# An All Binary Sub-Pixel Motion Estimation Approach and its Hardware Architecture

Anıl Çelebi, *Student Member*, IEEE, Orhan Akbulut, Oğuzhan Urhan, *Member*, IEEE, İlker Hamzaoğlu, *Member*, IEEE, and Sarp Ertürk, *Member*, IEEE

**Abstract** — *Motion estimation (ME) is the most computationally intensive part of a video coding system. Therefore it is very important to reduce its computational complexity. In this paper, a novel all-binary approach for reducing the computational complexity of sub-pixel accurate ME is proposed. An efficient hardware architecture for the proposed all-binary sub-pixel accurate motion estimation approach is also presented. The proposed hardware architecture has significantly low hardware complexity and therefore very low power consumption. It can process 720p video frames at 30 fps in a pipelined fashion together with the integer ME hardware. Therefore, it can be used in real-time low power video coding systems required by many mobile consumer electronics devices*[1].

**Index Terms — Video Coding, One-bit transform, Sub-pixel motion estimation, Source pixel based linear arrays.**

## I. INTRODUCTION

Video compression is used in many consumer electronics products to reduce the data rate for efficient storage or transmission. Video compression standards, such as MPEG-2 [1], H.263 [2] and H.264/AVC [3], make use of temporal and/or spatial prediction and transform coding for effective compression. Motion estimation is one of the most important parts of these compression standards because it enables high compression rates obtained by removing temporal redundancies in the video. The computational load of motion estimation is about 50-70% of the computational load of the entire video coding system. This is a particularly important aspect in real-time applications with power consumption and processing capacity constraints which is the case for many mobile consumer electronics devices.

Block-based motion estimation with Sum of Absolute Differences (SAD) matching of pixels is the most common approach used for motion estimation in video encoders. The one-bit transform (1BT) has been proposed in [4] to reduce the computational complexity of the matching process in motion estimation by transforming video frames into 1 bit/pixel representations and performing motion estimation using these binary representations. In [4], video frames are filtered using a multi-bandpass filter and the corresponding pixels of the filtered frames are used as thresholds to construct the binary representations used for motion estimation. In [5], a new multi-bandpass filter kernel is proposed for 1BT to facilitate a multiplication free transform for reduced transform complexity. A two-bit transform (2BT) is proposed to improve motion estimation accuracy compared to 1BT by constructing two bit-planes for each video frame and performing motion estimation using the 2BT representations in [6]. In [7] constrained 1BT (C-1BT) is proposed and it is shown that this approach can provide higher motion estimation accuracy compared to 2BT at lower complexity. It is possible to further speed up the low bit representation based ME approaches by combining them with fast search techniques as in [8] at the cost of a tolerable accuracy loss. In [9] an early termination scheme for binary motion estimation is presented. All of the low bit representation based ME schemes mentioned above simply utilize the hardware efficient exclusive-or (EX-OR) matching of bit-planes.

One common way to further improve the ME performance of the low bit representation based ME methods is to utilize full pixel resolution matching in a restricted search area which is centered at the binary search result, as presented in [10-13]. However these improvements destroy the all binary nature of the approach and also cause an increase in processing load as well as complexity and power consumption of the hardware implementation. Another way to improve the ME accuracy is to move to the sub pixel domain. Although there are many approaches proposed in the literature for sub-pixel accurate motion estimation using full bit-depth frames, such as [14,15], there is only limited research on sub-pixel accurate motion estimation using lower bit-depth representations. In [16] it is proposed to interpolate video frames at 8 bits/pixel bit-depth and then carry out the 1BT using up-sampled data, to facilitate 1BT based motion estimation at sub-pixel accuracy. However, the method proposed in [16] does not provide an all-binary sub-pixel ME scheme because the interpolation used to obtain sub-pixel data is performed at full bit-depth. Furthermore the multi-bandpass filtering complexity also increases in this case to accomplish the 1BT of up-sampled data.

There are many hardware implementations in the literature for algorithms using 8-bit/pixel based motion estimation on video frames and some examples can be found

[1] This work was supported by the Scientific and Technical Research Council of Turkey (TUBİTAK) and Korean Research Foundation (KRF) Cooperation Program under TUBITAK contract no. 107E179.

A. Çelebi, O. Akbulut, O. Urhan and S. Ertürk are with Kocaeli University Laboratory of Image and Signal processing (KULIS), Department of Electronics and Telecommunication Engineering, University of Kocaeli, 41040, Kocaeli, Turkey (e-mail: anilcelebi@kou.edu.tr, orhan.akbulut@gmail.com, urhano@ieee.org, sarp@ieee.org).

I. Hamzaoğlu is with the Faculty of Engineering and Natural Sciences, Sabanci University, 34956, Istanbul, Turkey (hamzaoglu@sabanciuniv.edu).

Contributed Paper
Manuscript received September 25, 2008

in [17-23]. In terms of the hardware architecture type, two approaches namely, systolic arrays and semi-systolic arrays are commonly used in the literature [17-19], because systolic arrays are very well suited for the implementation of regular motion estimation algorithms such as full search block matching. In [20], several VLSI architectures for 8-bit/pixel video frames are proposed for various motion estimation schemes. A systolic array based architecture with a reduced number of input and output pins is proposed in [21]. In [22], an FPGA implementation of the SAD based algorithm is presented. In [23], an early termination strategy to reduce the computational load is implemented.

Fast search algorithms are frequently adopted in software implementations to speed up the motion estimation process by reducing the search space at the cost of a suboptimal solution. However, the data dependencies in fast search approaches hinder parallel hardware designs and in most hardware implementations a full search strategy is chosen for reasons of regularity, lack of data dependencies and optimality [24].

The first hardware implementation of the 1BT based ME in the literature is presented in [4]. This hardware is similar to the hardware proposed in [18] which uses SAD matching of pixels that have a bit-depth of 8 bits. The main difference between these two designs is that in [4] the input data is an $M$ bit wide row vector instead of a single pixel. Therefore, the required memory size is $M$ times smaller in [4] in comparison to [18]. Since an entire row of the reference block is read from memory in one cycle instead of a single pixel, the 1BT based ME hardware proposed in [4] needs only $M^2$ cycles for computing the MV of an $M \times M$ block. Therefore, it is $M$ times faster than the ME hardware proposed in [18]. In addition, a processing element (PE) used in 8-bits/pixel based ME hardware has 24 full adders for addition and accumulation whereas a PE used in 1BT based ME hardware has only 15 full adders. Therefore, 1BT based ME hardware has also lower area than 8-bits/pixel based ME hardware.

An all binary ME approach using a hierarchical layer structure in the form of a binary pyramid has been proposed in [25] but this approach has a high complexity compared to simple 1BT. The hardware architectures presented in [4] and [25] are only presented for integer ME and they cannot be scaled directly for sub-pixel ME.

This paper first presents a new approach for sub-pixel accurate ME which is accomplished directly using 1-bit/pixel resolution video frames. In the proposed approach, the 1BT is applied to the original size video frames and the interpolation required for sub-pixel ME is performed in the 1BT domain, significantly reducing the computational complexity in comparison to [16] in which the interpolation is performed in the full bit-depth representation after which the 1BT is applied to up-sampled data. The proposed method is therefore particularly suitable for consumer electronics applications that require real time video encoders.

