VLSI Design, 2002 Vol. 14 (3), pp. 299–305

Taylor & Francis
Taylor & Francis Group

Maximizing Memory Data Reuse for Lower Power Motion Estimation

BO-SUNG KIM and JUN-DONG CHO*

Department of Electronic Engineering, SungKyunKwan University, Suwon 440-746, South Korea

(Received 15 March 2000; Revised 15 April 2000)

This paper presents a new VLSI architecture of the Motion Estimation in MPEG-2. Previously, a number of full search block matching algorithms (BMA) and architectures using systolic array have been proposed for motion estimation. However, the architectures have an inefficiently large number of external memory accesses. Recently, to reduce the number of accesses in one search block, a block matching method within a search area to reuse the search data is provided using systolic process arrays. To further reduce the data access and computation time during the block matching, we propose a new approach through the reuse of the previously-search data in two dimensions. Our new architecture in this paper is an extension from our previous work such that we reuse the previously-searches area not only between two consecutive columns but also between two consecutive rows, so as to entirely remove redundant memory accesses. Experimental results show that our architecture of increased area by 81% can reduce 98% of memory accesses. Total power reduction is 86% in power estimation by SPICE model.

Keywords: Low power motion estimation; VLSI architecture; Image processing; Parallel architecture

INTRODUCTION

A number of VLSI architectures have been proposed for the block-matching motion estimation [1–6]: (1) three-step search algorithm [7–8], (2) block match algorithm with pixel sub-sampling [11], (3) block matching algorithm using temporal/spatial correlation [12], and (4) some other hierarchical search algorithms [9–10]. These architectures and implementations have a wide range of applications, from H.261 to MPEG-2 and high-definition TV (HDTV). In the H.263 format, three step search algorithm is usually used for motion estimation because it calculate the motion vector in short time and low cost, but the motion vector calculated by three step search algorithm may be local minimum solution. The full-search block-matching motion estimation algorithm is very accurate and easy for hardware implementation. However, computation complexity and excessive external memory accesses are drawbacks against VLSI implementation. The computation complexity results system to consume high power and to be needed scheduling. Furthermore, the granularity of computation for ME is 6 times greater than that of any other kernel in MPEG system [14]. Therefore, it will be necessary to execute all the other kernels six times per each execution of motion estimation in order to process the same size of data. Moreover, the motion estimation block is the most power-hungry block in MPEG and consumes 50% of power in total system. In addition to this problem, the number of external memory access results in high power consumption because of high capacitance of external-load wire. To solve the aforementioned problems of full search block matching algorithm, we propose a modified architecture that has an extremely lower number of external memory access [13]. The main idea is to reuse the previously matched data with removing redundant external memory accesses so as to minimize the power consumption and processing time. In order to implement a fast full search algorithm, we adapt a parallel array architecture.

In this paper, we first present an overview of the block-matching algorithm in the second section. In the third section, we extend the algorithm of [13] to further reduce the memory accesses without employing an efficient architecture. Finally, fourth section presents an experimental result and conclusion.

*Corresponding author.

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/10655140290011096

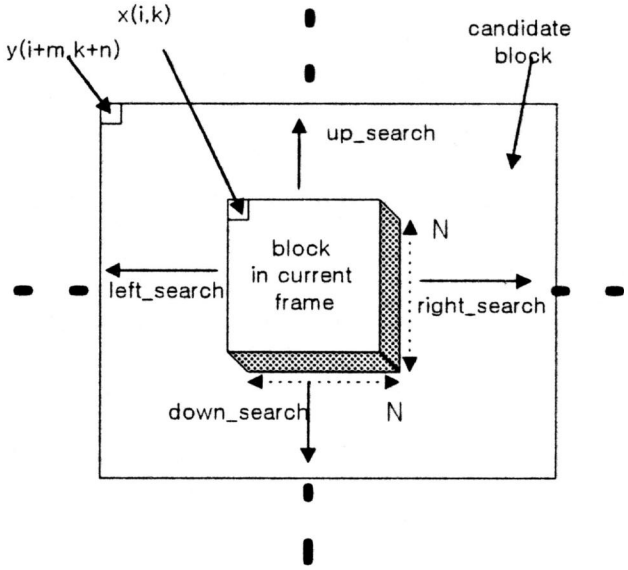

FIGURE 1 An instance of block matching.

