data organization style, the memory address control is also simplified. The difference of succeeding two addresses is only the height of Part_OO. For example, we assume that there are h addresses in each bar of Part_OO. In $2n$th cycle, one pixel row at address m of Part_OO is loaded, in the next cycle ($(2n+1)$th cycle), another pixel row from Part_EO is required for the APPSAD structure based on pattern 2 of Fig. 2. The address of this pixel row will be $(m+h)$. The address generation of NHMB case and HHMB's 2nd

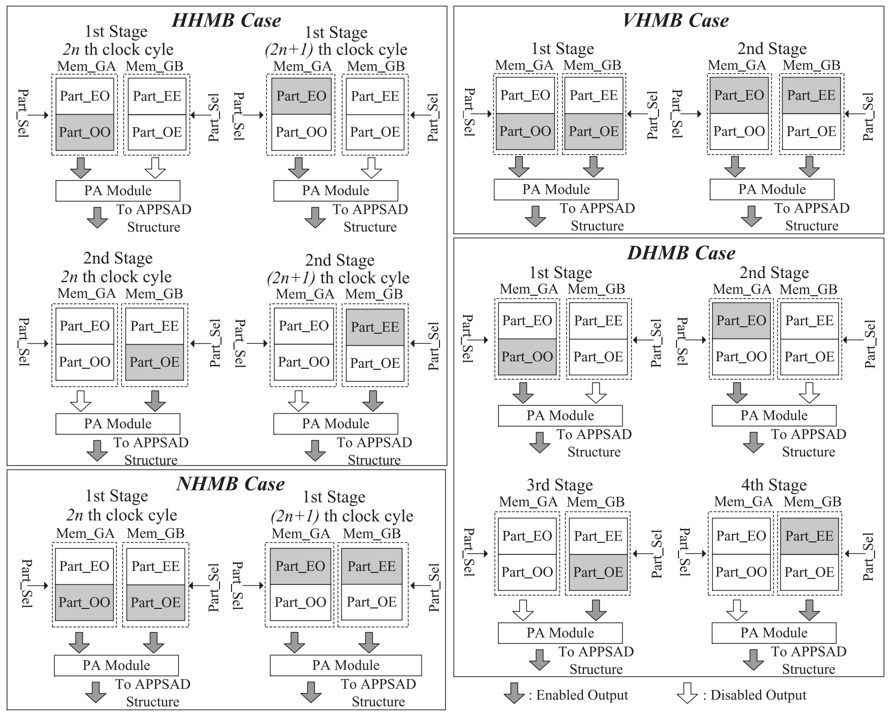


Fig. 7 Data flow of APPSAD architecture.

Stage can be traced by analogy. When all the pixels in Part_OO and Part_EO of HHMB case are loaded, it turns to 2nd Stage, during which only Part_EE and Part_OE are candidate parts and power dissipation for Mem_GA can be saved.

For VHMB case, it also consists of two stages. In each clock cycle, both memory groups are activated because the required number of pixels for APPSAD structure is doubled (16 pixels) according to pattern 3 of Fig. 2. Specifically, in 1st Stage, only Part_OO and Part_OE are candidate parts. Two pixel rows are loaded simultaneously from low address to high address cycle by cycle. When all the pixels are loaded, it turns to 2nd Stage, which only requires pixels from Part_EO and Part_EE. The pixel assemble module (PA Module) combines the two rows together and outputs the assembled 16 pixels to the APPSAD architecture.

For DHMB case, since subsampling is adopted both horizontally and vertically, the pixels of different types are loaded one part by one part. So, there are four stages in all. In each stage, the pixel row of specific part is loaded from low address to high address cycle by cycle.

As for NHMB case, only 1 stage exists based on full pixel pattern in Fig. 2. As shown in Fig. 7, in the $2n$th clock, the pixel rows from Part_OO and Part_OE are loaded simultaneously. In the succeeding $(2n+1)$th cycle, two rows from Part_EO and Part_EE are loaded. The whole process continues until all the pixels in the memory are loaded.

Furthermore, for HHMB and DHMB cases, the required number of pixels in each clock cycle is 8, which is half of the VHMB and NHMB cases. So, only one memory group is enabled within each stage and the power consumption of another group can be saved. Therefore, the proposed pixel organization can keep high data reuse while achieve lower memory power dissipation.

