Frame Rate Up-Conversion Technique Using Hardware-Efficient Motion Estimator Architecture for Motion Blur Reduction of TFT-LCD

Jonghee HWANG†a), Yongwoo CHOI†, Nonmembers, and Yoonsik CHOE††, Member

SUMMARY  Motion blur in TFT-LCD is caused by sample and hold characteristic, slow response time of liquid crystal, and the inconsistency between object tracking of the human eye and the actual object location. In order to solve this problem, a high frame rate driving method based on motion estimation and motion compensation has been applied to LCD products. However, as the required processing time of motion estimation increases in LCD TV and monitor systems, real-time video image processing becomes more difficult. Frame interpolation through the large macro block (MB) size has limitations to detect small objects. So, this paper proposes the efficient motion estimator architecture which uses seven kinds of macro blocks to enhance the accuracy of motion estimation and combines the parallel processing with pre-computation technology and hardware optimization for high-speed processing. Also, for increased efficiency in the hardware architecture, we employed an I2C (Inter Integrated Circuit) communication unit to control the key parameters easily through the personnel computer. Simulation results show that the critical path at the motion estimator is reduced by about 27.47% compared to the conventional structure. As a result, the proposed motion estimator will be applicable for the high-speed frame interpolation of variable video.

key words: motion blur, motion estimator, TFT-LCD, pre-computation

1. Introduction

Recently, the digital flat-panel TV market has recorded rapid growth due to the development of 3D image processing technology [1]. TFT-LCD among digital flat-panel displays has achieved faster growth than other displays and is leading the growth of the next generation display. However, video quality which is called as the base of TV cannot excel a cathode ray tube (CRT) TV. Although LCD TV display has achieved the level of the rapid development which other display cannot keep up with, we had to admit that its video quality is lower than CRT TV [2]. Motion blur has been used as a measure of video quality performance in the display industry. Here, motion blur represents the apparent streaking of rapidly moving objects in a still image or a sequence of images such as a movie or animation. Main causes of motion blur in TFT-LCD can be divided into two parts. First, it is caused by sample and hold driving characteristics. In other words, as shown in Fig. 1, if the voltage inputted from source driver integrated circuit (IC) is charged in storage capacitance ($C_{st}$) according to gate voltage ($V_{gs}$), motion blur occurs because TFT-LCD has a characteristic that adjusts the amount of light which is transmitted from the backlight to the panel by changing optical properties of liquid crystal during the time of one frame depending on the charged amount.

The second, it is caused by the inconsistency between object tracking of the human eye and the actual object location. As shown in Fig. 2, when people observe rapidly moving objects, the human eye moves along the object. However, the location of the eye movements does not match with the location of the object in the actual screen. The human eye movements are continuous along the object, whereas display device has the discrete movement as the video frame unit. So, original image in the retina is formed to inaccurate position because of this issue.

In order to remove the above causes of motion blur, a method to replace hold type driving with impulsive type driving has been used. As shown in Fig. 3, in terms of the LCD panel, the impulsive driving method that inserts black image data or gray image data in the middle of neighboring video frames exists. In terms of LCD backlight, the scanning backlight method that flashes on and off very quickly to
achieve the effect of a high frame rate exists. However, these methods result in inevitable side effects of the brightness loss and flicker increase. So, as shown in Fig. 4, frame rate up-conversion based on motion estimation and motion compensation (MEMC) has been applied to LCD displays. It has been found that the use of motion information in frame rate conversion reduces the motion judder or motion blur efficiently through the reduction of hold time [3]. When the response time for liquid crystal is sufficient, MEMC technology does not reduce the brightness of LCD displays. Generally the higher frame rate than input video frame rate can remove flicker on the screen [4].

Many studies have been published concerning MEMC. Most conventional MEMC algorithms can be divided into two main modules: the motion estimation module which searches motion vectors between two consecutive frames, and the motion compensated frame interpolation module which configures the intermediate frame data between neighboring frames using motion vectors of moving objects from motion estimation.

In particular, motion estimation constitutes a significant portion of the required processing resources in LCD TV and monitor systems. For example, a full search algorithm is the most convenient and effective way to perform motion estimation, but as this requires computing SAD (sum of absolute difference) values for all pixels within the search range to obtain motion vector data for macro blocks with different block sizes, many operations are required. And frame interpolation through the large macro block (MB) size has limitations to detect small objects. A full search algorithm in MPEG-2 system uses a fixed block size of $16 \times 16$ to estimate the motion. When the moving objects are small, the motion vector data can express an inadequate precision.

