Algorithm and Architecture Design of Multi-rate Frame Rate Up-conversion in QHD LCD System

Yung-Lin Huang, *Student Member, IEEE,* Fu-Chen Chen, and Shao-Yi Chien, *Member, IEEE*

***Abstract*—In current LCD systems, the frame size becomes larger even to Quad-HD (3840x2160) resolution, and the refresh rate becomes higher to 120Hz or more. However, available videos are usually at only 24FPS, 30FPS or 60FPS, which is much lower than the refresh rate of LCD. To fill the gap between videos and LCD systems, frame interpolation techniques are usually adopted. Although frame rate up-conversion is regarded as the most efficient method, many new design challenges are introduced in current high-resolution and high-frame-rate LCD systems. In this paper, we proposed a hardware-efficient multi-rate frame rate up-conversion technique to enhance the visual quality when converting Qual-HD video from 24FPS or 60FPS to 120FPS. Also, a hardware architecture for our proposed multi-rate frame rate up-conversion technique is proposed to support the current Qual-HD LCD systems. The experimental results show that our proposed techniques produce high visual quality video and have high hardware utilization.**

***Index Terms*—Frame rate up-conversion, motion blur reduction, motion estimation, motion compensation, Markov Random Field**

I. Introduction

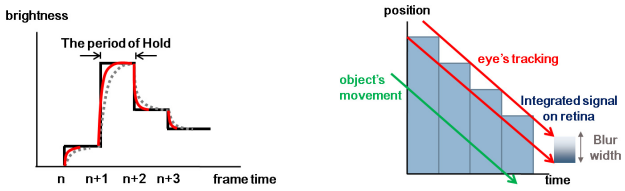
NOWADAYS, liquid crystal display (LCD), which is one of the important display techniques, is used widely for many applications. Also, the refresh rate of LCD systems is getting higher in order to enhance the visual quality. However, the video frame rate is still not as high as the growing LCD refresh rate.

Frame rate up-conversion (FRUC) is a technique that interpolates intermediate frames to increase video frame rate. For recent years, FRUC is applied on LCD systems for converting frame rate of input video stream to 120 frames per second (FPS) or higher in order to reduce the hold-type motion blur on LCD [1].

While the frame size of video systems becomes larger to Full-HD (1920x1080) or Quad-HD (3840x2160) resolution, many new design challenges are introduced. For example, huge computation, large bandwidth and large on-chip SRAM size requirements. Our goal is to develop a FRUC algorithm and architecture which fits with the current LCD system. In this paper, a multi-rate FRUC in Quad-HD LCD System is proposed. The capability of our design has 24FPS to 120FPS and 60FPS to 120FPS multi-rate up-conversion, and supports Quad-HD resolution for next LCD generation.

The rest of paper is organized as follows. First, the relation between motion blur on LCD and FRUC is shown in Sec. II, and then a general FRUC flow is given. Afterwards, the proposed algorithm and hardware architecture with performance evaluation are described in Sec. III and Sec. IV, respectively. Therefore, in Sec. V, the implementation results will be shown. Finally, a conclusion is given in Sec. VI.

Thanks...



(a) (b)

Fig. 1. (a) Hold-type display with slow response, (b) Direct evaluation of blur width.

II. Motion Blur on LCD and Frame Rate Up-conversion

A. Motion Blur on LCD

The visual quality of LCD suffers from motion blur due to the physical property of liquid crystal. In general, there are two types of motion blur occur on LCD [1].

The first type of motion blur is caused by the slow response of liquid crystal. As shown in Fig. 1(a), the black solid line is the targeting brightness and the dotted line is the actually displayed brightness. The smooth variation of brightness looks blurred by human eyes. To overcome this problem, a popular solution called overdrive is applied. That is, the voltage is first set higher (or lower) than the targeting brightness, and then set back to the ordinary value after the brightness is close to the target. The red solid line in Fig. 1(a) shows the resulting brightness, which reduce the smooth variation of brightness.

The second type is called hold-type motion blur. As shown in Fig. 1(a), the maintenance of brightness is called the period of hold which is equal to the inverse of frame rate. In Fig. 1(b), when human eyes track objects along their movement with velocity v, they integrate the intensity continuously, but the real intensity changes discretely. This divergence makes the integrated signal of the object's boundary on retina smoothly decrease (increase). The range of decreasing (increasing) is called blur width and can be directly expressed as

$$Blur\ width_{direct} = v/frame\ rate$$

Another way for evaluating hold-type motion blur is based on the sampling and reconstruction theory of integrated signal on retina [2]. In the case of idle display (without slow response), the blur width is equal to

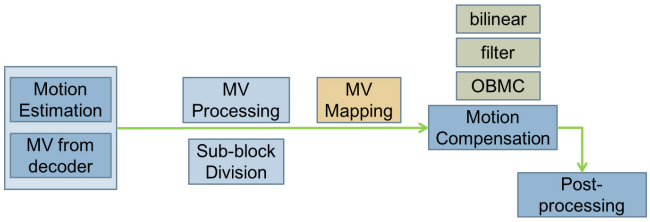


Fig. 2. A general FRUC flow.

$$Blur\ width_{theoretical} = 0.8 \times v/frame\ rate$$

Therefore, the blur width is inverse-proportional to the frame rate. Among the solutions of hold-type motion blur, up-converting the frame rate is regarded as the most efficient method since it can directly reduce the effect of motion blur without visual quality drop [1].

B. Frame Rate Up-conversion

For hold-type motion blur reduction on LCD, a FRUC algorithm and architecture which fits with the current LCD system is needed. Fig. 2 shows a general FRUC flow.

At first, the true motion vector fields (MVFs) between existing frames are required. To present more realistic and detail motion in the frames, the further motion vector (MV) processing may be operated. After the MVFs are retrieved, they are needed to be mapped from existing frames to targeting intermediate frames because of temporal mismatch. Afterwards, intermediate frames are interpolated according to mapped MVFs using motion compensation techniques such as overlapped block motion compensation (OBMC) [3]. Finally, the interpolated frames are post-processed to achieve better visual quality.

Discussion of related works at each part is listed as follows,

1) Motion Estimation: Unlike conventional motion estimation in video encoder, the purpose here is to find true motion presenting objects' movement [4], not just to reduce the residual energy of each block comparison. Many related works approximate the true MVFs using block-based motion estimation with spatial and temporal predictions [4] [5] [6]. However, the true MVF is hard to be estimated and the computational cost is usually high. On the other hand, it is possible to get MVF from video decoder [7] [8] [9] and then perform MV processing to optimize the rough MVFs. Nevertheless, the decoding information is not always available for FRUC in current LCD systems.

2) Motion Vector Processing: MV processing such as median filter [10] is often adopted because of the spatial and temporal coherence of true motion. Besides, more processing methods such as motion smoothing via global energy minimization [11] and 3D Markov Random Field modeling [12] are proposed to approximate the MVF to a true one. However, most of the algorithms are too expensive for hardware implementation and some of them are even heuristic.

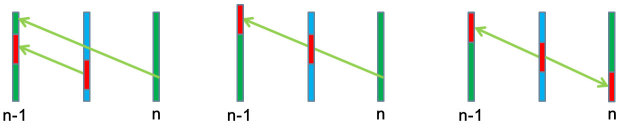


Fig. 3. Three general MV mapping methods: Tradition, forward and bilateral.

