Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation

Ozgur Tasdizen, *Student Member, IEEE*, Abdulkadir Akin, Halil Kukner, and Ilker Hamzaoglu, *Member, IEEE*

Abstract — *Motion Estimation (ME) is the most computationally intensive part of video compression and video enhancement systems. For the recently available High Definition (HD) video formats, the computational complexity of full search (FS) ME algorithm is prohibitively high, whereas the PSNR obtained by fast search ME algorithms is low. Therefore, in this paper, we present Dynamically Variable Step Search (DVSS) ME algorithm for processing high definition video formats and a dynamically reconfigurable hardware architecture for efficiently implementing DVSS algorithm. The simulation results showed that DVSS algorithm performs very close to FS algorithm by searching much fewer search locations than FS algorithm and it outperforms successful fast search ME algorithms by searching more search locations than these algorithms. The proposed hardware is implemented in VHDL and is capable of processing high definition video formats in real time. Therefore, it can be used in consumer electronics products for video compression, frame rate up-conversion and de-interlacing.* [1]

Index Terms — Motion Estimation, Video Compression, Video Enhancement, Hardware Implementation, FPGA.

I. INTRODUCTION

Motion Estimation (ME) is the most computationally intensive part of video compression and video enhancement systems. ME is used to reduce the bit-rate in video compression systems by exploiting the temporal redundancy between successive frames, and it is used to enhance the quality of displayed images in video enhancement systems by extracting the true motion information. ME is used in video compression standards such as MPEG4 and H.264 [1], and it is used in video enhancement algorithms such as frame rate conversion [2-3] and de-interlacing [4-5]. Therefore, ME hardware is used in consumer electronics products such as digital camcorders, DVD players and recorders, set-top boxes, LCD televisions, and video conferencing devices.

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. The SAD value of a search location defined by the motion vector $d(d_x,d_y)$ is calculated as in (1) where c is the current frame, r is the reference frame, and the coordinate (i, j) is the location of current and reference blocks of size NxN in current and reference frames respectively.

$$SAD(\vec{d}) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left|c(x+i,y+j) - r(x+i+d_x, y+j+d_y)\right| \quad (1)$$

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 Definition (HD) 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 Orthogonal Search (OS) [6], New Three Step Search (NTSS) [7], Four Step Search (FSS) [8], Block-Based Gradient Descent Search (BBGDS) [9], Diamond Search (DS) [10], Hexagon-Based Search (HEXBS) [11], Adaptive Rood Pattern Search (ARPS) [12], Adaptive Dual Cross Search (ADCS) [13] and Flexible Triangle Search (FTS) [14].

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

Therefore, in this paper, we propose Dynamically Variable Step Search (DVSS) ME algorithm for processing HD video formats and a dynamically reconfigurable systolic ME hardware architecture for efficiently implementing DVSS algorithm. The simulation results showed that DVSS algorithm performs very close to FS algorithm by searching

[1] O. Tasdizen is with Department of Electronics Engineering, Sabancı University, Tuzla 34956, Istanbul, Turkey and with Vestek Electronic Research & Development Corp., Maslak 34469, Istanbul, Turkey (e-mail: tasdizen@su.sabanciuniv.edu, ozgur.tasdizen@vestel.com.tr)

A. Akin, H. Kukner and I. Hamzaoglu are with Department of Electronics Engineering, Sabancı University, Tuzla 34956, Istanbul, Turkey (e-mail: abdulkadir@su.sabanciuniv.edu,shalil@su.sabanciuniv.edu, hamzaoglu@sabanciuniv.edu).

Contributed Paper
Manuscript received July 15, 2009

much fewer search locations than FS algorithm and it outperforms successful fast search ME algorithms by searching more search locations than these algorithms.

DVSS algorithm has a maximum of three different granularity search steps. First, the entire search window is searched with a coarse granularity search step. Then, two finer granularity search steps are performed around the search locations with minimum SAD. The number of steps and the search range of each step are determined for the current block based on the size and SAD value of previously found Motion Vector (MV) for the left neighboring block.

Therefore, for each block, the proposed ME hardware can be dynamically reconfigured to execute different number of steps and different search ranges for each step. The proposed hardware is implemented in VHDL and mapped to an XC3S1500-5 FPGA. It consumes 9128 slices (2282 CLBs) and 16 BRAMs. It works at 130MHz and is capable of processing HD video formats in real time. Therefore, it can be used in consumer electronics products for video compression, frame rate conversion and de-interlacing.

A small number of hardware architectures for fast search ME algorithms are proposed in the literature [14-16]. The proposed hardware consumes less area than the implementation of one of the best performing fast search ME algorithms in the same FPGA [14] and it has higher throughput than the fast search ME hardware presented in [15].

We proposed a ME algorithm for processing HD video formats and a systolic ME hardware architecture for efficiently implementing this ME algorithm in [16]. We proposed another ME algorithm for processing HD video formats and a reconfigurable systolic ME hardware architecture for efficiently implementing this ME algorithm in [17]. This ME algorithm obtains similar performance results by searching fewer locations than the ME algorithm in [16].

In this paper, we propose DVSS algorithm in order to obtain a performance very close to FS algorithm by searching even fewer search locations than the ME algorithms in [16, 17]. The dynamically reconfigurable systolic ME hardware architecture proposed in this paper is based on the systolic ME hardware architectures proposed in [16, 17]. The major differences between them are the proposed hardware is dynamically reconfigurable and it implements DVSS algorithm.