PREVIOUS WORK OF BLOCK MATCHING ALGORITHM

Most video coding schemes apply a block-matching algorithm for the motion estimation. This is a scheme to find the best matching part of each current macroblock in the reference frame. Matching is evaluated in terms of a distance between the macroblock; the relative position where the distance takes the minimum is selected as the best match.

The block matching algorithm is summarized as follows. We first divide a frame in a squared block into a set of blocks. Each block is compared with a block in the search area (in the previous frame), looking for the most similar one, as shown in Fig. 1. Full search algorithm exhaustively searches all the search blocks in the searched area. The search area in the previous frame is determined by *left_search*, *right_search*, *down_search,* and *up_search* spanned from coordinate center of a current frame.

This matching procedure determines the optimum of the selected cost function. During the matching procedure, whenever a block in a current frame moves to the next block, a previously-searched data are repeatedly accessed for generating a motion vector. The cost function for the motion estimation of MPEG-2 and H.263 is the Minimum Mean Absolute Error (MAE). For simpler computational complexity (because hardware multiplier is not required) than Mean Square Error (MSE), the MAE is widely adopted for motion estimation.

The cost function is presented in Eq. (1), where N is the dividend block size in a frame, x's are the pels in the reference block; and y's are the pels within the search area. The *left_*, *right_*, *up_*, and *down_search* represent the search range of the search block.

$$S(m,n) = \sum_{i=1}^{N}\sum_{k=1}^{N} |x(i,k) - y(i+m,k+n)|, \qquad (1)$$

where $left\text{search} \le m \le right\text{search}$, $up\text{search} \le n \le down\text{search}$, $left\text{search}, up\text{search} \le 0$, $right\text{search}, down\text{search} \ge 0$,

$$u = \min_{(m,n)} S(m,n) \qquad (2)$$

$$v = (m,n)|_u \qquad (3)$$

Here m is the number of row pels in a search range of the previous block, and n is the number of column pels in a search range of the previous block. The sum $S(m, n,$ in Eq. (1), of the absolute differences between corresponding pel of reference block data $x(i, k)$ in the current frame and search block data $y(i + m, k + n)$ of the previous frame are added for each search block. The minimum error u in Eq. (2) of all sums $s(m, n)$ within a search area denotes the

Algorithm 1 2-way reuse scheme

```
Function(left_search,right_search,up_search,down_search,
         coordinate of x_pixel,coordinate of y_pixel)
  for(m=left_search;m ≤ right_search;m++)
    for(n=up_search;n ≤ down_search;n++)
      for(i=1;i ≤ N;i++)
        for(k=1;k ≤ N;k++)
          s(m,n)=|x(I,k)-j(I+m,k+n)|;
/*main full search algorithm per full one frame*/
l_s =0      /* initial value of search range */
r_s =right_search
d_s =down_search
u_s =0
 for column=1 to column_number do
  for row=1 to row_number do
    Function(l_s, r_s, u_s, d_s,row+i,column+k)
   /* calculation sum of difference pixel between
        current block and search block*/
    Function(l_s, r_s, u_s, d_s,row+N+i,column+k)
   /* calculation sum of difference pixel between
        next current block and search block*/
 l_s = N×column + left_search
 r_s = N×column + right_search
 d_s = N×row + down_search
 u_s = N×row + down_search
   /*search range vector*/
```