3) Motion Vector Mapping: To our knowledge, there are three MV mapping methods in general, and the illustration of the three methods are shown in Fig. 3. The first one is called traditional MV mapping method, which maps the block MV of existing frames to the corresponding blocks of intermediate frames. However, the two blocks exist at the same position but different time, so the MVs of them are not exactly the same. The second one is called forward MV mapping [13], which maps through the direction of MV to the block pointed by it. There is no temporal mismatch using forward MV mapping, but in this case some positions may be pointed by many MVs or no MVs, the problems of overlap and hole are introduced. The third one is called bilateral MV mapping, which performs motion estimation on intermediate frames [14] [15]. There are no problems of overlap and hole mentioned above, but it usually fails to find the true MVs at flat regions .

4) Motion Compensation: Since the motion compensation is usually block-based, it is an important issue to avoid block artifacts. Applying adaptive weighted-interpolation for occlusion handling is also proposed [16]. Besides, because motion compensation should be performed for each interpolated frame, the bandwidth consumption becomes a problem when the number of interpolated frames increases.

5) Post-Processing: The visual quality of interpolated frame might suffer due to the incorrect MVs, wrong interpolation and so on, so it is a choice to refine the interpolated frame via post-processing [17] [16]. Nevertheless, it is hard to determine where the artifacts are and how to interpolate with better visual quality.

III. Proposed Multi-rate Frame Rate Up-conversion Algorithm

At first, a low-complexity true motion estimation algorithm is proposed. For hardware complexity consideration, the block size is set as 32x32, the matching criterion is 8x8 multilevel successive elimination algorithm (MSEA) [18], and the search range is set as $\pm$128x$\pm$128 for Full-HD videos. Furthermore, MV correction based on Markov Random Field (MRF) modeling is performed with our proposed simplified iterated conditional mode (ICM) minimization.

After true motion estimation, we propose a block-based forward MV mapping technique that determines MVFs of intermediate frames with both the benefits of forward MV mapping and bilateral MV mapping. Finally, the blocks detected by our proposed artifact detection method are divided into sub-blocks. For those sub-blocks with artifacts, new MVs are bilateral searched and re-interpolated with consideration of occlusion. The experiments show that the proposed algorithm performs well in both subjective and objective evaluation.

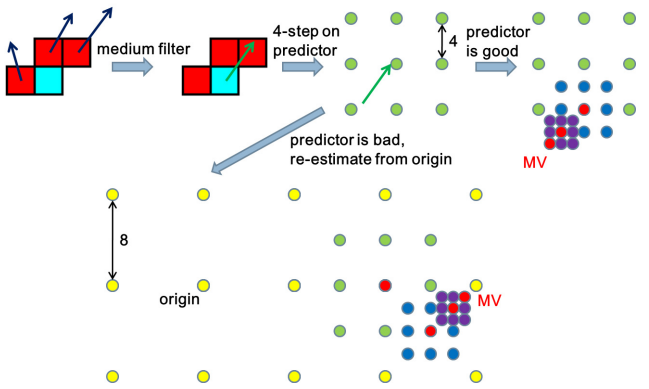


Fig. 4. A graphic illustration of our proposed predictive square search algorithm.

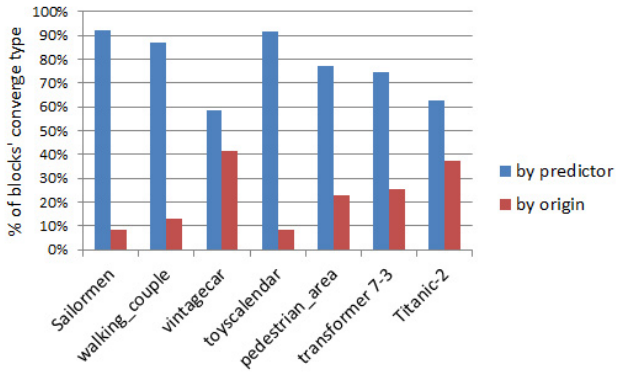


Fig. 5. Percentage of blocks' converge type at the worst cases of each Full-HD sequence.

A. Predictive Square Search Motion Estimation

This motion estimation algorithm is very similar to a hybrid search algorithm with four step search (4SS) [19] and three step search (3SS) [20]. A graphic illustration of our proposed predictive square search algorithm is demonstrated in Fig. 4.

First, the median of three neighboring (left, up and upper-right) MVs is calculated as a predictor. Afterwards, a 4-step square search pattern centering on the predictor is employed. If the minimum distortion appears at the center or its value is smaller than the threshold, the predictor will be regarded as good and proceed to apply 2-step and 1-step square patterns for converge, like 3SS. Otherwise, we go back to the origin and search MV like 4SS but with an 8-step square pattern. If the minimum distortion found at the center of a 8-step square pattern, 4-step, 2-step and 1-step square patterns are employed for converge.

The ability to reject predictor and re-estimate from origin can prevent wrong motion estimation due to wrong predictors. Besides, the proposed algorithm is cost efficient because of its early convergence. Fig. 5 shows the percentage of blocks' converging type at the worst cases of each sequence. For the most complex sequence "vintagecar", there are still up to 60% blocks converging around predictor.

The chosen matching criterion for block-matching is 8x8 MSEA. The 8x8 MSEA, which is usually calculated for fast full search at previous works, can be regarded as the down-sampled version of sum of absolute difference (SAD). However, we found that employing MSEA with square patterns reduces lots of computation and bandwidth cost in hardware design, especially when the step size matches the sub-block size.

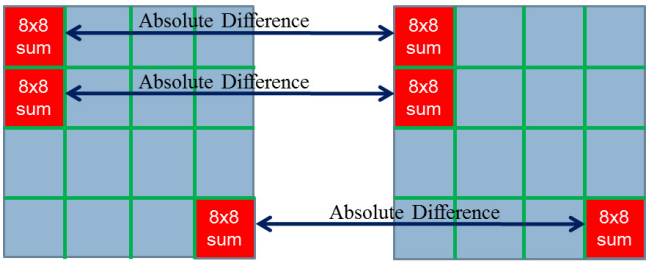


Fig. 6. Illustration of 8x8 MSEA.

The computation of 8x8 MSEA is shown in Fig. 6. Each 32x32 block is first divided into 16 8x8 sub-blocks, and each sub-block is sumed up. Therefore, the 16 absolute differences of the summed up sub-block pairs are accumulated to derive the 8x8 MSEA.

B. Markov Random Field Motion Vector Correction

After motion estimation, the rough MVF is formed, but some MV outliers exist. In this step, the blocks are processed in raster scan order to refine the MVs based on MRF modeling. MRF is a theoretical modeling method based on Bayesian's framework, applied to computer vision algorithm such as optical flow or true motion estimation [21] for many years [22]. The global energy minimization is a NP-complete problem so many fast algorithms are proposed [23]. For example, one of the well-known algorithms called belief propagation [24] is widely adopted, but the related hardware design costs extremely high [25].

