Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 186057, 12 pages
doi:10.1155/2012/186057

Research Article

DMPDS: A Fast Motion Estimation Algorithm Targeting High Resolution Videos and Its FPGA Implementation

Gustavo Sanchez,[1] Felipe Sampaio,[2] Marcelo Porto,[1, 2] Sergio Bampi,[2] and Luciano Agostini[1]

[1] *Group of Architectures and Integrated Circuits (GACI), Federal University of Pelotas (UFPEL), 96010-610 Pelotas, RS, Brazil*
[2] *Microelectronics Group (GME), Federal University of Rio Grande do Sul (UFRGS), 90040-060 Porto Alegre, RS, Brazil*

Correspondence should be addressed to Gustavo Sanchez, gustavofreitassanchez@gmail.com

Received 2 May 2012; Accepted 9 October 2012

Academic Editor: Michael Hübner

This paper presents a new fast motion estimation (ME) algorithm targeting high resolution digital videos and its efficient hardware architecture design. The new Dynamic Multipoint Diamond Search (DMPDS) algorithm is a fast algorithm which increases the ME quality when compared with other fast ME algorithms. The DMPDS achieves a better digital video quality reducing the occurrence of local minima falls, especially in high definition videos. The quality results show that the DMPDS is able to reach an average PSNR gain of 1.85 dB when compared with the well-known Diamond Search (DS) algorithm. When compared to the optimum results generated by the Full Search (FS) algorithm the DMPDS shows a lose of only 1.03 dB in the PSNR. On the other hand, the DMPDS reached a complexity reduction higher than 45 times when compared to FS. The quality gains related to DS caused an expected increase in the DMPDS complexity which uses 6.4-times more calculations than DS. The DMPDS architecture was designed focused on high performance and low cost, targeting to process Quad Full High Definition (QFHD) videos in real time (30 frames per second). The architecture was described in VHDL and synthesized to Altera Stratix 4 and Xilinx Virtex 5 FPGAs. The synthesis results show that the architecture is able to achieve processing rates higher than 53 QFHD fps, reaching the real-time requirements. The DMPDS architecture achieved the highest processing rate when compared to related works in the literature. This high processing rate was obtained designing an architecture with a high operation frequency and low numbers of cycles necessary to process each block.

1. Introduction

Nowadays digital video compression is really a relevant issue. It happens due to the growing development of applications that handle high definition videos, as smart phones, digital cameras, tablets, and so on. These applications would not be possible without video compression. The video bitstream must be drastically reduced to enable the transmission and storage, especially when high definition videos must be processed in real-time.

In a digital video there is a lot of redundant information, and this redundancy is explored in the current video coder standards. Neighbor blocks in a frame usually have very similar pixel colors and intensity and the intraframe prediction of the current video coders explore this type of redundancy. The transforms and quantization also contribute to reduce the intraframe redundancy, but in this case, in the frequency domain. In a set of neighbor frames, the information is also very similar among them, because the video is encoded at a frame rate of at least 24–30 frames per second, so neighbor frames tend to be very similar. This redundancy is explored by the interframe prediction operation. The redundancy on the bitstream representation is also explored by the current video coder standards through the entropy coding operation. Our work is focused on the interframe prediction.

Motion Estimation (ME) is the main operation of the interframe prediction and it represents about 80% of the total computational complexity of current video coders [1]. The ME must find the best matching in the reference frames for each block of the current frame, defining a motion vector indicating where the best matching was found. A search algorithm defines how the search is done and a similarity criterion is used to compare the candidate blocks with each original block. A search area is defined inside the reference frame around the collocated block to constrain the ME complexity. The collocated block is that block in the reference frame which is located at the same position of the original block that is currently being processed. The search for best vectors is known to be very expensive in terms of calculations and, consequentially, in terms of the processing time. The Full Search (FS) [2] algorithm explores all possibilities in a given search area, which implies in a very high computational cost, especially for high resolution videos, which requires the use of larger search areas.

The current video coding standards like MPEG2 [3] and H.264/AVC [4] and even the emerging HEVC (High Efficiency Video Coding) [5] standard do not restrict how the ME is done. Based on this fact, there is a vast space to explore new algorithm solutions for the ME. These solutions are evaluated according to the tradeoff between complexity and objective quality of the encoded digital video.