This paper then presents a novel low complexity and therefore low power hardware implementation of the proposed method for sub-pixel accurate ME. Some modifications to conventional sub-pixel approaches are performed to increase the efficiency of the hardware especially in the interpolation step. Because of the low resolution nature of the method, some of the arithmetic operations are transformed to simple logic operations and this significantly simplified the hardware architecture. For example, in the 8 bits/pixel bit-depth half-pixel interpolation hardware presented in [26] and [27], five additions and one subtraction at full bit-depth are performed for half-pixel interpolation. However, in the proposed binary half-pixel interpolation hardware only a read operation from a small look up table is performed for half-pixel interpolation.

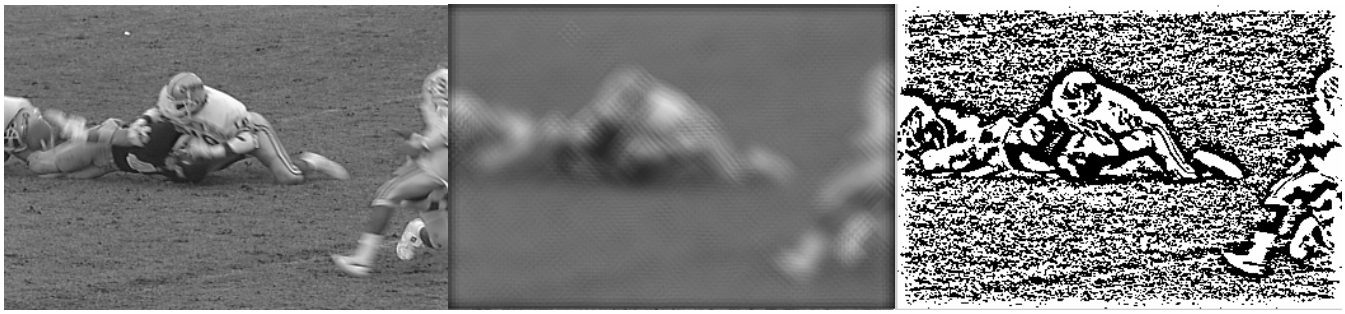
In addition, the 8-bits/pixel half-pixel ME hardware proposed in [28] occupies 3197 LUTs in a XCV3200E-7 FPGA for a block size of 16×16 pixels. When the proposed sub-pixel (half-pixel and quarter-pixel) ME hardware is synthesized to the same FPGA using the same synthesis tool (Xilinx XST), it only occupies 1734 LUTs for a block size of 16×16 pixels.

## II. MF1BT Based Block Motion Estimation

Standard video compression schemes typically use block matching based motion estimation algorithms. In block matching, the reference block, i.e. the current block, in the current image frame of time $t$ is searched for in a search window in typically the previous image frame of time $t$-1. For blocks encoded through temporal prediction, the motion vector for every current block is obtained and the current block is encoded with the motion vector and the residual data, remaining after motion compensation, is compressed via transform coding. The block size and search window size determine the computational complexity.

1BT and 2BT approaches present in the literature for low-complexity block matching are very suitable for hardware implementations, because the basic EX-OR structure is used in these approaches to perform matching on image frames with reduced bit-depth. In [4], a multi-band pass filter that has 25 non-zero elements is utilized to obtain filtered images. The filtered images are compared to the original image frames to create the one-bit images. In this case, non-integer operations are required for the normalization stage of the filtering which has comparatively higher computational complexity. In [5] a novel diamond shaped kernel, given in (1), is proposed to decrease the computational complexity of the filtering stage of 1BT. This new kernel contains a power of 2 non-zero elements and thus the multiplication operation becomes simple logical shift. For filtering, sixteen additions and one four-bit shift operation is performed. Therefore, this method is called multiplication free one-bit transform (MF1BT).

In MF1BT the standard bit-plane is obtained as in conventional 1BT, shown in (2). The number of non-matching point (NNMP) criterion given in [4] is then used to evaluate the match of two blocks as shown in (3).



(a) (b) (c)

**Fig. 1. (a) A sample frame of "Football" sequence, (b) Filtered version, (c) One-bit transform.**

$$K=\frac{1}{16}\begin{bmatrix}
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0
\end{bmatrix} \quad (1)$$

$$B(i,j)=\begin{cases}1, & \text{if } I(i,j)\geq I_F(i,j)\\ 0, & \text{otherwise}\end{cases} \quad (2)$$

$$NNMP(m,n)=\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}B^t(i,j)\oplus B^{t-1}(i+m,j+n) \quad (3)$$

where $\oplus$ denotes the binary EXOR operation.

The search location for which the smallest *NNMP* value is obtained is considered as the motion vector of the related current block. A sample frame of the Football sequence, its filtered version, and the 1BT result are shown in Fig. 1.

## III. Sub-Pixel Motion Estimation Based on MF1BT

Sub-pixel ME is an important feature of recent video coding standards such as H.264/AVC. Sub-pixel, i.e. half-pixel and quarter-pixel, motion estimation is used to improve integer motion estimation accuracy. The first step of sub-pixel motion estimation process is the interpolation operation. This operation is performed on the integer reference block for half-pixel motion estimation. Each half-pixel sample is interpolated from six neighboring integer pixels using a 6 taps FIR filter. A half-pixel interpolation example is shown in Fig. 2. For example, the half-pixel value **b** shown in Fig. 2 is calculated as

$$\mathbf{b}=round\left((E-5F+20G+20H-5I+J)/32\right) \quad (4)$$

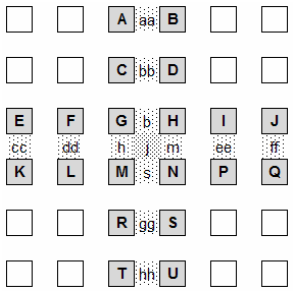



**Fig. 2. Half-pixel interpolation example.**

In the proposed MF1BT based sub-pixel motion estimation, the interpolation process is same as (4). However, since 1BT representations are used instead of full bit-depth frames, any pixel in the interpolation can only take a value of 0 or 1. Therefore, this interpolation operation is simpler than the 8 bits/pixel interpolation operation.