Among the energy minimization methods, we choose the simplest ICM with selected candidates. Only nine MVs adjacent to the processing block is chosen to be candidates instead of all 65,536 MVs in the $\pm$128x$\pm$128 search range because there is a very high probability for a block to find its true motion from nearby blocks' MVs [26]. As well, choosing only neighboring nine candidates can prevent over-smoothing, and the complexity is lower. For a block, eight neighboring MVs and itself MV are chosen as nine new MV candidates. Thus, the corresponding MRF energy for each candidate is calculated as follows,

$$MSEA_{8\times8} + weight \times \sum_{\forall neighbor} |MV_{candidate} - MV_{neighbor}|$$

Finally, the one with the smallest MRF energy is selected from these nine candidates as the new MV of this block as follows,

$$New\ MV = \underset{MV_{candidate}}{arg\ min}\ Energy_{candidate}$$

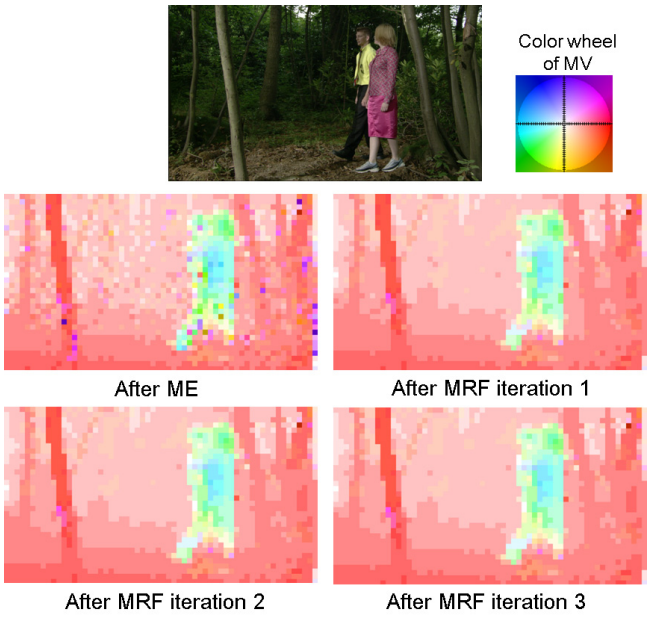


Fig. 7. Visualization of MVF after ME and MV correction.

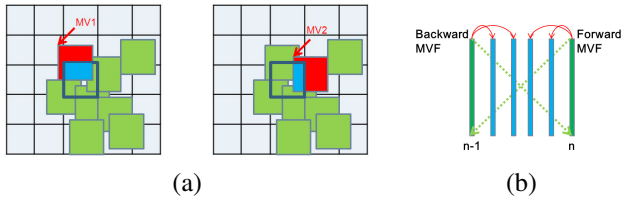


Fig. 8. (a) Proposed MV mapping: , (b) Proposed MC: .

Fig. 9. Labeled 16x16 sub-blocks in intermediate frame.

The weight is set as 48 in our design. Besides, running ICM for three iterations is enough to remove most MV outliers. Fig. 7 shows the visualization of MVF after ME and MV correction. The color of visualization presents the direction of MVs, and the intensity presents the magnitude of MVs [27]. The shape of objects in the frame are roughly formed in the MVF after ME, and the MV outliers are corrected through iterations.

C. Block-based Forward Motion Compensation

To avoid temporal mismatch, the forward MV mapping is operated to project block-based MV from existing frame to intermediate frame, as the green block shown in Fig. 8(a). However, it introduces overlap and hole problems as mentioned above. Here we divide intermediate frames into intermediate blocks with 32x32 block size, and then we can calculate how much one block-based MV contributes to one intermediate block, as the blue part of the block shown in Fig. 8(a).

In [16], the MV of intermediate block is calculated using the weighted sum of block-based MVs projected on it, and the weighting is equal to the overlapped area between intermediate block and each projected MV. Unfortunately, the MVF of intermediate frames may be over-smoothed using weighted sum operation. To prevent over-smoothing, we accumulate the total overlapped area of each projected MV and find the MV with the maximum overlapped area. If the maximum overlapped area is bigger than a half of block size, the MV with the maximum overlapped area is assigned to the intermediate block. If not, the MV of corresponding block at the same position on existing frames is assigned.

After all MVs in the intermediate frame are determined, the frame will be interpolated using block-based motion compensation. Thus, the block-based operation is hardware-friendly, and there are no temporal mismatch, overlap and hole problems.

To achieve multi-rate up-conversion, motion estimation is performed twice to retrieve forward and backward MVFs as the green dotted arrows shown in Fig. 8(b). Moreover, uni-directional interpolation is adopted to prevent blur and reduce complexity. In other words, the first and the second intermediate frames are interpolated using the pixel in frame n-1 with mapped backward MVF. Similarly, the third and the fourth intermediate frames are interpolated using the pixel in frame n with mapped forward MVF.

D. Sub-block Division with Artifact Detection

We observe that artifacts always appear in the interpolated frame when the MVs of neighboring blocks are discontinuous. Therefore, the sub-block division with artifact detection is operated after motion compensation. First, the boundary of MV discontinuity is determined using a block matching criterion. We use bilateral MSEA (bi-MSEA) which calculate 8x8 MSEA of two blocks pointed by forward and backward MV as matching criterion. Thus, blocks with higher bi-MSEA values on the boundary of MV discontinuity are detected and then divided into sub-blocks.

One result of sub-block division with artifact detection is shown in Fig. 9. Obviously, most of the labeled sub-blocks are located on the boundary of moving objects since there exists MV discontinuity.

The number of labeled sub-blocks is calculated for simulation, and the results in the frame with highest number of sub-blocks are listed in Table I. There are about 12% sub-blocks labeled for the following post-processing at most. Consequently, sub-block division with artifact detection discovers blocks needing to be refined efficiently without much computational overhead.

TABLE I
TOTAL NUMBER OF LABELED SUB-BLOCKS IN THE FRAME WITH HIGHEST NUMBER IN EACH FULL-HD SEQUENCE

Sequences	Total number of sub-blocks	Percentage
pedestrian_area	2,787	8.54%
Titanic-2	1,820	5.58%
Vintagecar	3,074	9.42%
ducks_take_off	1,340	4.11%
park_joy	2,474	7.58%
Tractor	1,969	6.03%
transformer 7-3	3,947	12.09%

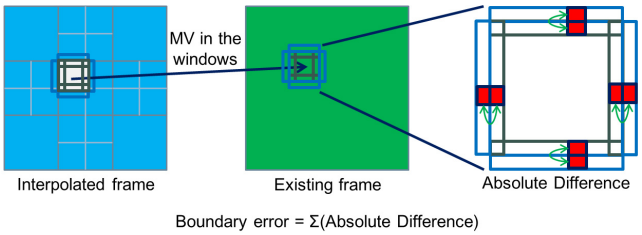


Fig. 10. MV refinement: Boundary error computation.

E. Sub-block Refinement with Bilateral Motion Vector Search and Overlapped Block Motion Compensation

To refine the sub-blocks with artifacts, corrected MVs and interpolation are required. Therefore, the side match technique [28] is employed as the matching criterion for bilateral MV search. For each labeled sub-block, two windows with range 8x8 around initial MV are searched. One is in the forward frame and another is in the backward frame using projection of forward and backward MV. One example is shown in Fig. 10. We compute the boundary error defined as the sum of absolute difference between each outside and inside pixel pairs on the sub-block boundary. Hence, the MV with the smallest boundary error is assigned to the sub-block.

After refined MVs of all the labeled sub-blocks are estimated, OBMC is performed on these sub-blocks as [7] [29]. This technique blurs the boundary of MV discontinuity and reduces the blocky artifacts. Fig. 11 shows some regions of interpolated frames before and after refinement. In brief, most of the labeled sub-blocks are corrected to be with better visual quality using proposed sub-block refinement and OBMC technique.

Fig. 11. Artifact reduction after sub-block refinement and OBMC.

Fig. 12. Full-HD sequences for experiment, from top-left to button-right: pedestrian_area, transformer 7-4, vintagecar, and titanic-2.

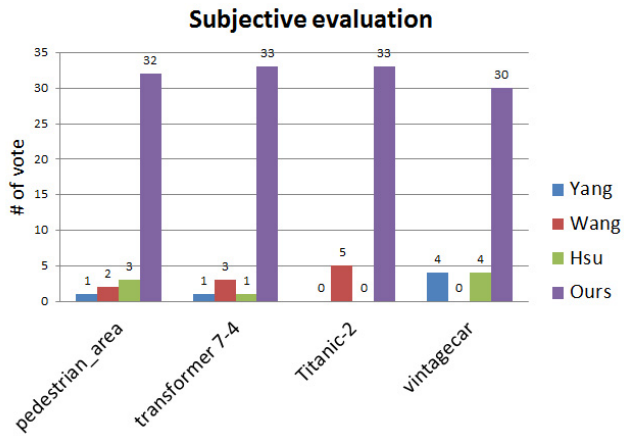


Fig. 13. Experimental results of subjective evaluation.

