A Hardware Architecture for Motion Compensated Video Frame Rate Up-Conversion

HUONG HO
Communications Research Centre Canada
3701 Carling Ave., Ottawa, Ontario, K2H 8S2
CANADA
huong.ho@crc.gc.ca http://www.crc.gc.ca

Abstract: - A hardware architecture for motion compensated, video frame rate up-conversion (MC-FRUC) applications is presented in this paper. The MC-FRUC architecture has been designed based on an advanced motion estimation (ME) and motion compensated frame interpolation (MCFI) algorithm to achieve interpolated frames with high level of quality. The proposed architecture is a flexible and highly parallel MC-FRUC that has been designed to support frame rate-up conversion (FRUC) for high definition (HD) video at high frame rate. The ME building block of the MC-FRUC circuit is a reconfigurable structure designed to support reconfigurations for single or multiple reference frames. The MCFI building block performs frame interpolation with the objective of minimizing of block artifacts, overlapping, and holes for the interpolated frames. Hardware implementations for the MC-FRUC design were carried out on FPGA where the circuit high performance FRUC capability has been validated.

Key-Words: - Frame rate up-conversion, Multi-frame motion estimation, Motion-compensated frame interpolation, Hardware architecture, FPGA implementation, Reconfigurable.

1 Introduction

Motion compensated frame interpolation techniques for FRUC applications have been the subject of research in recent years [1-7]. For video applications where large amounts of data need to be processed at high speed, a hardware solution with the ability to generate interpolated frames in real time is required. Moreover, it is essential that the interpolated pictures produced by the FRUC circuit have the highest quality possible.

Recently, a number of FRUC hardware architectures have been proposed in the literature [8-11]. The architectures presented in [8-9] use digital signal processors (DSP) for circuit implementations. The FRUC design presented in [10] is a reconfigurable circuit that can be configured to perform frame interpolation based on different frame interpolation (FI) techniques. The architecture presented in [11] describes the design of the ME engine where the details of the interpolation circuit were not provided. The commonality of these architectures is low hardware complexity. Design issues associated with quality level of the interpolated picture were not addressed in these papers.

A number of semiconductor companies have also introduced processors that support FRUC applications to the market over the last few years [12-18]. These companies claim that their circuits support HD pictures. However, there is no detailed data available regarding the quality of the interpolated frames supported by these circuits.

In general, a motion compensated FRUC architecture consists of two building blocks: the ME and the MCFI. The ME generates motion vector (MV) data that are used in the motion compensation (MC) and the FI processes. The accuracy of the MVs contributes to the quality of the interpolated frame [19]. As a result, ME techniques that generate accurate motion trajectories are needed for the design of a motion compensated FRUC circuit.

Normally, ME design based on multiple reference frames (MRF) offers more accurate motion trajectories than ME techniques that use a single reference frame (RF) [20]. Many ME architectures proposed to support MRF have been reported in the literature [21-25]. The architectures proposed in [21],[24] support computation of MV based on 4 RFs. The architecture proposed in [22] supports sequential computation of MV for 5 RFs. The architectures proposed in [23],[25] support parallel computation for 2 RFs. A number of reconfigurable ME architectures based on single reference frame have also been proposed in the literature [26-30]. These architectures support reconfiguration for power aware applications [26],

multiple video standards [27], or different ME algorithms [28-30]. The one bit transform (1BT) proposed in [21] performs block matching (BM) computations based on 1-bit pixels while the architectures illustrated in [22-23] perform BM computations using the luminance component of the pixel represented by 8-bit data. Motion estimation based on low bit depth (1BT) or using only the luminance value of the pixel leads to a reduction in computational complexity at the expense of MV accuracy. Furthermore, memory access schemes needed to retrieve pixels from input frames for BM computations have not been addressed by some of these architectures. Off-chip memory access is the dominant factor that affects the throughput performance of ME circuits. On-chip data buffer that supports fast retrieval of search window data is essential to achieve a high throughput ME circuit [31].

Many MCFI techniques propose the use of a temporal filter [32] or an adaptive overlapped block motion compensation [33] for frame interpolation. The use of a median filter for MVs correction has also been proposed to reduce the effect of artifacts on the interpolated frame [34]. The MVs correction technique proposed in [34] has been implemented in hardware while [32-33] provided only software simulation results of the interpolation algorithms.

In this paper, a MC-FRUC architecture that supports frame rate-up conversion of HD video in real-time at high frame rates is presented. The proposed architecture consists of two building blocks: the reconfigurable multi-frame motion estimation (RMF-ME) and the MCFI. The RMF-ME generates motion fields (MF) data based on an advanced MRF motion estimation technique [5]. The MF data shifted out of the RMF-ME also contains information on the occluded blocks that are useful for holes and overlap detection. The accurate motion estimation and information on occlusions provided by these MFs contribute to the high quality of the interpolated frame. The MCFI takes MF data generated by the RMF-ME as inputs to compute for the motion compensated interpolated frames. The MCFI design incorporates the irregular-grid expanded block weighted motion compensation (IEWMC) and the block-wise directional hole interpolation (BDHI) algorithms to achieve high quality level for the interpolated pictures [6].

The paper is organized as follows. In section 2, the design methodology that transforms the MC-FRUC algorithm and system specifications into an architecture represented by two main building blocks, namely, the RMF-ME and the MCFI, is described. Comparisons of peak signal noise ratio (PSNR) performance of several video sequences simulated based on the MC-FRUC technique to other algorithms are also presented in this section. In Section 3, the detailed architecture and design of the MC-FRUC circuit and its building blocks are illustrated. Hardware implementation results of the RMF-ME and the MCFI designs and comparisons of circuits throughput performance to other architectures are presented in Section 4. Finally, concluding remarks are presented in section 5.

2 Motion Compensated Video Frame Rate Up-Conversion Design Methodology

In this section, the design methodology for the main building blocks of the MC-FRUC is described. The illustrated technique is then used to evaluate several video sequences for PSNR performances. The PSNR data obtained are compared to results presented by existed FRUC techniques.

2.1 Multi-Frame Motion Estimation With Reconfigurability

The RMF-ME architecture proposed in this paper is a reconfigurable ME architecture that can be configured to perform ME based on single RF or MRF. The BM engine of the RMF-ME circuit has been designed based on a hierarchical block matching (HBM) algorithm to reduce computational complexity [5]. Compared to a full search technique, HBM requires less BM error computations but with comparable performance [35-36]. The HBM algorithm performs BM computations on input frames proceeding from lower to higher resolution sequentially. The input frames are first transformed into a pyramid consisting of several images of different size. The size of the images at lower level is four times smaller than the image immediately above it. Based on the result of performance simulations reported in [5], accurate MV can be obtained using 2RFs with the size of top-level image limited to 256x256 pixels. Similarly, the block size is set at 16x16 for all levels of the pyramid where the search range is limited to 6% of the image width for both horizontal and vertical directions. Simulation results presented in [5] also demonstrate that the errors of MVs based on these criteria are much smaller than those produced by full search or other hierarchical algorithms. As a result, the maximum search

distance at the top level image supported by the RMF-ME is set at ±16.