Many hardware architectures for FS ME algorithm are proposed in the literature [18-21]. The proposed ME hardware has a much higher throughput than these FS hardware. In addition, since it is searching a larger search range than these FS hardware, the proposed ME hardware obtains better PSNR results for HD video formats than these FS hardware. Larger search ranges are necessary for HD video formats, since there are fast and complex motions in them.

The rest of the paper is organized as follows. Section II explains the proposed DVSS algorithm and presents a performance comparison between DVSS algorithm and the other ME algorithms. The proposed dynamically reconfigurable systolic ME hardware architecture is described in Section III. Section IV concludes the paper.

II. Proposed Dynamically Variable Step Search Motion Estimation Algorithm

We proposed a ME algorithm for processing HD video formats in [16]. This ME algorithm is a generalization of the HEXBS ME algorithm and it checks all the search locations that can be checked by HEXBS algorithm during its iterations in a given search range. For the 32x16 search pattern shown in Fig. 1, this ME algorithm checks all the search locations that can be checked by HEXBS algorithm during 16 iterations in the horizontal direction and 8 iterations in the vertical direction in a (±32, ±16) pixel search range. The numbers seen in Fig.1 represent iterations in which these search locations would be checked by HEXBS algorithm.

This ME algorithm can be used with other search patterns as well depending on the application requirements such as 10x9 and 14x15 patterns [16]. The difference between these patterns and 32x16 pattern is that they have a gap of two pixels in the vertical direction compared to the one pixel gap of 32x16 pattern and these patterns have a search range of (±10, ±9) and (±14, ±15) respectively. Therefore, they have less computational complexity than 32x16 search pattern.

We proposed another ME algorithm for processing HD video formats in [17]. This ME algorithm can be seen as a generalization of NTSS algorithm. It has a maximum of 3 different granularity search steps; coarse, medium and fine. First, the entire search window is searched with a coarse granularity search step. Then, two finer granularity search steps are performed around the search locations with minimum SAD. In these steps, horizontal and vertical distances between search locations are 4, 2 and 1 pixels. The number of steps and the search range of each step used in this ME algorithm are determined based on the application requirements.

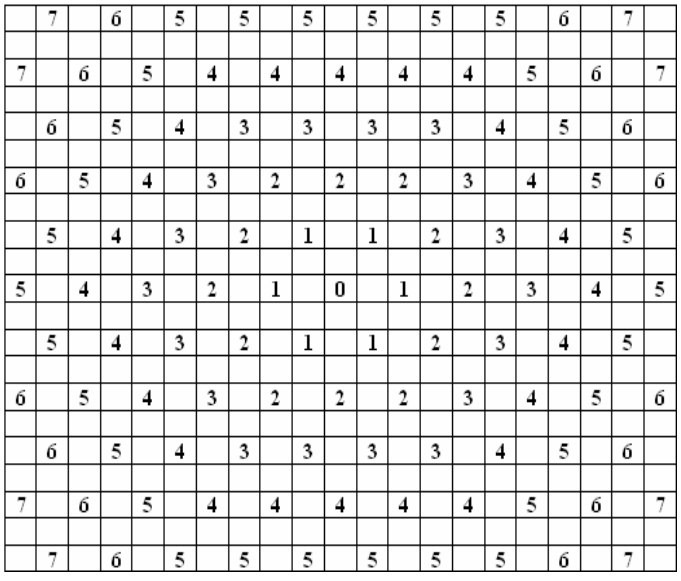

Fig. 1. Some of the Search Locations of 32x16 Search Pattern

In this paper, we propose DVSS algorithm in order to obtain a performance very close to FS algorithm by searching even fewer search locations than the ME algorithms proposed in [16, 17]. DVSS algorithm is based on the ME algorithm proposed in [17] and improves its performance by determining the number of steps and the search range of each step for the current block based on the size and SAD value of previously found Motion Vector (MV) for the left neighboring block. It is possible to use one of many different search patterns for a given block. Some of these search patterns, named as A1 [17], A2, A3, B and C, and the search patterns used in [16] are shown in Table I. As shown in the table, skipping the coarse and medium steps and doing the fine step on the entire search range is identical to FS algorithm.

The search pattern A1, as shown in Fig. 2, has 3 steps and the search ranges (SR) of coarse, medium, and fine steps are (±48,±24), (±6,±6), (±3,±3) pixels respectively. In Fig. 2, numbers represent the steps and shaded numbers show the search locations with minimum SAD for these steps. The search pattern A2 is the same as A1 except that the search range of its first step is (±24, ±12) pixels. The search pattern A3, as shown in Fig. 3, has only medium and fine steps.

The number of steps and sizes of search ranges for each step determine the computational complexity of a search pattern and Mean Absolute Difference (MAD) performance obtained by it. DVSS algorithm decreases the computational complexity by adaptively changing between search patterns A1, A2, A3 for each block based on the size and SAD value of the previously found MV for the left neighboring block, which is called as Left Neighboring Motion Vector (LNMV). As shown below, it uses FS, A3, A2, and A1 search patterns for small, medium, medium-to-large and large motions respectively.