3.3 Compressor Tree based Eight Stage Circuit Optimization

The proposed APPSAD architecture can realize adaptive subsampling algorithm by introducing some multiplexors and optimizing previous PE array. The hardware size will also be dilated compared with original PPSAD structure. In our previous work [5], we adopt circuit optimization to remove the adder within each PE. However, one adder still exists in generating partial result at each stage. In this work, we further improve our previous design by using 4-2 and 3-2 compressors to manually build up our compact architecture. Thus, all the unnecessary adders within each stage are removed.

$$\begin{cases} IS = A \oplus B \oplus C \\ ICO = (A \cdot B) + (A \cdot C) + (B \cdot C) \\ S = IS \oplus D \oplus ICI \\ CO = (IS \cdot D) + (IS \cdot ICI) + (D \cdot ICI) \end{cases} \tag{3}$$

$$\begin{cases} S = A \oplus B \oplus C \\ CO = (A \cdot B) + (A \cdot C) + (B \cdot C) \end{cases} \tag{4}$$

In TSMC library, compressor tree is a widely used design ware to achieve compact hardware structure. Generally, the 4-2 and 3-2 compressor trees are efficient

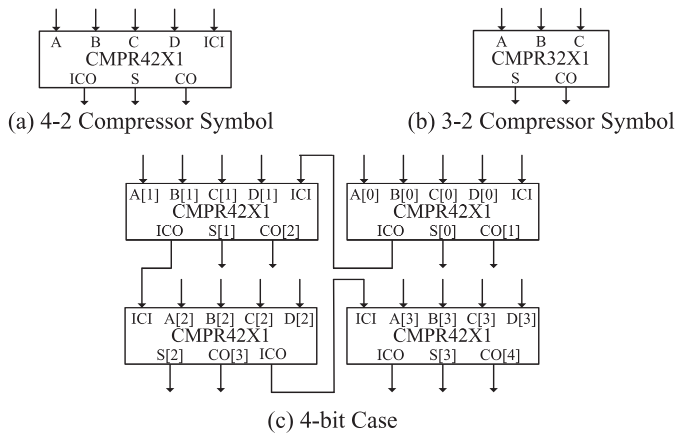


Fig. 8 Compressor tree symbols.

in generating superior structure. In our design, we use these two compressor trees to build up our compact architecture.

Figure 8 is the library symbol of these two compressor trees. For 4-2 compressor trees (called CMPR42X1), there are five inputs and three outputs. A, B, C, and D represent the 4 input bits data to be compressed. ICI is the immediate carry in flag of previous compressor. The 4-bits input data are compressed into 2-bits of partial product (S, CO). An immediate carry-out output bit to the following compressor unit is needed. In CMPR42X1, it also contains an internal sum (IS), which relays data during compression. The logic equation of CMPR42X1 is shown in Eq. (3). When multiple-bits width input is used, the CMPR42X1 can be configured to multiple-bits application. In Fig. 8 (c), an example of compressing 4-bit width input data by connecting four 1-bit width CMPR42X1 is shown. The principle of 3-2 compressor tree (CMPR32X1 in Fig. 8 (b)) is similar to CMPR42X1 and logic equation of CMPR32X1 is shown in Eq. (4). The configuration of multiple-bits width input on this design ware is also feasible.

Figure 4 (b) is one 8×8 PE block of Fig. 3 (left-up 8×8 block, for example), the original architecture has 64 PEs arranged in 8 rows (0th to 7th) and 8 columns (0th to 7th). Based on the absolute difference operation, each PE will generate one residue value between reference pixel and current pixel. Like Ref. 5), we do not directly get the result value from each PE. Instead, we only execute the subtraction operation and keep the most significant bit (MSB) and absolute difference result (abs) of each PE. So, like Ref. 5), the adders within each PE are removed in our design. However, as shown in Fig. 4 (b), one adder still exists in each 8×1 stage to accumulate not only the partial SAD of the current stage but also the result from upper stage. Considering the 8-stage based APPSAD structure, with the increase of stage, the hardware size and power consumption of these adders become serious. In fact, the adders from the 1st stage to the 7th stage are all redundant and only the final result at the 8th stage is the needed 8×8 SAD value. Thus, in our design, we propose a compressor tree based circuit optimization for 8×8 based APPSAD structure. For one 8×8 PE block in Fig. 4 (b), we have 64 abs (absXY: X=0 to 7, Y=0 to 7) and 64 MSB (renamed as cXY: X=0 to 7, Y=0 to 7) as input data. Here, absXY and cXY represent the abs and MSB value on the Xth row and Yth column of one 8×8 PE block. Based on these input data, we manually build up our adder tree circuit to remove all the unnecessary adders within the 8×8 PE block. The other three 8×8 PE blocks in Fig. 3 are optimized in the same way.