The HBM process performs BM computations on an image based on a pyramid structure illustrated in Fig. 1b. Full search is performed on blocks at the top-level image to obtain the coarse MF. For each block it calculates the matching error for every position within the configured search region and searches for the position that has the smallest matching error. The motion vector corresponding to this position is assigned to the block. The matching errors are measured by the sum of absolute difference (SAD) value over the block. To obtain accurate motion trajectories, the block matching errors have been computed using both the luminance and chrominance components of the images. On the next levels of the pyramid, the density of the motion vectors is increased. Since the images at all pyramid levels are partitioned into blocks of the same size, one block at level L_4 corresponds to four blocks at level L_3 and so on. The MVs on the upper level are used as search centers for the blocks on the lower level. Three MVs of the adjacent blocks on the upper level are used as search centers for each block on the lower level. Thus the BM process will run three times on three sets of candidate blocks for each block on the lower level. The MV associated with the smallest matching errors is selected as the final MV of the block.

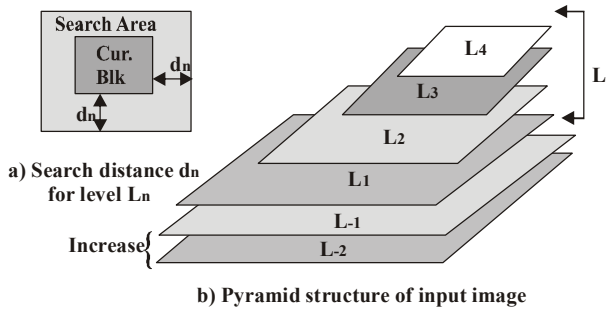

Fig. 1. Reconfigurable for number of level L and search distance d_n

The images at levels L_{-1} and L_{-2} belong to the iteration process where extra BM computations are performed to obtain MVs with more accuracy. The image at level L_{-1} is partitioned into 8x8 blocks and the image at level L_{-2} is divided into blocks of 4x4. The density of the motion vectors on levels L_{-1} and L_{-2} is increased due to smaller block sizes. The BM on each level of the iteration process also repeats three times for the three sets of candidate blocks.

The maximum number of pyramid levels L supported by the RMF-ME design is set to 4 plus 2 levels associated with the iteration process. Since the top-level image is limited to 256x256 pixels, a four level configuration is enough to support 1080P images. The number of levels L is configurable to enable the RMF-ME to support different video formats. For a target video format and a target search distance, the reconfigurability of L also enables lower or higher number of levels to be selected to reduce the computational complexity for the BM process. For example, if the search distance for a CIF video is set at [-16, +16], the RMF-ME circuit can be configured with a 2 level pyramid L_1 and L_2. On the other hand, a 3 level pyramid configuration is needed if the target search distance is set at [-32, +32]. The two levels L_{-1} and L_{-2} in the iteration process are also reconfigurable. To achieve more accuracy for the MVs, both levels can be selected. Otherwise, one MV per block of 8x8 pixels for video in the 720P or 1080P format is adequate for FRUC applications [5]. The search distance d_n is also reconfigurable where it can take values from 0 to 16.

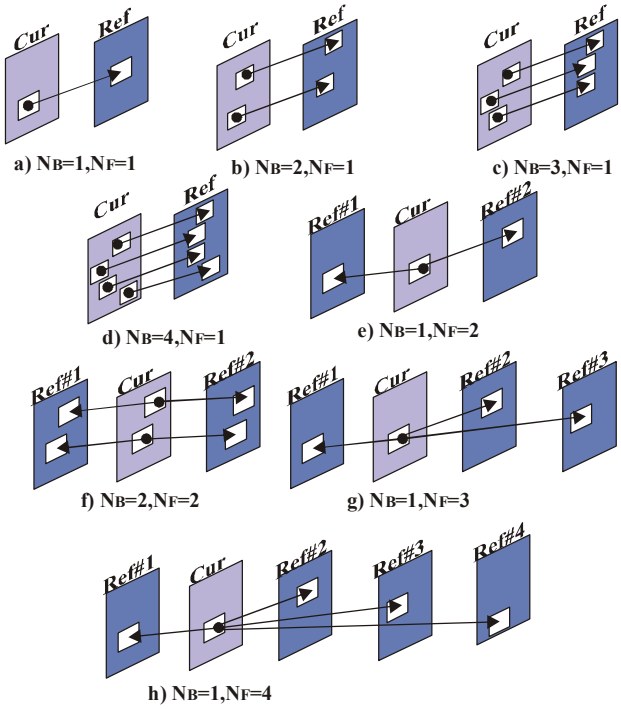

Fig. 2. Reconfiguration modes supported by the RMF-ME

The RMF-ME supports BM computations for single and multiple frames as depicted in Fig. 2. The RMF-ME supports parallel computations of up to 4 macro blocks (MB) and 4 RFs. The number of blocks N_B

and the number of frames N_F can be configured in real time.

2.2 Motion-Compensated Frame Interpolation

The MCFI hardware architecture proposed in this paper has been designed based on the algorithm in [6]. This MCFI algorithm computes the interpolated frames using MF data, which has been generated by the RMF-ME, based on two adjacent frames. The two generated intermediate frames FR_{IP1} and FR_{IP2}, based on these MFs, are located at the same time instance as the frame to be interpolated which is described in Fig. 3.

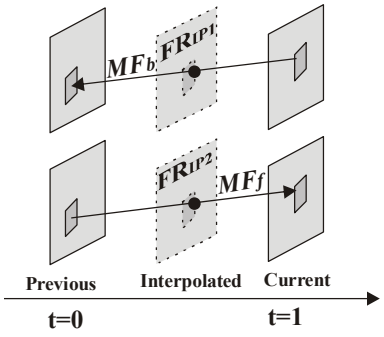

Fig. 3. Generated Interpolated frames based on MF_b and MF_f

The backward (MF_b) and forward (MF_f) MFs at the input of the MCFI block contain the motion information of blocks of 4x4 or 8x8 pixels that originates from the current and previous frames, respectively. The MCFI process consists of two parts: the generation of interpolated frames and the holes filling process where holes caused by occlusions and motion estimation errors are filled. The interpolation part generates interpolated frames based on the irregular-grid expanded block weighted motion compensation (IEWMC) technique. The hole filling part combines the two overlapped frames into one single frame. The pixels located in the hole areas of the combined frame are then constructed using the BDHI technique.