If there is no left neighboring block
  Do Pattern A1
Else if SAD value of LNMV exceeds the threshold (τ)
  Switch to next coarser pattern
Else
  If LNMV is within (±8, ±4) pixels
    Do FS in (±10, ±5) search range
  Else if LNMV is within (±16, ±8) pixels
    Do Pattern A3
  Else if LNMV is within (±24, ±12) pixels
    Do Pattern A2
  Else
    Do Pattern A1

If LNMV falls within a smaller search range, it decreases the search granularity and search range size, because for small motions doing the search in a smaller search range is sufficient and doing a finer granularity search in a smaller search range can give better MAD results. If the SAD value for LNMV is higher than a pre-determined threshold level (τ), it increases the search granularity and search range size. τ is set to 256 and to 1024 in our simulations. By setting τ to a higher value, many search locations can be skipped and higher processing speeds can be achieved with a slight decrease in MAD performance.

DVSS algorithm reduces the computational complexity of ME by adaptively changing the number of search locations searched for each block. Similar techniques are proposed in the literature [22, 23]. In [22], some of the candidate search locations are eliminated adaptively if their partial SAD value exceeds a dynamically modified threshold level. In [23], the size and SAD values of the MVs of the previous blocks are used to adaptively change the search window size of FS algorithm for each block.

TABLE I
SEVERAL SEARCH PATTERNS

Search Pattern	SR of First Step	SR of Second Step	SR of Third Step	Number of Search Locations
10x9 [16]	-	±10, ±9	±3, ±3	73
14x15 [16]	-	±14, ±15	±3, ±3	159
A1 [17]	±48, ±24	±6, ±6	±3, ±3	405
A2	±24, ±12	±6, ±6	±3, ±3	161
A3	-	±18, ±10	±3, ±3	249
32x16 [16]	-	±32, ±16	±3, ±3	553
B	±48, ±24	±12, ±12	±6, ±6	565
C	±48, ±24	±24, ±12	±12, ±6	793
48x24 [16]	-	±48, ±24	±3, ±3	1221
FS	-	-	±48, ±24	4753

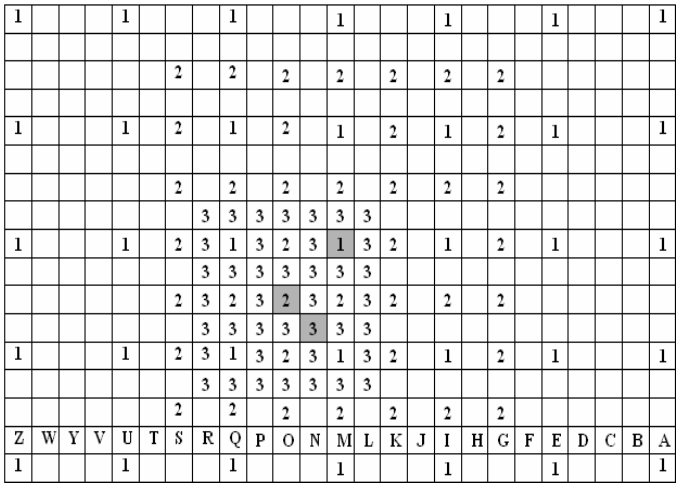

Fig. 2. Search Pattern A1

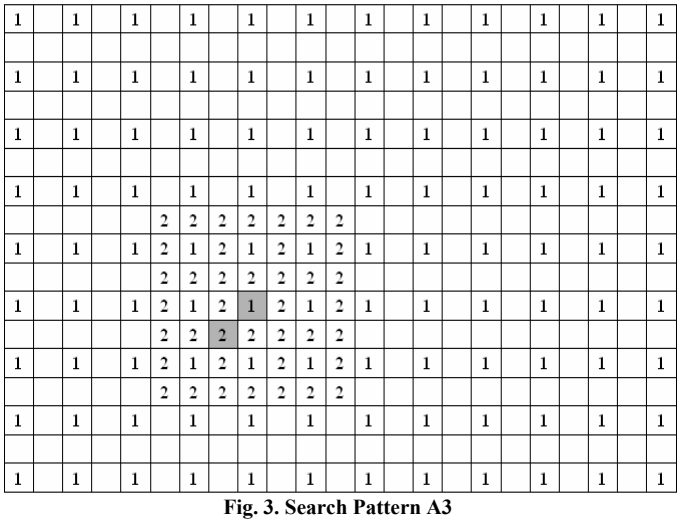

Fig. 3. Search Pattern A3

TABLE II
MAD SIMULATION RESULTS FOR FAST SEARCH ALGORITHMS

Video Sequence (Frame Size & Rate)	FS ±48,±24	FS ±16,±16	OS [6]	NTSS [7]	FSS [8]	BBGDS [9]	DS [10]	HEXBS [11]	ARPS [12]	ADCS [13]	FTS [14]
Spiderman (720x576, 25fps)	4.2086	6.9686	6.6708	10.7152	10.8176	7.4789	7.2072	7.3777	6.0776	6.2421	6.8786
Gladiator (720x576, 25fps)	2.8370	5.3812	5.2161	8.6823	8.7966	5.6802	5.4301	5.6132	3.9364	3.7342	6.0059
IRobot (720x576, 25fps)	2.9264	3.7108	4.7796	5.4830	5.5572	4.5329	4.3990	4.5108	3.8864	4.0393	4.8717
Susie (704x480, 15fps)	3.2262	3.421	4.2313	4.05	4.0861	3.8182	3.632	3.7153	3.6256	3.6241	3.9267
Flowers (704x480, 15fps)	8.3955	8.418	12.412	10.476	11.125	10.636	10.318	10.976	8.7037	8.9538	13.118
Table Tennis (704x480, 15fps)	3.4842	3.5824	3.9371	3.9738	4.0129	3.8621	3.8011	3.8367	3.7324	3.7478	3.8891
Foreman (352x288, 15fps)	4.1783	4.2327	6.013	4.8124	4.8613	4.5136	4.5664	5.0804	4.5463	4.6925	5.6903

