A High Performance Reconfigurable Motion Estimation Hardware Architecture

O. Tasdizen[1,2], H. Kukner[1], A. Akin[1], and I. Hamzaoglu[1]
[1]Faculty of Engineering and Natural Sciences, Sabanci University, 34956 Tuzla, Istanbul, Turkey
[2]Vestek Electronic Research & Development Corp., 34469 Maslak, Istanbul, Turkey

Abstract

Motion Estimation (ME) is the most computationally intensive part of video compression and video enhancement systems. For the recently available high definition frame sizes and high frame rates, the computational complexity of full search (FS) ME algorithm is prohibitively high, while the PSNR obtained by fast search ME algorithms is low. Therefore, in this paper, we propose a new ME algorithm and a high performance reconfigurable systolic ME hardware architecture for efficiently implementing this algorithm. The proposed ME algorithm performs up to three different granularity search iterations in different size search ranges based on the application requirements. Simulation results showed that the proposed ME algorithm performs very close to FS algorithm, even though it searches much fewer search locations than FS algorithm. It outperforms successful fast search ME algorithms by searching more search locations than these algorithms. The proposed reconfigurable ME hardware is implemented in VHDL and mapped to a low cost Xilinx XC3S1500-5 FPGA. It works at 130MHz and is capable of processing high definition and high frame rate video formats in real time. Therefore, it can be used in flat panel displays for frame rate conversion and de-interlacing, and in video encoders.

1 Introduction

Motion Estimation (ME) is used to reduce the bit-rate in video compression systems by exploiting the temporal redundancy between successive frames and to enhance the quality of displayed images in video enhancement systems by extracting the true motion information. It is the most computationally intensive part of video compression and video enhancement systems.

Block Matching (BM) is the most preferred method for ME. BM partitions current frame into non-overlapping NxN rectangular blocks and tries to find a block from a reference frame in a given search range that best matches the current block. Sum of Absolute Differences (SAD) is the most preferred block matching criterion because of its suitability for hardware implementation.

Among the BM algorithms, Full Search (FS) algorithm achieves the best performance since it searches all search locations in a given search range. However, the computational complexity of FS algorithm is very high, especially for the recently available high resolution and high frame rate video formats.

Several fast search ME algorithms have been developed for low bit-rate applications which use small frame sizes and require small search ranges. These algorithms try to approach the PSNR of FS algorithm by computing the SAD values for fewer search locations in a given search range. The most successful fast search algorithms are New Three Step Search (NTSS) [1], Diamond Search (DS) [2], Hexagon-Based Search (HEXBS) [3], Adaptive Rood Pattern Search (ARPS) [4], Adaptive Dual Cross Search (ADCS) [5] and Flexible Triangle Search (FTS) [6].

Fast search ME algorithms perform very well for low bit-rate applications such as video phone and video conferencing, because fast and complex movements are seldom in these applications. However, fast search algorithms do not produce satisfactory results for the recently available consumer electronic devices such as High Definition (HD) digital video broadcasting and high resolution & high frame rate flat panel displays, because there are fast and complex movements between successive frames in these applications.

Therefore, in this paper, we propose a new ME algorithm and a high performance reconfigurable systolic ME hardware architecture for efficiently implementing this algorithm. Simulation results showed that the proposed ME algorithm performs very close to ME algorithm in [7] and FS algorithm, even though it searches much fewer search locations than these algorithms. It outperforms successful fast search ME algorithms by searching more search locations than these algorithms.

The proposed ME algorithm is a generalization of the NTSS algorithm. It performs up to three different granularity search iterations. First, the entire search window is searched with a coarse search iteration. Then, two fine search iterations performed around the search locations with minimum SAD. One or two iterations can be skipped and search range of each iteration can be configured based on the application requirements.

The proposed reconfigurable systolic ME hardware architecture is based on the systolic ME hardware architecture proposed in [7]. The major differences between them are the proposed hardware is reconfigurable, implements three search iterations and has larger search ranges. The proposed ME hardware is statically configured to perform up to three different granularity search iterations in different size search ranges based on the application requirements in order to obtain a performance close to FS algorithm by searching even fewer search locations than the ME algorithm in [7]. For HD video frames, larger search ranges are necessary because of larger motion vectors (MV).

