Expert system based parallel multi-1D block matching algorithm with implementation for motion estimation

Shin-Yeu Lin a,*, Chong-Wei Su b, Jung-Shou Huang c

a Department of Electrical Engineering & Green Technology Research Center at Chang Gung University, Taoyuan, Taiwan, ROC
b Institute of Electrical and Control Engineering at National Chiao Tung University, Hsinchu, Taiwan, ROC
c Elan Electronics Corporation, Hsinchu, Taiwan, ROC

ABSTRACT

In this paper, we propose an expert-system based parallel multi-1-dimensional block matching algorithm (ESPM-1D-BMA) for motion estimation (ME). Instead of the conventional 2D block matching, we employ the parallel multi-1D blocks matching to improve the computing speed. To improve the ME accuracy, we design a knowledge base and inference engine to determine the true motion vector (MV) from the results of parallel multi-1D blocks matching. To speed up the computing speed further, we present a hardware architecture for implementing the ESPM-1D-BMA. We have demonstrated that the MV estimation accuracy achieved by the proposed ESPM-1D-BMA is much better than the comparing fast block matching algorithms and is close to the 2 dimensional full search block matching algorithm (2D-FSBMA). We also demonstrate that the computing speed of the proposed ESPM-1D-BMA is about two times as fast as the mixed-signal 2D-FSBMA (MS-2D-FSBMA).

1. Introduction

Two dimensional (2D)-block matching algorithm (BMA) is a commonly adopted method for searching the motion vector (MV) between two image frames namely the reference and the search frames, such that the MV is obtained when the best matched 2D-blocks, the reference and one search blocks are found. Among various BMAs, the 2D-full search BMA (2D-FSBMA) using the mean square error (MSE) criteria (Gharavi & Mills, 1990) is considered to be the most accurate algorithm for searching the MV. However, the 2D-FSBMA is computationally complex. Hence, some fast BMAs were proposed, such as the new three-step search (NTSS) (Li, Zeng, & Liou, 1994), diamond search (DS) (Zhu & Ma, 1997) and the hexagonal based search (HEXBS) (Zhu, Lin, & Chau, 2002). These algorithms use few search points to reduce computational complexity, however at the price of poor accuracy. Therefore, proposing a method to reduce the computational complexity of 2D-FSBMA while maintaining its accuracy in searching the MV is the purpose of this paper.

Instead of 2D-block matching, we will slice a 2D block into multiple, say \( K \), 1D blocks and employ a parallel multi-1D blocks matching to reduce the computational complexity in searching the MV. Due to the noise appearing in the search and reference frames, the 1D-block matching should be less accurate than the 2D-block matching in searching the MV. Additionally, the MVs determined in each of the \( K \) 1D-blocks matching may be different due to various noise contaminations in various 1D blocks. Therefore, to remedy the possible inaccuracy in searching the MV using parallel multi-1D blocks matching, we propose an expert system based parallel multi-1D-BMA (ESPM-1D-BMA).

For the purpose of real-time motion estimation (ME), we need to improve the computing speed further by implementing the proposed algorithm in hardware. Therefore, we will present the hardware implementation architecture of the proposed algorithm.

We organize our paper in the following manner. In Section 2, we will present the proposed ESPM-1D-BMA. In Section 3, we will present the hardware implementation architecture of the proposed algorithm. In Section 4, we will test the performance of the proposed algorithm and compare with other existing methods in terms of ME accuracy and the computing speed using comprehensive simulations. Finally, we will draw a conclusion in Section 5.

2. Expert system based parallel multi-1D block matching algorithm (ESPM-1D-BMA)

2.1. Review of 2D-FSBMA

We let \( I_n \) and \( I_{n+1} \) in Fig. 1(a) denote the reference and search frames, respectively; we let block \( A \) inside \( I_n \) denote the reference
block (RB) and let block B, which can be any block in \( I_{n-1} \), denote the search block (SB). The idea of 2D-FSBMA is to search all possible SBs and find the one that is most similar to A. This searching task is performed by computing the MSE between blocks A and B as described below. We assume that the sizes of the RB A (or SB B) and frame \( I_n \) (or \( I_{n-1} \)) are \( X \times Y \) and \( M \times N \) pixels, respectively, as shown in Fig. 1(b). The MSE between two blocks, RB A and SB B, induced in 2D-FSBMA, denoted by \( MSE_{2D} \), is defined as

\[
MSE_{2D} = \frac{1}{X \cdot Y} \sum_{i=1}^{X} \sum_{j=1}^{Y} [r(i,j) - s(i,j)]^2
\]

where \( r(i,j) \) and \( s(i,j) \) denote the \((i,j)\)th pixel values of the RB A and SB B, respectively. Therefore, the 2D-FSBMA will search through all possible SBs to find the one with smallest MSE, say B’. Then the MV is defined as the difference of position indices between RB A and SB B’ as shown in Fig. 1(a).