Figure 9 is the proposed compressor tree based circuit optimization. Here we show the first three stages as an example. Based on the pipeline feature of proposed architecture, the whole 8×8 PE array has to be built up in eight compression stages in similar way. The square dot in the graph represents the bit-insert in head, which means that one bit is inserted in the beginning of the byte. For example, in Fig. 9, by using CMPR42X1 to compress the Layer-1 in Stage-1, we obtain two output bytes ([8:1] and [7:0]) and one carry-out bit (ico). Here, ico is the 8th bit of output result; and we name [7:0] and [8:1] as CMPR42X1_S[7:0] and CMPR42X1_CO[8:1] respectively. As shown in the bottom right corner of Fig. 9, we combine the ico with CMPR42X1_S[7:0] to form CMPR42X1_S[8:0] as one input of next compression layer. The square dot is drawn here to indicate this operation. Similarly, the diamond dot in Fig. 9 represents the bit insert in tail, which indicates that the combination of one bit and one byte occurs in the end of this byte. For example, as shown in bottom right of Fig. 9, after the Layer-1 compression of Stage-1, we combine c02 with CMPR42X1_CO[8:1] to form CMPR42X1_CO[8:0] as the input of next compression layer. One diamond

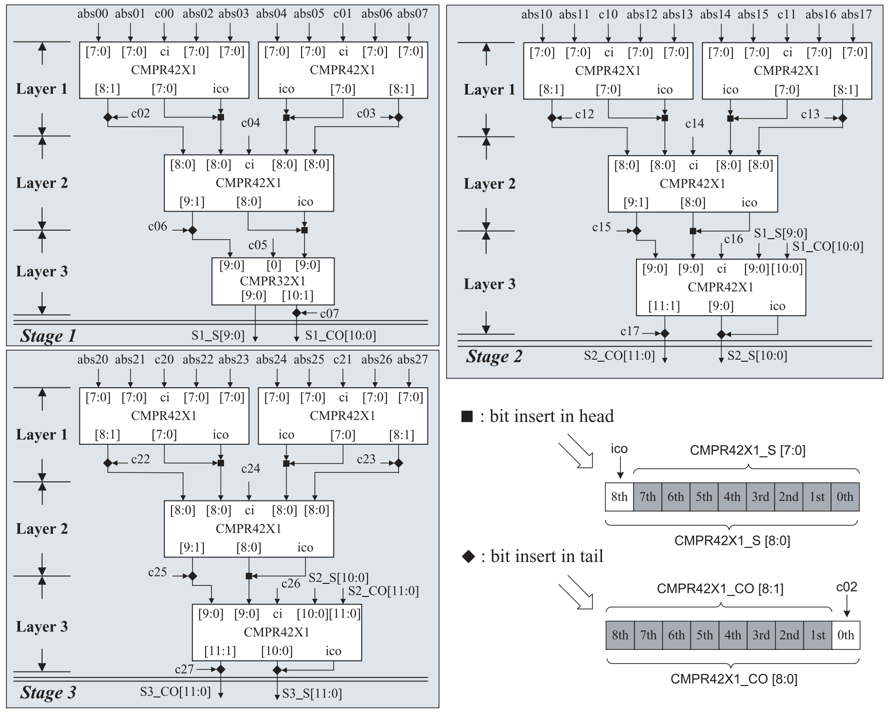


Fig. 9 8x8 PE circuit optimization.

dot is added here to indicate this operation.

In the Layer-1 of Stage-1, we use two CMPR42X1 to achieve the compression of abs00 to abs07, c00, and c01. In the Layer-2 and Layer-3, a CMPR42X1 and CMPR32X1 compressor are used to output the sum (O1_S[9:0]) and carry out (O1_C[8:0]) of the first stage. In Stage-2 and Stage-3, the circuits are also built on CMPR42X1 and CMPR32X1 based on similar way. From Stage-1 to Stage-7, two registers are used to store temporary results of each stage. An adder is applied at Stage-8 to output the final 8×8 SAD value.