There are many fast search algorithms in scientific literature. These algorithms deal with this complexity, at different levels of impact in objective quality (PSNR). Generally, these algorithms exploit the characteristic of locality among temporal correlated blocks and they can achieve good results in terms of numbers of calculations. However, these algorithms assume that the error function decreases monotonically on the surface of the frame, in order to speed up the algorithm. This assumption does not hold true sometimes, and the search might be trapped into a local minima.

The majority of the published ME search algorithms only considers low resolution videos, as QCIF and CIF, in its experiments. However, the quality results of the ME algorithms can significantly change with the increasing of the video resolution. For low resolution videos, the quality results for FS and many other fast algorithms are very close. The great amount of pixels in high definition videos may lead the fast algorithms to choose, more frequently, local minima as the best matching. Thus, the quality losses (in comparison with FS) are significant in this scenario. Techniques to avoid local minima falls must be explored to enhance the video quality without a significant increase in the ME computational complexity.

In previous works of our group, new algorithms and hardware architectures for ME on high definition videos were presented, as [6, 7]. The Multipoint Diamond Search (MPDS) and the hardware solution targeting real-time for HD 1080 p (1920 × 1080 pixels) videos were presented in [6]. The Dynamic Multipoint Diamond Search (DMPDS) algorithm and the initial hardware design for this algorithm were presented in [7]. This paper presents detailed results related to the ME behavior on high resolution videos, showing the growth on the quality losses of the traditional fast algorithms with the increasing of the video resolution. The DMPDS algorithm is also presented and evaluated with detailed information. The results showed that the DMPDS algorithms can significantly improve the video quality when compared with traditional fast algorithms like Diamond Search (DS) [8], especially when high definition videos are considered. Finally, this paper presents the hardware design for the DMPDS algorithm, including the synthesis results targeting an Altera Stratix 4 [9] and a Xilinx Virtex 5 FPGAs [10]. The hardware design of the DMPDS algorithm targeted real-time processing for Quad Full High Definition (QFHD) (2160 × 3840 pixels) video sequences.

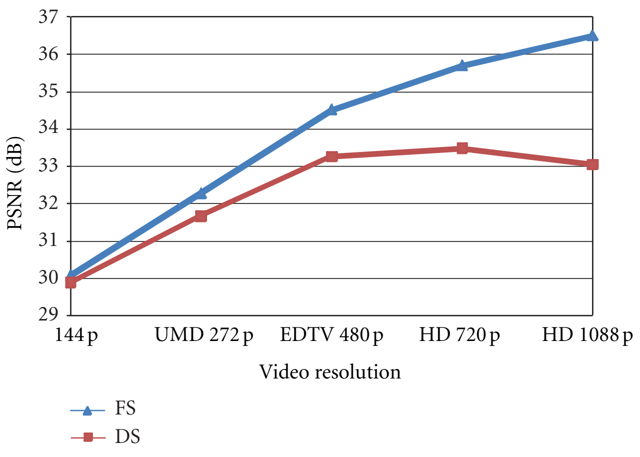

Figure 1: Average PSNR curves of DS and FS.

The paper is organized as follows. Section 2 presents an investigation about the ME in high definition videos. Section 3 explains the MPDS and DMPDS algorithms. Section 4 shows details about the architecture presented in this paper and Section 5 presents the results and comparisons with related works. Finally, Section 6 renders the conclusions.

2. Motion Estimation in High Definition Videos

The increasing in the video resolution can directly affect the ME results. The fast ME algorithms can be affected by this characteristic, generating different results, for the same video, in different resolutions. High resolution videos tend to present very similar neighboring pixels (much more than low resolution ones) and this fact contributes to increase the occurrence of local minima falls.

Diamond Search (DS) [8] and Full Search (FS) [2] algorithms were applied to ten HD 1080 p video sequences to demonstrate the influence of the video resolution growing in the ME quality. The used video sequences were blue_sky, man_in_car, pedestrian_area, hush_hour, station2, sunflower, riverbed, rolling_tomatoes, traffic, and tractor [11]. These sequences were resized for many lower resolutions: 256 × 144 pixels (144p—which is equivalent to QCIF resolution in a 16 : 9 aspect ratio); UMD 272 p (480 × 272); EDTV 480 p (854 × 480); HD 720 p (1280 × 720) and HD 1080 p (1920 × 1080). The search areas were defined proportionally with the resolution. All the experiments in this paper will consider the average results achieved for these ten video sequences.

Figure 1 presents the average PSNR curves of DS algorithm in red and FS algorithm in blue, considering the five different resolutions. The used block size was 16 × 16 pixels, and the search area grew proportionally to the video resolution growing.

