A Low-Power and Bandwidth-Efficient Motion Estimation IP Core Design Using Binary Search

Shih-Hao Wang, Shih-Hsin Tai, and Tihao Chiang, *Senior Member, IEEE*

***Abstract*— A new architecture design for motion estimation using binary matching criterion is proposed to achieve low power and bus bandwidth efficiency. Low power and high bus bandwidth efficiency are the two key issues for portable video applications. To address such issues, we first study an efficient algorithm called all binary motion estimation (ABME), and analyze its architecture issues in operational flow and bus access. Then, we propose an architecture for ABME with four new features: 1) macroblock level pre-processing; 2) efficient binary pyramid search structure; 3) parallel processing of 8×8 and 16×16 block searches; 4) parallel processing of bi-directional search. Such architecture leads to a superior performance in bus access, speed, and power. Our experiments show that the power consumption is as low as 763 μW for IPPPP CIF 30 frames/s and 896 μW for IPBPB CIF 30 frames/s. The bus bandwidth savings are 54.3% for P-frame search and 67.1% for B-frame search.**

***Index Terms*— Bandwidth efficient, low power, motion estimation (ME), MPEG-4.**

I. Introduction

MOTION ESTIMATION (ME) is the most computationally expensive module in multimedia compression standards such as MPEG-1/2/4 and H.26x. For portable video applications, low power and efficient bus access are two major design goals.

As compared to the prior low power ME designs [1]–[4], binary search [5], [6] has both advantages of low computational complexity and low bus bandwidth requirement. The reason is it reduces the pixel representation from eight bits to one bit for pattern matching. Such a search strategy can also be viewed as a kind of feature matching with binary images. Therefore, we develop our low-power and bandwidth-efficient ME design based on a binary pyramid search algorithm called all binary motion estimation (ABME) [7].

ABME [7] is a low-complexity and bandwidth-efficient algorithm, but not well optimized for VLSI implementation. The hardware design challenges are in 1) image preprocessing to form the binary image, 2) low power and bus bandwidth efficient architecture for binary pyramid search, and 3) support of bi-directional (or called B-frame) and 8×8 block searches. These design issues are addressed with three new features. First, a macroblock (MB) level preprocessing flow is proposed to replace the original frame level for the removal of the repeated bus access. Second, the data flow in the three layers of binary pyramid search structure is optimized for the data dependency removal and the efficient operations in the second layer of search. Finally, we address the design issues in B-frame search scheme and optimize the hardware architecture to enhance the processing throughput.

The contributions of this paper include the following.

1) Modified ABME algorithm for efficient VLSI implementation. We modify the ABME in the binary image generation and search method for efficient VLSI implementation. The power consumption for CIF 30 frames/s encoding only needs less than 1 mW.
2) MB level pipelining architecture for efficient bus access. The new processing flow integrates both binary image generation and binary motion search using MB level pipelining to avoid repeated bus access. The bus bandwidth saving achieved can be up to 67.1%.
3) Parallel B-frame search architecture. It reuses the same current search data to save on-chip memory access and power. Thanks to the simple binary image matching, the gate counts have increased twice but not as much as the conventional 8-bit designs.

The remainder of this paper is organized as follows. Section II depicts the algorithm, and its hardware architecture is proposed in Section III. In Section IV, experiments show the improved performance in power consumption and bus bandwidth loading. Section V gives the conclusions.

II. Algorithm

A. Review of ABME

ABME algorithm is composed of two major components: 1) frame level of pre-processing; 2) three layers of binary pyramid search. They are described as below.

1) Frame Level of Preprocessing: The frame level of preprocessing is used to generate binary images for the current frame. The original image $I_{M\times N}$ is downsampled by two twice to be $I_{(M/2)\times(N/2)}$ and $I_{(M/4)\times(N/4)}$. These original and downsampled images are then binarized to be three binary images $\tilde{I}_{M\times N}$ (referred to as LV3), $\tilde{I}_{(M/2)\times(N/2)}$, (referred to as LV2), and $\tilde{I}_{(M/4)\times(N/4)}$ (referred to as LV1) for search.