2.3 MC-FRUC Performance Evaluation

In this section, the performance of the MC-FRUC technique is evaluated. The simulation was performed on 3 video test sequences in CIF format using 2 RFs. The RMF-ME has been configured based on 2 RFs configuration as illustrated in Fig. 2f. The RMF-ME has also been configured to support 2-level pyramid and 2 levels of iteration for this simulation. The search range of [-16, 16] has been configured for a fair comparison with the algorithm proposed in [21]. The RMF-ME calculates the block matching errors based on the luminance and chrominance components of the images.

In the experiment, each of the 3 test video sequences has 31 frames. The even numbered frames are first removed from the sequences and then replaced by the interpolated frames. The average PSNR of the interpolated frames were calculated for each test sequence. Average PSNR results obtained from the MC-FRUC technique and the results of the 1BT algorithm reported in [21] are listed in Table 1. The PSNR results obtained by the 1BT algorithm based on 4 RFs are used for this comparison. As shown in Table 1, the PSNR values obtained from the MC-FRUC are higher than the results reported in [21] for the Mobile, Foreman and Football sequences. The PSNR performance of the 1BT algorithm is still not comparable to the MC-FRUC technique even if more than 4 RFs are used. In fact, simulation results for 6, 8 and 10 RFs based on the 1BT algorithm presented in [21] show PSNR performance lower than the results generated by the MC-FRUC for the Mobile, Foreman and Football sequences based on 2 RFs.

Table 1
AVERAGE PSNR GAIN (DB) OF SEVERAL VIDEO SEQUENCES RECONSTRUCTED BY MC-FRUC AGAINST THE 1BTALGORITHM

Sequences	Average PSNR		
	RMF-ME 2 RFs	1BT [21] 4 RFs	Gain
Football	24.27	23.61	0.66
Mobile	25.80	24.45	1.35
Foreman	35.88	33.94	1.94

3 Motion-Compensated Video Frame Rate Up-Conversion Circuit Design

In this section, the architecture and circuit design of the MC-FRUC and its building blocks are described. The top-level architecture of the MC-FRUC is depicted in Fig. 4.

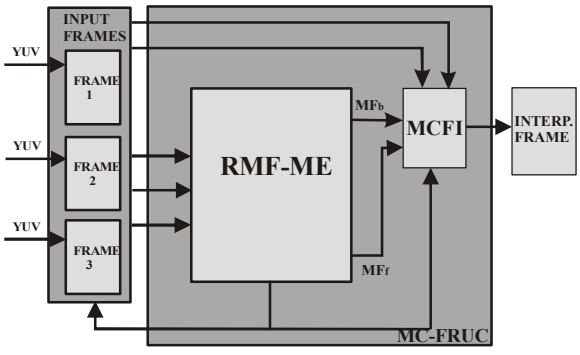

Fig. 4. Architecture of the MC-FRUC circuit

3.1 The RMF-ME Architecture and Circuit Configurations

The RMF-ME design consists of three main building blocks: the input buffer unit (IBU), the spiral blocks matching unit (SBMU) and the control unit (ME-CU). The block diagram of the RMF-ME circuit is depicted in Fig. 5.

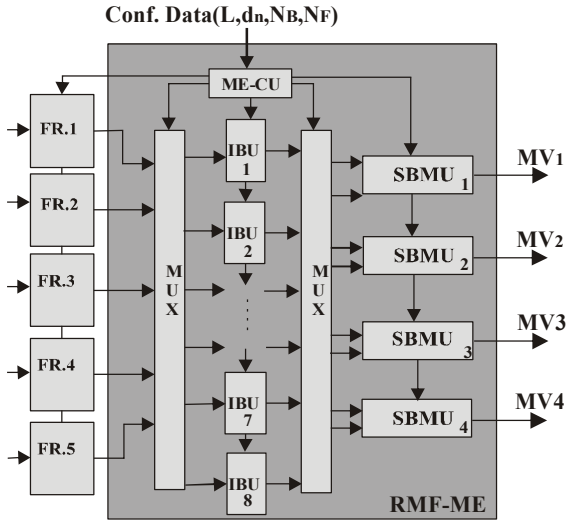

Fig. 5. Architecture of the RMF-ME circuit

The IBUs are ping-pong input buffers, each of which consists of two memory blocks (MEMB) designed to store the block of input pixels on-chip for BM computations. When one MEMB is involved in the BM computations, the other is ready to receive the next block of pixels shifting in from the input frame. This ping-pong architecture allows the RMF-ME circuit to perform BM computations continuously without having to wait for the next block of input pixels to be shifted in. The MEMBs have been designed using block RAM (BRAM) available on most FPGAs to allow simultaneous retrieval of a large block of pixels of the candidate block [25]. To reduce external memory access, the entire search area of the current block is shifted in and stored in the MEMB. At every clock cycle, each MEMB shifts out 64 pixels to the SBMU. Thus, one clock cycle is needed to shift out one candidate block at levels L_{-1} and L_{-2}. For other levels, a total of four clock cycles is required to shift out one single candidate block. This data reuse scheme enables the RMF-ME to compute the SAD values of one 16×16 candidate block in four clock cycles. Since the maximum search distance at a pyramid level is set at ±16, the MEMB is designed to be able to hold the data of 48×48 pixels. As three colour components of the pixel are involved in the BM process, a total of 48x48+2(24x24) bytes are required to store pixels data of the search area. Data of the search area and the current block stored in the MEMB has been replicated in sets of 96 RAM blocks to allow fast pixel retrieval. This arrangement enables the MEMB to shift out a block of 8×8 data pixels in every clock cycle. The set of 8 IBUs allows the search area of 8 MBs to be stored on chip for parallel BM computations. For the parallel computation of 4 MBs as described in Fig. 2d, the 8 IBUs enable blocks of pixels of 4 candidate blocks and 4 current blocks to be shifted out to the SBMUs simultaneously.

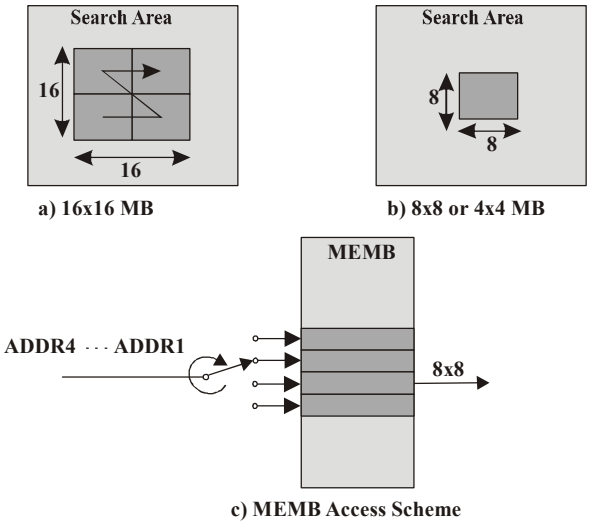

Fig. 6. Data reuse scheme for a) 16×16 MB; b) 8×8 or 4×4 MB and the MEMB memory access scheme

The data reuse scheme proposed for the design of the RMF-ME is depicted in Fig. 6a for a 16×16 MB and in Fig. 6b for 8×8 or 4×4 MB. Figure 6c describes the access scheme of the MEMB block.