So, this paper proposes the efficient motion estimator architecture which uses seven kinds of macro block to enhance the accuracy of motion estimation and combines the parallel processing with pre-computation technology and hardware optimization for high-speed processing; first, we applied parallel operations to blocks for SAD calculation. Second, to avoid critical path problems in the motion estimation detection (MED) unit and 41 SAD values calculation unit, we have replaced the most commonly used ripple carry adders with carry skip adders in the adder trees. Third, we employed a pre-computation technique contributing to the speed improvement in order to reduce the switching activity of the input signal in the minimum SAD calculation and MED units. Also, for increased efficiency in the hardware architecture, we employed an I2C (Inter Integrated Circuit) communication unit to control the key parameters easily through the personnel computer.

The rest of this paper is organized as follows. Section 2 explains the full search algorithm and reviews previous research. In Sect. 3, we propose the efficient motion estimator architecture for the full search algorithm with variable block sizes. Section 4 describes an implementation of the motion estimator using Verilog HDL (Hardware Description Language), and then describes the verified results using FPGA (Field Programmable Gate Array) and Synopsys design compiler. Finally, we conclude in Sect. 5.

2. The Motion Estimator through the Full Search Algorithm

2.1 The Full Search of Variable Block Sizes

Motion estimation methods include the optical flow method, pel-recursive method, phase correlation method and block matching method. Among them, the block matching method is the most popular because it is easy to implement in hardware and control memory data [5]. As shown in Fig. 5, the block matching method for full search finds the block with minimum matching error by matching the current $N \times N$ block in the current frame and all reference blocks in the search range of the previous frame, and then estimating the motion vector by using information from the matched block. It may be represented by equations (1).

$$SAD(i, j) = \sum_{k=1}^{N} \sum_{l=1}^{N} |C(k, l) - R(i + k, j + l)|$$

$$\text{Best\_match} = \min(SAD(i, j)) \quad (1)$$

![Fig. 3 Motion blur reduction technique in LCD module.](image)

![Fig. 4 Frame rate up-conversion through motion estimation and motion compensation.](image)

![Fig. 5 The motion estimation through block matching.](image)
Here, \(C(k, l)\) and \(R(i+k, j+l)\) represent the brightness value in the indicated pixel location of the current frame and reference frame respectively. Best match indicates the motion vector (MV\((i,j)\)) with minimum SAD value in the search range. The block matching is the process of calculating the similarity between two blocks; two successive frames will share a significant amount of information because of the very small time interval between frames. This shared information is called “temporal redundancy,” and is very important to interpolate data frames for motion blur reduction.

MPEG-2 employs motion compensation for a fixed block size of \(16 \times 16\) and MPEG-4 employs motion compensation for two block sizes, namely \(16 \times 16\) and \(8 \times 8\). On the other hand, we employ H.264 motion compensation to enhance the accuracy of motion estimation. It uses seven block sizes ranging from \(4 \times 4\) to \(16 \times 16\) as shown in Fig. 6. Thus, the output of minimum SAD and motion vector in the motion estimator require 41 total blocks: 16 of \(4 \times 4\) blocks, 8 of \(4 \times 8\) blocks, 8 of \(8 \times 4\) blocks, 4 of \(8 \times 8\) blocks, 2 of \(8 \times 16\) blocks, 2 of \(16 \times 8\) blocks and 1 of \(16 \times 16\) block. Macro block mode consists of \(8 \times 8\), \(8 \times 16\), \(16 \times 8\), and \(16 \times 16\) blocks, while sub macro block mode consists of \(4 \times 4\), \(4 \times 8\), \(8 \times 4\), and \(8 \times 8\). Note that, when the block size for motion compensation is small, more accurate prediction is possible.