4. Experiments and Comparisons

In this section, experiments and comparisons with other works will be given

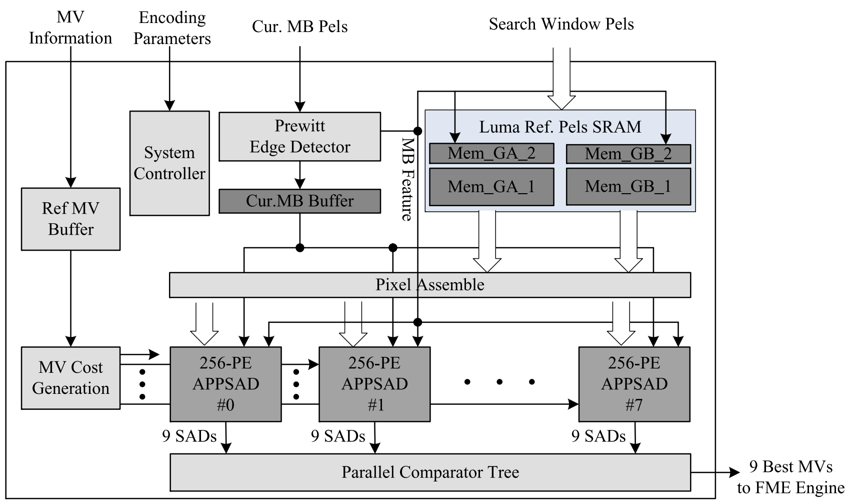


Fig. 10 IME engine with APPSAD architecture.

Table 2 Synthesis result and comparison.

Designs	SAD Tree [2)]	PPSAD [2)]	MRPPSAD [5)]	ours	ours
Technology	0.18 um	0.18 um	0.18 um	0.18 um	0.18 um
Frequency	110.8 MHz	110.8 MHz	110.5 MHz	110.5 MHz	150 MHz
PE Array & Cur.MB	88.6 k	81.5 k	68.7 k	70.4	73.3
Whole Engine	-	-	465 k	490 k	509 k
Optimized	-	-	-	481 k	498 k
Flexibility	No	No	No	Yes	Yes

out. Firstly, we set specification as HDTV 720p@30 fps as a case study. The search range is [−64,+63) in width and [−32,+31) in height. Fifteen bottom pixel rows are added for block matching of search points on the last row. One extra pixel row is included for hardware design. The final search window size is 144×80. **Figure 10** is the block diagram of whole IME engine based on our APPSAD architecture. Eight parallel APPSAD structures are used and we only adopt one reference frame. The Prewitt edge detector analyzes the homogeneity of input MB. Based on the throughput requirement of our specification, the minimum work frequency is 110.5 MHz . The synthesis result of our design is shown in **Table 2**. We pick 110.5 and 150 MHz as two synthesis points and compare with previous designs. The memory is excluded in this table. It is shown that compared with MRPPSAD architecture, the hardware cost of our design is increased by 2.47% for a single PE Array with Cur.MB part and 5.37% for the whole IME engine. Compared with full mode PPSAD and SAD Tree architectures, our design still outweighs them in hardware cost because of mode reduction method. Here, compressed tree based circuit level optimization is not adopted. As for the whole IME engine under 110.5 MHz work frequency, our design will incur 25 k gates mainly because of pixel assemble module, and extra control logic.

Secondly, we apply compressor tree based circuit optimization on APPSAD architecture. The optimized result of whole engine which consists of 8-set APPSAD architectures is shown in the second last line of Table 2. About 9 k and 11 k hardware can be reduced for 8 parallel APPSAD architectures under 110.5 and 150 MHz frequency points respectively. So the overall hardware increase of whole IME engine is reduced to only 3.44% compared with MRPPSAD architecture.

Thirdly, we pick two typical HDTV 720p format sequences to test gate level power consumption. **Table 3** and **Table 4** demonstrates the percentage of four types of MB within first 6 frames. It is shown that spatial homogeneity always exists in video sequences and there are more homogeneous MBs in Crew_720p than Stockholm_720p sequence. Since the HHMB and DHMB cases only have one memory group activated, with higher HHMB and DHMB ratio, more power can be saved in the memory part of Crew_720p sequence. **Figure 11** is the power consumption comparison between our work and previous design. In order to make a clear comparison, no speed-up algorithm such as coarse-to-fine search is adopted. The power consumption of SRAM is demonstrated individually besides the whole encoder's power dissipation. Since we rearrange the reference pixel data into two memory groups and only one memory group is enabled in case of HMMB and DHMB situation, the overall memory power consumption is lower than previous design which use all the memory bars. About 11.6% and 24.9% power consumption in memory part can be reduced for stockholm_720p and crew_720p, which is in accordance with the ratio of homogeneous MBs in two sequences. Apart from memory, the adaptive architecture can also adjust itself for MB with different homogeneous feature, which reduces power consumption in architecture level. Overall, 25.4% and 39.8% power dissipation is reduced for stockholm_720p and crew_720p sequences.