TABLE III
MAD SIMULATION RESULTS FOR PROPOSED SEARCH ALGORITHMS

Video Sequence (Frame Size & Rate)	10x9 [16]	14x15 [16]	32x16 [16]	48x24 [16]	A1 [17]	B	C	DVSS $\tau = 256$	DVSS $\tau = 1024$
Spiderman (720x576, 25fps)	9.3473	7.4848	5.5317	4.2296	4.2771	4.2657	4.2584	4.3990	4.5449
Gladiator (720x576, 25fps)	7,2923	5,8438	3,3246	2.8858	2.9797	2.9349	2.9207	3.1451	3.2622
IRobot (720x576, 25fps)	7.6989	6.9352	5.7265	3.0872	3.2350	3.1505	3.1045	3.2937	3.3346
Susie (704x480, 15fps)	3.9229	3.7203	3.4004	3.3334	3.4138	3.3482	3.3291	3.2979	3.2928
Flowers (704x480, 15fps)	8.898	8.7982	8.6225	8.6106	9.2686	9.065	8.9511	8.514	8.4891
Table Tennis (704x480, 15fps)	3.7986	3.6634	3.5656	3.517	3.572	3.5531	3.5409	3.554	3.5714
Foreman (352x288, 15fps)	5.0247	4.9595	4.6724	4.6626	4.8714	4.7015	4.6006	4.5191	4.3929

The performances of DVSS algorithm and the search patterns shown in Table I are compared with the performances of successful fast ME algorithms with respect to MAD criterion (2) and the results are shown in Table II and Table III.

$$MAD(\vec{d}) = \frac{1}{N^2}\sum_{y=0}^{N-1}\sum_{x=0}^{N-1}\left|c(x,y) - r(x+d_x, y+d_y)\right| \quad (2)$$

Seven 100 frame long 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 motions. "Susie", "Flowers", and "Table Tennis" video sequences are up-scaled versions of the commonly used CIF (352x288) sized benchmark videos. The up-scaling is done using the biqubic interpolation technique.

In our simulations, among the previously proposed fast search algorithms only NTSS and FSS have a search range of (±16, ±16) pixels and OS algorithm has a search range of (±24, ±24) pixels. The other fast search algorithms have a search range of (±48, ±24) pixels. FS is performed for both of the search ranges (±16, ±16) and (±48, ±24) pixels.

As it can be seen in Table II and Table III, DVSS algorithm clearly outperforms successful fast search ME algorithms and it performs very close to FS algorithm by searching much fewer search locations. For example, even though, FS algorithm with (±48, ±24) search range checks 4753 search locations in comparison to 405 search locations checked by the search pattern A1, its MAD performance is only 7.5% better than the performance of the search pattern A1 on the average. It can also be seen that the performance of FS algorithm with (±16, ±16) search range is very low for videos with large motion content.

The performance gap between other fast search algorithms and proposed search algorithms increase with increased video resolution and increased motion between consecutive frames. On the other hand, as it can be seen from "Foreman" video, when the resolution is very low and the motion can be detected within a search range of (±16, ±16) pixels, the performance gap decreases.

DVSS algorithm decreases the computational complexity significantly with a small decrease in the MAD performance. It even sometimes gives better MAD results than the pattern A1. The reason for this improvement is that search patterns with finer granularities perform better for small motions and DVSS algorithm dynamically decreases its granularity when small MVs are found for the previous blocks.

III. Proposed Reconfigurable ME Hardware

Top-level block diagram of the proposed ME hardware architecture is shown in Fig. 4. This ME hardware is based on the ME hardware we proposed in [17]. The hardware is highly pipelined and its latency is eight clock cycles; one cycle for synchronous read from memory, one cycle for shift registers, two cycles for the reconfigurable systolic Processing Element (PE) array and four cycles for the adder tree.

The proposed ME hardware finds a MV for a 16x16 MB based on the minimum SAD criterion in a maximum search range of (±48, ±24) pixels using the luminance data. The top-level controller takes the threshold level (τ) as an input and determines the number of search steps and their search ranges for each block adaptively. The control unit finds the MV for each block 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 step. On the other hand, the ME hardware proposed in [17] takes the number of search steps and their search ranges as input and performs same number of steps with same size search ranges for all blocks.

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 step ends after SAD values of the search locations in the bottom line of the search window are calculated. The next step around the search location with the minimum SAD is done in the same way.

16 BlockRAMs (BRAM) in the FPGA are used to store the search window. BRAMs are configured as dual port memories for overlapping ME of the current MB with loading of the search window of the next MB. The vertical rotator is used to align the outputs of the BRAMs and it has 32 identical rotators each 16 bit long. The reference MB data read from BRAMs must be matched with the current MB data, which is loaded into the 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 the fourth BRAM.

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.