The quarter-pixel interpolation scenarios for three possible quarter pixel positions are illustrated in Fig. 3. As it can be seen in the figure, some half pixels and some integer pixels are required in horizontal and vertical directions, but only half pixels are required in the diagonal direction in the interpolation scheme used in the H.264/AVC standard. The quarter-pixel interpolation operation is a simple linear interpolation as shown in (5) for quarter pixel **a** .

$$\mathbf{a}=round\left((G+\mathbf{b})/2\right) \quad (5)$$

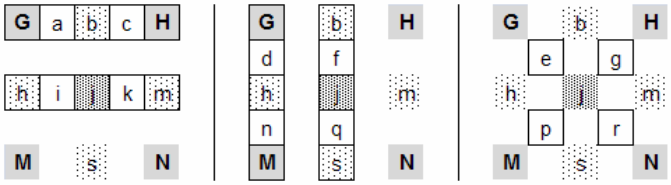



**Fig. 3. Quarter-pixel interpolation schemes a) In horizontal direction, b) In vertical direction, c) In diagonal direction.**

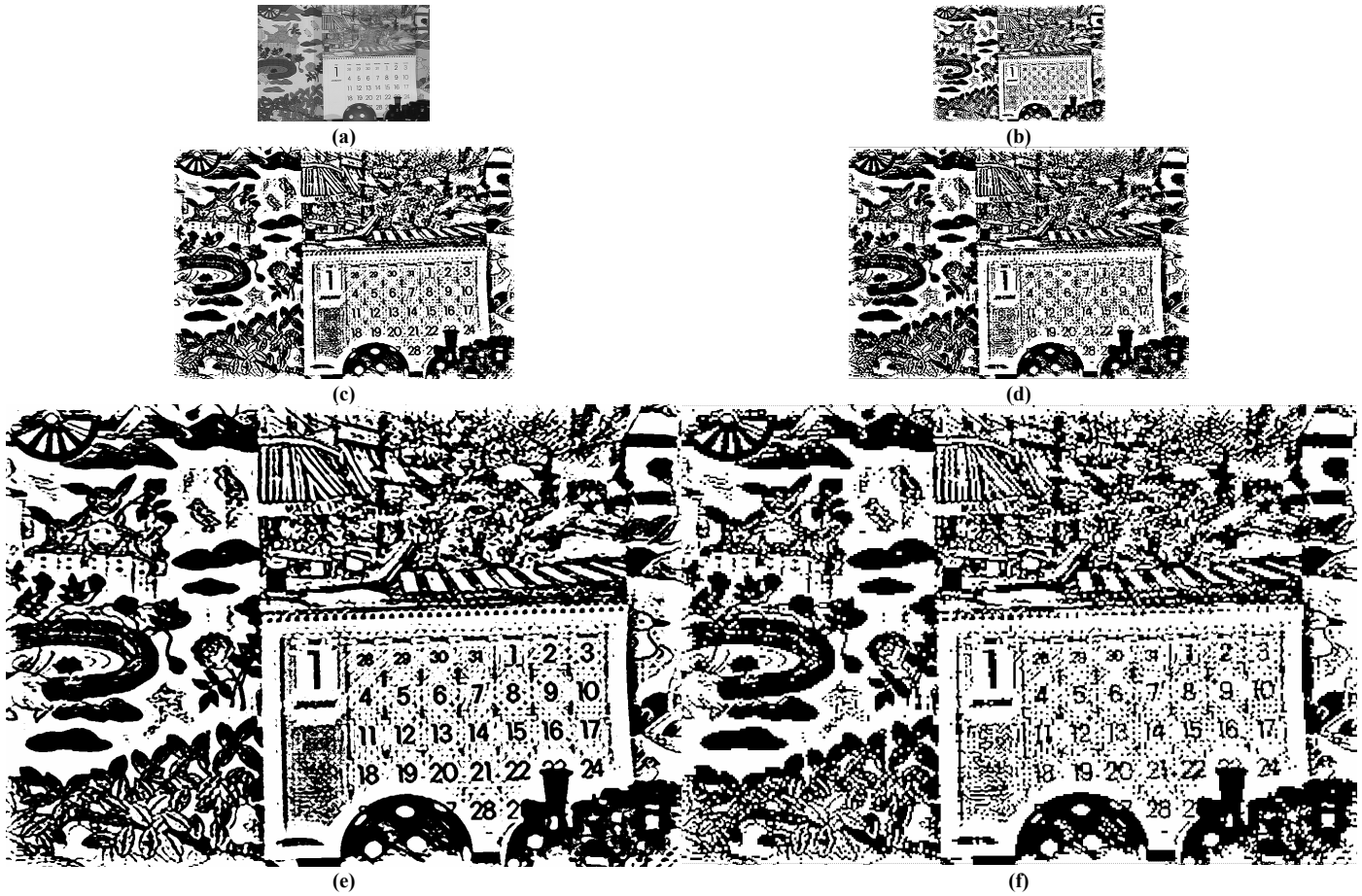



**Fig. 4. (a) Sample frame of the Mobile Sequence (b) 1BT result of the frame c) Half-pixel interpolation result of the method proposed in [16] d) Half-pixel interpolation result of proposed approach e) Quarter-pixel interpolation result of the method proposed in [16] f) Quarter pixel interpolation result of the proposed approach.**

In MF1BT based sub-pixel motion estimation method this arithmetic operation is transformed to simple logical OR operation, therefore the binary quarter-pixel interpolation equation for the same pixel is as shown in (6)

$$\mathbf{a} = G \,||\, \mathbf{b} \tag{6}$$

where || denotes logical OR operation. Therefore, a simple hardware architecture can be used for implementing binary quarter-pixel interpolation.

The 1BT results for the case where 1BT is performed after interpolation at 8 bits/pixel resolution as in [16], and for the proposed scheme where the interpolation is performed at 1 bit/pixel resolution after the 1BT are illustrated in Fig. 4. Fig. 4(a) shows a sample video frame of the Mobile sequence, and Fig. 4(b) shows the 1BT of this frame. Fig. 4(c) and Fig. 4(d) show the half-pixel frames for the approach presented in [16] and the proposed approach, respectively. Fig. 4(e) and Fig. 4(f) show the quarter-pixel frames for the approach presented in [16] and the proposed approach, respectively.

## IV. Hardware Architectures for MF1BT based ME with Sub-Pixel Accuracy

In this section, a novel hardware architecture for MF1BT based sub-pixel motion estimation is proposed. For illustration purposes, the hardware implementation of MF1BT based integer motion estimation is also presented, initially.

### *A. MF1BT Based Integer ME Hardware Architecture*

The MF1BT based integer ME hardware is based on the source pixel based linear array (SPBLA) architecture shown in Fig. 5. In Fig. 5, the *s1* and *s2* ports are combined into a single bus for simplicity. Since this architecture is quite similar to [29] and [30], it is only briefly explained in this section.

The hardware architecture in Fig. 5 is first proposed in [29] and its 1BT based implementation with a new data flow scheme that reduces the power consumption by about 50% is presented in [30].

The processing element (PE) architecture used in this ME hardware is shown in Fig. 6. The data flow scheme of this ME hardware is shown in Table 1. This MF1BT based integer ME hardware can compute integer motion vectors in 1039 clock cycles for a search range of [-16,15].