Furthermore, the architecture level's flexibility in our design is meaningful for power aware system according to different applications. Trade-off between video

Table 3 Ratio of homogeneous MB in Stockholm.

Stockholm	HHMB (%)	DHMB (%)	VHMB (%)	NHMB (%)
1st Frame	15.33	47.75	7.13	29.79
2nd Frame	15.44	47.41	6.77	30.38
3rd Frame	14.72	47.58	7.02	30.68
4th Frame	15.30	48.11	6.86	29.73
5th Frame	15.25	48.00	6.61	30.14
6th Frame	15.22	47.63	6.88	30.27
Average	15.21	47.75	6.88	30.17

Table 4 Ratio of homogeneous MB in Crew.

Crew	HHMB (%)	DHMB (%)	VHMB (%)	NHMB (%)
1st Frame	5.86	77.80	10.30	6.04
2nd Frame	5.25	81.80	8.58	4.37
3rd Frame	5.36	81.72	8.75	4.17
4th Frame	5.19	81.77	8.88	4.16
5th Frame	5.11	81.94	8.91	4.04
6th Frame	4.91	82.72	8.36	4.01
Average	5.28	81.29	8.96	4.47

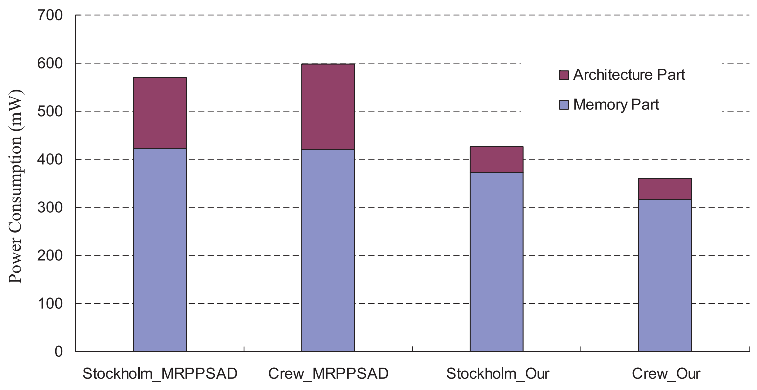


Fig. 11 Power consumption comparison.

quality and power consumption can be easily controlled by our proposed APPSAD architecture.

5. Concluding Remarks

The Prewitt edge detection is used to analyze the homogeneity of macroblocks and three hardware friendly subsampling patterns are adaptively selected for the block matching process of integer motion estimation. With adaptive subsampling algorithm, the computation complexity for homogeneous MB is relieved and the video quality is maintained compared with direct subsampling method. The proposed APPSAD architecture realize the adaptive subsampling algorithm by introducing four different processing elements. To improve the data reuse and reduce power consumption, the reference pixels are classified into four types and two memory groups are used. Furthermore, a compressor tree based circuit optimization is proposed in our architecture, which averagely reduces 10 k gates hardware for the whole IME engine. Based on our APPSAD architecture, an HDTV 720p@30 fps IME engine is implemented with 481 k logic gates and it can achieve 39.8% reduction in power dissipation at 110.5 MHz.

6. Acknowledgments and Appendices

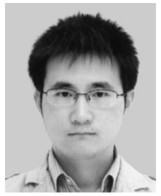
Acknowledgments This research is supported by CREST, JST. The research is also supported by the Ambient SoC Global COE Program of Waseda University under the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