The four SBMUs are the main computing engines of the RMF-ME circuit. The SBMU performs block matching error computations for a block of 16x16, 8x8 or 4x4 pixels based on an array of 4 processing elements (PE). Each PE unit calculates the SAD value of a block of 4x4 pixels before shifting it out to an adder tree where the SAD values are accumulated. The SAD value generated for the current candidate block is compared to the value generated for the previous candidate block and the one with smaller value is selected. This process is continued until the SAD value for the last candidate block has been calculated. The block diagram of the SBME architecture is depicted in Fig. 7.

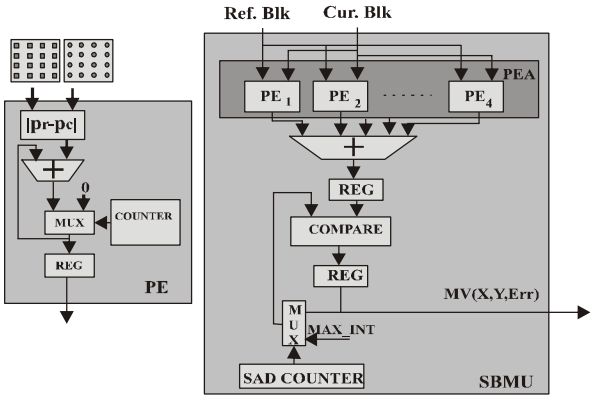

Fig. 7. Architecture of the SBMU building block

The ME-CU is the central control of the RMF-ME. It takes the configuration parameters and sends out control signals to the IBUs, MUXes, the SBMUs, and the input frames. The control signals sent to SBMUs include the total number of candidate blocks to be computed for each MV. The size of the candidate block and the total number of pyramid levels are also included in the set of control signals sent to the SBMUs from the ME-CU. The control signals sent to the input frames consist of the location of the candidate blocks and the decimation parameters of each level. The ME-CU routes the input pixels to the IBUs based on the configured number of blocks N_B and the number of frames N_F. Similarly, it routes the pixels of the candidate block and the current block to the SBMUs based on the values of N_B and N_F.

The RMF-ME can be configured to compute ME based on one single reference frame, one candidate block at a time. It also supports configuration for parallel processing of up to 4 candidate blocks of a single reference system as shown in Fig. 2b-2d. These configurations enable the system to shift out MV data at $1/2$, $1/3$, and $1/4$ the number of clock cycles compared to the system in Fig. 2a. The RMF-ME supports multiple RFs configurations as described in Fig. 2e-h. The 2, 3, and 4 RFs configurations depicted in Fig. 2e, Fig. 2g, and Fig. 2h enable the RMF-ME to process 2, 3, and 4 candidate blocks, one for each RF, in parallel. The 2 RFs configuration depicted in Fig. 2f enables the RMF-ME to process 4 candidate blocks, 2 for each RF, simultaneously.

3.2 The MCFI Design

The MCFI design consists of two main building blocks: The interpolation block (IP) and the hole filling (HF) block. The IP block takes the motion data contained in the MF_f and the MF_b motion fields to generate the interpolated frames from the two input frames. The HF block combines the two interpolated frames and then performs the filling of holes to construct the pixels located in the hole areas of the combined frame. The top level of the MCFI module is depicted in Fig. 8. The input frames, interpolated frames and the combined frame are assumed to be located on off-chip memory.

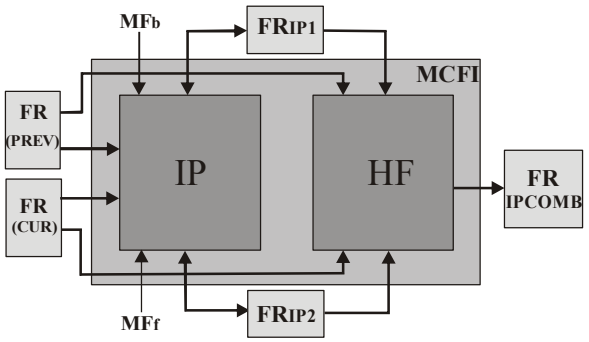

Fig. 8. Top-level block diagram of the MCFI architecture

The IP architecture consists of a bank of IEWMC processors that are the main computing engine of the interpolation process. The IEWMC processors perform weighted motion compensation on blocks of pixels extracted from the previous and current frames based on the motion data contained in the MF_f or MF_b motion field. The motion data contained in the MF_f (MF_b) motion field represent the motion estimated for blocks of pixels on the current (previous) frame with respect to the previous (current) frame. The IEWMC processors generate two intermediate frames FR_{IP1} and FR_{IP2} that are located at the same time instance as the frame to be interpolated as shown in Fig. 3.

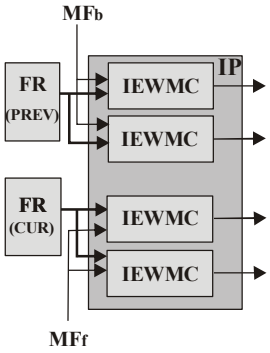

Fig. 9. Top level block diagram of the IP design

The IP design presented in this paper consists of four IEWMC processors that perform the

interpolation for four interpolated blocks of pixels in parallel. Each pair of IEWMC processors generates two interpolated blocks of pixels for the interpolated frames FR_{IP1} and FR_{IP2}, respectively. The top-level architecture of the IP building block is illustrated in Fig. 9.

The IEWMC processor performs overlap block motion compensation based on the IEWMC algorithm [6] and its detailed architecture is shown in Fig. 10.

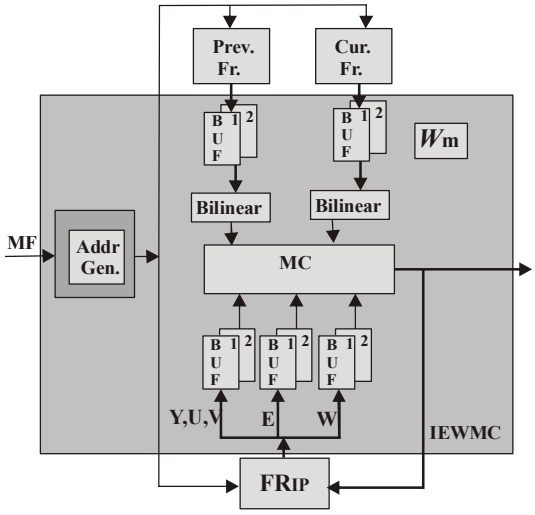

Fig. 10. The IEWMC processor architecture

In Fig. 10, the weight matrix $[W_m]$ used in the IEWMC process was predetermined using a bilinear function and stored in on-chip memory. The size of the $[W_m]$ matrix is equal to the size of the expanded block of pixels, which is 12x12 pixels if the block size is 4x4 or 16x16 pixels if the block size is 8x8. The blocks of pixels read out of the previous, the current, and the interpolated frames are stored in a set of ping-pong buffers. The ping-pong buffering of the input data enables faster computation of interpolated data. The pixels of the previous and the current frames shifted out from the ping-pong buffers are fed to the bilinear modules where the bilinear coefficients are generated. The motion compensation (MC) block performs pixels interpolation based on the data of the interpolated frame shifted in from the ping-pong buffers and the bilinear coefficients. Depicted in Fig. 10, E and W represent the values of the weighted motion compensation differences and the weight accumulated from the IEWMC computations for each block of pixels, respectively. These values are used in the normalization process to calculate the interpolated picture and the weight average of the motion compensation differences (WAMCD). The interpolated picture and the normalized WAMCD values are then shifted to the HF block to be used in the hole filling process.