**TABLE I.**
**DATA-FLOW SCHEME OF THE MF1BT BASED ME HARDWARE USING SPBLA ARCHITECTURE.**

| Cycle Number | Input Data | | | Inputs of PEs | | | | | LOCATION OF BLOCK DISTORTION THAT IS CALCULATED |
|---|---|---|---|---|---|---|---|---|---|
| | *R* | *S1* | *S2* | PE-0 | PE-1 | . . . | PE-14 | PE-15 | |
| 0 | R0 | S0,0 | | R0-S0,0 | | | | | |
| 1 | R1 | S1,0 | | [R0]-S1,0 | R1-S1,0 | | | | |
| 2 | R2 | S2,0 | | [R0]-S2,0 | [R1]-S2,0 | | | | |
| . | . | . | | . | . | | | | |
| . | . | . | | . | . | | | | |
| . | . | . | | . | . | | | | . |
| . | . | . | | . | . | ... | | | . |
| 14 | R14 | S14,0 | | [R0]-S14,0 | [R1]-S14,0 | ... | R14-S14,0 | | . |
| 15*** | R15 | S15,0 | | [R0]-S15,0 | [R1]-S15,0 | ... | [R14]-S15,0 | R15-S15,0 | D(0,0) |
| 16 | R0 | S16,0 | | [R0]-S16,0 | [R1]-S16,0 | ... | [R14]-S16,0 | [R15]-S16,0 | D(1,0) |
| . | . | . | | . | . | . | . | . | . |
| . | . | . | | . | . | . | . | . | . |
| . | . | . | | . | . | . | . | . | . |
| 31 | R15 | S31,0 | | [R0]-S31,0 | [R1]-S31,0 | ... | [R14]-S31,0 | [R15]-S31-0 | D(16,0) |
| 32 | R0 | S32,0 | S0,1 | [R0]-S0,1 | [R1]-S32,0 | ... | [R14],S32,0 | [R15]-S32,0 | D(17,0) |
| . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . |
| 46 | R14 | S46,0 | S14,1 | [R0]-S14,0 | [R1]-S14,0 | ... | [R14]-S14,1 | [R15]-S46,0 | D(31,0) |
| 47 | R15 | | S15,1 | [R0]-S15,1 | [R1]-S15,1 | ... | [R14]-S15,1 | [R15]-S15,1 | D(0,1) |
| ***All of the PEs are functional | | | | | | | | | |

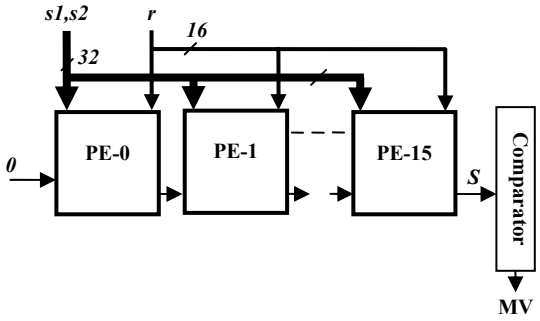



**Fig. 5. MF1BT based ME hardware using SPBLA Architecture.**

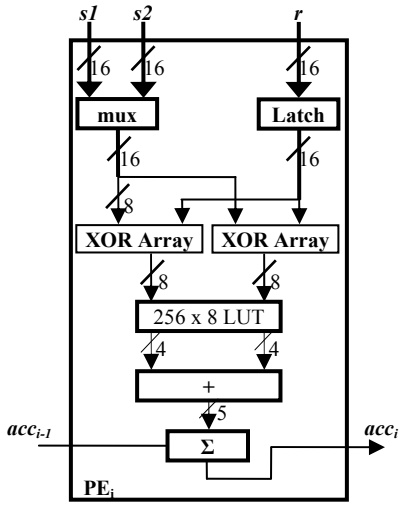



**Fig. 6. PE architecture of the MF1BT based ME hardware.**

## *B. MF1BT Based Sub-pixel ME Hardware Architecture*

The proposed MF1BT based sub-pixel motion estimation hardware architecture is shown in Fig. 7. The PE Array and the comparator are shared by half-pixel ME and quarter-pixel ME. They are first used by half-pixel ME and then by quarter-pixel ME. This reduces the hardware complexity.

Half-pixel interpolation and half-pixel search operations are performed in a pipelined fashion. Quarter-pixel operations start after half-pixel interpolation and search finish, because quarter-pixel search is performed around the location obtained by the half-pixel search.

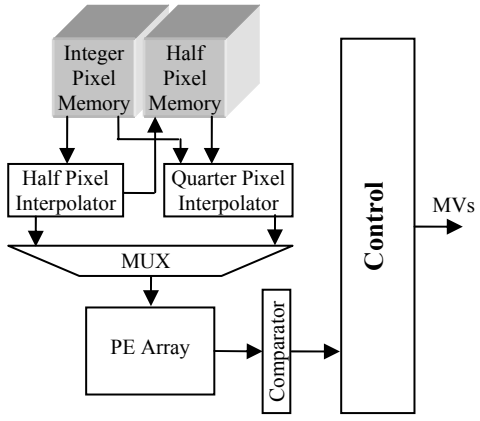



**Fig. 7. Proposed MF1BT sub-pixel motion estimation hardware architecture.**

The proposed sub-pixel ME hardware includes a 22×22 integer pixel search window memory and a 16×16 reference window memory which is not shown in the figure for simplicity. The half-pixel memory block shown in the figure includes an 18×17 memory for storing *A* type half pixels, a 17×18 memory for storing *B* type half pixels and a 17×17 memory for storing *C* type half pixels.

The first step of half-pixel ME is to obtain the interpolated image frames. An example half-pixel interpolation window for a 4x4 integer pixel block is shown in Fig. 8. The interpolation window contains *A, B* and *C* type pixels. This representation is used to simplify the hardware design. In this figure, the interpolation is performed for the 4x4 integer pixel block located inside the 7x7 window with dark lines, i.e. for the 4x4 block with integer pixels having horizontal and vertical indices between 3 and 6. In Fig. 8, the window with dashed lines is the half-pixel ME search window.

It is required to use 10 integer pixels in order to calculate 5 *A* type half-pixels in a row. In case of binary processing, the values of the 5 *A* type half-pixels in a row are represented using 5 bits, and the values of the 10 integer pixels required for computing these half-pixels are represented using 10 bits.

The proposed half-pixel interpolation hardware for a 4×4 integer pixel block is shown in Fig. 9. 5 LUTs are required for performing half-pixel interpolation for a 4×4 integer pixel block. In order to calculate all 5 *A* type half-pixels in a row during interpolation, the corresponding 10 integer pixels are read from memory in the same clock cycle and sent to five LUTs. Each LUT has 6 inputs and 1 output, and calculates one half-pixel by performing half-pixel interpolation. When the ME is performed for 16x16 integer pixel blocks, the interpolation window contains 22×22 integer pixels and therefore 17 LUTs are required for performing half-pixel interpolation.