The binarization process contains two steps: filtering and comparator. The filtering operation adopts a 3×3 two-dimensional filter H_A as shown in (1). The comparator constructs the three layers of binary pyramid data as shown

Manuscript received November 6, 2006; revised August 26, 2007. First version published March 16, 2009; current version published June 10, 2009. This work was supported by the National Science Council under Grant NSC 95-2221-E-009 -074 -MY3. This paper was recommended by Associate Editor M. Comer.

The authors are with the Department of Electronics Engineering, National Chiao Tung University, 30010, Hsinchu, Taiwan (e-mail: shwang.ee90g@nctu.edu.tw; tchiang@mail.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2009.2017416

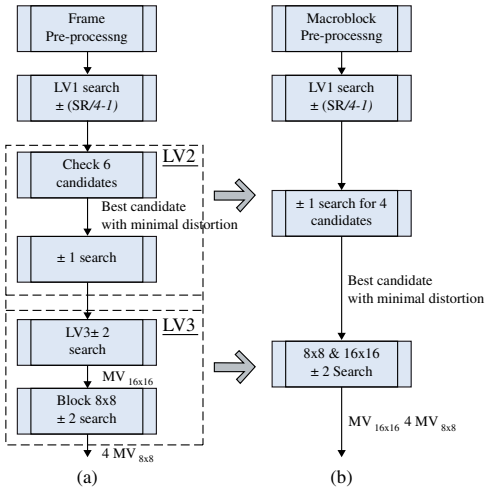

Fig. 1. Processing procedure of ABME algorithm: (a) original flow in [7]; (b) modified ABME flow.

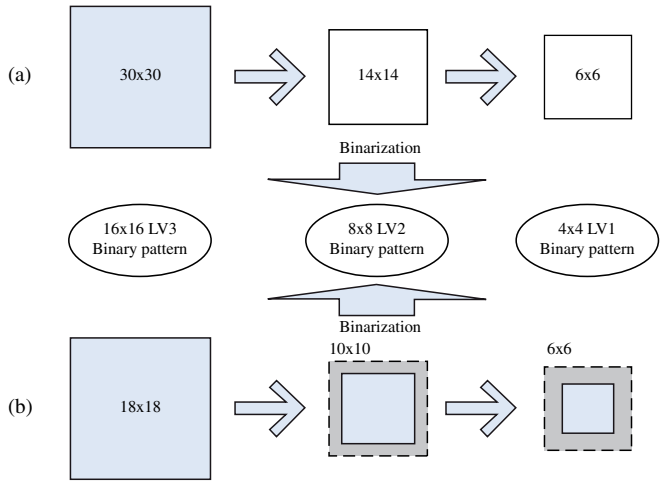

Fig. 2. Preprocessing flow in macroblock preprocessing unit. (a) $K = 30$ (b) $K = 18$ (The shaded area is padding pixels).

in (2)

$$H_A = \frac{1}{4} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{1}$$

$$\tilde{I}_{M\times N}(x,y) = \begin{cases} 1 & \text{if } H_A(I_{M\times N}(x,y)) \geq I_{M\times N}(x,y)) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

2) Three Layers of Binary Pyramid Search: The ABME processing flow as shown in Fig. 1(a) adopts a three-layer binary pyramid search structure. The first layer and third layer implement a small range of full search. The second layer performs the multiple candidates check first followed by another ±1 full search based on the selected candidate. The matching criterion is simplified from sum absolute difference (SAD) to sum of difference (SOD) as in (3) with equal results due to the binary data format

$$SOD = \sum_{y=0}^{L-1}\sum_{x=0}^{L-1} \left| \tilde{I}_C(x,y) \oplus \tilde{I}_R(x+x_0, y+y_0) \right| \tag{3}$$

where the symbol I_C is the current frame, the symbol I_R is the reference frame, the symbol $\oplus$ denotes the XOR operation, and L is 16 for LV3, 8 for LV2, and 4 for LV1.