A. Reconfigurable Systolic PE Array

The reconfigurable systolic PE array is shown in Fig. 5. This PE array is based on the PE array we proposed in [17]. 256 PEs are used to calculate the SAD of a 16x16 MB. A PE is used to calculate the absolute difference between a current pixel and the corresponding reference pixel. The latency of the PE array is two clock cycles, since reference and current pixel inputs and the absolute difference output are registered.

The reconfiguration 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 [16] is not reconfigurable, these multiplexers bring a slight area overhead in comparison to the PE array in [16], but they do not affect the clock frequency since they are not placed on the critical path. In Fig. 5, interconnects used for implementing 4, 2 and 1 shift amounts are illustrated with dashed, thin and bold lines respectively. Interconnects marked with "m" are connected to the memory.

The reference pixels for the first search location in a line of the search window are loaded in 4 clock cycles. After the SAD value of the first search location is calculated, the SAD value of the next search location is calculated in 1 cycle. After the SAD value of the first search location is calculated, 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 type of the step; coarse, medium or fine respectively.

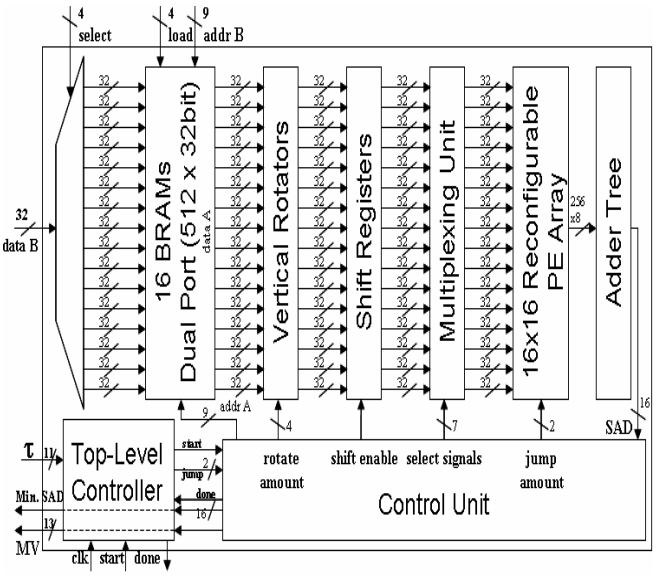

Fig. 4. Top-level Block Diagram

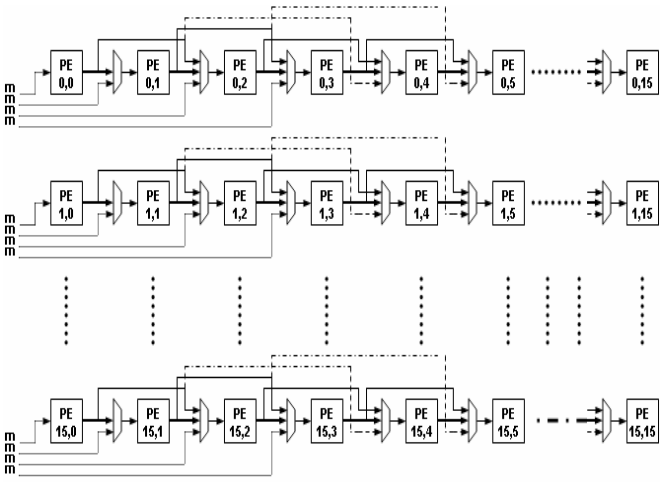

Fig. 5. Reconfigurable Systolic PE Array

TABLE IV
DATAFLOW THROUGH RECONFIGURABLE SYSTOLIC PE ARRAY

Clock Cycle	Processing Elements															
	(0,0) to (0,15)	(1,0) to (1,15)	(2,0) to (2,15)	(3,0) to (3,15)	(4,0) to (4,15)	(5,0) to (5,15)	(6,0) to (6,15)	(7,0) to (7,15)	(8,0) to (8,15)	(9,0) to (9,15)	(10,0) to (10,15)	(11,0) to (11,15)	(12,0) to (12,15)	(13,0) to (13,15)	(14,0) to (14,15)	(15,0) to (15,15)
1	D	C	B	A												
2	H	G	F	E	D	C	B	A								
3	L	K	J	I	H	G	F	E	D	C	B	A				
4	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
5	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
6	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
7	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G

The data flow through the reconfigurable systolic PE array is shown in Table IV. Let A – Z shown in Fig. 2 denote all pixels in these columns respectively. Assuming that "P" is the first search location, 4 clock cycles will be required to feed the reference data for this search location to the PE array, because regardless of the search pattern during loading of reference pixels for the first search location multiplexing unit feeds first four columns of the PE array. Assuming that after "P", the search pattern continues with search locations "R, T and V" (two pixel gap between consecutive search locations), multiplexing unit will feed only first two columns of the PE array. Therefore, reference pixels for these search locations will be in the PE array in 5th, 6th and 7th clock cycles respectively.

B. Memory Organization

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 the 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 an FPGA without data-reuse. The ME hardware proposed in [16] 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 same as the ME hardware proposed in [17]. The proposed ME hardware 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. Storing a line of reference pixels uses 28 address locations; therefore, addresses 0-111 are occupied to store four lines of reference pixels. Fig. 6 shows the layout of the reference pixels in the first BRAM, which stores 0th, 16th, 32th and 48th lines of the reference pixels in four distinct regions. The remaining BRAMs have the same organization.