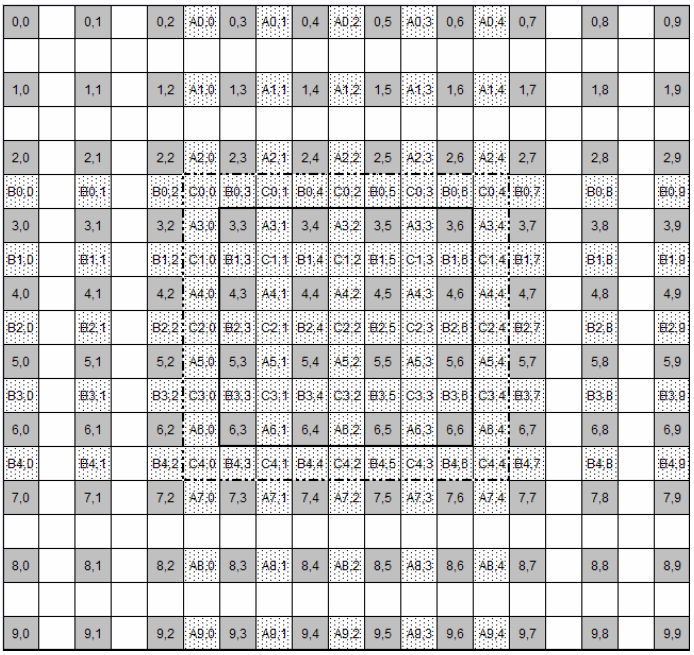



**Fig. 8. Half-pixel interpolation window for a 4×4 integer pixel block.**

Therefore, in the proposed binary half-pixel interpolation hardware, in order to calculate the 17 *A* type half-pixels in a row, 22 integer pixels are read from memory in the same clock cycle and sent to 17 LUTs. Each LUT calculates one half-pixel by performing half-pixel interpolation.

Since integer pixels are read from memory row by row, in order to calculate *B* type half-pixels, the integer pixels are shifted serially through shift registers and the pixels are sent to the interpolation LUTs in parallel from these shift registers. Each integer pixel in the same row is sent to a separate shift register and the integer pixels in the same column are sent to the same shift register. Therefore, 16 6-bit shift registers are required for a block size of 16×16 pixels. From each shift register, 6 bits are sent to the interpolation LUTs. Therefore, 16*6=96 bits are sent to *B* type half-pixel interpolation array.

*C* type half-pixels are calculated from *A* type half-pixels instead of integer pixels, for simplicity. Since A type half-pixels are obtained before B type half-pixels, *C* type half-pixels are calculated using *A* type half-pixels instead of B type half-pixels.

Serial to parallel conversion of A type half-pixels is done using shift registers in order to calculate C type half-pixels. For this purpose, *A* type half-pixels are fed into a shift register array and the outputs of these shift registers are sent to C type half-pixel interpolation array.

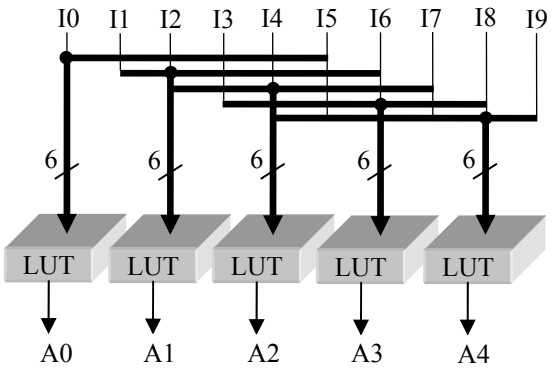



**Fig. 9. Proposed half-pixel interpolation hardware for a 4x4 integer pixel block.**

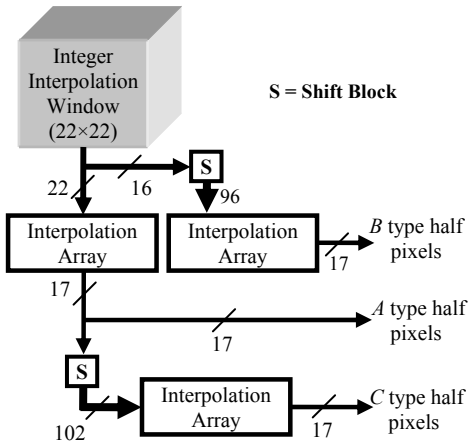



**Fig. 10. Proposed half-pixel interpolation hardware architecture.**

The proposed half-pixel interpolation hardware architecture is shown in Fig. 10. The half-pixel interpolation hardware first calculates the *A* type half-pixel. Because of the shift operations, *B* and *C* type half-pixels are calculated with latency. The whole half-pixel interpolation operation takes 22 clock cycles which is defined by the size of the integer interpolation window memory depth.

Since there are 8 different search locations for half-pixel search, an array of 8 PEs is used for a fully parallel search process. The architecture of the PE used in the MF1BT based sub-pixel ME hardware is shown in Fig. 11. The search operation starts immediately after the first half-pixel vector is available on the output ports of the half-pixel interpolation hardware. Thus, the NNMP values of the last half-pixel vector are calculated with the rising edge of 24$^{th}$ cycle. The half-pixel comparison operation starts with the rising edge of 21$^{st}$ clock cycle where as the NNMP values of first search location is computed. Half-pixel motion vectors are written to the output ports on the rising edge of the 27$^{th}$ clock cycle. Quarter pixel interpolation operation is completed on the rising edge of the 47$^{th}$ clock cycle and finally quarter pixel motion vectors are written to the output ports of the comparator on the rising edge of the 49$^{th}$ clock cycle.

For the MF1BT based quarter-pixel ME, there are 9 possible quarter-pixel interpolation locations depending on the half-pixel ME result. An example quarter-pixel interpolation window for a 2×2 integer pixel block is shown in Fig. 12. The pixels that are used in quarter-pixel interpolation are shown in Fig. 12. Both integer pixels and half-pixels are required for quarter-pixel interpolation. In Fig. 12, the dark outlined window includes all 9 possible half-pixel search locations around the integer pixel i11. The window with dashed lines includes all 9 possible quarter-pixel search locations for all 9 half-pixel search locations. Therefore, it includes a total of 81 quarter-pixel search locations.

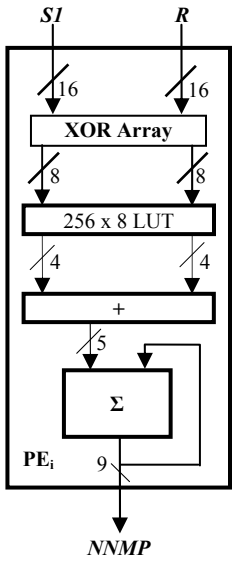



**Fig. 11. The architecture of PE used for sub-pixel search operation.**

The quarter-pixel interpolation strategy is illustrated in Table II. Because of the binary nature of MF1BT, quarter-pixel interpolation is performed simply by a two input OR operation. As seen in the table, the quarter pixels for 8 different quarter-pixel search locations (SL0-SL7) should be calculated depending on the half-pixel ME result. The quarter-pixel interpolation operands are different for 9 possible outcomes of the half-pixel ME. For example, if the half-pixel ME result is (-1, -1), the quarter-pixel interpolation operands for the quarter-pixel search location SL3 are a20 and c10.