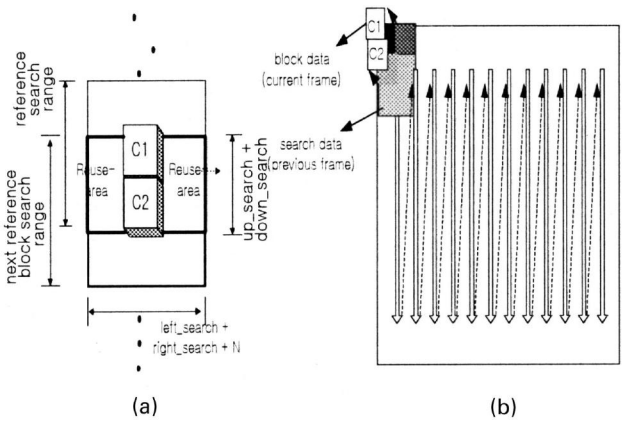

FIGURE 2 An instance of data reuse as in [13].

TABLE I Clock latency

Architecture	Clock latency
[3]	$(N + \|left\_search\| + right\_search) \times (N + \|up\_search\| + down\_search)$
Two cascaded [1]	$2 \times (N \times N)$
[13]: Our previous work	$(N + left\_search + \|right\_search\|) \times (N)$
Ours	$(N + right\_search) \times (N)$

position v of the best fitting search block which provides the displacement vector v in Eq. (3).

This method is inefficient due to the excessive data access. Therefore, the problem of generating motion vectors is how to avoid accessing the overlapping (i.e. reusable) search data. Recently, to reduce the number of accesses in one search block (which is a search area of a reference block), [8] used a block matching method within a search area to reuse the search data using three-step hierarchical search block-matching algorithm and [6] used one-dimensional processing element PE array and two data-interlacing shift-register array. Another scheme uses an embedded memory [15]. However, [8] often generates a local minimum solution and the implementation of hardware is difficult. Computational time of [6] is slow. The architecture of embedded memory is inefficient in area and implementation cost. To further reduce the data access and computational time during the block matching, in this paper, we propose a new approach through the reuse of the previously-searched data in four adjacent search blocks (called four-way reuse).

Recently, [13] introduced a reuse scheme with reusing the column-wise search data by computing motion vector simultaneously between a search block and two consecutive search blocks. Our new architecture in this paper is an extension from [13]. The difference is that we reuse the previously-searched data not only between two consecutive columns but also between two consecutive rows so as to reduce the memory data accesses. Whereas, the reuse scheme in [13] reuse the searched data between two consecutive rows. Figure 2 shows a data reuse scheme in [13]. In this case, the number of overlapped pels is $(up\text{search} + down\text{search}) \times (N + left\text{search} + right\text{search}) + N \times (left\text{search} + right\text{search})$ per each current block.

The idea is to reduce the number of accessed data by reusing those previously accessed data. During computation in a column, as shown in Fig. 2b, the number of data accesses in an overlapped area is decreased by reusing the data in *up_search* area and *down_search* area.

In addition to reduction of memory access, a total computational time is reduced because two functions compute parallely, motion vectors of two current block for one search candidate block. To be compared with conventional algorithm [3] of matching procedure, the search range vector reduced to $N \times row + down\text{search}$ by parallel computation. As a result, a clock latency of [3] is $(N + |left\text{search}| + right\text{search}) \times (N + |up\text{search}| + down\text{search})$. But a clock latency of algorithm 1 [13] is $(N + |left\text{search}| + right\text{search}) \times (N)$.

The data reuse is done by using two subtracters in one modified Process Element (PE). That is, two current block (c1 and c2 in Fig. 2b) is parallely searched to compute the motion vector with the same search data. The result is transferred to the next PE. As shown in Fig. 3, while the first subtractor in the modified PE computes the difference between the current search block and current reference