2.2 Review of Previous Research

Many hardware structures and algorithms for motion estimation have been proposed. From the viewpoint of hardware structure, architectures to enable low-power and high-speed processing have comprised parallel arrangements of the unit blocks in the motion estimator [6], [7]. Next, various search range scanning methods to maximize the recycling of memory data [8] and search range simplification based on the experiment result that most block matching is determined by conditions near the origin of the search range [9], [10] have been proposed. The ability to identify the matching block with minimum SAD by using such methods can reduce required computational resources, enabling operation at higher speed. Moreover, the recycling of overlapped data has the effect of reducing memory bandwidth. From an algorithmic perspective, fast search algorithms have been proposed with a view to reducing computational complexity [11]–[13]. These algorithms only search for a few pixels in the specified search range or select macro and sub macro blocks according to given image characteristics. Although these methods may save processing time and offer lower-cost motion estimation, they may also lead to problems such as irregular memory control and degradation of quality. Therefore, the full search algorithm is widely used owing to its simple structure, regularity and excellent quality in practice.

3. The Proposed Motion Estimator Architecture

3.1 The Overall Architecture of the Proposed Motion Estimator

The overall architecture of the proposed motion estimator, as shown in Fig. 7, consists of external memory, embedded SRAM, a hardware logic part for motion estimation, a data control unit and an interface circuit unit for I2C communication. The external memory stores the video data of the current and reference frames, and the embedded SRAM stores the partitioned data as a \(16 \times 16\) macro block in the current frame as well as the data of the maximum search area in the reference frame. The address generator and data control unit load data into internal memory from external memory according to the size of the search area. The interface circuit unit for I2C communication enables the control of the main parameters from an external PC. In this architecture, SRAM was implemented in built-in form into an FPGA (Field Programmable Gate Array) to perform real-time data processing. Even if the proposed motion estimator were manufactured as an ASIC (Application Specific Integrated Circuit), the SRAM could be implemented in built-in form. The hard-
hardware logic unit for motion estimation consists of a motion estimation detection unit, a calculation unit for the 41 SAD values, and a minimum SAD calculation and motion vector generation unit. The motion estimation detection unit calculates each 4 × 4 SAD value between the current block and reference block by dividing the 16 × 16 block into 16 blocks of 4 × 4 size. The calculation unit for 41 SAD values computes SAD values for seven block sizes, i.e., 4 × 4, 4 × 8, 8 × 4, 8 × 8, 8 × 16, 16 × 8 and 16 × 16, by using the 16 calculated 4 × 4 SAD values. The final unit calculates the minimum SAD and generates the motion vector over the entire search area. In Fig. 7, we present the data flow for the entire structure described above. The hardware logic part for motion estimation and the interface circuit unit for I2C communication will be described in detail in the next section.

3.2 The Memory Allocation Method for Motion Detection

In a full search on a given search area, the data of the current 16 × 16 block maintains the initial input values from the SRAM until full search is complete. However, when block matching is performed on the current block by moving it pixel by pixel within the search area, the search area data corresponding to the 16 × 16 block must be updated. Here, we exploited the reuse method of memory data and the scan method, shown by the direction of the arrow in Fig. 8, to efficiently recycle a large amount of duplicate pixel data. The reuse method of memory data reduces the input count of image data by rearranging memory location, eliminating the need to receive new input image data when the next image data to be entered is similar to the 16 × 16 block after motion estimation [10], [15]. In order to apply this method, suppose the current block moves in the left or right direction, depending on scan direction, the motion estimation detection (MED) unit in Fig. 7 receives data corresponding to only 16 × 1 pixels. On the other hand, if the current block moves down, it will receive data corresponding to only 1 × 16 pixels and the data for the other 15 × 16 pixels will be moved into the MED unit to be recycled in the next operation. Therefore, we divide the total SRAM into 16 small SRAMs to enable real-time processing; the MED unit then receives 1 × 16 or 16 × 1 pixels of data per clock to perform block matching for the 16 × 16 macro block in the search area. Pixel data retained from the previous clock are moved through the scan direction and I/O port, and then are reused within arranged unit blocks to calculate the SAD value.

3.3 The Determination of Reference Frames for Motion Estimation

When performing motion estimation, the goal is to find the current block that best matches in the multiple reference frames. However, the above method increases the computational complexity and required memory capacity because it is directly proportional to the number of reference frames. Therefore, when we implement the hardware module of the motion estimator, we need a way to lower the hardware cost and to simultaneously maintain the overall image quality.