The proposed hardware is implemented in VHDL and mapped to a low cost Xilinx XC3S1500-5 FPGA. It consumes 9083 slices (2271 CLBs) and 16 BRAMs. It works at 130MHz and is capable of processing HD high frame rate video formats in real time. Therefore, it can be used in flat panel displays for frame rate conversion and de-interlacing, and in video encoders.

Only a small number of hardware architectures for fast search ME algorithms are proposed in the literature [6, 7]. The proposed

978-3-9810801-5-5/DATE09 © 2009 EDAA

hardware consumes less area than the implementation of one of the best performing fast search ME algorithms in the same FPGA [6]. Because of the overhead of reconfigurability, it consumes slightly more area than the hardware proposed in [7] in the same FPGA. To the best of our knowledge, no hardware architecture is proposed in the literature for the ME algorithm proposed in this paper.

The rest of the paper is organized as follows. Section 2 explains the proposed ME algorithm and presents a comparison of its performance with the performance of other ME algorithms. The proposed reconfigurable systolic ME hardware architecture is described in Section 3. Section 4 concludes the paper.

2 Proposed Motion Estimation Algorithm

The proposed ME algorithm performs up to three iterations; coarse, medium and fine. In these iterations, horizontal and vertical distances between search locations are 4, 2 and 1 pixels. One or two iterations can be skipped and search range of each iteration can be configured based on the application requirements. The number of iterations and sizes of search ranges determine the computational complexity of a search pattern and PSNR obtained by it. Several search patterns are shown in Table 1. As shown in the table, skipping the coarse and medium iterations and doing the fine search on the whole search range is identical to performing FS algorithm. The table also includes ME algorithm proposed in [7].

Fig. 1 shows a portion of search pattern A. Search ranges of coarse, medium, and fine iterations of pattern A are (±48,±24), (±6,±6), (±3,±3) pixels respectively. In Fig. 1, numbers represent the iterations and dark shaded numbers show the search locations with minimum SAD for these iterations.

Table 1. Several Search Patterns

Search Pattern	SR of First Iteration	SR of Second Iteration	SR of Third Iteration	Number of Search Locations
A	±48, ±24	±6, ±6	±3, ±3	405
B	±48, ±24	±12, ±12	±6, ±6	565
C	±48, ±24	±24, ±12	±12, ±6	793
[7]	-	±48, ±24	±3, ±3	1221
FS	-	-	±48, ±24	4753

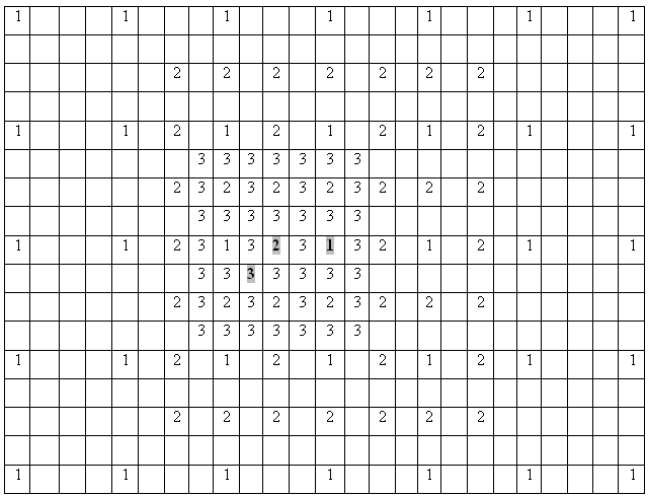

Fig. 1. Search Pattern A

The performances of the search patterns shown in Table 1 are compared with the performances of successful fast ME algorithms for the same search range of (±48, ±24) pixels with respect to Mean Absolute Difference (MAD) criterion and the comparison results are shown in Table 2. Six 100 frame video sequences are used for the comparison. “Spiderman”, “Gladiator” and “Irobot” video sequences are taken from “Spiderman II”, “Gladiator” and “Irobot” movies where there are fast and complex motion.

As it can be seen in Table 2, these search patterns clearly outperform successful fast search ME algorithms and they perform very close to ME algorithm in [7] and FS algorithm by searching much fewer search locations. For example, even though, FS searches 4753 search locations in comparison to 405 search locations searched by pattern A, its performance is only 5.36% better than the performance of pattern A on the average.