2.2. Motivation

To improve the computing speed of 2D-FSBMA, we will slice the 2D block into multi-1D blocks and apply a parallel multi-1D blocks matching algorithm. However, the MVs determined in each of the multi-1D blocks matching may be different due to various noise contaminations in various 1D blocks. Therefore, we will use the expert system concept to help determine the true MV. In the following, we will describe the proposed algorithm step by step.

2.3. The 1D block matching

We let B denote the number of pixels in 1D block, then a 1D block is formed by the B consecutive pixel values from a row of the frame. We let \( r(i), i = 1, \ldots, B \) and \( s(i), i = 1, \ldots, B \) denote the B pixel values of the 1D reference block and the 1D search block, respectively. Then, the MSE between \( r(i), i = 1, \ldots, B \) and \( s(i), i = 1, \ldots, B \) can be computed as follows:

\[
MSE(SBC) = \frac{1}{B} \sum_{i=1}^{B} (r(i) - s(i))^2
\]

where SBC represents the 1D search block coordinate (SBC), which is identified by the coordinate of the first pixel of the 1D search block, \( s(1) \).

2.4. Parallel multi-1D blocks matching

To improve the ME accuracy of 1D block matching, while keeping its computational efficiency, we can use a parallel multi-1D blocks matching. The multi-1D blocks matching consists of \( K \) 1D reference blocks, and each 1D reference block represents one independent 1D reference block in the reference frame as shown in Fig. 2, in which we assume \( K = 8 \).

We let \( r_k(i), i = 1, \ldots, B \) denote the kth 1D reference block in \( I_n \) and let the \( MSE_k(SBC) \) denote the MSE between \( r_k(i), i = 1, \ldots, B \) in \( I_n \) and the search 1D block, \( s(i), i = 1, \ldots, B \) in \( I_{n-1} \), then \( MSE_k(SBC) \) for \( k = 1, \ldots, K \) can be computed by

\[
MSE_k(SBC) = \frac{1}{B} \sum_{i=1}^{B} (r_k(i) - s(i))^2
\]

which can be performed independently and in parallel for each k.

2.5. Expert system based parallel multi-1D blocks matching algorithm (ESPM-1D-BMA)

As described earlier that the searched MV based on each of the \( K \) 1D-blocks matching may be different due to various noise

![Fig. 1. Motion vector determination using 2D-FSBMA. (b) Example reference block and image frame.](image1)

![Fig. 2. A diagram of the \( K \) independent 1D blocks in a reference frame.](image2)
contamination in the search and reference 1D blocks. However, we can view the MSEs computed from the 1D block matching for a reference block in \( L_a \) as a result determined by an expert. Therefore, the MSEs resulted from \( K \) 1D blocks matching can be viewed as a result determined by \( K \) experts. Subsequently, to determine the true MV, we can employ the concept of expert system (Liao, 2005; Li & Sun, 2009; Sun & Li, 2008) to construct the knowledge base (KB) and the inference engine (IE) for the parallel multi-1D blocks matching as follows.

The input of the employed expert system is the overall resulted MSEs of the \( K \) 1D-blocks matching. For each of the \( K \) 1D-blocks matching, the true MV should be among the MVs with top smallest MSEs. Therefore, the employed KB of the proposed algorithm can be stated as follows. For each of the \( K \) 1D blocks matching, we select the \( P \) search blocks that correspond to the top \( P \) smallest MSEs and assign them with the marked numbers (MNs) \( P, P - 1, \ldots, 1 \), such that the selected search block with smaller MSE is marked by a larger MN. For example, the MN assigned to the search block with smallest MSE is \( P \). Since each selected search block may conclude an MV, there will be \( K \times P \) MVs resulted from the parallel multi-1D blocks matching, and each MV is associated with the MN of the corresponding search block.