Conventionally, when we used two reference frames for all of the video images, we can decide the optimal trade-off point between the hardware cost and the PSNR (peak signal-to-noise ratio). In the case using three reference frames, although some performance improvement is observed, the hardware cost is increased too much. And when using four or five reference frames, only minimal performance improvement can be seen. Through this result, we found that most of the information for the current frame was present in the first of the reference frames. Considering hardware cost and performance improvement, it is reasonable to use two reference frames. In addition, when we implement the hardware module in order to minimize performance variation due to the selection of two reference frames, we have enabled users to change the search area of the first and second frames externally through I2C communication.

3.4 The Motion Estimation Detection Unit

The motion estimation detect (MED) unit enables parallel processing by allocating pixel data for 16 × 16 of the current and reference block to 16 MED_X blocks. This increases the area occupied by logic, but decreases the processing time. A MED_X block calculates the 4 × 4 SAD for the smallest block size in a variety of motion compensation blocks, and the results of 16 MED_X arranged in parallel are transmitted to the calculation unit for 41 SAD values. Each MED_X block is composed of 16 AD (absolute difference) blocks and a 32-bit I/O port with adder tree, as shown in Fig. 9. An AD block with an 8-bit I/O port performs the absolute value calculation of a pixel unit. The AD_x, 4, 8, 12 blocks receive new search area data from SRAM through the upper block of MED_X and transmit the previously received data through the output port. The AD blocks receiving new data from SRAM and the transmission direction for the existing data are set differently depending on the selected signal of the motion estimator describing the scan direction. If the scan direction is to the right, the reference pixel data are transmitted in the direction of AD_x, 4, 8, 12 → AD_x, 5, 9, 13 → AD_x, 6, 10, 14 → AD_x, 7, 11, 15. However, the received data in the current block will be maintained until block matching within search area is complete. The motion
Fig. 9  (a) The I/O port of a MED_X block, (b) the hardware structure of a MED block.

Fig. 10  The conventional structure of a $4 \times 4$ SAD calculator.

Fig. 11  The proposed AD structure.

Fig. 12  The structure of an 8-bit CSA.

Table 1  Relative comparisons for 8/16-Bit adder types.

<table>
<thead>
<tr>
<th>Adder Type</th>
<th>Area(mm$^2$)</th>
<th>Delay(ns)</th>
<th>Avg.Power(uW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple Carry</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carry Skip</td>
<td>1.23</td>
<td>1.45</td>
<td>0.82</td>
</tr>
<tr>
<td>Carry Select</td>
<td>1.33</td>
<td>1.63</td>
<td>0.78</td>
</tr>
<tr>
<td>Look ahead</td>
<td>1.41</td>
<td>1.61</td>
<td>0.74</td>
</tr>
</tbody>
</table>

3.4.1 The Structures of AD Block and $4 \times 4$ SAD Adder Tree

An AD block calculates the absolute value based on the current and previous pixels. It accomplishes a positive-result subtraction by comparing two received values as input. Next, the $4 \times 4$ SAD value is obtained using 16 difference values through the structure of the adder tree. As shown in Fig. 10, the adder tree which is composed of multiple stages is implemented using eight ripple carry adders at stage 1, four at stage 2, two at stage 3 and one at stage 4. As the ripple carry adder (RCA) generates a carry signal for the next bit operation through the operation of a bit unit, it requires a significant amount of processing time. However, this technique is widely used because of its simple structure. Generally, when a MED unit and an AD block are arranged in parallel and the video image is processed in the architecture of the motion estimator, the data processing time of the AD block and the adder tree, including the calculation unit of the 41 SAD values, becomes the critical path for hardware logic. The structures of the AD block and the adder tree must be optimized to improve the processing speed of the entire system.

Thus, first of all, this paper proposed pre-computation to perform high-speed frame interpolation. It presents a technique to identify only a subset of the input which can be used to describe the output. In other words, in the case of calculating the absolute value using the proposed pre-computation technique, we perform subtractions in advance for two cases of RB-CB and CB-RB values, as shown in Fig. 11. Here, RB and CB represent reference block and current block respectively. Next, the most significant bits (CB[7], RB[7]) of the current and previous pixels are identified as the input of the comparator, and then the AD block have predicted previous to the output using this result. If the CB[7] and RB[7] values are the same, we can calculate the absolute value with the existing structure. The proposed pre-computation structure as is mentioned above enables high-speed implementation by reducing the switching activity.