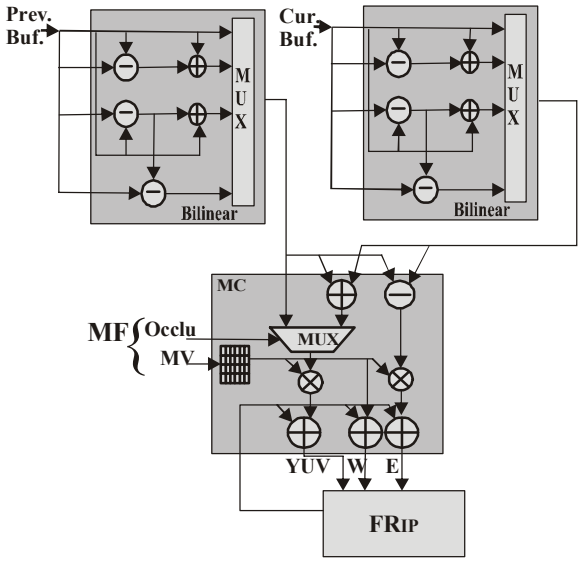

Fig.11. The bilinear and the MC circuits

The MC block is the heart of the IEWMC processor and its detailed architecture as well as the architecture of the bilinear blocks is described in Fig. 11. The bilinear unit computes the bilinear coefficients based on blocks of 2x2 pixels which are the closest to the pixel to be interpolated. If the location of the pixel to be interpolated is located in an occluded area, the generated bilinear coefficient based on a block of pixels from the current frame is selected. Otherwise, the values shifted out of the two bilinear blocks are combined to produce the interpolated pixel instead. The motion compensation difference is calculated as the difference between two coefficients shifted out of the two bilinear units. The interpolated pixel and the motion compensation difference are weighted separately by a weighting factor to generate the weighted pixel YUV and the weighted motion compensation difference E. The YUV pixel is then accumulated with the pixel shifted into the MC from the interpolated frame. Similarly, E is accumulated with the previously calculated motion compensation difference and W is accumulated with the weight values shifted into the MC from the interpolated frame.

The two interpolated frames that have been generated based on the motion fields MF_b and MF_f are then shifted to the HF block to be combined into a single frame. The pixels in the hole areas of the combined frame that have been missed by the IEWMC process are then constructed. The HF architecture consists of the combine, the gradient (GRAD), and the directional hole

interpolation (DHI) building blocks is depicted in Fig. 12.

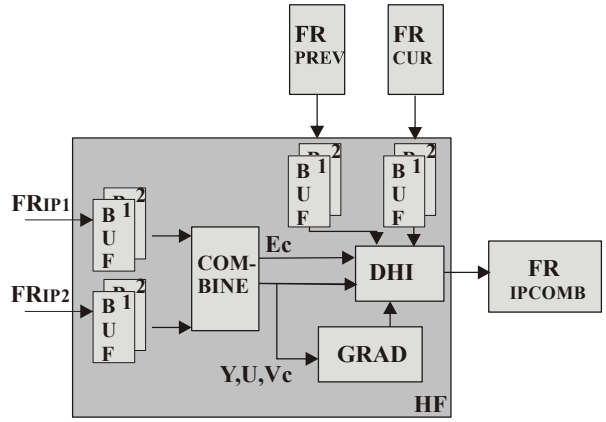

Fig.12.The HF building block

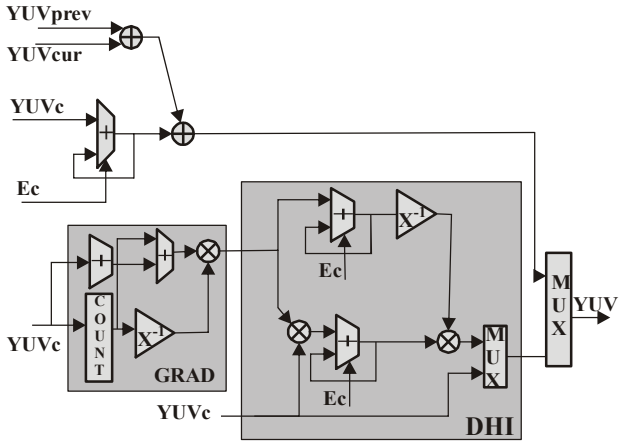

Fig.13. The architecture of the GRAD and the DHI circuits

The combination process combines two normalized pictures and two WAMCD values to generate the combined frame YUVc and the weight average motion compensation differences Ec, respectively. The GRAD block performs the computation to determine the orientation of the edge and texture of the holes. The linear interpolation process is then performed by the DHI block to compute the pixel value for each hole based on the computed orientation. The detailed architectures of the GRAD and the DHI building blocks are shown in Fig. 13.

4 Circuit Implementations Results

In this section, hardware implementation results of the RMF-ME and the MCFI designs are illustrated. Throughput performances of the RMF-ME design are then used to compare to other ME designs. For circuit implementations of the RMF-ME and the MCFI designs, two's complement number representation and fixed-point arithmetic have been used throughout. Hardware implementations for the RMF-ME and MCFI designs have been carried out using Xilinx's XC6VSX315-3 FPGA device.

4.1 RMF-ME Circuit Implementation

For the RMF-ME design, the luminance and chrominance components of the input pixels are shifted into the circuit via a 24-bit data bus. The pixels from all input frames are shifted into the RMF-ME circuit in parallel. The intermediate result of the SAD computation is represented by a 25-bit data bus. The MV data and the block matching errors are shifted out of the RMF-ME via the MV buses. The MV and the block matching error are represented by a 12-bit and 20-bit data bus, respectively.

Table 2
RMF-ME CIRCUIT IMPLEMENTATION RESULTS

Logic Resources/ Clock Rates	RMF-ME Used/Available
Slice Registers	16896 / 393600 (4%)
Slice LUT	54416 / 196800 (27%)
Block RAM	640 / 704 (90%)
I/Os	626 / 720 (86 %)
Clock Rate (MHz)	91

The FPGA implementation results illustrated in Table 2 show that the RMF-ME operates at a clock rate of 91 MHz. The performance of the RMF-ME circuit configured for different video formats based on a 91 MHz operating frequency is illustrated in Table 3.