B. Modified ABME Algorithm

Although ABME [7] is a low-complexity and bandwidth-efficient algorithm, it is not well optimized for hardware implementation. The modified ABME algorithm addresses its design issues as below. Fig. 1(b) shows the processing procedure of modified ABME algorithm. The first modification is to replace the original frame preprocessing with macroblock level preprocessing. The second modification is to simplify the LV2 search flow of ABME. The third modification is to support parallel processing of 8 × 8 and 16 × 16 LV3 block search.

TABLE I
COMPARISON OF DIFFERENT K VALUES FOR MACROBLOCK-BASED PREPROCESSING UNIT

Block size ($K \times K$)	16 × 16	18 × 18	20 × 20	30 × 30
Required data (bits)	256	324	400	900
PSNR loss (dB)	−0.50	−0.14	−0.10	−0.00

1) Macroblock Preprocessing Unit (MBPPU): The MBPPU removes the repeated bus access. As opposed to frame-level implementation of the preprocessing module, the MBPPU is integrated with the binary search module to generate MB-level binary search block for the current frame. The three layers of binary search blocks are then stored back to external memory as reference picture for the next frame. To integrate the preprocessing module with the MB-level pipelining, a straightforward approach is to implement at macroblock level as shown in Fig. 2(a). This approach needs to transmit 30×30 image data to generate the 4×4 LV1 binary block due to H_A, a 3×3 down-sampling filter. An alternative approach is to replace the 30×30 image data with smaller $K \times K$ image data and pad the missing image pixels. The value of K has no effect on the search range because such a simplification is applied for the current search block. Fig. 2(b) shows an example for $K = 18$. We pad the boundary pixels to fill the missing pixels at LV2 and LV1 to generate three layers of binary search blocks. We experiment with various K values, and Table I shows the PSNR results. From this Table, $K = 18$ is selected, as it has a PSNR loss around 0.1dB which is a tolerable penalty in quality.

2) Efficient LV2 Search: The modified LV2 search is to remove the redundant on-chip memory access in the LV2 flow of the original ABME algorithm. In Fig. 1(a), the original LV2 flow is a software-efficient flow that checks six candidates sequentially and then performs ±1 search from the best candidate with minimal distortion. Repeated on-chip memory access happens for the first candidate if the first candidate is selected for ±1 search finally. The modified LV2 flow is efficient for hardware as shown in Fig. 1(b). It reduces the number of checked candidates to avoid longer processing cycles. It also removes the conditional check of the candidates

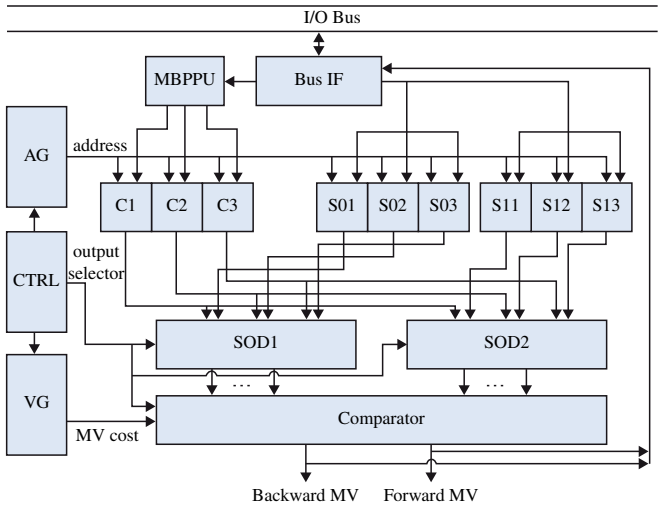

Fig. 3. System architecture for the modified ABME.

to improve parallelism and avoid repeated on-chip memory access for the selected candidate.

3) Parallel Processing of 8×8 *and* 16×16 *Block Searches:* In our MPEG-4 software [8], ± 2 of 8×8 block search is performed after 16×16 search is completed at integer pixel resolution. To support 8×8 block search, the memory bandwidth will be increased by 17%[1] while it is beneficial for area with complex motion. To balance the tradeoff between quality and bus bandwidth, a modified search method is used. To minimize bus access, we restrict the 8×8 block search to start at the same search center as the 16×16 LV3 search so that the search range is still within $[-16, +15]$. It enhances parallelism, and only suffers PSNR loss of at most 0.1dB.