Second, we applied a carry skip adder (CSA) structure to enable high-speed processing in the adder tree. In the case that we calculate a number of adders used to perform a block matching task for a $16 \times 16$ current block in the search area, a total of 240($8+4+2+1)\times 16$ adders are required to obtain 16 values of $4 \times 4$ SAD in the MED block. Table 1 shows chip area, gate delay, and average power consumption according to the type of adder [16], [17], using a 0.8 $\mu$m standard cell process. The figures are normalized so that the result of the ripple carry adder is ‘1’. In this paper, we have applied the CSA structure for the adder tree on the basis of the result in Table 1 because high-speed processing and minimization of power consumption were possible with a relatively small increase in chip area.

Figure 12 represents the structure of an 8-bit CSA. The
 CSA provides four full adders configuring a 4-bit RCA with the input data and determines whether or not the carry signal is skipped through the AND gate operation for the carry signal of each full adder. So, it has reduced the time delay due to carry signal propagation by forecasting the generated carry signal in a 4-bit RCA unit. As shown in Table 1, if the input of the CSA increases from 8-bit to 16-bit, the logic delay time will be much shorter.

3.5 The Calculation Unit for the 41 SAD Values

The calculation unit for the 41 SAD values generates SAD values for the macro block mode of 8 × 8, 8 × 16, 16 × 8, 16 × 16 and the sub macro block mode of 4 × 4, 4 × 8, 8 × 4 using the 16 values of the 4 × 4 SAD generated in the MED unit. It was implemented through the adder tree structure like 4 × 4 SAD calculation, and we applied the CSA structure connecting two 6-bit RCAs in series. The 4 × 8 and 8 × 4 SAD values are obtained using 16 12-bit adders at stage 1 through the multistage adder tree, and we can obtain the 8 × 8 SAD values using four 13-bit adders at stage 2. Likewise, the 8 × 16, 16 × 8, and 16 × 16 SAD values can be obtained respectively using four 14-bit adders at stage 3 and one 15-bit adder at stage 4.

3.6 The Minimum SAD Calculation and Motion Vector Generation Unit

The minimum SAD calculator unit is composed of 41 parallel blocks extending the lower dotted area shown in Fig. 13(a) and receives 41 SAD values for seven kinds of macro and sub macro blocks per clock. The generation process of the minimum value is as follows. First, we store 41 SAD values in the register set of the current SAD, and the stored values are compared with the register set of minimum SAD values. If the current SAD value is smaller than the minimum SAD, a 1-bit multiplexer outputs ‘1’, and then the minimum SAD value is updated by the current SAD value. Otherwise, as the 1-bit multiplexer outputs ‘0’, the existing minimum SAD value is maintained. In the case in which the search area is (−16, −16) ~ (15, 15), as there are 256 possible block matches, the minimum value operation is performed 256 times. The comparator of the minimum SAD calculator is implemented using the pre-computation technique as shown in the proposed AD structure of Fig. 11. So, it can reduce the switching activity of the input signal.

Next, to generate the motion vector, we can obtain the motion vector by counting the number of current block movements within the search area, as shown in Fig. 13(b). As the current block moves from the upper-left side, its count value is set to ‘1’. When it has reached the end of the search area, the count value will be ‘256’. The motion vector value corresponding to the count value is stored as the LUT (look-up table) and is output according to the count value. We compare the new input of the 41 SAD values per clock with the existing 41 SAD values in the comparator block, and if the input values are smaller than the existing values, the motion vector value replaces the new value by outputting “Enable Signal” to the LUT maintaining the motion vector values. Also, if the count value points to 256, the count and motion vector values are initialized in the next clock signal.

3.7 The Interface Circuit Unit for I2C Communication

Typically, I2C (inter integrated circuit) represents a bi-directional serial bus specification to control peripheral devices. Only two bus lines are required to use I2C. One is a serial data line (SDA), and the other is a serial clock line (SCL) [18]. I2C was used to control the key signal through the PC acting as a master. The slave block in Fig. 7 receives the PC’s commands through the RS-232 terminal using the SDA and SCL lines which are connected with pull-up resistors. The interface circuit enables the data from the internal register values to be read or data to the internal register values to be written. Controls such as search range, LUT generation of the motion vector, and the movement direction for the current macro block can be performed using this method. When changing the control signal mentioned above, the results can be immediately identified without a logic modification of the hardware configuration.
4. Implementation and Verification Results