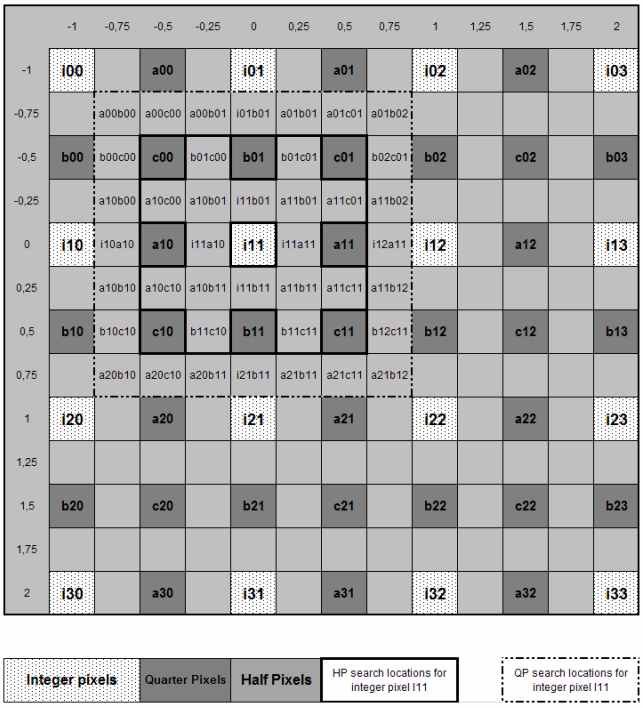



**Fig. 12. Quarter-pixel interpolation window for a 2×2 integer pixel block.**

**TABLE II.**
**QUARTER-PIXEL INTERPOLATION STRATEGY**

| *Calculation combinations of quarter pixels according to the half-pel search results* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| QP search locations | Half-pel motion vector locations for integer pixel i11(x,y) | | | | | | | | |
| | (-1,-1) | (0,-1) | (1,-1) | (-1,0) | (0,0) | (1,0) | (-1,1) | (0,1) | (1,1) |
| SL0 | b10,c10 | b11,c10 | b11,c11 | i10,a10 | i11,a10 | i11,a11 | b00,c00 | b01,c00 | b01,c01 |
| SL1 | b11,c10 | b11,c11 | b12,c11 | i11,a10 | i11,a11 | i12,a11 | b01,c00 | b01,c01 | b02,c01 |
| SL2 | a10,c10 | i11,b11 | a11,c11 | a10,c00 | i11,b01 | a11,c01 | a00,c00 | i01,b01 | a01,c01 |
| SL3 | a20,c10 | i21,b11 | a21,c11 | a10,c10 | i11,b11 | a11,c11 | a10,c00 | i11,b01 | a11,c01 |
| SL4 | a10,b10 | a10,b11 | a11,b11 | a10,b00 | a10,b01 | a11,b01 | a00,b00 | a00,b01 | a01,b01 |
| SL5 | a10,b11 | a11,b11 | a11,b12 | a11,b00 | a11,b01 | a11,b02 | a00,b01 | a01,b01 | a01,b02 |
| SL6 | a20,b10 | a21,b11 | a21,b11 | a10,b10 | a10,b11 | a11,b11 | a10,b00 | a10,b01 | a11,b01 |
| SL7 | a20,b11 | a21,b11 | a21,b12 | a10,b11 | a11,b11 | a11,b12 | a10,b00 | a11,b01 | a11,b02 |

| | | |
|---|---|---|
| SL4 | SL2 | SL5 |
| SL0 | HP | SL1 |
| SL6 | SL3 | SL7 |

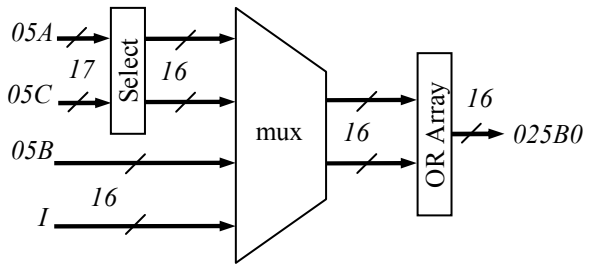



**Fig. 13. Quarter-pixel interpolation datapath used for quarter-pixel search location SL3.**

In the proposed hardware, a different quarter-pixel interpolation datapath is used for each quarter-pixel search location. Therefore, a total of 8 quarter-pixel interpolation datapaths are used. The quarter-pixel interpolation datapath used for the quarter-pixel search location SL3 is shown in Fig. 13. The datapaths used for other quarter-pixel search locations are similar to this datapath.

The select block selects the proper part of the input variable according to the horizontal position of the half-pixel motion vector. The selected parts of *A* and *C* type half pixels and the *B* type half pixels and integer pixels are sent to a multiplexer. The multiplexer selects the pixels that will be used in the interpolation depending on the horizontal component of the half-pixel motion vector. If the horizontal component of the half-pixel motion vector equals zero multiplexer selects *B* and *I*, otherwise it selects *A* and *C*.

## V. Experimental Results

Both open and closed loop coding performance of the proposed all binary sub-pixel ME approach is assessed. In the open loop test scheme the current video frame is estimated from the previous one and the distortion between the original and the reconstructed frame is measured using the peak signal to noise ratio (PSNR) criterion.

Table III shows the average open loop ME performance of the proposed method along with other 1BT based approaches that perform integer ME. As seen from these results, the proposed MF1BT based sub-pixel ME (MF1BT-SP) approach improves the performance of MF1BT based integer pixel ME by up to 0.7 dB because of the sub-pixel accuracy obtained with all binary operations. The proposed all-binary sub-pixel ME approach outperformed the performance of MAD based 8bit/pixel integer ME for the Mobile video sequence, which is caused by the highly detailed texture information of the image frames.

The closed loop performance of the proposed method is evaluated on H.264/AVC using the JM v12.4 reference software [31]. The experimental results for the closed loop test of the proposed method are illustrated in Fig. 14 for the Foreman, Mobile, Coastguard, and Tennis sequences. As seen from these results, the proposed MF1BT based all binary sub-pixel ME approach significantly improved the performance of the MF1BT based integer pixel ME in H.264 video coding. Furthermore, the proposed method also gives better results compared to MAD based integer 8bit/pixel ME for mid and high bit-rates. Note that the residual coding performance of the H.264/AVC can balance small estimation errors caused by binary ME.

The proposed MF1BT based sub-pixel ME hardware architecture and the MF1BT based integer pixel hardware architecture are implemented in Verilog HDL and verified with simulations using Mentor Graphics ModelSim. The proposed sub-pixel ME hardware has very low complexity because of its all-binary nature. The number of clock cycles needed for the entire sub-pixel ME operation is 49 clock cycles which is very small compared to the 1039 clock cycles needed for integer ME.

Sub-pixel ME hardware is synthesized to XC2VP30 FPGA using Synplicity's Synplify Pro at 192 MHz clock frequency. Integer ME hardware is synthesized to the same FPGA using Synplicity's Synplify Pro at 187 MHz clock frequency. The MF1BT based sub-pixel ME hardware occupied 8 of the 136 BlockRAMs and 1385 (4 %) LUTs of the XC2VP30 FPGA.

A pipelined configuration of the integer and sub-pixel ME hardware is capable of processing 720p resolution video sequences at a frame rate of 30 fps at quarter pixel accuracy where [28] can process 16CIF images in real-time with half-pixel accuracy. Required clock frequency for 720p motion estimation performance is only, [(720*1280)/(16*16)]*(1039+49)*30fps=117,504MHz while the proposed design is verified at 185MHz.