In low resolution videos, DS and FS algorithms obtained almost the same PSNR results. However, through the analysis of Figure 1 curves it is possible to notice that the difference in PSNR results between DS and FS algorithms grows significantly with the video resolution increasing. The FS continuously increases the PSNR gains with the video resolution growing. This happens because the search area also grows and the FS algorithm can explore all candidate blocks in the search area. Analyzing the DS algorithm PSNR curve, the growing in resolution does not mean that the DS quality increases. In fact, the DS got a worse quality for 1088 p than when encoding 720 p or 480 p. It is explained because the DS can easily be trapped into a local minimum. The efficiency of the DS is reduced a lot for high definition videos, because there is a lot of similar information among neighbor pixels (and blocks).

These results show that the DS algorithm is efficient for low resolution videos, since it has PSNR results which are very similar to FS results, with a significant computational cost reduction. For high definition videos, the PSNR losses become more significant, and the relevance of the algorithm is only related with computational cost reduction. This experiment demonstrates that quality results for fast algorithms in low resolution videos cannot be extrapolated for a high definition video scenario.

The increase of local minima falls in high definition videos is the main reason why the DS algorithm loses its efficiency. The increase in the search area for the DS does not present much gain because in average the DS iterates only 5 times. Also, with the increase in the search area, the ME optimum candidate block can be far away from the center of the search area. Fast algorithms, like DS, can be easily trapped in local minima, before they achieve the optimum result.

A widely used similar criterion is the Sum of Absolute Difference (SAD). To perform the SAD, the block that has been encoded is subtracted from a candidate block and the absolute value of this operation is added. SAD maps are presented in Figure 2, to illustrate the growing of the local minima occurrences in high definition videos. In Figure 2 SAD maps are presented for a search area in the sun_flower video. Each map represents the same region of the frame, with a different number of pixels. Figures 2(a)–2(e) represents the SAD maps for the resolutions 144 p, UMD 272 p, EDTV 480 p, HD 720 p, and HD 1080 p, respectively. The images represent the SAD magnitude for 16×16 blocks, where dark blue represents lower SAD values, and light orange represents higher SAD values.

In Figure 2(a) it is possible to see that good SAD results can be achieved around the center of the search area; traditional fast algorithms like DS would easily reach the global minimum. With the increase of resolution, some regions, dark blue regions, that is, good SADs results, can be found within some distance from the central block. This proves that the video resolution increase is a problem if the used search algorithm cannot deal with local minima falls. More dark regions can be seen in Figure 2(c); however, in Figures 2(a)–2(c) the global minima can be visually identified. However, in Figure 2(d) and Figure 2(e), there are a lot of dark blue regions and it is impossible to visually identify the global minima.

The analyses of the images presented in Figure 2 can explain the results presented in Figure 1. Both DS and FS algorithms choose similar candidate blocks (or even the same) at low resolution videos. As presented in Figure 2(a), there are only a few candidate blocks with good SAD results in a low resolution video, and all around the center of the search area. Even if the DS does not reach the optimal candidate block, it will choose a closer block with a similar SAD value (average difference of 0.19 dB). For high resolution videos, like HD 1080 p, the higher number of local minima significantly increases the differences between PSNR results for DS and FS algorithms (more than 3.4 dB). This DS fragility to local minima falls should be explored by new algorithms to achieve a good quality result without increasing a lot the complexity like FS would.

For a better analysis about local minima, Figure 3 presents the SAD for every candidate block in a search area of 128×128 samples in the HD 1080 p sun_flower sequence. This picture represents a 3D view of the SAD magnitude for blocks with 16x16 samples, where valleys represent lower SAD values, and peaks represent higher SAD values.

In Figure 3 it is also possible to see that there are a lot of peaks and valleys in this search area. The global minimum is the valley with the lowest SAD. This scenario is different for low resolution videos, where the number of peaks and valleys is much lower than that presented in Figure 3. In high resolution videos, fast iterative algorithms (DS for example) are unable to transpose some peaks around the center to achieve a global minimum. Then fast algorithms are easily trapped in local minima around the center of the search area.

Due to the FS algorithms complexity, their implementation for high definition videos is very computationally expensive. The performance requirements to achieve real-time processing in this kind of video are extremely high. Hardware solutions for FS algorithm must massively explore parallelism to reach real-time, increasing the hardware recourses utilization and also power consumption. Fast algorithm implementations use much lower hardware resources and achieve a much better processing rate, but at a cost of an expressive loss in quality.