III. Hardware Architecture

Fig. 3 shows the system architecture of the proposed design that implements the modified ABME algorithm with the parallel hardware support for B-frame search as described in Section II. For B-frame search, two sets of hardware are used to enable forward and backward search in parallel. The 8-bit data of current search block is passed to the preprocessing module (MBPPU) to generate three layers of binary search blocks. The three layers of binary data are stored in the on-chip memories C1–C3. The binary data of forward and backward reference frames are stored in two on-chip memories S01–S03 and S11–S13, respectively. The memory blocks of C1, S01, and S11 are for LV1. The memory blocks of C2, S02, and S12 are for LV2. The memory blocks of C3, S03, and S13 are for LV3.

For each layer of block search, the address generator (AG) controls the access of C0–C3, S01–S03, and S11–S13 to provide the necessary reference and target block data to the shared SOD processing units for block matching (SOD1 for forward and SOD2 for backward). The shared processing units first decide which layer of binary data to be used for calculation according to the control signal from controller (CTRL). Then, it computes the SOD between the selected current and reference search blocks. The matching results are sent to the comparator for final motion vector selection considering motion vector cost input from the motion vector generator module (VG).

[1] $(18 \cdot 2 + 16)^2 / (16 \cdot 2 + 16)^2 = 117\%$.

TABLE II
Comparison for Serial and Parallel Architecture

Architecture	Serial		Parallel	
	8-bit	1-bit	8-bit	1-bit
Memory size	$(8 \cdot M)$	M	$2 \cdot (8 \cdot M)$	$2 \cdot M$
Memory bandwidth	$8 \cdot (B_1 + B_2)$	$(B_1 + B_2)$	$8 \cdot (B_1 + 2 \cdot B_2)$	$(B_1 + 2 \cdot B_2)$
Running cycles	$2 \cdot C$	$2 \cdot C$	C	C
Hardware cost	$8 \cdot Q$	Q	$8 \cdot (2 \cdot Q)$	$2 \cdot Q$

For the forward-only P-frame search, the parallel architecture leaves half the hardware idle. Such an issue is addressed with a parallel P-frame search scheme. In the parallel P-frame search mode, the forward search data from S01–S03 are mirrored to S11–S13. The original forward search path through SOD1 module handles search for the odd positions, while the backward search path through SOD2 module handles search for the even positions. The AG/CTRL/VG/Comparator modules will send appropriate addresses and signals for each path. Thus, both paths are busy with half execution cycles.

Compared to the serial architecture, which processes forward and backward search sequentially, this parallel architecture enjoys five major advantages.

1) *Less on-chip memory access*: Parallel architecture reuses the current block search data, removes redundant on-chip memory access, and thus saves power. This is particularly important for the pyramid search structure since the current block data are changed for each layer of search.
2) *Higher overall hardware utilization*: In addition to full hardware utilization for ME, all the other modules in the pipeline enjoy higher utilization. Typically ME module takes the longest execution cycles as compared to other modules such as transform or motion compensation (MC), so it can become the design critical path in the overall system. The execution cycles are doubled for B-frame search, which leads to more idle cycles for the other modules such as entropy coding or transform. Thus, parallel architecture can not only halve the ME execution cycles but also reduce the idling of other modules. Compared to serial architecture, our parallel architecture decreases execution cycles for both P-frame search and B-frame search.
3) *Lower working frequency*: Lower working frequency is a key factor leading to a low-power design. Similar to the previous item, the B-frame ME search is typically the slowest module. In that case, it is the dominant factor to decide the system frequency. The parallel B-frame architecture, as opposed to serial architecture, improves the worst case scenario leading to the lowest system frequency. Combined with the voltage scaling technique, parallel architecture can achieve further power reduction.