Multiplexing unit shown in Fig. 7 is used to feed the correct data to the 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 the 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 the 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.

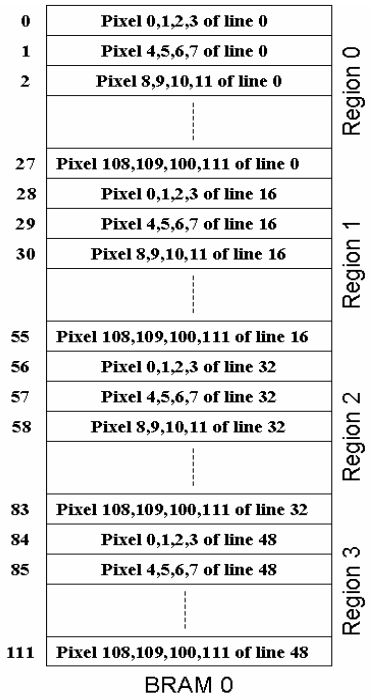

Fig. 6. Memory Organization

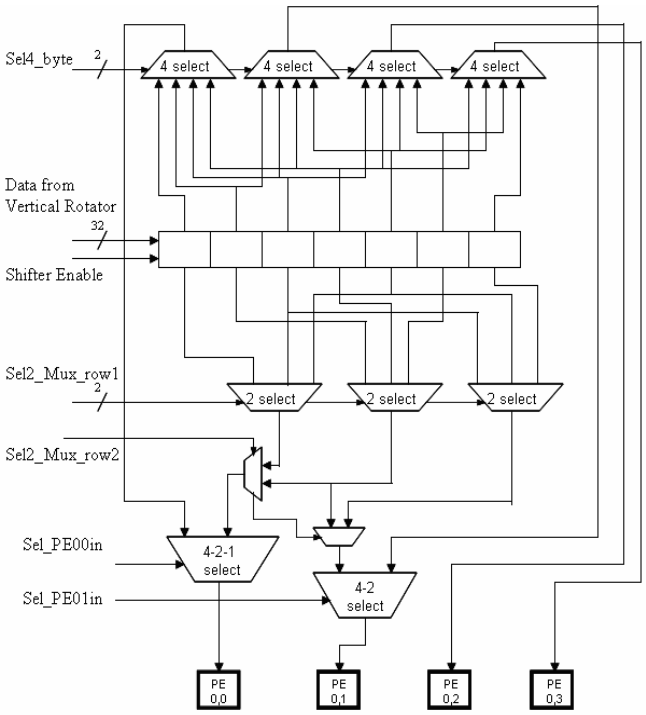

Fig. 7. Multiplexing Unit

The data received from the vertical rotator is captured in a 56 bit long shift register. If the enable signal of the shift register is high, it shifts its content 32 bits to right. If the distance between two search locations is four pixels, "4 select" multiplexers otherwise "2 select" multiplexers are used to select the corresponding reference pixels from the shift register.

C. Implementation Results

The proposed hardware architecture is implemented in VHDL, verified by simulation using Modelsim 6.3c, and mapped to an XC3S1500-5 FPGA using Synplify Pro 8.9 and ISE 10.1. The proposed hardware works at 130MHz and consumes 9128 slices (2282 CLBs) and 16 BRAMs. The reconfigurable systolic PE array with the adder tree consumes 7510 slices.

The number of clock cycles per MB required by the proposed hardware depends on the search pattern. Starting a step has a start-up cost of 15 clock cycles, which is called *step 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 (3). The performance of proposed ME hardware for several search patterns are calculated based on (3) and shown in Table V.

$$\sum_{1}^{n_s}\left(\tau_s+\left(n_{sad}-1\right)\times n_{line}+\left(n_{line}-1\right)\times\tau_{line}\right) \qquad (3)$$

In (3), "n_s, n_{sad}, n_{line}" are the number of steps, search locations per line and lines per step respectively. "τ_s" and "τ_{line}" are step and line latencies respectively. Based on this equation, for the coarse, medium and fine steps the start-up latency is 45 clock cycles. For these three steps, there is 192 clock cycles of line latency and 396 clock cycles are required for remaining search locations. Therefore, pattern A1 requires 633 clock cycles to find the MV of a MB. Patterns A2 and A3 requires 357 and 380 clock cycles, respectively. FS with a search range of (±10, ±5) pixels requires 304 clock cycles.

The performance of DVSS algorithm on the proposed ME hardware for different threshold values is shown in Table VI. DVSS algorithm achieves much better real-time performance, with a small decrease in the MAD performance, since it adaptively changes the search patterns and uses pattern A1 only for large motions, patterns A2 and A3 for medium motions and FS only for small motions. As it can be seen in the table, increasing the threshold value increases the supported frame rate.