In Table 3, the performance of the RMF-ME for CIF video has been calculated based on the configured search distance of ±16. The performances for the 720P and 1080P video have been calculated based on the search distances of ±64. Two levels of the pyramid L_1, L_2, and two iteration levels L_{-1} and L_{-2} have been configured for the CIF video. The search distances for the four levels L_2, L_1, L_{-1} and L_{-2} are 6, 2, 1, and 0, respectively. Four levels of pyramid, L_1, L_2, L_3, L_4 and one iteration level L_{-1} have been configured to compute for the search distance of ±64 for the 720P and 1080P video frames. For this configuration, the search distances for levels L_4, L_3, L_2, L_1 and L_{-1} are 7, 1, 1, 2, and 0, respectively. Thus, the total number of clock cycles needed to compute the CIF picture at one frame per second (fps) based on the configuration depicted in

Fig. 2a is 210K clock cycles. A total of 105K, 70K and 52.5K clock cycles are required to support CIF video at 1 fps based on the configurations depicted in Fig. 2b, Fig. 2c, and Fig. 2d, respectively. The number of clock cycles required to compute one frame of 720P and 1080P video based on the configuration depicted in Fig. 2a and a search distance of ±64 is 1304K and 2942K, respectively. Frame rate performances and the average number of clock cycles required to process a single 16×16 MB for 1, 2, and 4 RFs configurations are illustrated in Table 3.

Table 3
PERFORMANCE OF RMF-ME CIRCUIT FOR DIFFERENT VIDEO FORMATS AND CONFIGURATIONS

Video Formats (Search Dist.)	Frame Rates (fps)/Average Clock. Cycles per MB		
	No. RFs = 1 (Fig. 2d)	No. RFs = 2 (Fig. 2f)	No. RFs = 4 (Fig. 2h)
CIF (±16)	1733 / 159	866 / 318	433 / 636
720P (±64)	280 / 90	140 / 180	70 / 361
1080P (±64)	123 / 90	62 / 180	31 / 361

The performance of the RMF-ME circuit configured to support single RF compared to several existing RME circuits is presented in Table 4. The circuit in [28] is a 3-level hierarchical based design. The circuit in [27] is a full search based design whereas the architecture in [21] has been designed based on the 1BT technique. The RMF-ME circuit configured to compute 4 candidate blocks in parallel as depicted in Fig. 2d has been used for this comparison. At the configured search range of ±64, the RMF-ME performs at higher throughput compared to other circuits. In term of hardware usage, the RMF-ME requires more resources than the designs presented in [21] and [28]. This is due to the fact that the RMF-ME design uses all three colour components of the input pixels in the BM process. Even though the RMF-ME design requires more logic resource for circuit implementation, it offers better video quality than the 1BT design as illustrated in Table 1. There is no PSNR performance reported by the architecture presented in [27]. The authors in [28] did not provide any PSRN data but chose to present the performance of their algorithm based on the mean absolute difference instead. Motion vector estimated with the criteria of minimizing the motion compensation difference might be very different from the true motion vector [37]. As a result, this algorithm may not be suitable for FRUC applications where dense MFs and accurate motion trajectories are required.

The use of newer FPGA technology for circuit implementation contributes to the RMF-ME high throughput performance compared to other FPGA-based designs as shown in Table 4. However, a combination of the advanced HBM algorithm and the highly parallel architecture used for circuit implementation has also contributed to the high throughput performance of the RMF-ME design. As illustrated in Table 4, the number of clock cycles required to process one single 16×16 MB by the RMF-ME design is lower than the FPGA-based designs in [21] and [28].

Table 4
PERFORMANCE COMPARISONS WITH OTHER SINGLE FRAME ARCHITECTURES

Single Frame	[28]	[21]	[27]	RMF-ME
Tech.	FPGA XC3S1500-5	FPGA XC2VP30-7	ASIC	FPGA XC6VSX 315T-3
Search Range	±48,±24	±16	±16	±64
No. of PEs	256	1024	528	256
Block sizes	16×16	16×16	All	16×16 8×8 4×4
Pixel Res.	8-bit	1-bit	8-bit	24-bit
720P (fps) / Av. Clock. Cycles per MB		189 / 282	30 / 1459	280 / 90
1080P (fps) / Av. Clock. Cycles per MB	25 / 633	84 / 282		123 / 90

Performance comparisons of the RMF-ME design with several existing RME circuits for multiple RF configurations are presented in Table 5. The circuit in [22] has been designed based on a subsample-by-four over a search range of ±64 for 5 RFs. The results of the design presented in [22] show only hardware requirements and throughput performance of the SAD computing unit. No information was provided for the implementation of the line buffers and the off-chip memory bandwidth usage, which is a performance bottleneck in ME circuit designs. The search ranges of the 1BT based design and the design in [23] are set at ±16 and ±64, respectively. The 1BT design used 4 RFs whereas the design in [23] uses 2 RFs for circuit implementations. The RMF-ME has been configured to compute for search distances of ±64 for the 720P and 1080P video. The performances of the RMF-ME circuit configured for 2 and 4 RFs have been illustrated for this comparison. The results presented in Table 5 show that the proposed design configured to support 2 RFs requires smaller number of clock cycles to compute for a 16×16 MB compared to the designs presented in [21-23]. The design in [21] with 4096 PEs supports higher frame rates compared to the RMF-ME circuit with 256 PEs. However, the search range supported by [21] is ±16 compared to the

search range of ±64 supported by the RMF-ME circuit.

Table 5
PERFORMANCE COMPARISONS WITH OTHER MULTI-FRAME ARCHITECTURES

Multiple Frames	[21]	[22]	[23]	RMF-ME	
Technology	FPGA	ASIC	ASIC	FPGA	
Search Range	±16	±64	±64	±64	
No. of PEs	4096	64	4096	256	
Block sizes	16×16	All	All	16×16 8×8 and 4×4	
Pixel Resolution	1-bit	8-bit	8-bit	24-bit	
Num. Ref. Frames	4	5	2	2	4
720P (fps) / Av. Clock Cycles per 16×16 MB	189/282	84/512		140/180	70/361
1080P (fps) / Av. Clock Cycles per 16×16 MB	84/282	118/512	30/512	62/180	31/361

Implementation results shown in Table 4 and Table 5 demonstrate that FPGA-based MEs are suitable for HD video applications. These performance results show that FPGA-based ME circuits perform at comparable frame rates compared to the ASIC-based designs. Among the FPGA-based designs illustrated in Table 5, the RMF-ME architecture uses more logic resources for circuit implementation. The higher level of hardware usage required by the proposed architecture is due to the fact that the RMF-ME circuit uses 24-bit pixels for BM computation. The 1BT design [21] computes BM based on 1-bit pixels whereas the ASIC-based designs [22-23] use 8-bit pixels for BM computation. The trade-offs for higher hardware cost is that the RMF-ME offers higher video quality. Furthermore, the hardware usage reported in [21] did not include the logic and memory resources required by the binary conversion of the input frames. This one-bit conversion process involved the filtering of the input frames by a 17×17 multiband-pass filter kernel. The 1-bit frame is then determined based on a comparison between the filtered frame and the original frame [38]. If logic resources used for the filtering of the input pixels and extra memory used to store the filtered and the 1-bit frames were included, the overall hardware requirement for the 1BT design would increase significantly. Moreover, the 1BT design has been implemented based on full search algorithm that supports a search range of ±16. The BM computational load of the 1BT design would increase significantly for a search range larger than ±16. For applications that require a large search range, the high computational load will make the 1BT more costly in terms of hardware usage. High level of hardware usage can also affect the circuit throughput performance, which makes it less suitable for real-time video applications.