However, some of the \( K \times P \) MVs may be the same. In general, the frequently appearing MVs and the MV with smaller MSE have higher probability to be the true MV. Consequently, if an MV appears \( q \) times in the \( K \times P \) MVs, it will associate with \( q \) MNs. Therefore, the IE of our expert system can be stated as follows. For each distinct MV with \( q \) copies in the resulted \( K \times P \) MVs, we will sum the \( q \) MNs and define the resulting value as the accumulated MN (AMN) of the MV. Consequently, the MV with largest AMN is determined to be the true MV.

For the sake of illustration, we use the following example to explain the proposed ESPM-1D-BMA. We assume \( K = 8 \) and \( P = 3 \). In Table 1, the first column shows the \( K \) 1D reference blocks, and the second, the third and the fourth columns show the \( P (=3) \) MVs with \( P \) smallest MSEs resulted from the 1D block matching for each reference block. Then the MVs in columns 2, 3, and 4 are assigned with MNs 3, 2, and 1, respectively. Calculating the AMNs of the seven distinct MVs presented in Table 1, we find that (3,2) has the largest AMN, 17, and is considered to be the true MV.

### Table 1
The top 3 MV for each 1D reference block.

<table>
<thead>
<tr>
<th>Index of reference block</th>
<th>MV with smallest MSE</th>
<th>MV with 2nd smallest MSE</th>
<th>MV with 3rd smallest MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3,2)</td>
<td>(5,8)</td>
<td>(1,5)</td>
</tr>
<tr>
<td>2</td>
<td>(3,2)</td>
<td>(5,5)</td>
<td>(8,9)</td>
</tr>
<tr>
<td>3</td>
<td>(5,8)</td>
<td>(3,2)</td>
<td>(8,8)</td>
</tr>
<tr>
<td>4</td>
<td>(8,8)</td>
<td>(3,2)</td>
<td>(5,8)</td>
</tr>
<tr>
<td>5</td>
<td>(8,9)</td>
<td>(6,3)</td>
<td>(5,5)</td>
</tr>
<tr>
<td>6</td>
<td>(5,8)</td>
<td>(6,3)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>7</td>
<td>(3,2)</td>
<td>(3,5)</td>
<td>(1,5)</td>
</tr>
<tr>
<td>8</td>
<td>(3,2)</td>
<td>(6,3)</td>
<td>(1,5)</td>
</tr>
</tbody>
</table>

**Fig. 3.** The hardware implementation architecture of the 1D block matching using mixed-signal approach.

### 3. Implementation for motion estimation

For the purpose of real-time ME, we can speed up the computing speed of the proposed algorithm further by hardware implementation. However, a pure digital circuit implementation (Hsieh & Lin, 1992; Yang, Wolf, & Vijaykrishan, 2005) may suffer from some implementation problems, such as high power consumption and large chip size. To overcome these implementation problems, a mixed-signal approach that uses simple current-summation circuit to circumvent computationally complex digital MSE computation should be a good choice (Panovic & Demosthenous, 2006). In the following, we will present the implementation of the proposed algorithm using mixed-signal approach step by step.

#### 3.1. Implementing 1D-block matching using mixed-signal approach

First of all, we transformed the sensed pixel values into voltages within the range \([0 \text{ V}, 2.5 \text{ V}]\) by dividing the range of pixel values between black and white into 255 grey levels, which are represented by 255 least significant bits (LSBs), such that black and white correspond to 0 and 255 LSB, respectively. We let \( V_s \) denote the transformed voltage of a pixel value of \( x \) LSB, such that \( V_{s} = \frac{2.5V - 0V}{255} \times x \). We let voltages \( V_r(i) \) and \( V_s(i) \) denote the transformed voltages of \( r(i) \) and \( s(i) \), respectively. Assuming \( B = 8 \), the mixed signal approach for the 1D block matching hardware implementation architecture is presented in Fig. 3, in which we preload the transformed voltage \( V_s(i) \) of the pixel value of the 1D reference block \( r(i), i = 1, \ldots, B \) in \( L_s \) into the reference block memory (RBM). Then the transformed voltage \( V_s(i) \) of the pixel value of the search frame \( L_s - 1 \) is serially fed into the search block memory (SBM) from left to right, from top to bottom, then the transformed voltage of the pixel value of the 1D search block will be fetched from the SBM as shown in Fig. 3. The two clock signals \( t_c \) and \( t_r \) are used...
3.2. Implementing parallel multi-1D blocks matching