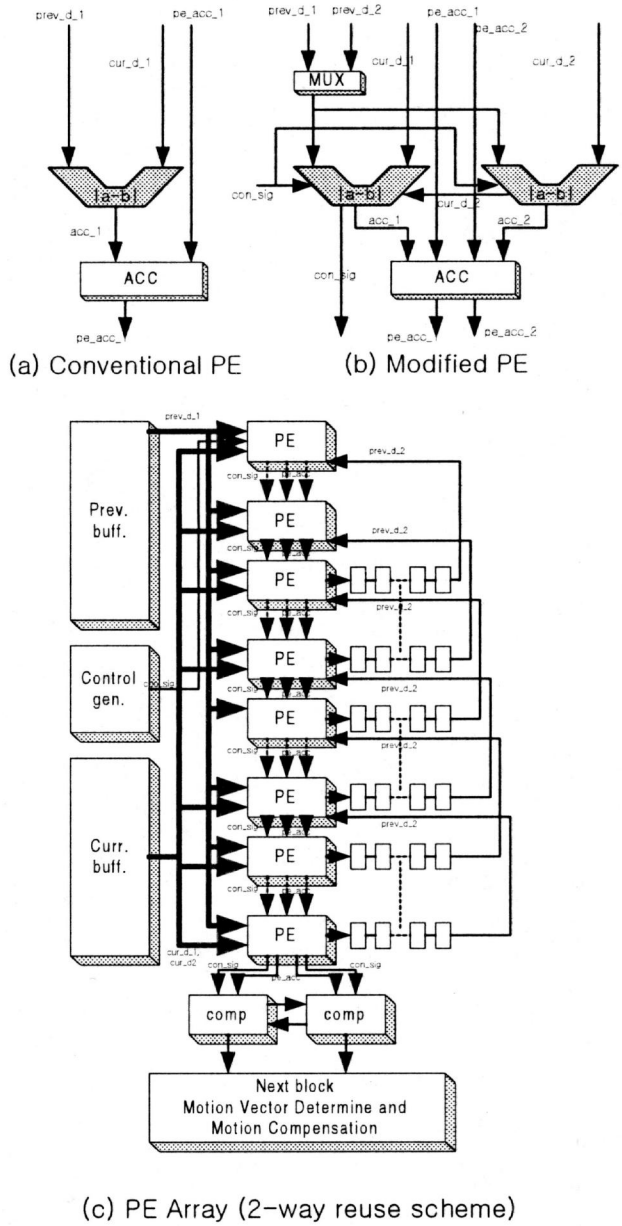

FIGURE 3 A conventional PE and a modified PE with two-way reuse scheme.

TABLE II The number of data accesses of [1,13], and ours (147 × 176 pels/frames)

Architecture	R1	R2	R3
[1]	161,847	174,090	173,538
[13]	44,247	47,565	49,290
Ours	6,192	3,024	1,440

R1: No of current blocks = 4 × 4, *up_search* region = 4, *down_search* region = 3, *right_search* region = 3, *left_search* region = 4;
R2: No of current blocks = 8 × 8, *up_search* region = 8, *down_search* region = 7, *right_search* region = 7, *left_search* region = 7;
R3: No of current blocks = 16 × 16, *up_search* region = 16, *down_search* region = 15, *right_search* region = 15, *left_search* region = 16;

block, the second subtractors simultaneously computes the difference between the next search block and next reference block in a single cycle to reuse the data of searching area in the next cycle time. That is, the modified PEs compute matching criterion between two reference blocks and one search block at one cycle time. Therefore, computational time is reduced by half and the number of data accesses is decreased proportional to the size of search region. The architecture is much efficient than other systolic array architectures in terms of speed and power consumption. However, the total area is inferior of other architecture [13].

OUR 4-WAY REUSE SCHEME

In this paper, in order to further remove the redundancy of row-wise searched data as well as column-wise searched data, we extend the algorithm of [13]. During the processing of one searched data, four subtractors calculate the difference of luminance between four current data and one previous data in parallel. We refer to the new algorithm as *four-way reuse scheme* while [13] is referred to as *two-way reuse scheme*.