The proposed motion estimator structure was designed using Verilog HDL. After we performed the gate-level structure analysis using the Synopsys design compiler and FPGA, we completed the verification for the motion estimator structure through a timing simulation. The CIF level of variable image at 30 frames per second was used as the input data of embedded SRAM into FPGA by converting two successive frames of image data to a text file, and the search areas were set as $(-16, -16) \sim (15, 15)$. During the first 16 clocks, data for the $16 \times 16$ current and previous blocks are input to the 16 MED blocks which are arranged in the MED unit. In other words, 16 SRAMs output $16 \times 1$ pixel data simultaneously as an 8-bit bus size, according to the system clock. Therefore, the motion estimator only receives inputs from the SRAM for 16 clocks in order to initiate block matching. It then starts the motion estimation in real time by receiving the $16 \times 1$ or $1 \times 16$ data inputs for the search area depending on the scan direction.

Looking at the data flow within the motion estimator, the MED unit receiving inputs for 16 clocks calculates and outputs 16 values of $4 \times 4$ SAD for the reference block, then 41 SAD values corresponding to seven kinds of block size are generated through the adder tree using the result from the 17th clock. The minimum SAD calculation and motion vector generation unit updates the minimum by comparing the received 41 SAD values with the previously stored minimum. Therefore, it takes 18 clocks to establish the initial motion estimator output. Also, as the entire search area is $(-16, -16) \sim (15, 15)$, it takes $18 + (16 \times 16) = 274$ clocks to identify the best matching block corresponding to the current block.

Next, we carry out the FPGA implementation and verification to measure the delay time of the critical path in each unit and the number of gate counts for the designed structure. The used FPGA is a Virtex 4 family of XC4VLX200, with a package type of FF1513 and a speed grade of −11. The synthesis information is set equally to accurately compare the delay time of the proposed and conventional structures. In the proposed motion estimator’s structure, the pre-computation technique, the hardware optimization through addition alteration and the interface circuit unit for I2C communication are applied. On the contrary, in the conventional structure, it was designed using high-speed processing architecture with a parallel array of the unit block in the reference [6], [14]. In other words, it uses a ripple carry adder and the structure to obtain the absolute value after performing an operation on all input bits instead of the prediction method of input data through pre-computation. Also, it does not include an I2C interface block because the search area is fixed. The critical path affects the operating frequency for the whole architecture because it has a maximum delay time.

In the motion estimation structure, the critical path occurs between the MED unit and the calculation unit for the 41 SAD values, and the main logic consists of an operation block for the absolute value and two adder trees. In other words, it will appear as the computation speed of the motion estimation detection unit. These are shown in Fig. 14. Each adder tree is composed of the basic structure with two 4-bit RCAs and two 6-bit RCAs. Therefore, in the motion estimator including adders with a variety of the input bits, the variation in combinational logic delay will depend on the degree of carry skip. Figure 14 indicates the minimum period according to the occurrence of the carry skip. The conditions for case 1 ~ case 5 are listed below:

- Case1: pre-computation, carry skip occurs in an 8-bit adder
- Case2: Case1 + carry skip occurs in a 9-bit adder
- Case3: Case2 + carry skip occurs in a 10-bit adder
- Case4: Case3 + carry skip occurs in a 12-bit adder
- Case5: Case4 + carry skip occurs in a 13-bit adder

The delay of critical path represents the distribution of $9.443 \sim 14.614$ ns as shown in Fig. 14. As the carry skip increases, the delay is proportionally reduced. In terms of operating frequency, the conventional structure has the operating frequency of 58.048 MHz (1/17.227 ns). On the other hand, in the proposed structure, we were able to obtain an operating frequency of 80.032 MHz (1/12.495 ns) by computing the average minimum period for the variable adder inputs.

Practically, in order to analyze the necessary time for the motion estimation, we define the following quantities:

- Size of the frame: $N_h \times N_v$ (352 $\times$ 288: CIF video)
- Size of the block: $N \times N$ ($16 \times 16$)
- Number of candidate blocks to be matched for one current block: $N_p (16 \times 16$: search range $(-16, -16) \sim (15, 15))$
- Number of clocks to establish the initial motion estimator output: $N_{clk}$ (18: number of clocks for data loading from SRAM)
- Minimum period: $T_{clk}$ (17.227 ns: the conventional architecture, 12.495 ns: the proposed architecture)

Therefore, the necessary time $T_{frame}$ for the motion estimation per one frame is expressed as

$$T_{frame} = \frac{N_h \times N_v}{N^2} \times (N_p + N_{clk}) \times T_{clk}$$  \hspace{1cm} (2)$$

$T_{frame}$ of the conventional motion estimator is 1.869 ms. On the contrary, $T_{frame}$ of the proposed motion estimator is 1.356 ms. Based on this result, a CIF video at
Table 2  Areas for the conventional and proposed Architectures.