The parallel computing architecture for the $K$ 1D blocks matching is presented in Fig. 4. The $K$ 1D reference blocks are preloaded into RBM$_k$, $k = 1, \ldots, K$, and the search block in the frame $I_{n-1}$ is serially fed into SBM. The MSE$_k$, $k = 1, \ldots, K$ are computed in parallel to obtain MSE$_k$(SBC), $k = 1, \ldots, K$. The detailed structure of SBM, RBM$_k$, and MSE$_k$ in Fig. 4 are the same as that presented in Fig. 3.

3.3. Implementing ESPM-1D-BMA

To implement the ESPM-1D-BMA, we need a PWC to select the $P$ search blocks that correspond to the top $P$ smallest MSEs resulted from the 1D block matching for each 1D reference block. The circuit of PWC for the $k$th 1D reference block, denoted by PWC$_k$, is presented in Fig. 5, which is designed based on a sorting logic; the solid lines and dotted lines in this figure represent the transmission of data and signals, respectively.

At the very beginning, the $P = 3$ sample and hold circuits (SHs) are reset to a default value, which is the largest value that SH can take; similarly, the corresponding P digital memories (DMs) are reset to null values. Then, for each incoming MSE$_k$(SBC), we will compare it with the MSE values stored in the $P$ SHs as indicated by the P comparators, denoted by COMP, $i = 1, \ldots, P$, shown in Fig. 5. COMP$ _i$ will generate an enable signal $en_i$, whose value depends on the comparison result such that if MSE$_k$(SBC) $< $ MSE$_{SH_i}$, then $en_i = 1$; otherwise $en_i = 0$. The combination of $en_1, \ldots, en_P$ will indicate which of the following ranges that MSE$_k$(SBC) lies: $[0, MSE_{SH_1}]$, $(MSE_{SH_1}, MSE_{SH_2}]$, $\ldots$, $(MSE_{SH_{P-1}}, MSE_{SH_P}]$, or $(MSE_{SH_P}, \infty]$ as presented in the illustrative table in Fig. 5, in which we set $P = 3$. From the value range of the incoming MSE$_k$(SBC), we can easily update the $P$ winners as follows. If $en_i = 1$, MSE$_{SH_i}$, the content of SH$_i$, should be replaced by either MSE$_{SH_{i-1}}$ for the case $en_{i-1} = 1$ or MSE$_k$(SBC) for the case that $en_{i-1} = 0$ or $i = 1$, and the content of the corresponding DM$_i$ will be replaced by the proper SBC accordingly. To implement the above $P$ winners updating logic, we will design a selection signal, sel, to determine what to replace the contents of SH and DM, when the enable signal $en_1 = 1$ in the following manner. If $en_1 = 1$ and $sel = 1$, the contents of SH$_1$ and DM$_1$ will be replaced by SH$_{i-1}$ and DM$_{i-1}$, respectively. If $en_1 = 1$ and $sel = 0$, the contents of SH$_1$ and DM$_1$ will be replaced by MSE$_k$(SBC) and the corresponding SBC, respectively. If $en_1 = 0$, SH$_1$ and DM$_1$ remain unchanged. The values of $sel_i, i = 2, \ldots, P$, which are determined based on the values of $en_i, i = 1, \ldots, P$, are presented in the illustrative table of Fig. 5, and they can be generated using AND gates.

![Fig. 4. The computing architecture of parallel multi-1D block matching.](image-url)

![Fig. 5. The circuit of PWC and illustrations.](image-url)
3.4. Hardware Implementation Architecture of ESPM-1D-BMA for ME

The average ME accuracies of the 5000 estimated MVs.