3 Proposed Reconfigurable ME Hardware

Top-level block diagram of the proposed hardware is shown in Fig. 2. The reconfigurability is achieved with the multiplexing unit and PE array. The hardware finds a MV for a 16x16 Macroblock (MB) based on minimum SAD criterion in a maximum search range of (±48, ±24) pixels using the luminance data. The hardware is highly pipelined and its latency is eight clock cycles; one cycle for synchronous read from memory, one cycle for multiplexing unit, two cycles for reconfigurable systolic Processing Element (PE) array and four cycles for adder tree.

Table 2. MAD Simulation Results

Video (Frame Size & Rate)	FS	DS[2]	HEXBS[3]	ARPS[4]	ADCS[5]	FTS [6]	48x24[7]	A	B	C
Spiderman (720x576, 25fps)	4.2086	7.2072	7.3777	6.0776	6.2421	8.3311	4.2265	4.2771	4.2657	4.2584
Gladiator (720x576, 25fps)	2.8370	5.4301	5.6132	3.9364	3.7342	5.2116	2.8858	2.9797	2.9349	2.9207
IRobot (720x576, 25fps)	2.9264	4.3990	4.5108	3.8864	4.0393	6.6888	3.0125	3.2350	3.1505	3.1045
Susie (704x480, 15fps)	3.4268	4.0875	4.1583	4.0491	3.9970	4.8050	3.5163	3.5418	3.5046	3.4901
Flowers (704x480, 15fps)	8.2963	9.7922	10.3324	8.6041	8.8182	16.5934	8.4660	9.1652	8.9850	8.8105
Table Tennis (704x480, 15fps)	4.3926	4.7276	4.7535	4.7449	4.6636	4.9372	4.4186	4.4418	4.4284	4.4219

Top-level controller takes SR4, SR2 and SR1, the sizes of search ranges for coarse, medium and fine iterations respectively, as inputs. Control unit finds the MV by generating required address and control signals to compute the SAD values of the search locations in the search window determined by the top-level controller for each iteration. After the MV in an iteration is found, top-level controller determines the search window for the next iteration based on this MV and the size of the search range.

The search locations in a search window are searched line by line. First, SAD values of the search locations in the top line of the search window are calculated starting from the right most search location in the top line. Then, SAD values of the search locations in the next line of the search window are calculated starting from the right most search location in the next line. The first iteration ends after SAD values of the search locations in the bottom line of the search window are calculated. The next iteration around the search location with minimum SAD is done in the same way.

16 BlockRAMs (BRAM) in the FPGA are used to store the search window of current MB. BRAMs are configured as dual port memories for overlapping ME of current MB with loading of search window of next MB. Vertical rotator is used to align the outputs of the BRAMs and it has 32 identical rotators each 16 bit long. Reference MB data read from BRAMs must be matched with the current MB data, which is loaded into PE array previously, by rotating the data lines. For example, for the search locations in the fourth line of the search window, the rotate amount will be four so that first line of reference data will be read from fourth BRAM.

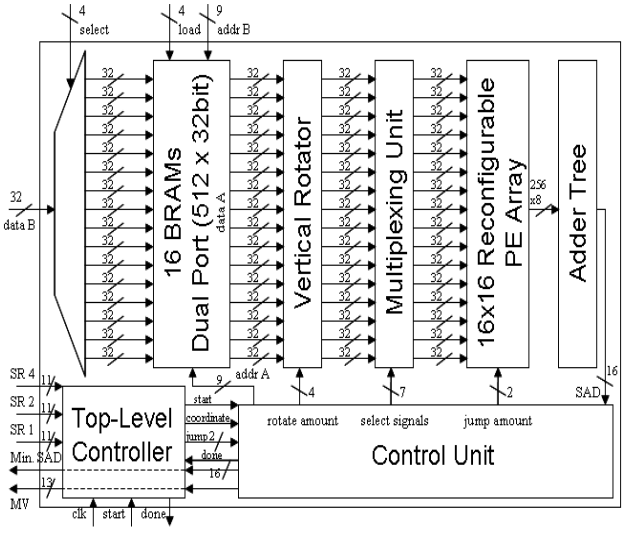

Fig. 2. Top-level Block Diagram

The SAD value for a search location is calculated by summing the outputs of all 256 PEs in the reconfigurable PE array by an adder tree. The adder tree has four pipeline stages; SAD values of 4x4 blocks are calculated in the first two clock cycles, in the third clock cycle SAD values of 8x8 blocks are calculated and in the fourth clock cycle SAD value of 16x16 MB is calculated.