The development of new fast ME algorithms, focused in high definition videos, is very important to achieve a good tradeoff between quality and computational cost. The hardware implementation is also very important, mainly for real-time applications on portable devices. Fast algorithms must be easily implemented in hardware, translating their computational cost reduction to hardware resources reduction, and also power in savings.

3. The DMPDS Algorithm

The DMPDS algorithm is based on the MPDS algorithm [6]; however, DMPDS introduces the dynamic control of the multi point search engines position. The basic engine used by the DMPDS algorithms is the well-known DS algorithm. A multi point approach is used to find the best matching in five different positions of a search area, and the dynamic control defines where the search engines will be placed, according to the video motion activity characteristics. With this strategy it is possible to reduce the local minima falls and to increase the quality results.

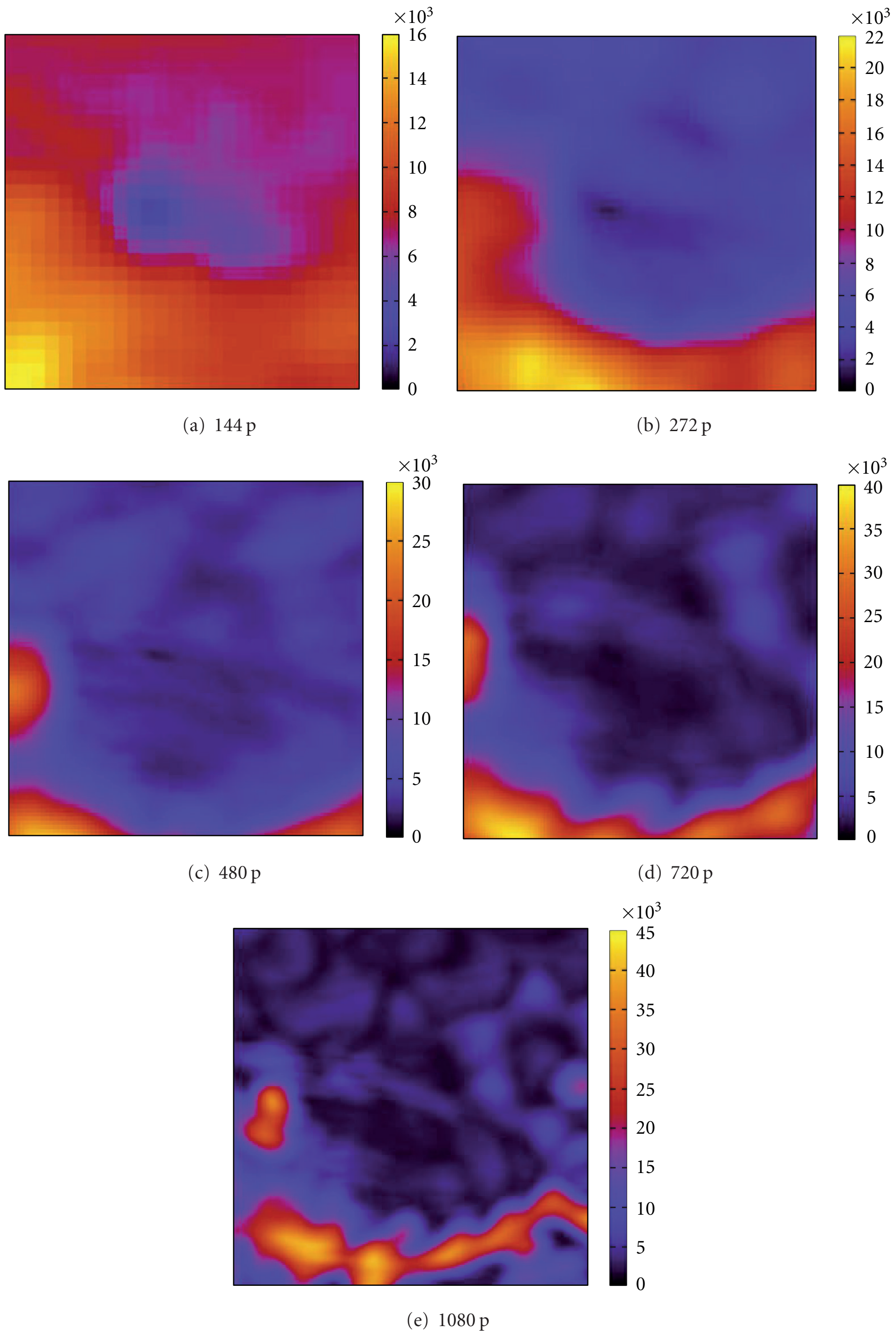

Figure 2: SAD maps for a sun_flower frame.