TABLE V
PERFORMANCE OF PROPOSED HARDWARE FOR SEVERAL PATTERNS

Search Pattern	Required Clock Cycles per MB	Processed MBs per Second	Supported Frame Size & Rate
A1 [17]	633	205371	1920x1080, 25.3 fps
B	957	135841	1366x768, 33.1 fps
C	1221	106470	1366x768, 25.9 fps
10x9 [16]	122	1180327	1920x1080, 145.7 fps
14x15 [16]	236	610169	1920x1080, 75.3 fps
32x16 [16]	672	214285	1920x1080, 26.4 fps
48x24 [16]	1425	101052	1366x768, 24.6 fps
FS	5103	25475	720x576, 15.7 fps

TABLE VI
PERFORMANCE OF PROPOSED HARDWARE FOR DVSS ALGORITHM

Video Sequence	Threshold (τ)	Required Cycles for 100 Frames	MBs per Frame	Average Cycles per MB	Supported 1920x1080 fps
Spider	256	96094246	1620	594	27.0
Spider	1024	90284377	1620	558	28.7
Gladiator	256	87299334	1620	539	29.7
Gladiator	1024	80952068	1620	500	32.1
Irobot	256	77966499	1620	482	33.3
Irobot	1024	74177157	1620	458	35.0
Susie	256	59212520	1320	449	35.7
Susie	1024	51666864	1320	392	41.0
Flowers	256	52181938	1320	396	40.5
Flowers	1024	49586582	1320	376	42.7
TableTennis	256	53382291	1320	405	39.6
TableTennis	1024	47136775	1320	358	44.9
Foreman	256	15926153	396	403	39.9
Foreman	1024	14250681	396	360	44.5

TABLE VII
COMPARISON OF ME HARDWARE ARCHITECTURES

Architecture	Algorithm	Technology	MB size	Number of PEs	Search Range	Area	Speed [MHz]	Cycles per 16x16 MB	Supported 1920x1080 [fps]
Proposed	DVSS	XC3S1500-5 FPGA	16x16	256	(±48, ±24)	2282 CLBs	130	467 ($\tau = 256$)	34.3 ($\tau = 256$)
[14]	FTS	XC3S5000 FPGA	16x16	16	(±16, ±16)	6142 CLBs	74	202	45.2
[15]	FS & DS	Unknown	8x8, 16x16	Dedicated HW	(-16, +15) in both axis	9K gates	50	2879 (average)	2.1
[16]	32x16 [16]	XC3E1200E-5 FPGA	16x16	256	(±32, ±16)	1692 CLBs	144	672	26.4
[17]	A1 [17]	XC3S1500-5 FPGA	16x16	256	(±48, ±24)	2271 CLBs	130	633	25.3
[18]	FS	0.25µm CMOS 1P5M	16x16	256	(-16, +15) in both axis	16.07 mm^2	36	1421	3.1
[19]	FS	0.6µm SPTM CMOS	8x8, 16x16, 32x32	64	(±32, ±32)	267K gates	60	4209	1.7
[20]	FS	XC4VLX100 FPGA	16x16	Dedicated HW	(±16, ±16)	380 LUTs	221	1111	24.5
AS1 [21]	FS	XC40250 FPGA	16x16	33	(±16, ±16)	1214 CLBs	24	25344	0.1
AB2 [21]	FS	XC40250 FPGA	16x16	256	(±16, ±16)	948 CLBs	30	1584	2.3
AS2 [21]	FS	XC40250 FPGA	16x16	528	(±16, ±16)	3732 CLBs	22	768	3.5

The proposed ME hardware is compared with several ME hardware implementations presented in the literature in Table VII. A small number of hardware architectures for fast search ME algorithms are proposed in the literature [14-17]. The proposed ME hardware consumes less area than the implementation of one of the best performing fast search ME algorithms in the same FPGA [14]. The MAD performance of this hardware is lower than the MAD performance of the proposed ME hardware, since it implements FTS algorithm. In [15], a hybrid architecture supporting both FS and DS is presented. This architecture speeds up FS by successively eliminating some of the search locations. In addition, it is suitable for the irregular data flow of fast search algorithms and it consumes less area than the dedicated FS systolic array implementations. However, it has lower throughput than the proposed ME hardware.

The proposed ME hardware is based on the systolic ME hardware proposed in [16, 17]. The major differences between them are the proposed hardware is dynamically reconfigurable and it implements DVSS algorithm. Because of the overhead of reconfigurability and additional complexity of the control unit, the proposed ME hardware consumes 2363 slices more than the ME hardware proposed in [16] in the same FPGA. 1136 slices are used by the multiplexing unit, 836 additional slices are used by the multiplexers in the PE array and the remaining additional slices are used by additional complexity of the control unit. Because of the overhead of the dynamic reconfigurability, which is implemented in the top-level controller, the proposed ME hardware consumes slightly more area than the ME hardware proposed in [17] in the same FPGA.

Many hardware architectures for FS algorithm are proposed in the literature [18-21]. The throughput of the proposed ME hardware is much higher than the FS hardware implementations in [18-19]. An ASIC implementation of FS algorithm utilizing 256 PEs in 0.25µm CMOS technology is given in [18]. This architecture is a modified version of the AB2 type systolic array. Another ASIC implementation of FS algorithm is given in [19]. The throughput of this architecture is low, because it has only 64 PEs, it is optimized for low power consumption and it is implemented in an older technology.

A real-time ME hardware implementing FS algorithm for HD video is given in [20]. However, since this hardware is implemented on a high-end FPGA, it is not suitable for consumer electronics products. The FPGA implementations of the systolic architectures AS1, AB2, AS2 are presented in [21]. Despite using large number of PEs, the throughputs of these ME hardware are much lower than the throughput of the proposed ME hardware, since they are implementing FS algorithm. The area results presented in [21] include only the datapath and do not include the control unit and memory.