F. Performance Evaluation

Three related works are selected for performance evaluation. Yang [7] extracts MVF from H.264 decoding information (using reference software JM 15.1), performing OBME and OBMC. Wang [9] also extracts MVF from H.264 decoding information, ignoring MVs that are perceptually unapparent. Hsu [17] performs global motion estimation and sub-block refinement.

We choose several HD video sequences including sport game lives and movies for experiment. Due to the deficiency of Quad-HD video sequences, we choose Full-HD video sequences instead. Four Full-HD video sequences shown in Fig. 12 are used at the following evaluations.

1) Subjective Evaluation: There are thirty-eight people for subjective evaluation. Besides, twenty-two people are expert on image and video processing, and the others are not. The up-converted sequences generated from four different algorithms are displayed simultaneously. After seeing the sequence twice, people choose the best one among four sequences. The experimental results are shown in Fig. 13, and we can see that at least 79% people choose the sequence generated from our proposed algorithm as the best one.

2) Objective Evaluation: We compute the difference between original even frames and even frames interpolated by

odd frames using FRUC techniques. Peak signal-to-noise ratio (PSNR) is used in our evaluation. Although PSNR value can not totally represents perceptual quality, the frame with higher PSNR value still looks better at most time. Fig. 14 shows the results, and it indicates that our proposed algorithm outperforms other three algorithm at most frames.

IV. Hardware Architecture Design

There are three design challenges for the hardware architecture of the proposed multi-rate FRUC algorithm in Qual-HD video systems. The first challenge encountered is the requirement of large on-chip SRAM. For example, to support ±128x±128 search range, the on-chip SRAM arrangement will be significant. Second, since we interpolate multiple frames, the required bandwidth for motion estimation and motion compensation is huge. Third, to achieve 24FPS to 120FPS and 60FPS to 120FPS multi-rate up-conversion, the cycles for data fetching must be as less as possible. Consequently, to utilize the available hardware resource efficiently is very important.

To the best of our knowledge, there is often a specific DRAM belongs to the FRUC algorithm without sharing with other modules to satisfy the huge bandwidth requirement of it. In the proposed Quad-HD LCD systems, the path between FRUC and DRAM is assumed that there is fifty cycle latency with uncertainty. In addition, the clock frequency is assumed 300MHz and the I/O width of system bus is 16 bytes per cycle.

To achieve the target Quad-HD 120Hz video systems, we choose 4 pieces of 16bit x 1Gb DDR3-1333 DRAM, and each bank consists of 2 pieces. Pixels in DRAM are arranged as four successive pixels per address in two banks with raster scan order, thus it is possible to get 16x1 or 8x2 pixels at one time. Furthermore, for the selected DDR3 DRAM, the theoretical maximum bandwidth is equal to 1333MHz x 4pieces x 16bits/8bits = 10,666MB per second. Under the assumption of 65% probability of DRAM request failure, the real bandwidth available is 10,666MB x 35% = 3,733.3MB per second. For the operations on each frame in 24FPS to 120FPS up-conversion, the maximum bandwidth available is 3,733.3MB/24FPS = 15.6MB, and the maximum cycles available is 300MHz/24FPS = 12.5M cycles. Besides, in 60FPS to 120FPS up-conversion, the maximum bandwidth available is 3,733.3MB/60FPS = 62.2MB, and the maximum cycles available is 300MHz/60FPS = 5M cycles.

Fig. 15 shows the overview of the proposed architecture. The FRUC control is composed of control units of each procedure. Write request, read request and read receive are responsible for communicating with DRAM. For the uncertainty of bus latency, there is job queue for request pushing and popping. In addition, SRAM of our design is shared by all of the modules.

The flowchart is shown in Fig. 16 and the procedures of our proposed FRUC algorithm are performed frame by frame. Notice that the frames are down-sampled from 3840x2160 to 1920x1080, and all pixel comparisons are operated at the 1920x1080 scale such as 8x8 MSEA, MRF energy, bi-MSEA and boundary error.

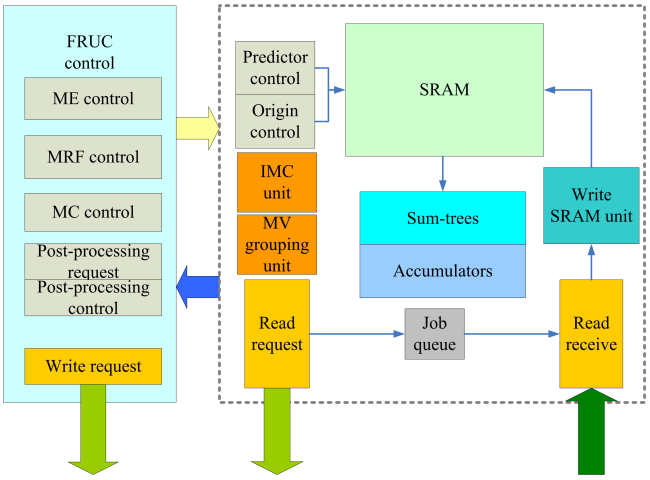


Fig. 15. The overview of the proposed architecture.

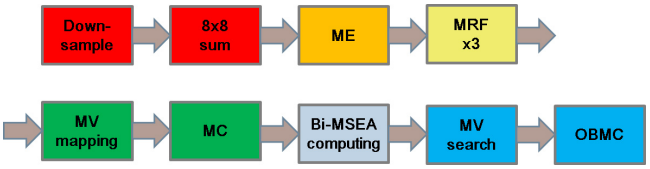


Fig. 16. Flowchart of the proposed FRUC algorithm.

A. Architecture of Motion Estimation

First, the on-chip SRAM size will be large in order to support ±128x±128 search range. Even if scheme C data reuse is adopted [30], the SRAM size is still (128 + 128 + 32) x (128 + 128 + 32) = 82,944 Bytes. Second, the required cycles for data fetching are (128 + 128 + 32) x 32 / 16 = 576 cycles per block. Third, the bandwidth consumption is (128 + 128 + 32) x 32 x 2,040 (total number of blocks) = 18.8MB for one frame in order to fetch data for the next block. All of the resource requirements above are too much to be feasible.

1) On-chip SRAM Reduction: The resource requirement can be greatly reduced by employing the characteristics of the proposed motion estimation algorithm [31]. That is, we only read the pixels needed for 4, 2, 1-step convergence and 8x8 sub-block summation values (SUM) for the 8-step square pattern with a huge amount of bandwidth and SRAM size reduction. A set of SRAM called M1 with range 8x8 is prepared to store the possible required pixels when performing 4, 2 and 1-step convergence in the algorithm. On the other hand, a set of SRAM called O1 is prepared for the 8x8 SUM with scheme C data reuse when performing 8-step re-estimation from origin with 8x8 MSEA criterions. Therefore, the bandwidth consumption is reduced to around 7.2MB for one frame for a worst-case video sequence. Fig. 17 shows the required size of M1 and O1 memory.