<table>
<thead>
<tr>
<th>$K$</th>
<th>$P$</th>
<th>Football</th>
<th>Greens</th>
<th>Concord</th>
<th>Fabric</th>
<th>Hestain</th>
<th>Pears</th>
<th>Tissue</th>
<th>Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>91.70</td>
<td>98.84</td>
<td>89.98</td>
<td>96.62</td>
<td>98.60</td>
<td>81.16</td>
<td>98.04</td>
<td>98.34</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>95.11</td>
<td>98.96</td>
<td>92.12</td>
<td>98.31</td>
<td>99.14</td>
<td>86.20</td>
<td>97.82</td>
<td>98.76</td>
</tr>
<tr>
<td>3</td>
<td>96.04</td>
<td>99.00</td>
<td>93.18</td>
<td>98.89</td>
<td>99.30</td>
<td>99.02</td>
<td>99.97</td>
<td>97.92</td>
<td>98.80</td>
</tr>
<tr>
<td>4</td>
<td>96.51</td>
<td>99.36</td>
<td>94.18</td>
<td>99.02</td>
<td>99.20</td>
<td>91.84</td>
<td>97.82</td>
<td>98.70</td>
<td>97.67</td>
</tr>
<tr>
<td>5</td>
<td>97.36</td>
<td>99.04</td>
<td>94.12</td>
<td>99.17</td>
<td>98.92</td>
<td>91.30</td>
<td>98.04</td>
<td>98.72</td>
<td>98.72</td>
</tr>
<tr>
<td>4</td>
<td>98.27</td>
<td>99.60</td>
<td>94.32</td>
<td>99.59</td>
<td>99.54</td>
<td>93.74</td>
<td>98.66</td>
<td>98.72</td>
<td>98.72</td>
</tr>
<tr>
<td>2</td>
<td>98.83</td>
<td>99.36</td>
<td>95.18</td>
<td>99.53</td>
<td>99.16</td>
<td>95.96</td>
<td>98.30</td>
<td>98.98</td>
<td>98.72</td>
</tr>
<tr>
<td>3</td>
<td>99.08</td>
<td>99.44</td>
<td>95.40</td>
<td>99.59</td>
<td>99.04</td>
<td>97.24</td>
<td>98.28</td>
<td>98.97</td>
<td>98.72</td>
</tr>
<tr>
<td>4</td>
<td>99.39</td>
<td>99.42</td>
<td>96.20</td>
<td>98.87</td>
<td>99.20</td>
<td>97.60</td>
<td>98.32</td>
<td>98.96</td>
<td>98.72</td>
</tr>
<tr>
<td>5</td>
<td>99.56</td>
<td>99.48</td>
<td>96.10</td>
<td>99.70</td>
<td>99.12</td>
<td>97.80</td>
<td>98.12</td>
<td>98.72</td>
<td>98.72</td>
</tr>
</tbody>
</table>

We let $T_\text{IE}$ denote the computation time or propagation time of unit $\text{IE}$. In addition to $T_\text{IE}$, there are other time delays need be considered such as (i) the time delay incurred from DC to guarantee a safety margin of controlling the action of the $K$ units of same type of components in parallel and (ii) the wire delay between units. However, comparing with $T_\text{IE}$, these time delays are negligible. Therefore, the estimated time needed to generate an MV of the ESPM-1D-BMA can be stated in the following: 

$$ (M \times N)(T_{\text{SH}} + T_{\text{MSE}} + T_{\text{PWC}}) + T_{\text{IE}} $$ 

where $T_{\text{SH}}$ denotes the time for loading a pixel value into RBM or SBM, $T_{\text{MSE}}$ denotes the time required in computing the MSE of 1D block matching, which is the propagation time of the circuit.
presented in Fig. 3, $T_{PWC}$ denotes the propagation time of the PWC$_k$ presented in Fig. 5 and $T_{IE}$ denotes the processing time of the IE presented in Fig. 8.