IV. CONCLUSION

In this paper, DVSS ME algorithm for processing HD video formats and a dynamically reconfigurable systolic ME hardware architecture for efficiently implementing DVSS algorithm are proposed. For each block, the proposed ME hardware is dynamically reconfigured to execute different number of steps and different search ranges for each step based on the previously found MV for the left neighboring block. The simulation results showed

that DVSS algorithm performs very close to FS algorithm by searching much fewer search locations than FS algorithm and it outperforms successful fast search ME algorithms by searching more search locations than these algorithms. The proposed ME hardware consumes less area than the implementation of one of the best performing fast search ME algorithms in the same FPGA. The proposed ME hardware is capable of processing HD video formats in real time and its throughput is much higher than the FS hardware implementations reported in the literature. Therefore, DVSS ME algorithm and the proposed dynamically reconfigurable ME hardware can be used in consumer electronics products for video compression, frame rate conversion and de-interlacing.

REFERENCES

[1] I. Richardson, *H.264 and MPEG-4 Video Compression,* Wiley, 2003.

[2] 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, pp. 1759-1767, Nov. 2007.

[3] 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. Consumer Electronics*, vol. 54, no.2, pp. 863-869, May 2008.

[4] Y.-Y. Jung, S. Yang, and P. Yu, "An effective de-interlacing technique using two types of motion information," *IEEE Trans. Consumer Electronics*, vol. 49, no.3, pp. 493-498, Aug. 2003.

[5] S.-G. Lee and D.-H. Lee, "A motion-adaptive de-interlacing method using an efficient spatial and temporal interpolation," *IEEE Trans. Consumer Electronics*, vol. 49, no.4, pp. 1266-1271, Nov. 2003.

[6] A. Puri, H. M. Hang, and D. L. Schilling, "An efficient block matching algorithm for motion compensated coding," *ICASS*P, pp. 1063-1066, April 1987.

[7] R. Li, B. Zeng, and M.L. 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.

[8] 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, Jun. 1996.

[9] L. K., Liu and E. Feig, "A block-based gradient descent search algorithm for fast block-matching motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no.4, pp. 419–422, Aug. 1996.

[10] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *IEEE Trans. on Image Processing*, vol. 9, no.2, pp. 287–290, Feb. 2000.

[11] C. Zhu, X. Lin, and L. P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no.5, pp. 349–355, May 2002.

[12] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Trans. on Image Processing*, vol. 11, no. 12, pp. 1442–1449, Dec. 2002.

[13] 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.

[14] 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*, Greece, May 2006.

[15] W. M. Chao, C. W. Hsu, Y. C. Chang, and L. G. Chen, "A novel motion estimator supporting diamond search and fast full search," *IEEE ISCAS*, Arizona, U.S.A., May 2002.

[16] 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*, Rhodes, Greece, Oct. 2008.

[17] O. Tasdizen, H. Kukner, A. Akin, and I. Hamzaoglu, "A high performance reconfigurable motion estimation hardware architecture", *IEEE DATE Conference*, Nice, France, Apr. 2009.

[18] N. Roma and L. Sousa, "Efficient and configurable full-search block-matching processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1160-1167, Dec. 2002.

[19] J.-F. Shen, T.-Chich Wang, and L.-G. Chen, "A novel low-power full-search block-matching motion-estimation design for H.263+," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no.7, pp. 890–897, Jul. 2001.

[20] A. Saha and S. Ghosh, "A speed-area optimization of full search block matching hardware with applications in high-definition TVs (HDTV)," HiPC 2007, LNCS 4873, pp. 83-94, 2007.

[21] A. Ryszko, K. Wiatr, "An assesment of FPGA suitability for implementation of real-time motion estimation," *EUROMICRO Conference on DSD,* Warsaw, Poland, pp. 364-367, Sep. 2001.

[22] V. G. Moshnyaga, "A New computationally adaptive formulation of block-matching motion estimation," *IEEE Trans.Circuits Syst. Video Technol.*, vol. 11, no. 1, pp. 118–124, Jan 2001.

[23] S. Saponara and L. Fanucci, "Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems," *IEE Proceedings - Computers and Digital Techniques,* vol. 151, no 1, pp. 51-59, Jan. 2004.

Özgür Taşdizen received B.S. degree in Electrical and Electronics Engineering from Istanbul Technical University, Istanbul, Turkey, in June 2003. He received M.S. degree in Electronics Engineering from Sabanci University, Istanbul, Turkey, in August 2005. From September 2005 to August 2006 he worked in TUBITAK-UEKAE, Izmit, Turkey, as a Digital Design Engineer. In September 2006, he joined Vestek Electronic R&D Corp., Istanbul, Turkey, where he is currently working as a Senior Digital Design Engineer. He is currently also studying towards a Ph.D. degree in Electronics Engineering at Sabanci University, Istanbul, Turkey. His research interests include ASIC and FPGA design for video compression and video enhancement, and low power digital design.

Abdülkadir Akın received B.S. degree in Electronics Engineering from Sabanci University, Istanbul, Turkey in July 2008. He is currently working towards an M.S. degree in Electronics Engineering at Sabancı University, Istanbul, Turkey. His research interests include digital hardware design for video compression and video enhancement.

Halil Kükner received B.S. degree in Electronics Engineering from Sabanci University, Istanbul, Turkey in July 2008. He is currently working towards an M.S. degree in Electronics Engineering at Delft University of Technology, Netherlands. His research interests include digital hardware design for video compression and video enhancement.

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