2) Ping-pong Two-way Scheduling: The data and operation dependencies will introduce pipeline bubbles to lower the hardware utilization with direct scheduling, as shown in Fig. 18. Thus, a ping-pong two-way scheduling for dependency elimination is proposed [31], where an additional SRAM pair of M2 and O2 like original M1 and O1 is

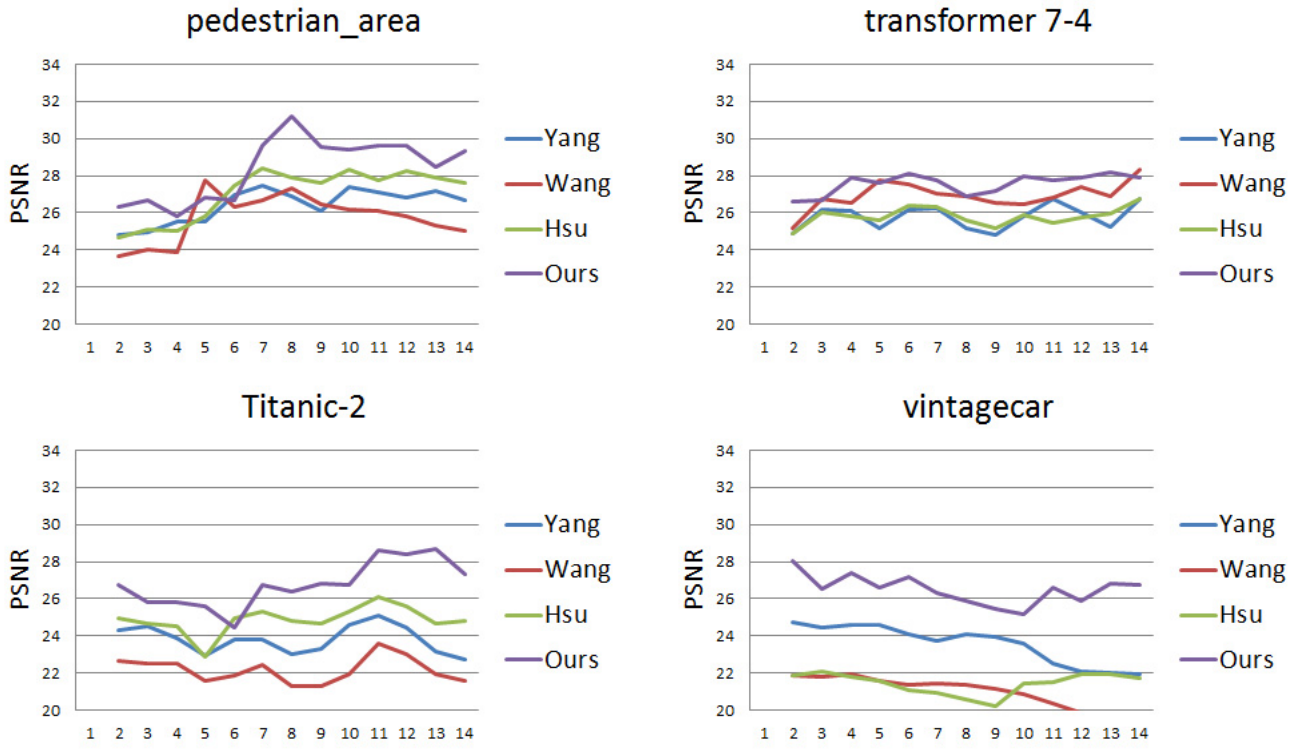


Fig. 14. Experimental results of PSNR evaluation.

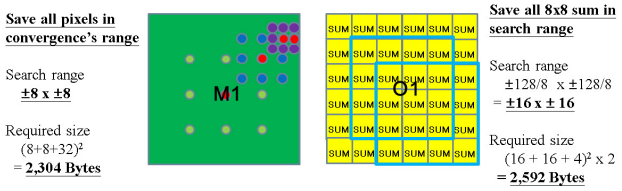


Fig. 17. SRAM usage for motion estimation.

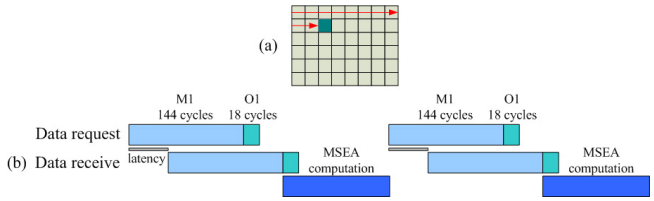


Fig. 18. Direct scheduling. (a) Raster scan order. (b) Pipeline bubbles.

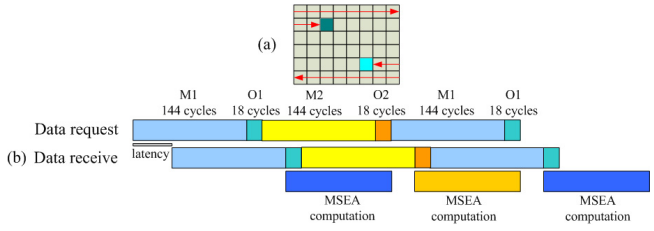


Fig. 19. Ping-pong two-way scheduling. (a) Two-way scan order. (b) Ping-pong usage without pipeline bubbles.

introduced for data interleaving. The ping-pong means that one of the pair is used for computing while the other is used for data pre-fetching. The two-way means one of the pair is running raster scan and the other is running inverse raster scan, as shown in Fig. 19(a). Consequently, the pipeline bubbles are eliminated with ping-pong data pre-fetching, as shown in Fig. 19(b).

The final synthesized SRAM size is 48 address x 128 bits x 3 banks x 2 = 4,608 Bytes for M1 and M2, and 84 address x 16 bits x 16 banks x 2 = 5,376 Bytes for O1 and O2. That is, the SRAM size is reduced from 82,944 Bytes to 4,608 + 5,376 = 9,984 Bytes, and the bandwidth is reduced from 18.8MB to 7.2MB for one frame. Also, for the best balance between data fetching and MSEA computing, the cycles consumed for 4, 2 and 1-step convergence should be about 18 + 144 = 162 cycles.

B. Architecture of Markov Random Field Motion Vector Correction

To compute MRF energy for a block, the 8x8 MSEA value of nine candidates and distance between nine candidate MVs are needed. The 8x8 MSEA value of each block is written out to DRAM after motion estimation, but the 8x8 MSEA values of the other eight candidates may not be the same and need to be calculated. However, if we fetch pixels candidate-by-candidate for these 8x8 MSEA computing, it will consume

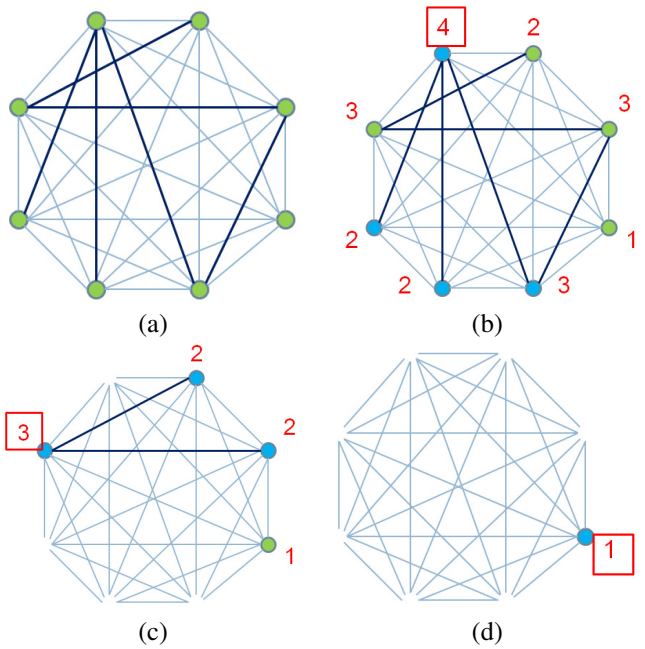


Fig. 20. An illustration of our proposed MV grouping algorithm. (a) 8 nodes with 28 edges. (b) Result of group 1. (c) Result of group 2. (d) Non-grouped node.