Our *four-way reuse scheme* is described in Algorithm 2. In addition to reduction of memory access in row wise, a memory access in column wise is reduced because four functions compute parallely motion vectors of four current

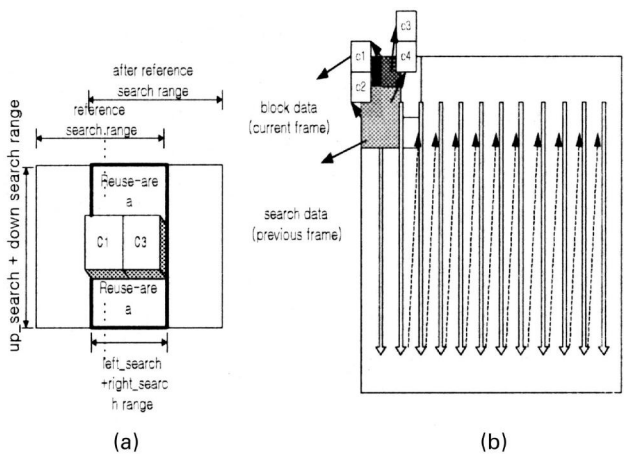

FIGURE 4 Four-way data reuse scheme. (a) row-wise data reuse. (b) four-way data reuse.

TABLE III Clock latency(4 × 4 current frame)

Architecture	4/-3 search	8/-7 search	16/-15 search
[3]	121	3600	4340
[1]	32 (cascade 2)	32 (cascade 4)	32 (cascade 8)
[13]	46	76	140
Ours	28	44	76

(No of clock)

block for one search candidate block. To be compared to algorithm 1 of matching procedure, the search range vector reduced to $N \times row + down$search and $N \times column + right$search by parallel computation. As result, a clock latency of algorithm 2 is $(N + right\text{search}) \times (N)$. The clock latency of other architecture is Table I. As shown in Fig. 4b, we compute the motion vector in parallel between a search data in a previous frame and four adjacent reference blocks in the current frame. Figure 5 represents the block diagram of our top-level system. In the figure, the shift register of search-data is used for reusing of search data in each search window because the column-wise data is reused by the PE in one search area. The shift register is 8-bit because of the bit-value of luminance. We match a search block with a current bock in a row-wise manner as shown in Fig. 4a. After processing entire column in a search area, we move to the next column and repeat the same procedure.

During this procedure, two PE arrays search individually the different column area. That is, our architecture shows a concurrent procedure of Figs. 3a and 4a. The front PE array (left-hand side in Fig. 5) generates a corresponding motion vector and the behind PE array (right-hand side) generates a partial solution of a corresponding motion vector.

The partial solution of motion vector is shifted to the front comparator unit through shift register. The number of shift registers is determined by a number of row-pixels. If the number of row-pixels is 144, the number of required shift registers is $144/N$. By applying four-way reuse scheme, computation time is reduced by four times. The number of memory accesses is reduced accordingly.

Our architecture is much efficient than any other systolic array architectures [1,2,13] in terms of speed and power consumption. Furthermore, the size of address-generating block becomes small, and the number of external pin is decreased because this architecture reuses search data. However, the total area is inferior of other architectures [2,13]. For our extended architecture, an additional buffer is used for the current data. The number of additional buffers is required due to manipulating the

TABLE IV The number of gates in VLSI implementation of [1,13] and ours (4 × 4 current block)

Architectures	No of gates in a frame
[1]	5,136
[13]	13,184
Ours	27,072

current data concurrently. Our scheme is to use two one-dimension arrays and one-dimensional shift register array in parallel.

Note that the shift register is used for data transfer without external memory access of the data in the overlapped data reusable regions. Therefore, in the next reference computation, the column-wise data access duplication becomes zero. However, we were not able to exhaustively remove entire row-wise redundancy. In order entirely to remove the redundancy, we need more hardwares which are not desirable.