Based on these results it is clear that the proposed all binary sub-pixel ME approach significantly improves the estimation accuracy compared to integer-pixel binary ME, and it can be used in the consumer electronics devices with tight processing resource and power constraints such as mobile phones and camcorders.

## VI. Conclusion

In this paper, a novel all binary sub-pixel accurate ME approach and an efficient hardware architecture for this sub-pixel ME approach are presented. The experimental results showed that the proposed all binary sub-pixel 1BT ME performance is better than integer 1BT ME and the proposed hardware architecture has very low hardware complexity. Therefore, the proposed hardware is especially suitable for many mobile consumer electronics devices such as mobile phones and camcorders.

## References

[1] B. G. Haskell, A. Puri, and A. N. Netravali, Digital video: an introduction to MPEG-2, New York: Chapman & Hall, 1997

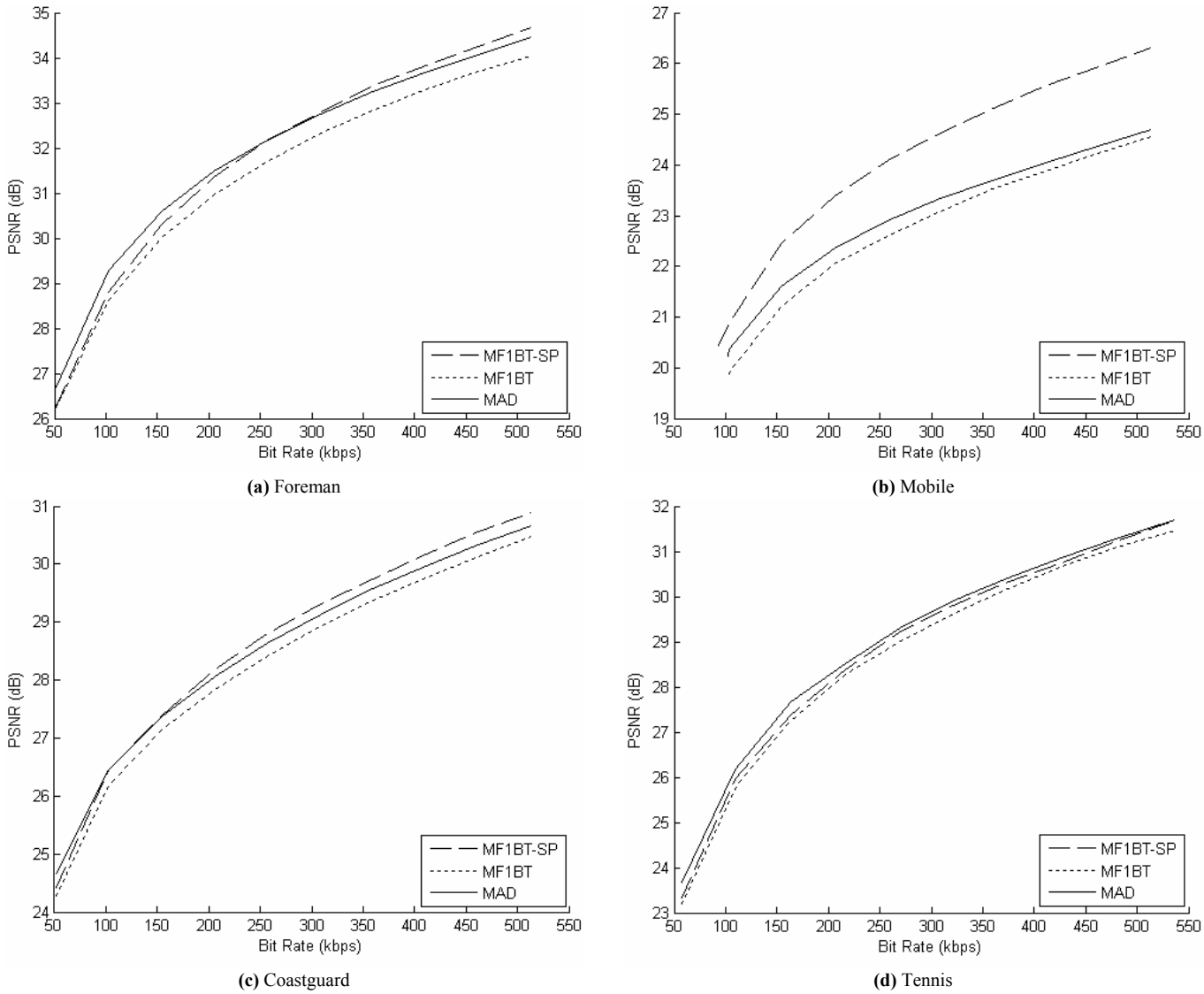
[2] ITU-T Recommendation H.263, "Video coding for low bit rate communication," 1996.

[3] J.V. Team, "Draft ITU-T recommendation and final draft international standard of joint video specification," ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.

[4] B. Natarajan, V. Bhaskaran, K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 702-706, Aug. 1997.

**TABLE III. AVERAGE PSNR (DB) OF SEVERAL SEQUENCES RECONSTRUCTED BY VARIOUS MOTION ESTIMATION TECHNIQUES, WITH FULL SEARCH AND A BLOCK SIZE OF 16×16 PIXELS WITH A MOTION VECTOR SEARCH RANGE OF 16 PIXELS.**

| Method | Video Sequences (Frame Size, Sequence Length) | | | | | |
|---|---|---|---|---|---|---|
| | Football (352×240) (125 frames) | Foreman (352×288) (300 frames) | Tennis (352×240) (150 frames) | Garden (352×240) (115 frames) | Mobile (352×240) (300 frames) | Coastguard (352×288) (300 frames) |
| MAD | 22.88 | 32.09 | 29.45 | 23.79 | 23.94 | 30.48 |
| 1BT [6] | 21.83 | 30.32 | 28.11 | 23.31 | 23.61 | 29.83 |
| MF-1BT [11] | 21.81 | 30.38 | 28.18 | 23.26 | 23.63 | 29.88 |
| **MF1BT-SP** | **22.03** | **30.62** | **28.39** | **23.67** | **24.31** | **29.99** |




**(a)** Foreman **(b)** Mobile

**(c)** Coastguard **(d)** Tennis

**Fig. 14. H.264/AVC results for the proposed method.**

[5] S. Ertürk, "Multiplication-Free One-Bit Transform for Low-Complexity Block-Based Motion Estimation," *IEEE Signal Process. Lett.*, vol. 14, no. 2, pp. 109-112, Feb. 2007.

[6] Ertürk and S. Ertürk, "Two-Bit Transform for Binary Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 938-946, July 2005.

[7] O. Urhan and S. Ertürk, "Constrained One-Bit Transform for Low Complexity Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, pp. 478-482, Apr. 2007.

[8] O. Urhan, "Constrained One-Bit Transform Based Motion Estimation using Predictive Hexagonal Pattern," *Journal of Electron. Imaging*, vol. 16, no. 3, article no: 033019, July-Sep. 2007.