4.2 MCFI Circuit Implementations

The Xilinx's XC6VSX315-3 device has also been used for circuit implementation of the MCFI design. For the MCFI circuit implementation, the data bus for the input pixels of the input frames is 24-bit wide. The input/output data bus of the interpolated and the combined frames is 48-bit wide. The coefficients of the weight matrix have been predetermined and stored as a look up table using on-chip distributed RAM (DRAM). A look up table containing the pre-computed values of the 1/x division has been used to implement the divisions required by the normalization and the hole filling process. This look up table is also stored in the blocks of DRAM available on the FPGA. The use of a look up table for division computations helps to reduce required hardware resources, and eases the throughput bottlenecks. The ping-pong buffers on the IP and the HF designs have been implemented using on-chip BRAM to speed up the interpolation computations.

The implementation results for the IP and the HF circuits are listed in Table 6 and Table 7, respectively. The IP circuit shows an operating frequency of 166 MHz and the maximum operating frequency supported by the HF circuit is 84 MHz. Since one motion vector represents the motion estimated of an 8x8 block of pixels for HD picture, a total of 135x240 vectors is required to support the 1080P video. Inside the IEWMC processor, each block of pixels is expanded from 8x8 to 16x16. Thus, a total of 256 clock cycles are required to compute the weighted motion compensation for each 8x8 block of pixels. To support an up conversion of 1:2 rate at 30 frames per second (fps) for 1080P picture; the IEWMC needs to compute a total of $\left(\left(\frac{1080}{8}\right)*\left(\frac{1920}{8}\right)*256*30\right)$ clock cycles or 249 MHz. Since the IP circuit generates two pairs of interpolated pixels simultaneously, the required frequency for the 1080P pictures is 125 MHz. Thus, at 166 MHz operating frequency, the IP circuit supports the frame rate required by the 1080P video at 30 fps.

Table 6
IP CIRCUIT IMPLEMENTATION RESULTS

Logic Resources/ Clock Rates	RMF-ME Used/Available
Slice Registers	4495 / 393600 (1%)
Slice LUT	16538 / 196800 (8%)
Block RAM	130 / 704 (18%)
I/Os	658 / 720 (91%)
Clock Rate (MHz)	166

Table 7
HF CIRCUIT IMPLEMENTATION RESULTS

Logic Resources/ Clock Rates	RMF-ME Used/Available
Slice Registers	2300/ 393600 (1%)
Slice LUT	10099/ 196800 (5%)
Block RAM	74/ 704 (10%)
I/Os	309 / 720 (42%)
Clock Rate (MHz)	84

To support 1080P video at 30fps, the HF circuit needs to perform a total of 1080x1920x30 clock cycles or 63 MHz. With the operating frequency of 84 MHz, the HF circuit supports the 1080P video at the required frame rate.

As illustrated in Table 2, Table 6 and Table 7, the XC6VSX315-3 device has enough registers and LUT resources to hold complete MC-FRUC design. However, the requirements of on-chip memory and I/Os for circuit implementation of the MC-FRUC design have exceeded the device capacity. Therefore, the RMF-ME, the IP and the HF building blocks have been implemented on FPGA separately. Implementation results of these building blocks demonstrate the MC-FRUC circuit ability to support high-resolution pictures at high frame rate.

5 Conclusion

In this work, a highly parallel and flexible MC-FRUC hardware architecture has been designed and implemented on FPGA. Simulation results show that the MC-FRUC supports frame rate-up conversion for HD video where high quality pictures can be achieved. Due to the limit of on-chip memory and I/O resources, one FPGA is required for circuit implementation of each building block of the MC-FRUC design. To reduce cost and power consumption, the MC-FRUC can be implemented on ASIC instead of a multi-FPGA system. However, the high frame rates performance of the FPGA-based implementation demonstrates that the MC-FRUC circuit is also suitable for real-time HD video applications.

References:

[1] S. J. Kang, K. R. Cho, and Y. H. Kim, "Motion compensated frame rate up-conversion using extended bilateral motion estimation," *IEEE Trans. Consumer Electronics*, Vol. 53, No. 4, Nov. 2007, pp. 1759–1767.

[2] B. D. Choi, J. W. Han, C. S. Kim, and S.J. Ko, "Frame rate up-conversion using perspective transform," *IEEE Trans. on Consumer Electronics*, Vol. 52, No. 3, Aug. 2006, pp. 975–982.

[3] Y. T. Yang, Y S. Tung, and J. L.Wu, "Quality enhancement of frame rate up-converted video by adaptive frame skip and reliable motion extraction," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 17, No. 12, Dec. 2007, pp. 1700–1713.

[4] A. M. Huang and T. Nguyen, "Correlation-based motion vector processing with adaptive interpolation scheme for motion-compensated frame interpolation," *IEEE Trans. on Image Processing*, Vol. 18, No. 4, Apr. 2009, pp. 740–752.

[5] D. Wang, L. Zhang, and A. Vincent, "Motion-compensated frame rate up-conversion – Part I: fast multi-frame motion estimation," *IEEE Trans. on Broadcasting*, Vol. 56, No. 2, June 2010, pp. 133-141.

[6] D. Wang, A. Vincent, P. Blanchfield, and R. Klepko, "Motion-compensated frame rate up-conversion – Part II: New Algorithm for Frame Interpolation," *IEEE Trans. on Broadcasting*, Vol. 56, No. 2, June 2010, pp. 142-149.

[7] R. Rodrigo, Z. Chen, and J. Samarabandu, "Energy Based Video Synthesis," *Proc. of the 5th WSEAS International Conference on Telecommunications and Informatics*, May 2006, pp. 198-202.

[8] J. W. van de Waerdt, S. Vassiliadis, E. B. Bellers, and J. G. Janssen, "Temporal Video Up-Conversion on a Next Generation Media-Processor," *Proc. of the 7th IASTED International Conference on Signal and Image Processing*, Aug. 2005, pp. 434-441.