Algorithm 2 4-way reuse scheme

```
Function(left_search,right_search,up_search,down_search,
         coordinate of x_pixel,coordinate of y_pixel)
for(m=left_search;m ≤ right_search;m++)
  for(n=up_search;n ≤ down_search;n++)
    for(i=1;i ≤ N-1;i++)
     for(k=1;k ≤ N-1;k++)
        s(m,n)=|x(i,k)-j(i+m,k+n)|;
/*main full search algorithm per one frame*/
l_s =0        /* initial value of search range */
r_s =right_search
d_s =down_search
u_s =0
 for column=1 to column_number do
   for row=1 to row_number do
    Function(l_s,r_s,u_s,d_s,row+i,column+k)
    /* calculation sum of difference pixel between
      current block and search block*/
    Function(l_s,r_s,u_s,d_s,row+i,column+k+N)
    /* calculation sum of difference pixel between
     next block and column-wise N shift search block*/
    Function(l_s,r_s,u_s,d_s,row+I+N,column+k)
    /* calculation sum of difference pixel between
     next block and row-wise N shift search block*/
    Function(l_s,r_s,u_s,d_s,row+I+N,column+k+N)
    /* calculation sum of difference pixel between
    next block and row and column-wise N shift search block*/
l_s = N×column + right_search
r_s = N×column + right_search
d_s = N×row + down_search
u_s = N×row + down_search
    /*search range vector*/
```

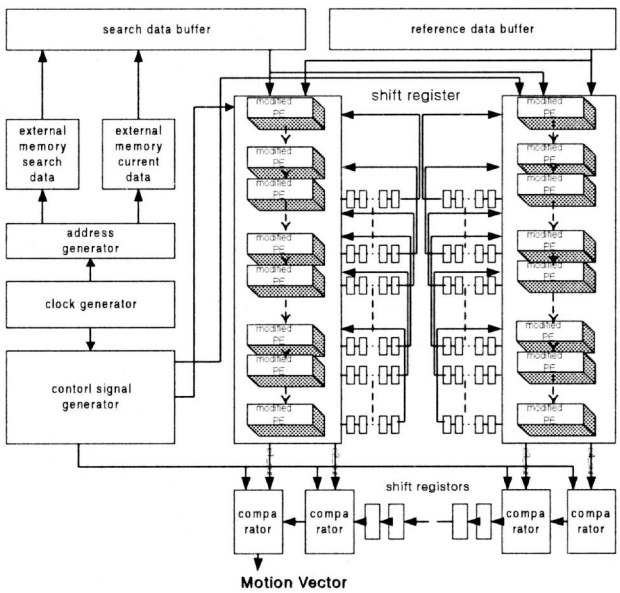

FIGURE 5 Our top-level block diagram for four-way data reuse scheme.

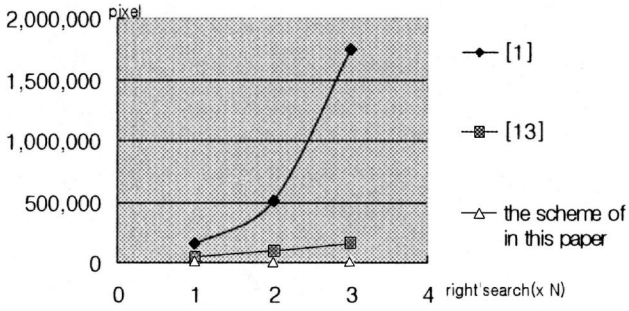

FIGURE 6 The number of overlapped data for various search ranges.

EXPERIMENTAL RESULTS

We implemented and verified the proposed algorithm and architecture using Synopsys™. Table II shows the number of redundant memory references compared with one of [1,13] and ours. Figure 6 is a simulation result of the number of data overlapping on various search ranges. In Fig. 6, we can see that the number of overlapped pixel is increased as to search range increasing. But the overlapped pixel number of our architecture and [13] is increased smaller than [1]. Because of reducing the number of memory access, the power consumption of our architecture is lower than [1,13].

Table III is clock latency for various search range. Our architecture is faster than [3,13]. Because the architecture of [1] uses current block data for propagated data in a PE array, in proportion to search area, the PE array must be cascaded. So in a difference PE array, a number of memory access is very high and area cost is very high.