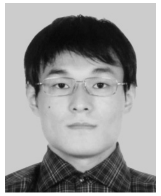
1) Bjontegaard, G.: Calculation of average psnr differences between rd-curves, *VCEG-M33* (2001).
2) Chen, C., Chien, S., Huang, Y., Chen, T., Wang, T. and Chen, L.: Analysis and architecture design of variable block size motion estimation for h.264/avc, *IEEE Transactions on Circuits and Systems I*, Vol.53, No.3, pp.578–593 (2006).
3) Chen, Z., Zhou, P. and He, Y.: Fast Integer Pel and Fractional Pel Motion Estimation for JVT, *JVT-F017.doc, 6th Meeting: Awaji, Island, JP*, pp.5–13 (2002).
4) Huang, Y., Liu, Z., Goto, S. and Ikenaga, T.: Adaptive Edge Detection Pre-Process Multiple Reference Frames Motion Estimation in H.264/AVC, *International Conference on Communications, Circuits and Systems*, pp.787–791 (2007).
5) Huang, Y., Liu, Z., Song, Y., Goto, S. and Ikenaga, T.: Parallel Improved HDTV720p Targeted Propagate Partial SAD Architecture for Variable Block Size Motion Estimation in H.264/AVC, *IEICE Transactions on Fundamentals*, Vol.E91-A, No.4, pp.987–997 (2008).
6) Huang, Y., Chen, T., Tsai, C., Chen, C., Chen, T., Chen, C., Shen, C., Ma, S., Wang, T., Hsieh, B., Fang, H. and Chen, L.: A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications, *IEEE International Solid-State Circuits Conference 2005*, pp.128–130 (2005).
7) Huang, Y., Hsieh, B., Chien, S., Ma, S. and Chen, L.: Analysis and Complexity Reduction of Multiple Reference Frames Motion Estimation in H.264/AVC, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.16, No.4, pp.507–508 (2006).
8) Huang, Y., Wang, T., Hsieh, B. and Chen, L.: Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264, *Proceedings of the 2003 International Symposium on Circuits and Systems*, Vol.2, pp.796–799 (2003).
9) Liu, Z., Song, Y., Ikenaga, T. and Goto, S.: A Fine-Grain Scalable and Low Memory Cost Variable Block Size Motion Estimation Architecture for H.264/AVC, *IEICE Transactions on Electronics*, Vol.E89-C, No.12, pp.1928–1936 (2006).
10) Liu, Z.Y., Song, Y., Shao, M., Li, S., Li, L.F., Ishiwata, S., Nakagawa, M., Goto, S. and Ikenaga, T.: A 1.41 W H.264/AVC REAL-TIME ENCODER SOC FOR HDTV1080P, *VLSI Symposium '07. Proc. 2007 VLSI Symposium*, pp.12–13 (2007).
11) Pan, F., Lin, X., Rahardja, S., Lim, K., Li, Z., Wu, D. and Wu, S.: Fast mode decision algorithm for intraprediction in h.264/avc video coding, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.15, No.6, pp.813–821 (2005).
12) Tuan, J., Chang, T. and Jen, C.: On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.12, No.1, pp.61–72 (2002).
13) Wang, C., Yang, S., Liu, C. and Chiang, T.: A hierarchical decimation lattice based on N-queen with an application for motion estimation, *IEEE Signal Processing Letters*, Vol.10, No.8, pp.228–231 (2003).
14) Wiegand, T., Sullivan, G., Bjontegaard, G. and Luthra, A.: Overview of the H.264/AVC Video Coding Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.13, No.7, pp.560–576 (2003).
15) Wikipeida: http://en.wikipedia.org/wiki/Prewitt.

(Received November 17, 2008)
(Revised February 20, 2009)
(Accepted April 13, 2009)
(Released August 14, 2009)

(Recommended by Associate Editor: *Yusuke Matsunaga*)



Yiqing Huang was born in 1982. He received the B.E. and M.E. degrees in communication and information engineering from Shanghai University, China in 2004 and 2007, respectively. In 2007, he received the M.E. degree in LSI systems from Waseda University, Japan. He is currently a Ph.D. candidate and a research associate of the Graduate School of Information, Production and Systems, Waseda University, Japan. His research interests include hardware-friendly video coding algorithms and the associated very large scale integration (VLSI) architecture.



Qin Liu was born in Nanjing, China. He received the M.S. degree in software engineering from Nanjing University (NJU), Nanjing, China, in 2006. He is currently pursuing a Ph.D. degree at the Graduate School of Information, Production and Systems, Waseda University. His research interests include video compression algorithms and their VLSI hardware architecture.

Takeshi Ikenaga received his B.E. and M.E. degrees in electrical engineering and the Ph.D. degree in information and computer science from Waseda University, Tokyo, Japan, in 1988, 1990, and 2002, respectively. He joined LSI Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 1990, where he undertook research on the design and test methodologies for high-performance ASICs, a real-time MPEG2 encoder chip set, and a highly parallel LSI as well as system design for image-understanding processing. He is presently an Associate Professor in the system LSI field of the Graduate School of Information, Production and Systems, Waseda University. His current interests are application SoCs for images, security, and network processing. In particular, he is conducting research on the H.264 encoder LSI, JPEG2000 codec LSI, LDPC decoder LSI, UWB wireless communication LSI, public key encryption LSI, and object recognition LSI. Dr. Ikenaga is a member of the IEICE, IIEEJ and the IEEE.