### 4. Test results and comparisons

In this section, we will demonstrate the ME accuracy achieved by the proposed ESPM-1D-BMA in comparison with the 2D-FSBMA and some fast block matching algorithms (Li et al., 1994; Zhu et al., 2002; Zhu & Ma, 1997) using extensive simulations. We will also compare the computational efficiency of the proposed ESPM-1D-BMA with the 2D-FSBMA. We use Matlab and their eight pictures, football, greens, concord, fabric, hestain, pears, tissue and board, as our simulation tool and test bed, respectively. The eight tested pictures were all formatted as grey-level image in eight-bit. We set our simulation tool and test bed, respectively. The eight tested pictures, we arbitrarily pick an image frame of $X = 8$ and $Y = 8$ for 2D-FSBMA, and $B = 8$ for ESPM-1D-BMA. However, we will use various combinations of $K$ and $P$ to test the performance of ESPM-1D-BMA. For each tested picture, we randomly generate an MV ranging from $-\frac{X}{2}$ to $\frac{X}{2}$ and from $-\frac{Y}{2}$ to $\frac{Y}{2}$ in $x$ and $y$ directions, respectively, and apply it to $I_n$ to form a noise free $I_{n-1}$. For each pixel in the noise free $I_{n-1}$, we randomly generate a noisy signal based on a normal distribution with mean 0 LSB and variance 3 LSBs and add it to the pixel. The resulted frame will serve as the test frame $I_{n-1}$. For each one of the eight tested pictures, we prepare 5000 ($I_{n-1}, I_n$) pairs based on the above process.

#### 4.1. Comparisons of ME accuracy

Now for each of the eight tested pictures, we use the prepared 5000 ($I_{n-1}, I_n$) pairs to test the three methods, the proposed ESPM-1D-BMA and the 2D-FSBMA, the average accuracy of the 5000 estimated MVs for each picture and for various combinations of $K$ and $P$ are presented in Table 2. From Table 2, we can observe that the larger the values of $K$ and $P$ in the proposed ESPM-1D-BMA, the more accurate the MV estimation will be. We can also observe that when $K \geq 8$ and $P \geq 4$, the proposed algorithm is at most 1% less accurate than the 2D-FSBMA on the average.

Putting the test results of ESPM-1D-BMA for $K = 8$ and $P = 3$ presented in Table 2 in the second row of Table 3 for reference, we also use the same 5000 prepared ($I_{n-1}, I_n$) pairs to test the three fast block matching algorithms, the DS, the NTSS, the HEXBS. The average ME accuracy resulted by these three methods are presented in Table 3. From Table 3, we can observe that the proposed ESPM-1D-BMA is far better than the three fast block matching algorithms in ME accuracy.

#### 4.2. Comparison of computational efficiency

Now, to evaluate the computing time of ESPM-1D-BMA and the mixed-signal approach 2D-FSBMA (MS-2DFSBMA), we need to review the computational complexity of the latter first. Similar to the

Inference Engine(
{
  \% - initialization - \\
  D_{MSE} = 0, d = 1, ..., KP; AMN_{d} = 0, d = 1, ..., KP; \\
  wMV = 0; wAMN = 0; D = 1; \\
  for (k = 1 to K) \\
  { \\
    for (p = 1 to P) \\
    { \\
      d' = D; \\
      smv = m_{sv}; \\
      for (d = 1 to KP) \\
      { \\
        if (D_{MSE} = smv) d' = d; \\
        if (d' = D) new = 1; \\
        else new = 0; \\
      } \\
      \% - step 2 - \\
      AMN_{old} = AMN_{d} + AMN_{new} + MN_{d}; \\
      \% - step 3 - \\
      if (AMN_{new} > wAMN) win = 1; \\
      else win = 0; \\
      \% - step 4 - \\
      D_{MSE} = smv; \\
      AMN_{d} = AMN_{new}; \\
      if (win == 1) \\
      { \\
        wMV = smv; \\
        wAMN = AMN_{new}; \\
        if (new == 1) D = D + 1; \\
      } \\
    } \\
  }
Fig. 7. The pseudo-code of IE.

1 × $B$ current summation circuit employed in 1D block matching presented in Fig. 3, the MS-2D-FSBMA employed a $X \times Y$ current-summation circuit to obtain the 2D-MSE (Panovic & Demosthenous, 2006). Instead of PWC$_k$, $k = 1, \ldots, K$, the MS-2D-FSBMA need only one single winner comparator (SWC), which consists of a comparator, COMP, a SH, and a DM to identify the coordinate of the SB with minimum MSE. In the case of a frame with size $M \times N$, a block with size $X \times Y$, the computing time of the MS-2D-FSBMA for computing an MV can be calculated by:

$$
2 \times X \times Y \times T_{SH} + (M - X)(N - Y + 1)(Y \times T_{SH} + T_{MSE2D} + T_{COMP}) + (N - Y)(X \times T_{SH} + T_{MSE2D} + T_{COMP})
$$

where $T_{SH}$ is the same as that in (4); $T_{MSE2D}$ denotes the time for performing a 2D-MSE computation; since the most time consuming