Figure 7 is the number of gates used for our algorithm implementation compared with the conventional algorithm [1] using various search ranges. Our architecture requires additional subtractors in PE to reduce the memory access redundancy. If *right_search* is *N*, then the number of subtractors required in PE is two in order to remove the column-wise redundancy and the number of PE arrays required is two in order to reduce the row-wise redundancy. If a *right_search* is 2*N*, then the number of subtracters required in PE is four and the number of PE arrays required is three. Also, if *right_search* is *3N*, the number of subtracters required in PE is six and the number of needed PE array is four.

Our architecture efficiently removed the memory access redundancy, thus it is time and power-efficient. However, the number of gates is increased in our architecture. Table IV shows the comparison of the number of gates for 4×4 current-block between our algorithm and ones of [1,13]. Table V is a comparison of power consumption also see Fig.8. The capacitance-power consumption per gate is 0.058 pJ and the load power consumption is 0.365 pJ. In Fig. 5 is shown that the architecture of reused memory is very efficient for power consumption.

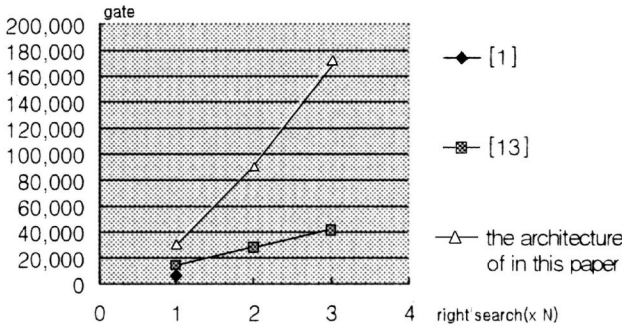

FIGURE 7 The number of gates required for our architecture using various search ranges.

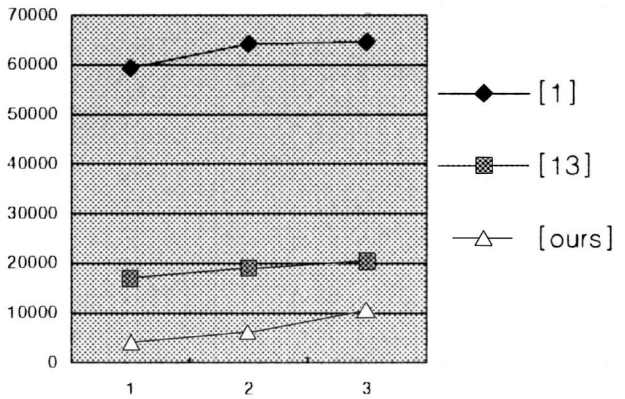

FIGURE 8 The comparison of power.

The area of our architecture is increased by one PE array, a shift register array to propagate the partial solution to the next column matching and two additional comparators. Note that in the computation block, the number of gates in our architecture is increased, but the number of gates in the address-generating-block is decreased because the number of accessing search data has been decreased. In addition, the speed of our architecture is much faster than that of other architectures in [1,6,13]. For example, our architecture is faster than one of [1] by four. Due to the speed enhancement, our architecture is effectively adaptive for HDTV.

TABLE V The comparison of power estimation ([1,13],ours)

Architectures	R1	R2	R3
[1]	59,405	64,204	64,665
[13]	16,968	18,985	20,349
Ours	4075	6333	10,495

(power per gate = 0.058(pJ), power per access = 0.365(pJ)).

References

[1] Sanz, Cesar, Garrido, Matias J. and Meneses, Juan M. (1996) "VLSI architecture for motion estimation using the block-matching algorithm", *Proceedings of the European Design \& Test Conference*, 0.

[2] Chen, Sheu-Chih and Hang, Hsueh-Ming (1997) "A comparison of block-matching algorithms", *IEEE Transactions on Circuits and Systems for Video Technology* **7**(5), 0.

[3] Komarek, Thomas and Pirsh, Peter (1989) "Array architectures for block matching algorithms", *IEEE Transactions on Circuits and Systems* **36**(10), 0.