4) *Less penalty in hardware cost*: It is less expensive to use parallel architecture for the binary search. If the system were to use full pixel (8-bit) for matching, parallel architecture suffers from more increase of hardware. Although the increase in percentage is the same, the binary search algorithm has smaller increase of hardware cost.
5) *Flexibility for joint optimization of B-frame search*: Joint optimization of B-frame search is a widely used encoding technique that jointly considers cost and distortion based on the forward and backward search results to provide better motion vectors. For a serial architecture, it needs to finish the forward and backward searches first. Then, the joint optimization can start. Parallel architecture can save cycles and the memory for storing first-pass results.

Table II summarizes the difference between the serial architecture and our parallel architecture. From this table, it shows the binary parallel search architecture has great advantages in memory size, memory bandwidth, running cycles, and hardware cost.

IV. Experimental Results and Analysis

A. Rate-Distortion (R-D) Performance Evaluation

Table III shows the R-D performance comparison for full search (FS), prior ABME work [6] (ABME1), and modified ABME (ABME2). Five commonly used MPEG test sequences are tested with group of pictures (GOP) structures of IPPPP and IPBPB. The search range is $[-16, +15]$. From this table, the PSNR loss for the modified ABME algorithm is up to 0.45 dB for IPPPP and 0.75 dB for IPBPB compared to full search. Compared with ABME1, up to 0.14 dB PSNR loss is observed for IPBPB.

TABLE III
R-D Performance for Full Search (FS), ABME Algorithm (ABME1) [6], and the Modified ABME (ABME2) Algorithm at the Bitrate of 512 kilo bps ($N = 300$)

Sequence	Method	IPPP 30 frames/s($M = 1$)		IBPBP 30 frames/s ($M = 2$)	
		PSNR_Y (dB)	ΔPSNR	PSNR_Y (dB)	ΔPSNR
Foreman	FS	34.18		34.62	
	ABME1	33.82	−0.36	34.01	−0.61
	ABME2	33.79	−0.39	33.87	−0.75
Akiyo	FS	43.33		43.47	
	ABME1	43.31	−0.02	43.39	−0.08
	ABME2	43.33	−0.00	43.52	+0.05
Flower	FS	26.11		26.56	
	ABME1	25.75	−0.36	26.29	−0.27
	ABME2	25.66	−0.45	26.30	−0.26
Mobile	FS	26.18		27.69	
	ABME1	26.10	−0.08	27.59	−0.10
	ABME2	26.07	−0.11	27.33	−0.36
Tempete	FS	28.78		29.89	
	ABME1	28.78	−0.00	29.67	−0.22
	ABME2	28.79	+0.01	29.55	−0.34

B. Hardware Design Performance

The hardware gate count is 62.6 kilogates and the on-chip memory size is 8.64 kilobits using TSMC 0.18 μm CMOS technology. For CIF 30 frames/s with GOP of IPPPP, the working frequency is 1.67 MHz and the power consumption is 763 μW measured by Synopsys PrimePower. The power consumption for CIF 30 frames/s with GOP of IPBPB is 896 μW with a working frequency of 1.94 MHz. The functionality of this IP core design has been verified on FPGA.

To compare with the state-of-the-art designs, the proposed design shows the advantage in power consumption. Table IV summarizes the design information for [1], [2], [4], [6]. Considering the designs with different CMOS technology, frequency, voltage, etc., the normalized power [9]

$$\text{normalized power} = \text{power} \times \frac{0.18^2}{\text{Process}^2} \times \frac{1.8^2}{\text{Voltage}^2} \quad (4)$$

is provided. From this table, it shows the proposed design can provide very low power consumption.

C. Bus Bandwidth Analysis

The bus bandwidth has been the bottleneck for modern SoC design, especially for data-intensive designs such as video. The 8-bit ME designs need whole search window to complete the motion search such as in [1], [2], [4]. They propose solutions to reduce gate counts or on-chip memory bandwidth, but not the bus bandwidth. In our case, the required data for movement are as follows and are also summarized in Table V.