16MB per iteration and 512 cycles per block, which is not feasible for computing and memory bandwidth.

1) Motion Vector Grouping: As we observed, the neighboring MVs are similar and many required pixels are overlapped in the MVF generated by our proposed true motion estimation. Therefore, if we can group the candidate MVs and fetch all required pixels for 8x8 MSEA computing simultaneously, it reduces bandwidth consumption largely compared to fetching pixels candidate-by-candidate. As a result, a MV grouping technique using center MV for our proposed SRAM M1(M2) is proposed. That is, all the MVs with distances to the center MV smaller or equal to eight (search range of M1 and M2) can be calculated at the same time. Notice that the group size must be bigger than two for bandwidth gain, and there are at most two groups in our design.

An illustration of our proposed MV grouping algorithm is shown in Fig. 20. The eight candidates are regarded as eight nodes with twenty-eight edges for all possible connections between them. For each edge, the corresponding MV distance which is needed for MRF energy computation is calculated. Then, the edges with distance smaller or equal to eight are labeled. The node with maximum number of connected labeled edges is marked as the center MV of the first group, and the other connected nodes are regarded as the members of this group. Hence, the group is removed, and the next group is generated similarly. In the end, the nodes without being grouped are marked as non-group.

2) Architecture of Grouping: The proposed hardware architecture for MV grouping is shown in Fig. 21. At first, the MV distance is computed one-by-one, and the corresponding edge registers are labeled if the distance is smaller or equal to eight. Afterwards, they are accumulated into Total MV dis. Registers for MRF energy computing. The grouped MVs are pushed into

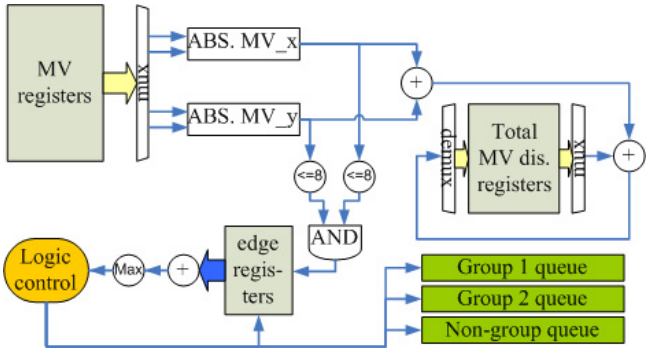


Fig. 21. Proposed architecture of MV grouping.

three types of queues after performing the proposed grouping algorithm. Besides, the proposed MV grouping algorithm and its architecture are also used for distance energy computation and median filter.

3) Markov Random Field Energy Computing: To compute MRF energy, we only have to compute 8x8 MSEA of each MV candidate because the MV distance is already computed during MV grouping. For the first MRF iteration, the 8x8 MSEA of each candidate is computed one by one using ping-pong two-way scheduling. Afterwards, each 8x8 MSEA is written out to DRAM for further re-use.

Table II shows the grouping results after the first MRF iteration. Note that the total number of blocks is 2,040, and the total number of MV candidates is 2,040 x 8 = 16,320. Take sequence "transformer 7-3" for example, the cycle consumption is (32 + 4) x Grouped + 64 x Non-grouped = (32 + 4) x (16,320 - 2,338) + 64 x 2,338 = 652,984 cycles, and then 652,984 / 2,040 = 320 cycles per block. Furthermore, the bandwidth consumption is 48 x 48 x (Number of Group1 + Number of Group2) + 32 x 32 x (Number of Non-grouped) = 48 x 48 x (1931 + 152) + 32 x 32 x 2338 = 4.8 MB + 2.4 MB = 7.2 MB.

At the second and third MRF iterations, nine computed 8x8 MSEA results at the first MRF iteration are loaded for re-use because the MVF changes little. However, fetching more pixels to compute 8x8 MSEA is needed for the candidates whose MV is not equal to one of the nine MV candidates at the first MRF iteration. In our simulation, the worst case takes 82 cycles and 1.1MB more for the second iteration, and 57 cycles and 0.3MB more for the third iteration.

Consequently, the cycle is reduced from 512 x 3 = 1,536 to 320 + 82 + 57 = 459 cycles per block for three MRF iterations in the worst case. Also, the bandwidth is reduced from 16MB x 3 = 48MB to 7.2MB + 1.1MB + 0.3MB = 8.6MB for three MRF iterations.

C. Architecture of Motion Compensation

In general, motion compensation is operated block by block on the intermediate frame. Nevertheless, for 24FPS to 120FPS up-conversion, it consumes bandwidth and cycles extremely when reading existing frames and writing intermediate frames. That is, the bandwidth consumption is (3840 x 2160 x 1.5) x

TABLE II
GROUPING RESULTS AFTER THE FIRST MRF ITERATION. AVERAGE NUMBER OF BLOCKS IN THE SEQUENCES IS SHOWN IN THE COLUMN "GROUP1" AND "GROUP2" WITH THE SIZE OF GROUP, AND AVERAGE NUMBER OF NON-GROUPED MV CANDIDATES IS SHOWN IN THE COLUMN "NON-GROUP".

Sequences	Group1						Group2		Non-grouped
	Size8	Size7	Size6	Size5	Size4	Size3	Size4	Size3	Size1
park_joy	1291	113	109	242	135	112	17	160	1912
ducks_take_off	1771	130	66	35	21	12	1	8	521
vintagecar	998	237	209	274	191	112	20	202	2267
tractor	1314	196	140	198	135	53	32	172	1269
pedestrian_area	1328	174	147	151	127	74	13	98	1761
transformer 7-3	1283	132	98	194	108	116	6	146	2338
Titanic-2	945	312	213	286	152	95	13	222	2268

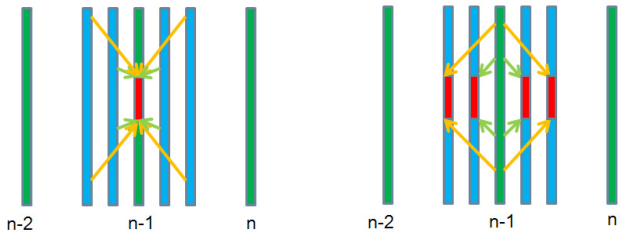


Fig. 22. The same architecture for overlapped region derivation. The left one is for inverse motion compensation, and the right one is for MV mapping.

(4 + 4) = 99.5MB, and the cycle consumption is 99.5M / 16 = 6.5M cycles.

1) Inverse Motion Compensation Scheduling: To achieve a reasonable bandwidth and cycle consumption, we proposed an inverse motion compensation that is operated block by block on the existing frame instead. As shown in Fig. 22, the existing frame is used to interpolate its four nearby intermediate frames.

At first, the pixels in one block of existing frame are read. After projecting the blocks of intermediate frame using their MVs to the existing frame, the overlapped regions between projected blocks and the block of existing frame are derived. Therefore, the pixels in the overlapped regions are written out and used to interpolate the intermediate frames. In this case, only one existing frame is read, and four intermediate frames are written out, which consumes the resources at the minimum requirement.

2) Architecture of Inverse Motion Compensation Unit: The operation of inverse motion compensation is similar to block-based forward MV mapping but with different overlapped region derivation, as shown in Fig. 22. Consequently, our proposed architecture shown in Fig. 23 is able to support both inverse motion compensation operation and MV mapping operation. In addition, the SRAM here is also used in our proposed ping-pong two-way scheduling.

At first, corner coordinates of the overlapped region are derived. For MV mapping, the corner coordinates are used to compute area of the region, and then the MV with the largest accumulated area is assigned to the block of intermediate frame. For inverse motion compensation, the corner coordinates are used for SRAM address generator and SRAM rotate unit, and then the required pixels are read.