[4] Yang, K.M., Sun, M.T. and Wu, L. (1986) "A family of VLSI designs for the motion compensation block-matching algorithm", *IEEE Transactions on Circuits and Systems* **36**, 1317–1325.

[5] Uramoto, S.I., Takabatake, A., Sakurai, M. and Yoshimoto, M. (1993) "A half-pel precision motion estimation processor", *Proceeding of the IEEE Custom Integrated Circuits Conference*, 11.2.1–11.2.4.

[6] Lai, Yeong-Kang and Chen, Lian-Gee (1998) "A data-interlacing architecture with two-dimensional data-reuse for full-search block-matching algorithm", *IEEE Transactions on Circuits and Systems for Video Technology* **8**(2), 0.

[7] Me, Z. and Lion, M.I. (1994) "A new array architecture for motion estimation", *Proceeding of the IEEE Workshop on Visual Signal Processing and Communications*, 0.

[8] Long, H.M., Chen, I.G. and Chiueh, T.D. (1994) "Parallel architectures for 3-step hierarchical search block-matching algorithm", *IEEE Transactions on Circuits and Systems for Video Technology* **4**, 407–416.

[9] Jiang, Li, Ito, Kazuhito and Kunieda, Hiroaki (1997) "Bits truncation adaptive pyramid algorithm for motion estimation of MPEG", *IEICE transactions on Fundamentals* **E80-A**(8), 1438–1445.

[10] Li, R., Zeug, B., Lion, M.I. and New, A. (1994) "Three-step search algorithm for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology* **4**, 438–442.

[11] Liu, B. and Zaccarin, A. (1993) "New fast algorithms for the estimation of block motion vectors", *IEEE Transactions on Circuits and Systems for Video Technology* **7**(3), 148–157.

[12] Chalidabhongse, J. and Kuo, C.C.J. (1997) "Fast motion vector estimation using multiresolution-spatio-temporal correlations", *IEEE Transactions on Circuits and Systems for Video Technology* **7**(3), 0.

[13] Kim, B.S. and Cho, J.D. (1999) "VLSI architecture for low power motion estimation using high data access reuse", *The first IEEE Asia Pacific Conference on ASIC* **1**(8), 162.

[14] Maestre, R., Kurdahi, F.J., Bagherzadeh, N., Singh, H., Herminda, R. and Fernandez, M. (1999) "Kernel scheduling in reconfigurable computing", *Design, Automation and Test in Europe, Conference and Exhibition 1999. Proceedings*, 90–96.

[15] Tuan, J.C. and Jen, C.W. (1998) "An architecture of full-search block matching for minimum memory bandwidth requirement", *Proceedings of the Great Lakes Symposium on VLSI'*, 98.

Authors' Biographies

Bo-sung Kim is with the Electronics and Computer Engineering Department of Sung kyun kwan University, Korea. He studies in Low Power VLSI Design and Algorithmic Low Power Architecture for image processing, multimedia processing, and communication algorithm. He is an IEEE Student Member.

Jun-dong Cho received the B.S degree in Electrical Engineering from sung Kyun Kwan University, Seoul, Korea, in 1980, the M.S degree from Polytechnic University, Brooklyn, NY, in 1989, and the PhD degree from Northwestern University, Evanston, IL, in 1993, both in computer science. He was a Senior CAD Engineer at Samsung Electronics, Co., Ltd, Buchun, Korea. He is now an Assocaite Professor of Electronic Engineering, Sung Kyun Kwan University, Suwon, Korea. His research interests are in the area of VLSI CAD algorithms and lower power designs. He received the Best paper award at the 1993 Design Automation conference in the area of

Physical Design. He has a book High-Performance Design Automation of MCMs and Packages, World Scientific, 1996, and the co-author of an invited chapter in Encyclopedia of Electrical and Electronics Engineering in the area of GLSI Circuit Layout. He has been a guest editor of VLSI DESIGN for the several special issues. He also served on the technical committee of several International Conferences. He is an IEEE Senior Member.