1) Input current block data: 16 × 16 × 8 bits.
2) Input reference block data: For search range of $[-SR, +(SR-1)]$, we have the following.
   a) $(16 + 2 \times SR) \times (16 + 2 \times SR) \times 8$ bits if search window reuse is not considered.
   b) $16 \times (16 + 2 \times SR) \times 8$ bits if search window reuse is considered.
   c) $16 \times (16 + 2 \times (SR + 2)) \times 8$ bits if one reuses search window and applies 8 × 8 block search with ±2 range. If B-frame search is supported, it is doubled;
3) Output motion vectors: 80 bits for five pairs of motion vectors if 16 bits are assumed for a pair of motion vectors.

The total needed data movement for one block search is 8784 bits for P-search and 14 336 for B-search under SR is 16.

Considering our proposed design, the required data for movement are summarized in Table V. The input current block data are (18 × 18 × 8) bits for ABME2 and (30 × 30 × 8) for ABME1. The input reference block data with a reuse scheme are $16 \times (16 + 2 \times SR) \times 1$ bits for LV3, $8 \times (8 + 2 \times (SR/2)) \times 1$ bits for LV2, and $4 \times (4 + 2 \times (SR/4)) \times 1$ bits for LV1. If B-frame search is supported, the total bits are doubled. For outputting binarized data as reference picture for the next frame, there are 4 × 4 bits for LV1, 8 × 8 bits for LV2, and 16 × 16 bits for LV3. The output motion vectors (MV) are 80 bits for the five motion vectors if 16 bits are used for a pair of motion vectors. For the B-frame search, the MV bits are doubled due to the forward and backward directions.

In our design, the required data movement for one MB search is 4016 bits, which is 45.7% of the bandwidth for the 8-bit ME designs for the P-frame search. For the B-frame search, the data movement is 5104 bits, which is 32.9% of the bandwidth for 8-bit ME designs. Table V summarizes the bandwidth analysis results. It shows that the bus bandwidth

TABLE IV
PERFORMANCE COMPARISON WITH STATE-OF-THE-ART DESIGNS

Design	FS [4]	GME [2]	GDS [1]	ABME1 [6]	ABME2
Architecture	1-D systolic	Global elimination	Gradient search	Binary search	Parallel Binary search
Cycle/MB	4096	1784	568	283	148(P) 177(B)
On-chip memory (kilobits)	n.a.[1]	24.08	40	9.80	8.64
PSNR loss (dB)	0.00	0.08	0.10	0.19	0.23
Search range	$[-16, +15.5]$	$[-16, +15]$	$[-16, +15.5]$	$[-16, +15]$	$[-16, +15]$
Search block size	8×8 and 16×16	8×8 and 16×16	8×8 and 16×16	16×16	8×8 and 16×16
Process (μm)	0.60	0.35	0.13	0.18	0.18
Gate count (kilogates)	66.8	89.39	250	68.5	62.6
Power (mW)[2]	353	160	2.5	2.21	0.763
Normalized power (mW)[3]	4.12	12.59	15.53	2.20	0.763

[1]not available.
[2]mW for CIF 30 frames/s.
[3]normalized power $=$ power $\times$ $(0.18^2/\text{Process}^2) \times (1.8^2/\text{Voltage}^2)$ [9].

TABLE V
BUS BANDWIDTH ANALYSIS FOR 8-BIT ME SCHEME AND THE PROPOSED DESIGN (SEARCH RANGE $= [-16, +15]$)

Scheme	In/out data	8-bit	AMBE1 [6]	ABME2
P-frame	Current block	2048	7200	2592
	Ref. block	6656	1032	1008
	Binarized data	0	336	336
	Motion vectors	80	80	80
	Bandwidth[1]	8784	8648	4016
	Percentage[2]	100%	98.5%	45.7%
B-frame	Current block	2048	7200	2592
	Ref. block	13312	2064	2016
	Binarized data	0	336	336
	Motion vectors	160	160	160
	Bandwidth	15520	9760	5104
	Percentage	100%	62.9%	32.9%

[1]Data in bits for one MB search.
(= cur. block + ref. block + binarized data + motion vectors)
[2]Bandwidth of this paper/Bandwidth of 8-bit design.

savings for the proposed design are 54.3% and 67.1% for P-frame and B-frame searches, respectively.