The reconfigurable systolic PE array is shown in Fig. 3. 256 PEs are used to calculate the SAD of a 16x16 MB same as the systolic PE array presented in [7]. A PE is used to calculate the absolute difference between a current pixel and the corresponding reference pixel. The latency of the reconfigurable PE array is two clock cycles, since reference and current pixel inputs and absolute difference output are registered.

The reconfigurability of the PE array is achieved with the multiplexers placed between the PEs that process the same line in a MB. Since the PE array in [7] is not reconfigurable, these multiplexers bring a slight area overhead in comparison to the PE array in [7]. But they do not affect the clock frequency since they are not placed on the critical path.

The reference pixels for the first search location in a line of the search window are loaded in four clock cycles. After the SAD value of the first search location is calculated in four clock cycles, SAD value of next search location is calculated in one clock cycle. When the SAD value of the first search location is obtained, reference data is shifted to the right in the PE array in each consecutive clock cycle and shift amount can be 4, 2 or 1 pixels depending on the iteration number. In Fig. 3, interconnects used for implementing 4, 2 and 1 shift amounts are illustrated with dashed, thin and bold lines respectively. Interconnects marked with "m" indicate connections to the memory.

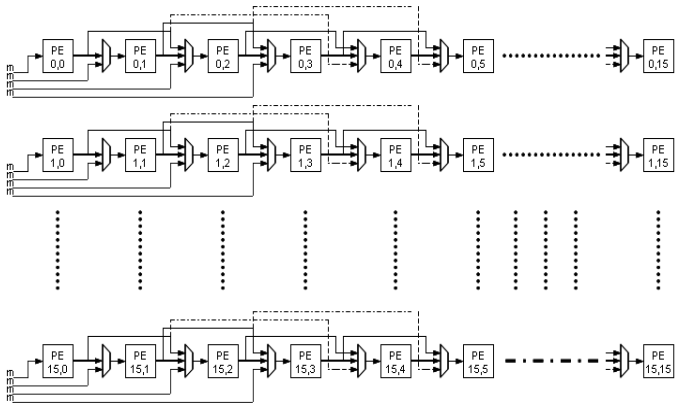

Fig. 3. Reconfigurable Systolic PE Array

In order to calculate the SAD values of search locations at the rate of one SAD value per clock cycle, pixels for a particular search location must be brought to PE array in one clock cycle, and this requires many accesses to the memory in the same clock cycle. This memory requirement cannot be satisfied by a low cost FPGA family without data-reuse. The ME hardware proposed in [7] reduces the internal memory bandwidth by applying data-reuse and it uses only 16 BRAMs for storing the reference pixels of a search window for a search range of (±32, ±16) pixels. BRAMs are configured as 16 bit wide because of the 2 pixel distance between consecutive search locations.

The ME hardware proposed in this paper also applies data-reuse. However, it uses only 16 BRAMs for storing the reference pixels of a search window for a search range of (±48, ±24) pixels and it further reduces the internal memory bandwidth by feeding only 64 PEs from BRAMs, the remaining PEs receive reference pixels from neighboring PEs. BRAMs are configured as 32 bit wide and they are connected to the four left end columns of the PE array. Therefore, loading the reference pixels for the first search location into the PE array takes four clock cycles.

Each BRAM stores four lines of reference pixels. The first BRAM stores 0th, 16th, 32th and 48th lines of the reference pixels. The second BRAM stores 1st, 17th, 33th and 49th lines of the reference pixels. The remaining BRAMs have the same organization. Storing a line of reference pixels uses 28 address locations; therefore, addresses 0-111 of a BRAM are occupied to store four lines of reference pixels.

Multiplexing unit is used to feed the correct data to PE array. In order to support horizontal distances of 1, 2 and 4 between consecutive search locations, multiplexing unit is designed to feed first one, two or four left end columns of PE array. Independent from the search pattern, reference pixels for the first search location are loaded by feeding the four columns. Therefore, four clock cycles are required to fill PE array with the reference pixels for the first search location. The reference pixels for the next search location will be available in the next clock cycle.

Multiplexing unit consist of shift registers and multiplexers. The data received from the vertical rotator is captured in a 56 bit long shift register. The shift register has an enable signal. When it is enabled, the register shifts its content 32 bits to right. While loading the first search location and performing the first iteration, the shift register will be enabled in each clock cycle. While performing the second and third iterations, the shift register will be enabled twice and once in every four clock cycles, respectively. Multiplexers are used to select the corresponding reference pixels from the shift register.