#### Table 3
Comparisons of ESPM-1D-BMA with the three fast block matching algorithms.

<table>
<thead>
<tr>
<th>Football</th>
<th>Greens</th>
<th>Concord</th>
<th>Fabric</th>
<th>Hestain</th>
<th>Pears</th>
<th>Tissue</th>
<th>Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPM-1D-BMA ($K = 8, P = 3$)</td>
<td>99.58</td>
<td>99.54</td>
<td>97.20</td>
<td>99.57</td>
<td>98.90</td>
<td>99.28</td>
<td>98.52</td>
</tr>
<tr>
<td>DS</td>
<td>45.93</td>
<td>69.04</td>
<td>81.79</td>
<td>82.88</td>
<td>89.04</td>
<td>75.54</td>
<td>79.24</td>
</tr>
<tr>
<td>NTSS</td>
<td>48.69</td>
<td>81.13</td>
<td>86.87</td>
<td>86.22</td>
<td>87.89</td>
<td>75.01</td>
<td>87.69</td>
</tr>
<tr>
<td>HEBXBS</td>
<td>38.15</td>
<td>63.02</td>
<td>73.89</td>
<td>67.10</td>
<td>74.53</td>
<td>62.13</td>
<td>74.69</td>
</tr>
</tbody>
</table>
component in SWC is the COMP, the time needed for comparison in SWC is \( T_{COMP} \).

Based on the existing circuit for SH (Chen, Gu, Shen, Wu, & Hsu, 1998), MSE2D (Panovic & Demothenous, 2004) and COMP (Razavi & Wooley, 1992), we obtain the following computing time using hspice simulation: \( T_{SH} = 50 \) ns, \( T_{MSE2D} = 10 \) ns, and \( T_{COMP} = 100 \) ns. Although the analog circuit for computing MSE presented in Fig. 3 is simpler than computing MSE2D in MS-2D-FSBMA, the operations of all SEs in both circuits are carried out in parallel. Therefore, \( T_{MSE} \approx T_{MSE2D} = 10 \) ns. Similarly, the operations of PWCs (Fig. 5), \( k = 1, \ldots, K \) are also carried out in parallel, and the most time consuming component in PWCs is COMPk, therefore, \( T_{PWC} \approx T_{SWC} = T_{COMP} = 100 \) ns. According to the hardware implementation architecture of the IE presented in Appendix A, the processing time for IE is \( T_{IE} = K \times P \times T_{clock} \), where \( T_{clock} = 13.1 \) ns is the critical path delay of IE. Therefore, for \( K = 8, P = 3 \), we have \( T_{IE} = 314.4 \) ns. Consequently, based on the parameters employed in our tests, the computing time needed for an MV estimation of ESPM-1D-BMA with \( K = 8, P = 3 \) and MS-2D-FSBMA are \( 92474.4 \) ns and \( 153,280 \) ns, respectively. This demonstrates that the proposed ESPM1D-BMA uses only 60% of computing time of the MS-2D-FSBMA and achieves almost same ME accuracy. Notably, the critical point of the computing efficiency achieved by the ESPM-1D-BMA is the small amount of time spending on loading the pixel values of the frame \( I_{k-1} \) into the SBM. The part of loading time in (4) and (5) for ESPM-1D-BMA and MS-2D-FSBMA are \( M \times N \times T_{SBM} \) and \( 2 \times X \times Y \times T_{SBM} + (M - X)(N - Y + 1)(Y \times T_{SBM}) + (N - Y)(X \times T_{SBM}) \), respectively, which constitutes 31.14% and 79.33% of the corresponding total computing time, respectively.

5. Conclusion

In this paper, we have proposed a hardware implementable ESPM-1D-BMA and demonstrated that its ME accuracy is close to the 2D-FSBMA and better than the three comparing fast block matching algorithms. Above all, the computing speed of ESPM-1D-BMA is about two times as fast as the MS-2D-FSBMA.