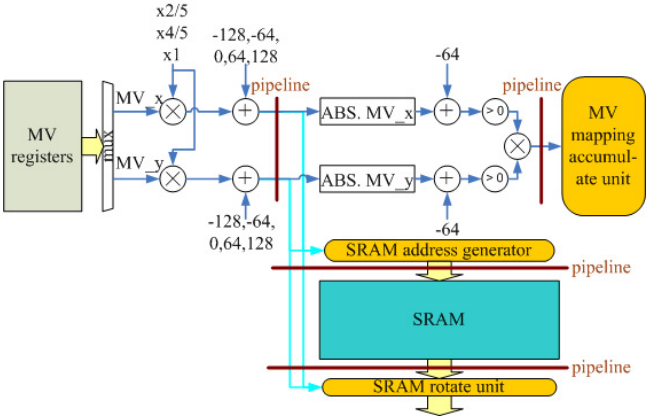


Fig. 23. Proposed architecture of inverse motion compensation unit.

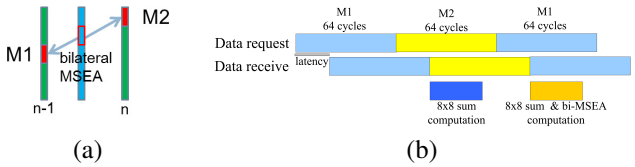


Fig. 24. (a) SRAM usage for bi-MSEA computation. (b) Scheduling for bi-MSEA computation.

D. Architecture of Sub-block Refinement

As mentioned above, bi-MSEA is computed only on some of the sub-blocks with MV discontinuity. Furthermore, bilateral MV search and OBMC are operated only on sub-blocks labeled by artifact detection. As a result, queues are created in DRAM for pushing and popping information of sub-blocks.

1) Bilateral MSEA Computing: To compute bi-MSEA, the required pixels are fetched into M1 and M2 SRAM in ping-pong two-way scheduling as shown in Fig. 24. The 8x8 sums of M1 is used for motion estimation at first, and then it is calculated with 8x8 sums of M2 for bi-MSEA computation.

2) Bilateral MV Searching and OBMC scheduling: After all of the above operations, the cycles left for labeled sub-blocks are shown in Table III by simulation. In the worst case, there are only 1,064 cycles left for bilateral MV search nad OBMC on one labeled sub-block. To achieve the target, the bilateral MV search performs $\pm$8x$\pm$8 even point search instead of full search. The cycle consumption for bilateral MV

TABLE III
CYCLES LEFT FOR LABELED SUB-CLOCKS IN THE WORST CASE.

Sequence	Cycle Left	# of Sub-block	For One Sub-block
park_joy	4,462,482	2,474	1,804
ducks_take_off	4,718,998	1,340	3,522
vintagecar	4,258,462	3,074	1,385
tractor	4,641,346	1,969	2,357
pedestrian_area	4,410,932	2,787	1,583
transformer 7-3	4,199,558	3,947	1,064
Titanic-2	4,246,090	1,820	2,333

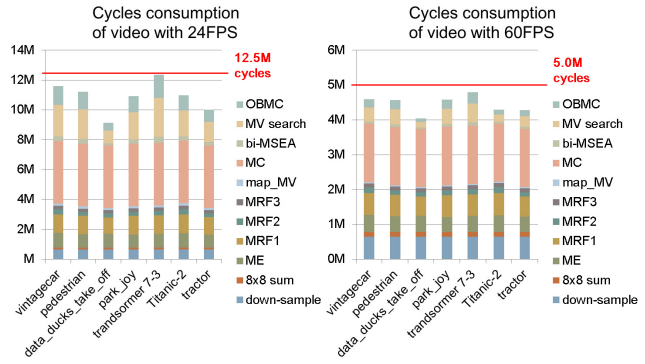


Fig. 25. Total cycle consumption of video converted from 24FPS to 120FPS and from 60FPS to 120FPS.

search becomes $((8 + 8) / 2 + 1)^2$ x (64 / 16) x 2 = 648 cycles per sub-block. Since the least cycle consumption for OBMC is (32 x 32 x 1.5 x (3 + 1)) / 16 = 384 cycles, 384 + 648 = 1,032 cycles are close to the target.

V. IMPLEMENTATION RESULT

We use Verilog-HDL for hardware implementation, and synthesize it by SYNOPSIS Design Compiler with UMC 90nm cell library. It works at 300MHz in frequency with a 128-bit bus. To summarize, it provides 24FPS to 120FPS and 60FPS to 120FPS multi-rate frame rate up-conversion, supporting video systems with Quad-HD resolution.

The overall resource saving and characteristics of the proposed architecture are shown in Table IV. Obviously, lots of cycles, bandwidth and on-chip SRAM are saved using our proposed hardware design techniques.

A. Cycle and Bandwidth Consumption

We simulate the cycle and bandwidth consumption on seven different sequences. Fig. 25 shows the simulation results of cycle consumption. With our proposed scheduling method, all of the cycles consumption are lower than the available resource.

Fig. 26 shows the simulation results of bandwidth consumption. All of the bandwidth consumption are lower than the available resource except the sequence "transformer 7-3" converted from 60FPS to 120FPS. It is acceptable since the sequences for simulation are actually at 24FPS. If the sequences are really at 60FPS, they will take less cycles and bandwidth because of the smaller MVs between frames.

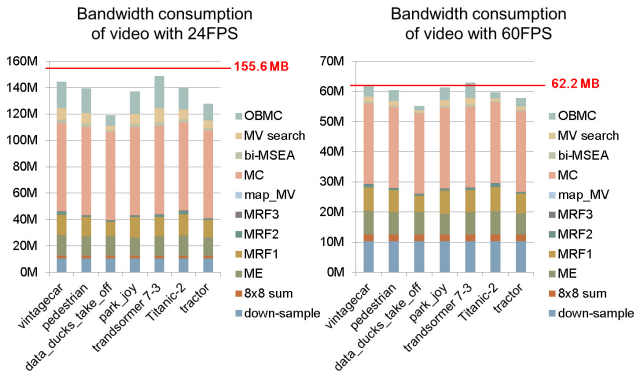


Fig. 26. Total bandwidth consumption of video converted from 24FPS to 120FPS and from 60FPS to 120FPS.

TABLE V
SPECIFICATION OF IMPLEMENTATION.

Specification	
Technology	UMC 90nm
Clock rate	300MHz
Bus width	128 bits/cycle
DRAM	DDR3-1333
Gate count	537,652
SRAM size	single port 9984 Bytes
FRUC mode	24FPS to 120FPS, 60FPS to 120FPS
Frame size	3840x2160
Search range	±128x±128

TABLE VI
COMPARISON OF NORMALIZED SPECIFICATIONS.

	Wang	Hsu	Ours
Technology	UMC 90nm	UMC 90nm	UMC 90nm
Clock rate	200MHz	133MHz	300MHz
Gate count	292,732	1,627,900	537,652
Gate count without SRAM	212,582	1,301,464	273,845
SRAM size (Byte)	3,036	12,365	9,984
FRUC mode (FPS)	60 to 120	60 to 120	24 to 120 60 to 120
Frame size	1920x1080	1920x1080	3840x2160

B. Chip Design and Specification

coming soon...

C. Hardware Efficiency Evaluation

We also compare the hardware efficiency with two related works [32]. The specifications are listed in Table VI and the gate count is normalized regarding all SRAM as single port with 3.3 gate count per bit. Our proposed design provides four times resolution and additional 24FPS to 120FPS up-conversion compared with other two implementations.

VI. CONCLUSION

In this paper, a multi-rate FRUC technique in Quad-HD LCD system is proposed. After introducing the motion blur

TABLE IV
OVERALL RESOURCE SAVING OF THE PROPOSED ARCHITECTURE.