3.1 Implementation Results

The proposed ME hardware architecture is implemented in VHDL, verified by simulation using Mentor Graphics Modelsim 6.3c, synthesized with Synplicity Synplify Pro 8.9, and mapped to a low cost Xilinx XC3S1500-5 FPGA using Xilinx ISE 9.2.04. The proposed hardware works at 130MHz and consumes 9083 slices (2271 CLBs) and 16 BRAMs. The reconfigurable systolic PE array with the adder tree consumes 7510 slices.

In the same FPGA and for the same MB size, the ME hardware proposed in [6] consumes 6142 CLBs, works at 74MHz and requires 202 clock cycles per MB. The proposed ME hardware uses 2318 slices more than the systolic ME hardware in [7]. 1136 slices are used by the multiplexing unit, 836 additional slices are used by the multiplexers in the PE array and remaining additional slices are used by additional complexity of the control unit.

The number of clock cycles per MB required by the proposed hardware depends on the realized search pattern. Starting an iteration has a start-up cost of 15 clock cycles, which is called iteration latency, and starting the search on a line has a start-up cost of 8 clock cycles, which is called line latency. The total number of clock cycles per MB required to complete a search pattern is given by the equation below. The performance of proposed ME hardware for several search patterns are calculated based on this equation and shown in Table 3.

$$(n_{it} \times \tau_{it}) + (n_{sad}-1) \times n_{line} + ((n_{line}-1) \times \tau_{line})$$

In this equation, "n_{it}, $n_{sad,}$ n_{line}" are the number of iterations, search locations per line and lines per iteration respectively. "τ_{it}" and "τ_{line}" are iteration and line latencies respectively.

Table 3. Performance of Proposed ME Hardware

Search Pattern	Required Clock Cycles per MB	Processed MBs per Second	Supported Frame Size & Rate
A	633	205371	1920x1080, 25 fps
B	957	135841	1366x768, 33 fps
C	1221	106470	1366x768, 25 fps
48x24 [7]	1425	101052	1366x768, 24 fps
FS	5103	25475	1366x768, 6 fps

4 Conclusions

In this paper, a new ME algorithm and a high performance reconfigurable systolic ME hardware architecture for efficiently implementing this algorithm are proposed. The proposed ME hardware is statically configured to perform up to three different granularity search iterations in different size search ranges based on the application requirements. The simulation results showed that proposed ME algorithm outperforms successful fast search ME algorithms and it performs very close to the performance of FS algorithm by searching much fewer search locations. The proposed ME hardware is implemented on a low cost Xilinx FPGA and it is capable of processing HD high frame rate video formats in real time.

5 Acknowledgements

This work is supported in part by TUBITAK (The Scientific and Technological Research Council of Turkey) and Vestek Electronic Research & Development Corp., Istanbul, Turkey.

References

[1] R. Li, B. Zeng, and M.L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. CAS for Video Technology, vol. 4, pp. 438–442, 1994.

[2] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast blockmatching motion estimation," IEEE Trans. Image Processing, vol. 9, pp. 287–290, 2000.

[3] C. Zhu, X. Lin, and L.P. Chau, "Hexagon-based search pattern for fast block motion estimation," IEEE Trans. CAS for Video Technology, vol. 12, pp. 349–355, 2002.

[4] Y. Nie, K.-K. Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation," IEEE Trans. on Image Processing, vol. 11, pp. 1442–1449, 2002.

[5] X-Q Banh and Y-P Tan, "Adaptive Dual-Cross Search Algorithm for Block-Matching Motion Estimation," IEEE Trans. on Consumer Electronics, vol. 50, no. 2, pp. 766-775, May 2004.

[6] M. Rehan, M. W. El-Kharashi, P. Agathoklis, and F. Gebali, "An FPGA Implementation of the Flexible Triangle Search Algorithm for Block Based Motion Estimation," IEEE ISCAS, May 2006.

[7] O. Tasdizen, A. Akin, H. Kukner, I. Hamzaoglu, and H. F. Ugurdag, "High Performance Hardware Architectures for a Hexagon-Based Motion Estimation Algorithm," 16th IEEE / IFIP International Conference on VLSI - SoC, October 2008.