<table>
<thead>
<tr>
<th>Area(gate)</th>
<th>Conventional</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Detection</td>
<td>226,685.81</td>
<td>241,990.09</td>
</tr>
<tr>
<td>SADs Calculation</td>
<td>7,372.55</td>
<td>8,465.78</td>
</tr>
<tr>
<td>Minimum SAD Calculation</td>
<td>14,846.75</td>
<td>14,994.15</td>
</tr>
<tr>
<td>Motion Vector Generation</td>
<td>191.73</td>
<td>401.24</td>
</tr>
<tr>
<td>Slave</td>
<td>-</td>
<td>948.78</td>
</tr>
<tr>
<td>Cfgreg Control</td>
<td>-</td>
<td>1,824.33</td>
</tr>
<tr>
<td>Total Motion Estimator</td>
<td>249,096.84</td>
<td>268,624.37</td>
</tr>
</tbody>
</table>

30 frames per second could be processed faster than using the existing structure. Target frequency of the entire system for the display resolution larger than CIF resolution can be satisfied by a parallel array of the proposed motion estimator architecture. Here, $T_{frame}$ for large-size LCD TV having the standard image size of FHD (Full High Definition: $1920 \times 1080$) can be explained as follows. For example, as the number of pixels for a FHD image shows with approximately 20.45 times than the number of pixels for a CIF image, the motion estimation should be performed within 8.35 ms (1/120) per a frame to meet the timing requirement for a FHD video of 120Hz per second. So, the proposed structure with the parallel structure of two or four stages should be used. If the parallel structure of four stages has been used, $T_{frame}$ of the conventional motion estimator is $9.5574$ ms ($1.869 \times 20.45$)/4). On the contrary, $T_{frame}$ of the proposed motion estimator is $6.9329$ ms ($1.356 \times 20.45$)/4).

Next, we performed the synthesis of the gate-level circuits using a Synopsys design compiler with an MC $0.18 \mu m$ CMOS standard cell library and then measured chip area. Typically, after the area of the 2-input NAND gate is normalized to 1, the total area information is calculated by representing a different cell as the comparison figure. As shown in Table 2, the simulation result showed that the conventional motion estimator has a gate count of 0.249M(249,096.84). On the contrary, the proposed motion estimator has a gate count of 0.269M(268,624.37). That is, as the proposed AD structure and high-speed adder is used to shorten the critical path, the area has increased by 7.00%. Here, in the case of a FHD video, as the proposed structure is used repeatedly as the same number of stages for the parallel structure, the number of gate count is increased. Namely, the conventional motion estimator has a gate count of 0.996M. On the contrary, the proposed motion estimator has a gate count of 1.066M.

5. Conclusion

For motion blur reduction in TFT-LCD, we proposed the motion estimation and motion compensation structure having a hardware-efficient motion estimator of variable block sizes to increase frame rate. So, we could perform accurate motion estimation fast. In addition, as we enabled users to easily control key variables such as control signaling of search range, the efficiency of the hardware structure could increase. The proposed hardware module can be included in the chips that are used to control images in the products with the flat panel display or the timing controller that provides the driver IC with the control signals. Simulation and verification results show that the critical path at the motion estimator was reduced by about 27.47% compared to that of the conventional structure by sacrificing an area increase of 7.00%. The proposed motion estimator successfully verified that it can interpolate the data frame in real-time with an 80.032 MHz operation clock and 0.269M logic gates. Therefore, the proposed motion estimator could be used widely in case of a system in which high-speed frame interpolation is an important factor.