Appendix A

To implement the IE, we will first describe the pseudo code for carrying out the IE then present the hardware architecture for implementing the pseudo code. We let \( m_{vk} \) and \( MN_{k,p} \), \( p = 1, \ldots, P \) denote the \( P \) MVs that correspond to the top \( P \) smallest MSEs for the \( k \)th 1D block matching and the corresponding MN, respectively; we let \( D \) denote the number of distinct MVs among the \( K \times P \) \( m_{vk} \)’s and let \( D_{mvk} \), \( d = 1, \ldots, D \) denote the \( D \) distinct MVs; we let \( AMN \) denote the AMN of \( D_{mvk} \) and let \( wMV \) and \( wAMN \) denote the resulting best-so-far MV and the corresponding AMN during the process, respectively. Based on the above notations, the pseudo code for carrying out the IE is presented in Fig. 7.

The operations of the pseudo code can be summarized in the following. In the initialization step, we reset all the variables. In step 1, we check whether the incoming MV, \( m_{vk,p} \) matches any one of the stored distinct MVs, \( D_{mvk} \). If it matches, we identify the matched distinct MV and output a flag \( new = 0 \); otherwise, we denote the incoming MV as a new distinct MV and set \( new = 1 \). In step 2, we compute the AMN of the identified distinct MV. In step 3, we check whether \( AMN_{new} \) of the distinct MV identified previously greater than the \( wAMN \) of \( wMV \) and output a flag \( win = 1 \) if the result is positive; otherwise we set \( win = 0 \). In step 4, we update the distinct MV and the associated AMN and update \( wMV \) and \( wAMN \) if \( win = 1 \); furthermore, if the flag \( new \) resulted in step 1 is \( 1 \), we set \( D = D + 1 \) that is to increase the number of distinct MVs by 1.

The hardware implementation architecture of the pseudo code is presented in Fig. 8. In the leftest part of this architecture, we use a counter, \( counter_{kp} \) marked by (ii) in Fig. 8, to generate the index \( p = 1, \ldots, P \) for each \( k = 1, \ldots, K \) sequentially and use a KP-to-1 Multiplexer, denoted by MUX and marked by (i) in Fig. 8, to select \( m_{vk,p} \) based on the generated index; this corresponds to the statement \( smv = m_{vk,p} \) in step 1 of Fig. 7. The \( smv \) will be input to the MV identifier marked by (iii) in Fig. 8, which will compare the \( smv \) with the distinct MVs stored in the registers denoted by Distinct MV and AMN. The counter, counter_D marked by (iv) in Fig. 8, will generate the index \( D \), such that if \( smv \) does not match any existing distinct MVs, this \( smv \) will be the \( D \)th distinct MV, and the MV identifier will set the enable signal \( en_{D} = 1 \) to activate the Distinct MV and AMN registers, marked by (v) in Fig. 8, to store the current \( smv \) and the corresponding AMN. In the meantime, the MV identifier also sets \( new = 1 \) to increase counter_D by 1. We will then use an adder marked by (vi) in Fig. 8 to update the AMN register for the current \( smv \), which may be one of the existing distinct MVs or a new distinct MV, as follows. Add the \( MN_{k,p} \) corresponding to the current \( smv \) to the \( AMN_{old} \) then output \( AMN_{new} \), which will be fed back to the
Distinct MV and AMN registers. Notably, the $MN_k$ is generated by the counter_MN marked by (vii) in Fig. 8. Furthermore, the AMN_new will compare with wAMN through the comparator, $>$, marked by (viii) in Fig. 8. If AMN_new > wAMN, we set $w_{AMN} = 1$, which will update wAMN by AMN_new and wMV by $smv$ as shown by the blocks marked by (x) and (ix), respectively. This completes the implementation of the pseudo code presented in Fig. 7.

We design the clock period to be long enough such that the signal $m_{vkp}$ can travel through the critical path, which includes the propagation of counter_kp, KP-to-1 MUX, and MV identifier, the access of the Distinct MV and AMN registers, the processing time of adder and comparator, and the update of wMV and wAMN. Based on Taiwan Semiconductor Manufacturing Corporation (TSMC) 0.18 μm CMOS technology, the critical path's delay of the IE presented in Fig. 8 can be within 13.1 ns. That means we can design the clock period for IE as $T_{clock} = 13.1$ ns. Then, the time required for processing the IE is $T_{IE} = K \times P \times T_{clock}$. Therefore, for $K = 8$, $P = 3$, the IE can obtain the true MV within 314.4 ns.

References