[9] E. B. Bellers, J. G. Janssen, and M. Penners, "Efficient Architecture for Full HD 120 Hz Frame Rate Conversion," *Proc. of the 2008 International Conf. on Consumer Electronics ICCE'08*, Jan. 2008, pp. 1-2.

[10] O. Tasdizen and I. Hamzaoglu, "A Reconfigurable Frame Interpolation Hardware Architecture for High Definition Video," *Proc. of the 12th Euromicro Conference on Digital System Design / Architectures, Methods and Tools*, 2009, pp. 714-719.

[11] Y. L. Lee and T. Nguyen, "Method and Architecture Design for Motion Compensated Frame Interpolation in High-Definition Video Processing," *Proc. of the IEEE Int'l Symp. on Circuits and Systems ISCAS'09*, May 2009, pp. 1633-1636.

[12] Tensilica Inc, "Xtensa Microprocessor Overview Handbook for Xtensa V (T1050) Processor Cores," 2002.

[13] Integrated Device Technology. [Online]. Available: http://www.idt.com/go/HQVmotionSMART

[14] Imagination Technologies Ltd. [Online]. Available: http://www.imgtec.com/powervr/powervr-frc

[15] Zoran Corp. [Online]. Available: http://www.zoran.com/SupraFRC-301-High-Performance-100

[16] Trident Microsystems. [Online]. Available: http://www.tridentmicro.com/producttree/tv/dtv/frc/frc-94x9q

[17] Toshiba Inc. [Online]. Available: http://www.toshiba.com/taec/components/docs/ProdBrief/07H02_TC90240XBG_Frme_ProdB.pdf

[18] Teranex Systems Inc. [Online]. Available: http://www.teranex.com

[19] T-H Tsai, H-G Chen, and H-Y Lin, "Frame Rate Up-Conversion Using Adaptive Bilateral Motion Estimation," *Proc. of the 8th WSEAS International Conference on Applied Computer and Applied Computational Science ACACOS'09*, May 2009, pp. 199-202.

[20] Y. Su and M-T. Sun, "Fast Multiple Reference Frame Motion Estimation for H.264/AVC," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 16, No. 3, March 2006, pp. 447-452.

[21] A. Akin, G. Sayilar, and I. Hamzaoglu, "A reconfigurable hardware for one bit transform based multiple reference frame motion estimation," *Proc. of the Conference on Design, Automation and Test in Europe DATE'10*, March 2010, pp. 393-398.

[22] S. Warrington, S. Sudharsanan, and W-Y Chan, "Architecture for Multiple Reference Frame Variable Block Size Motion Estimation," *Proc. of the IEEE Int. Symp. on Circuits and Syst.*, May 2007, pp. 2894–2897.

[23] C-Y Kao and Y-L Lin, "A Memory-Efficient and Highly Parallel Architecture for Variable Block Size Integer Motion Estimation in H.264/AVC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 18, No. 6, June 2010, pp. 866-874.

[24] T. Moorthy, P. Chen, and A. Ye, "A Scalable Architecture for H.264/AVC Variable Block Size Motion Estimation on FPGAs," *WSEAS Transactions on Signal Processing*, Vol. 7, No. 1, January 2011, pp. 23-33.

[25] Huong Ho, "Design and Implementation of a Fast Multi-Frame Hierarchical Motion Estimation Circuit," *Proc. of the 29th IEEE International Conf. on Consumer Electronics ICCE'2011*, Jan. 2011, pp. 533-534.

[26] Y-N. Pan, D-Y. Shen, and T-H. Tsai, "Reconfigurable Motion Estimation Architecture Design on H.264/AVC for Power Aware Mobile Application," *Proc. of the Int'l SOC Design Conf. ISOCC'08*, Nov. 2008, pp. I246-I249.

[27] L. Lu, J. McCanny, and S. Sezer, "Reconfigurable Motion Estimation Architecture for Multi-standard Video Compression," *Proc. of the IEEE Int'l Conf. on App. Specific Systems, Architectures and Processors*, July 2007, pp. 253-259.

[28] O. Tasdizen, A. Akin, H. Kukner, and I. Hamzaoglu, "Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation," *IEEE Trans. on Consumer Electronics*, Vol. 55, No. 3, Aug. 2009, pp. 1645-1653.

[29] M. Ribeiro and L. Sousa, "A Run-time Reconfigurable Processor for Video Motion Estimation," *Proc. of the Int'l Conf. on Field Programmable Logic and Applications FPL 2007*, Aug. 2007, pp. 726-729.

[30] Konstantinos Babionitakis et al., "A real-time motion estimation FPGA architecture," *J. Real-Time Image Proc.*, Vol. 3, No. 1, March 2008, pp. 3-20.

[31] J. C. Tuan, T. S. Chang, and Chein-Wei Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 12, No. 1, January 2002, pp. 61-72.

[32] G. Dane and Truong, Q. Nguyen, "Optimal Temporal Interpolation Filter for Motion-Compensated Frame Rate Up Conversion," *IEEE Trans. on Image Pro*cessing, Vol. 15, No. 4, April 2006, pp. 978-991.

[33] Byeong-Doo Choi, Jong-Woo Han, Chang-Su Kim, and Sung-Jea Ko, "Motion-Compensated Frame Interpolation Using Bilateral Motion Estimation and Adaptive Overlapped Block Motion Compensation, " *IEEE Trans. on*

Circuits and Systems for Video Technology, Vol. 17, No. 4, April 2007, pp. 407-416.

[34] Suk-Ju Kang, Dong-Gon Yoo, Sung-Kyu Lee, and Young Hwan Kim, "Design and Implementation of Median Filter based Adaptive Motion Vector Smoothing for Motion Compensated Frame Rate Up-Conversion," *Proc. of the 13th IEEE International Symposium on Consumer Electronics ISCE'2009*, May 2009, pp. 745-748.

[35] John C-H. Ju, Yen-Kuang Chen, and S.Y. Kung, "A Fast Algorithm for Rate Optimized Motion Estimation," *Proc. of the International Symposium on Multimedia Information Processing*, Dec. 1997, pp. 472-477.

[36] M. Tun, K. Loo, and J. Cosmas, "Fast Motion Estimation using Semi-Hierarchical Approach for the Dirac Video Codec," *Proc. of the 8th WSEAS International Conference on Multimedia Systems and Signal Processing MUSP'08*, April 2008, pp. 273-279.

[37] Y-L Chan and W-C Siu, "An Efficient Search Strategy for Block Motion Estimation Using Image Features," *IEEE Trans. on Image Processing*, Vol. 10, No. 8, Aug. 2001, pp. 1223-1238.

[38] S. Ertürk, "Multiplication-Free One-Bit Transform for Low-Complexity Block-Based Motion Estimation*," IEEE Signal Processing Letters*, Vol. 14, No. 2, Feb. 2007, pp. 109–112.