References


Jonghee Hwang was born in Chung-nam, South Korea in 1975. He received the B.S. and M.S. degrees from the school of Semiconductor Engineering from the INHA University, Incheon, Korea, in 2001 and 2005, respectively. He is currently studying toward the Ph.D. degree in electrical and electronic engineering at Yonsei University, Seoul, Korea. From 2001 to 2003, he was with LG Electronics Co., Ltd., where he involved in the circuit design of video, DVD, and audio products. He is currently working at LG Display Co., Ltd. from 2005, where he involved in preceding development for LCD image quality improvement. His research interests include the image enhancement for flat panel displays, the high frame rate driving methods and circuits, and image processing.

Yoonsik Choe received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea in 1979, and the M.S.E.E. in systems engineering, M.S. and Ph.D. degrees all in electrical engineering from the Case Western Reserve University, Cleveland, OH, the Pennsylvania State University, University Park, PA and the Purdue University, West Lafayette, IN, in 1984, 1987, and 1990, respectively. From 1990 to 1993 he was a technical staff at the industrial electronics research center in the Hyundai Electronics, Co. Ltd. Since 1993 he has been with the Department of Electrical and Electronic Engineering at the Yonsei University, Seoul Korea, where he is Professor. His research interests include video coding, video communication, statistical signal processing and digital image processing.

Jonghee Hwang was born in Chung-nam, South Korea in 1975. He received the B.S. and M.S. degrees from the school of Semiconductor Engineering from the INHA University, Incheon, Korea, in 2001 and 2005, respectively. He is currently studying toward the Ph.D. degree in electrical and electronic engineering at Yonsei University, Seoul, Korea. From 2001 to 2003, he was with LG Electronics Co., Ltd., where he involved in the circuit design of video, DVD, and audio products. He is currently working at LG Display Co., Ltd. from 2005, where he involved in preceding development for LCD image quality improvement. His research interests include the image enhancement for flat panel displays, the high frame rate driving methods and circuits, and image processing.

Yongsik Choe received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea in 1979, and the M.S.E.E. in systems engineering, M.S. and Ph.D. degrees all in electrical engineering from the Case Western Reserve University, Cleveland, OH, the Pennsylvania State University, University Park, PA and the Purdue University, West Lafayette, IN, in 1984, 1987, and 1990, respectively. From 1990 to 1993 he was a technical staff at the industrial electronics research center in the Hyundai Electronics, Co. Ltd. Since 1993 he has been with the Department of Electrical and Electronic Engineering at the Yonsei University, Seoul Korea, where he is Professor. His research interests include video coding, video communication, statistical signal processing and digital image processing.

Yoonsik Choe received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea in 1979, and the M.S.E.E. in systems engineering, M.S. and Ph.D. degrees all in electrical engineering from the Case Western Reserve University, Cleveland, OH, the Pennsylvania State University, University Park, PA and the Purdue University, West Lafayette, IN, in 1984, 1987, and 1990, respectively. From 1990 to 1993 he was a technical staff at the industrial electronics research center in the Hyundai Electronics, Co. Ltd. Since 1993 he has been with the Department of Electrical and Electronic Engineering at the Yonsei University, Seoul Korea, where he is Professor. His research interests include video coding, video communication, statistical signal processing and digital image processing.

Yongsik Choe received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea in 1979, and the M.S.E.E. in systems engineering, M.S. and Ph.D. degrees all in electrical engineering from the Case Western Reserve University, Cleveland, OH, the Pennsylvania State University, University Park, PA and the Purdue University, West Lafayette, IN, in 1984, 1987, and 1990, respectively. From 1990 to 1993 he was a technical staff at the industrial electronics research center in the Hyundai Electronics, Co. Ltd. Since 1993 he has been with the Department of Electrical and Electronic Engineering at the Yonsei University, Seoul Korea, where he is Professor. His research interests include video coding, video communication, statistical signal processing and digital image processing.

Yongsik Choe received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea in 1979, and the M.S.E.E. in systems engineering, M.S. and Ph.D. degrees all in electrical engineering from the Case Western Reserve University, Cleveland, OH, the Pennsylvania State University, University Park, PA and the Purdue University, West Lafayette, IN, in 1984, 1987, and 1990, respectively. From 1990 to 1993 he was a technical staff at the industrial electronics research center in the Hyundai Electronics, Co. Ltd. Since 1993 he has been with the Department of Electrical and Electronic Engineering at the Yonsei University, Seoul Korea, where he is Professor. His research interests include video coding, video communication, statistical signal processing and digital image processing.