D. Summary

The proposed architecture leads to a superior performance in bus access and power consumption.

The low-power advantage is attributed to the following:

1) the low complexity using binary SOD in ABME;
2) hardware efficient binary pyramid search structure, especially the hardware oriented LV2 search;
3) parallel processing of 8×8 and 16×16 LV3 block searches to minimize additional search cycles for 8×8 block search;
4) hardware support of B-frame parallel processing to reuse the current search block data and remove repeated on-chip memory access.

On the other hand, the high bandwidth efficiency is achieved by the following:

1) binary search structure to use binary data instead of 8-bit data for search;
2) MB level preprocessing unit to reduce the amount of bus access for generation of the three layers of binary pyramid structure;
3) parallel processing of 8×8 and 16×16 LV3 block searches to save additional 17% bus bandwidth for search range of $[-16, +15]$.

V. CONCLUSION

In this paper, we have proposed new ME hardware architecture to achieve low power and high bandwidth efficiency. The proposed design is developed from a very low complexity ME algorithm called all binary motion estimation [7]. It integrates several important features including: 1) MB based pre-processing; 2) support of B-frame parallel search; 3) parallel processing of 8×8 and 16×16 LV3 block searches; 4) shared processing units to reduce the hardware cost; 5) efficient LV2 search to reduce the latency. We also analyze how low power and high bandwidth efficiency can be achieved with the proposed design. Experiments show that the power consumption can reach as low as 763 μW for IPPPP CIF 30 frames/s and 896 μW for IPBPB CIF 30 frames/s. The bus bandwidth saving that can be achieved is up to 54.3% for P-frame only forward search and 67.1% for B-frame search.

REFERENCES

[1] M. Miyama, J. Miyakoshi, Y. Kuroda, K. Imamura, H. Hashimoto, M. Yoshimoto, "A sub-mW MPEG-4 motion estimation processor core for mobile video application," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1562–1570, Sep. 2004.
[2] Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, "Global elimination algorithm and architecture design for fast block matching motion estimation," *IEEE Trans. Circuit and Syst. Video Technol.*, vol. 14, no. 6, pp. 898–907, Jun. 2004.
[3] H.-M. Jong, L.-G. Chen, and T.-D. Chiueh, "Parallel architectures for 3-step hierarchical search block-matching algorithm," *IEEE Trans. Circuit and Syst. Video Technol.*, vol. 4, no. 4, pp. 407–416, Aug. 1994.
[4] 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. Circuit and Syst. Video Technol.*, vol. 11, no. 7, pp. 890–897, Jul. 2001.
[5] M. M. Mizuki, U. Y. Desai, I. Masaki, and A. Chandrakasan, "A binary block-matching architecture with reduced power consumption and silicon area requirement," in *Proc. IEEE ICASSP*, vol. 6. Atlanta, GA, May 1996, pp. 3248–3251.

[6] S.-H. Wang, W.-L. Tao, C.-N. Wang, W.-H. Peng, T. Chiang, "Platform-based design of all binary motion estimation with bus interleaved architecture," in *Proc. IEEE Int. Symp. VLSI-DAT*, Apr. 2005, pp. 241–244.

[7] J.-H. Luo, C.-N. Wang, and T. Chiang, "A novel all-binary motion estimation (ABME) with optimized hardware architectures," *IEEE Trans. Circuit and Syst. Video Technol.*, vol. 12, no. 8, pp. 700–712, Aug. 2002.

[8] *Final Committee Draft*, MPEG01/N4025, ISO/IEC 14496-5:2001.

[9] T.-C. Chen, Y.-H. Chen, S.-F. Tsai, S.-Y. Chien, L.-G. Chen, "Fast algorithm and architecture design of low-power integer motion estimation for H.264/AVC," *IEEE Trans. Circuit and Syst. Video Technol.*, vol. 17, no. 5, pp. 568–577, May 2007.