[9] H. Lee and J. Jeong, "Early Termination Scheme for Binary Block Motion Estimation," *IEEE Trans. Consumer Electron.*, vol. 53, no. 4, pp. 1682-1686, Nov. 2007.

[10] P. H. W. Wong and O. C. Au, "Modified one-bit transform for motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 1020-1024, Oct. 1999.

[11] B. Demir and S. Ertürk, "Block motion estimation using modified two bit transform," *Lect. Notes in Computer Science (LNCS)*, vol. 4263, pp. 522-531, 2006.
[12] B. Demir and S. Ertürk, "Block Motion Estimation Using Adaptive Modified Two-Bit Transform", *IET Image Process.*, vol. 1, no. 2, pp. 215-222, June 2007.
[13] H.-Y. Oh, D.-H. Kim, O. Urhan, T.-G. Chang, "Modified Constrained One-Bit Transform Based Fast Block Motion Estimation", *IEEE Trans. Consumer Electron.*, Vol. 53, No. 3, pp. 1093-1097, Aug 2007.
[14] J.W. Suh and J. Jechang, "Fast sub-pixel motion estimation techniques having lower computational complexity," *IEEE Trans. Consumer Electron.*, vol. 50 , pp. 968-973, 2004.
[15] J.F. Chang and J.J. Leou, "A Quadratic Prediction Based Fractional-Pixel Motion Estimation Algorithm for H.264," IEEE Int'l Symposium. Multimedia, pp. 491-498, 2005.
[16] O. Akbulut, O. Urhan, S. Ertürk, "Fast Sub-Pixel Motion Estimation by means of One-Bit Transform", *Lecture Notes in Computer Science (LNCS)*, vol. 4263, pp. 503-510, Nov 2006.
[17] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1301-1308, Oct. 1989.
[18] Y.-H. Yeh, C.-Y. Lee, "Cost-effective VLSI architectures and buffer size optimization for full-search block matching algorithms," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 7, no. 3, pp. 345-358, Sep 1999.
[19] S. Yalcin, H. Ates, I. Hamzaoglu, "A High Performance Hardware Architecture for an SAD Reuse based Hierarchical Motion Estimation Algorithm for H.264 Video Coding", Int. Conf. on Field Programmable Logic and Applications, pp. 509–514, August 2005.
[20] K.-M. Yang, M.-T. Sun, L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1317-1325, Oct. 1989.
[21] H. Yee and H.Y. Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, pp. 407-416, Oct. 1995.
[22] S. Wong, S. Vassiliadis, S. D. Cotofana, "A Sum of Absolute Differences Implementation in FPGA Hardware", in Proceedings of the 28th EUROMICRO Conference, pp. 183-188, Dortmund, Germany, Sept. 2002.
[23] D. Larkin, V. Muresan, N. O'Connor, "A low complexity hardware architecture for motion estimation," in Proceedings of the 2006 IEEE International Symposium on Circuits and Systems, pp. 2677-2680, May 2006.
[24] B.M.H. Li and P.H.W. Leong, "Serial and parallel FPGA-based variable block size motion estimation processors," *Journal of Signal Process. Syst.*, vol. 51, no. 1, pp. 77-98, Apr. 2008.
[25] J.-H. Luo, C.-N. Wang, and T. Chiang, "A novel all-binary motion estimation (ABME) with optimized hardware architectures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, pp. 700-712, Aug. 2002.
[26] R. Wang, M. Li, J. Li, Y. Zhang "High Throughput and Low Memory Access Sub-pixel Interpolation Architecture for H.264/AVC HDTV Decoder," *IEEE Trans. Consumer Electron.*, vol. 51, no. 3, pp. 1006-1013, Aug 2005.
[27] S. Yalcin, I. Hamzaoğlu, "A High Performance Hardware Architecture for Half-Pixel Accurate H.264 Motion Estimation", IFIP International Conference on VLSI-SoC, October 2006.
[28] T. Dias, N. Roma, L. Sousa, "Efficient Motion Vector Refinement Architecture for Sub-Pixel Motion Estimation Systems," IEEE Workshop on Signal Processing System Design and Implementation, pp. 313-318, 2005.
[29] V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards: Algorithms and Architectures. 2nd. Kluwer Academic Publishers, 1997.
[30] A. Çelebi, O. Urhan, İ. Hamzaoğlu, S. Ertürk, "Novel Hardware Implementations of Constrained One-Bit Transform Based Motion Estimation," *IEEE Signal Process. Letts.*, Submitted for publication, 2008.
[31] H.264/AVC JM reference software. [online: http://iphome.hhi.de/suehring/tml/]

**Anıl Çelebi** (S'00) was born in Ordu, Turkey. He received the B.Sc. and M.Sc. degrees in electronics and communication engineering from Kocaeli University, Kocaeli, in 2002 and 2005 respectively. He is currently working toward the Ph.D. degree at the Graduate School of Natural and Applied Sciences, Kocaeli University. His major research interests include very large scale integration (VLSI) design and implementation for analog/mixed signal systems, image processing systems, and video coding systems.

**Orhan Akbulut** was born in Kütahya, Turkey. He received the B.Sc. and M.Sc. degrees in electronics and telecommunication engineering from Kocaeli University in 2005 and 2007 respectively. He is currently working towards the Ph.D. degree at the Graduate School of Natural and Applied Sciences, Kocaeli University. His major research interests are image and video coding systems.

**Oğuzhan Urhan** (S'02-M'06) received his B.Sc., M.Sc., and Ph.D. degrees in electronics and telecommunication engineering from the University of Kocaeli, Kocaeli, Turkey, in 2001, 2003, and 2006, respectively.
Since 2001 he has been with the Department of Electronics and Telecommunications Engineering, University of Kocaeli, Turkey, where he is currently assistant professor. He was a visiting professor at Chung-Ang University, Korea, from 2006 to 2007. His research interests include digital signal and image processing, in particular, image and video restoration and coding.

**İlker Hamzaoğlu** (M'00) received his B.Sc. and M.Sc. degrees in Computer Engineering from Bogazici University, Istanbul, Turkey in 1991 and 1993 respectively. He received his Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign, IL, USA in 1999. He worked as a Senior and Principle Staff Engineer at Multimedia Architecture Lab, Motorola Inc. in Schaumburg, IL, USA between August 1999 and August 2003. He is working as an Assistant Professor at Sabanci University, Istanbul, Turkey since September 2003. His research interests include SoC ASIC and FPGA design for digital image and video processing and coding, low power digital SoC design, digital SoC verification and testing.

**Sarp Ertürk** (M'99) received his B.Sc. in Electrical and Electronics Engineering from Middle East Technical University, Ankara in 1995. He received his M.Sc. in Telecommunication and Information Systems and Ph.D. in Electronic Systems Engineering in 1996 and 1999 respectively from the University of Essex, U.K. From 1999 to 2001 he carried out his compulsory service at the Army Academy, Ankara. He is currently appointed as Full Professor at Kocaeli University, where he worked as Assistant Professor between 2001 and 2002, and Associate Professor between 2002 and 2007. His research interests are in the area of digital signal and image processing, video coding, remote sensing and digital communications.