	Motion Estimation			MRF Iteration x3			Motion Compensation		
	Direct	Proposed	Reduction	Direct	Proposed	Reduction	Direct	Proposed	Reduction
Cycles per block	576	266	54%	1,536	459	70%	6.2M (24FPS)	4M (24FPS)	35%
Bandwidth	18.8MB	7.2MB	62%	48MB	8.6MB	82%	99.5MB	64.8MB	35%
SRAM	82,944Bytes	9,984Bytes	88%	Shared by all modules					

problem on LCD and the general steps of FRUC technique, an algorithm and architecture implementation for 24FPS to 120FPS and 60FPS to 120FPS up-conversion in Quad-HD LCD system is proposed.

For the algorithm design, several techniques are proposed to achieve the target multi-rate FRUC algorithm. At first, we propose a predictive square search algorithm for true motion estimation with 32x32 block size, 8x8 MSEA criterion and $\pm$128x$\pm$128 search range. The experiments show that more than 60% blocks converge at their predictor, which indicates low-complexity and hardware-efficiency. In addition, MRF model is employed to perform MV correction. The proposed ICM method reducing energy computation from 65,536 to 9 and preventing over-smoothing is used for energy minimization. Furthermore, we propose a motion compensation with block-based forward MV mapping combining both the benefits of forward MV mapping and bilateral MV mapping. Finally, to enhance the visual quality of interpolated frames, sub-block division with artifact detection using 8x8 bi-MSEA criterion and OBMC refinement with bilateral MV Search using boundary error criterion are proposed.

For the architecture design, lots of hardware architecture optimizations are proposed to utilize the limited resources efficiently. For hardware sharing, the SRAM is shared with all of the modules in our proposed FRUC architecture. Also, the architecture of grouping is shared with MRF energy computing and median filter, and the architecture of inverse motion compensation is shared with MV mapping. With the careful arrangement of SRAM, there are totally 88% on-chip SRAM size reduction. Furthermore, our proposed ping-pong two-way scheduling eliminates the dependencies between blocks to achieve tight scheduling. There are 54% cycle reduction and 62% bandwidth reduction in ME, 70% cycle reduction and 82% bandwidth reduction in MRF MV correction, and 35% cycle reduction and 35% bandwidth reduction in motion compensation

In the experimental results, the subjective evaluation shows more than 79% subjects vote the sequences of our proposed algorithm as the best one. Also, the proposed algorithm has 0.63 to 5.47 PSNR gain compared to other algorithms in the objective evaluation. The implementation results show that our design, which consumes about 274k gate count and 10k byte single port SRAM, is most hardware-efficient compared to related works. In brief, the proposed algorithm and architecture consumes reasonable amount of resources but still maintains well performance.

Some possible future works are listed in the following. Since it easily introduce artifacts when interpolating between the frames with different scenes, the scene change detection may be adopted to increase robustness. On the other hands, for visual quality enhancement, the perceptual concepts may be taken into account for operations such as MV refinement and motion compensation.

REFERENCES

[1] H. Pan, X.-F. Feng, and S. Daly, "Quantitative analysis of LCD motion blur and performance of existing approaches," *SID Symposium Digest of Technical Papers*, vol. 36, no. 1, pp. 1590–1593, 2005.

[2] ——, "LCD motion blur modeling and analysis," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2, Sept. 2005, pp. II – 21–4.

[3] M. Orchard and G. Sullivan, "Overlapped block motion compensation: an estimation-theoretic approach," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 693–699, Sept. 1994.

[4] G. de Haan, P. W. A. C. Biezen, H. Huijgen, and O. A. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 5, pp. 368–379, Oct. 1993.

[5] J.Wang, D. Wang, and W. Zhang, "Temporal compensated motion estimation with simple block-based prediction," *IEEE Trans. Broadcast.*, vol. 49, no. 3, pp. 241–248, Sept. 2003.

[6] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *SPIE Visual Communications and Image Processing (VCIP)*, Feb. 2002, pp. 1069–1079.

[7] 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. Circuits Syst. Video Technol.*, vol. 17, no. 12, pp. 1700–1713, Dec. 2007.

[8] A.-M. Huang and T. Nguyen, "A multistage motion vector processing method for motion-compensated frame interpolation," *IEEE Trans. Image Process.*, vol. 17, no. 5, pp. 694–708, May 2008.

[9] Y.-N. Liu, Y.-T. Wang, and S.-Y. Chien, "Motion blur reduction of liquid crystal displays using perception-aware motion compensated frame rate up-conversion," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2011, pp. 84–89.

[10] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, Apr. 1990.

[11] G. Dane and T. Nguyen, "Smooth motion vector resampling for standard compatible video post-processing," in *Asilomar Conference on Signals, Systems and Computers*, vol. 2, Nov. 2004, pp. 1731–1735.

[12] D. Wang, L. Zhang, and A. Vincent, "Motion-compensated frame rate up-conversion - part i: Fast multi-frame motion estimation," *IEEE Trans. Broadcast.*, vol. 56, no. 2, pp. 133–141, June 2010.

[13] B.-W. Jeon, G.-I. Lee, S.-H. Lee, and R.-H. Park, "Coarse-to-fine frame interpolation for frame rate up-conversion using pyramid structure," *IEEE Trans. Consum. Electron.*, vol. 49, no. 3, pp. 499–508, Aug. 2003.

[14] B.-T. Choi, S.-H. Lee, and S.-J. Ko, "New frame rate up-conversion using bi-directional motion estimation," *IEEE Trans. Consum. Electron.*, vol. 46, no. 3, pp. 603–609, Aug. 2000.

[15] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko, "Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, pp. 407–416, Apr. 2007.

[16] Y. Ling, J. Wang, Y. Liu, and W. Zhang, "A novel spatial and temporal correlation integrated based motion-compensated interpolation for frame rate up-conversion," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, pp. 863–869, May 2008.

[17] K.-Y. Hsu and S.-Y. Chien, "Frame rate up-conversionwith global-to-local iterative motion compensated interpolation," in *IEEE International Conference on Multimedia and Expo (ICME)*, Apr. 2008, pp. 161–164.

[18] X. Gao, C. Duanmu, and C. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 501–504, Mar. 2000.
[19] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, June 1996.
[20] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.
[21] K. P. Lim, A. Das, and M. N. Chong, "Estimation of occlusion and dense motion fields in a bidirectional bayesian framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 712–718, May 2002.
[22] S. Z. Li, *Markov Random Field Modeling in Image Analysis*, 3rd ed. Springer Publishing Company, Incorporated, 2009.
[23] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, 2008.
[24] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comput. Vision*, vol. 70, no. 1, pp. 41–54, 2006.
[25] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 80–87.
[26] Y.-L. Huang, Y.-N. Liu, and S.-Y. Chien, "MRF-based true motion estimation using h.264 decoding information," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2010, pp. 99–104.
[27] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Proc. 11th Int. Conf. on Computer Vision (ICCV)*, Oct. 2007, pp. 1–8.
[28] J. Zhang, J. Arnold, and M. Frater, "A cell-loss concealment technique for mpeg-2 coded video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 4, pp. 659–665, June 2000.
[29] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko, "Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, pp. 407–416, Apr. 2007.
[30] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching vlsi architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.
[31] F.-C. Chen, Y.-L. Huang, and S.-Y. Chien, "Hardware-efficient true motion estimator based on markov random field motion vector correction," in *International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*, 2012, pp. 1–4.
[32] K.-Y. Hsu and S.-Y. Chien, "Hardware architecture design of frame rate up-conversion for high definition videos with global motion estimation and compensation," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2011, pp. 90–95.