This section presents the Diamond Search and the Multipoint Diamond Search algorithms which are the basis for the Dynamic Multipoint Diamond Search proposed in this paper.

3.1. Diamond Search Algorithm. Diamond Search [8] is a fast and well-known search algorithm for ME. As previously explained, the DS achieves quite impressive results for low resolution videos; however, when dealing with high resolution videos, the quality degradation becomes higher.

The DS algorithm has two search patterns: the Large Diamond Search Pattern (LDSP) and the Small Diamond Search Pattern (SDSP) [8], as presented in Figure 4. The first used pattern is the LPDS. If a best match is found in an edge or a vertex of the LDSP, a new LDSP is formed, considering the best match of the previous LDSP as the center. This

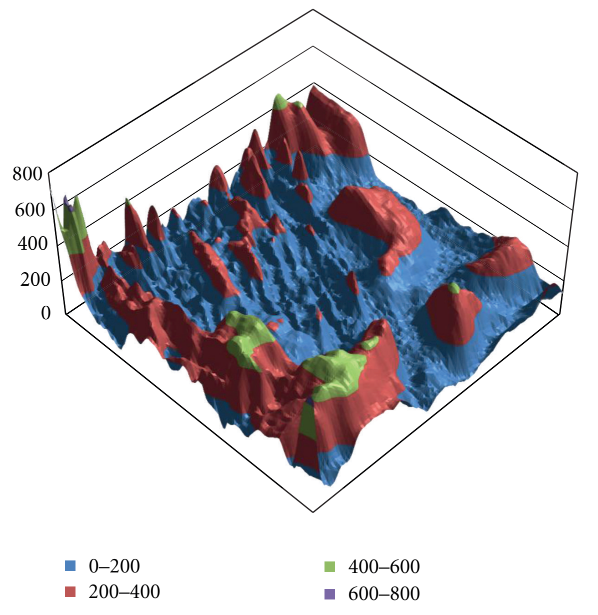

Figure 3: 3D SAD map for a 1080 p sun_flower frame.

process can be repeated many times. When the best match of an LDSP is found at the center, then all positions of the SDSP are evaluated to refine the result, considering the previous best result of LDSP as the center. The SDSP is applied just once while LDSP may be used many times and the best position found in the refinement is used to generate the motion vector.

3.2. Multipoint Diamond Search Algorithm. The main motivation to develop new fast algorithms to ME is related to the behavior of the traditional approach used in fast ME algorithms which does not reach good results for high resolution videos. The single iterative approach, as the one used in DS and other algorithms, is strongly susceptible to local minima falls when processing high resolution videos. Then new approaches are mandatory if the quality increase is intended in this type of applications. The MPDS algorithm was a first solution developed in our group [6] dealing with this challenge. The MPDS algorithm intends to generate quality results close to those generated by FS with a much lower number of calculations. In low resolution videos, the DS achieves good results ruining alone; however, the MPDS increases the DS computational complexity and it is able to achieve high quality on high definition videos.

The MPDS finds the best block matching in five different positions of a search area. Each position is the start point of a DS iterative search. Beyond the central point, other four points are defined, each one inside of a sector (A, B, C, and D), as presented in Figure 5. The MPDS is not restricted to only one start point, exactly to avoid the same local minimum reached by DS. In the worst case, the MPDS algorithm will reach the same results as the DS algorithm. This multi point approach is the technique that is used by MPDS to obtain a better quality when a high definition video has been encoded.

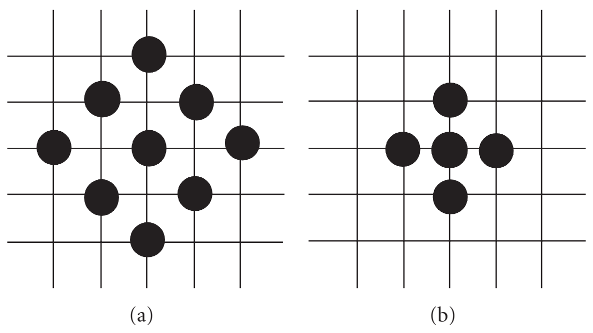

Figure 4: LDPS at the left and SDSP at the right [8].

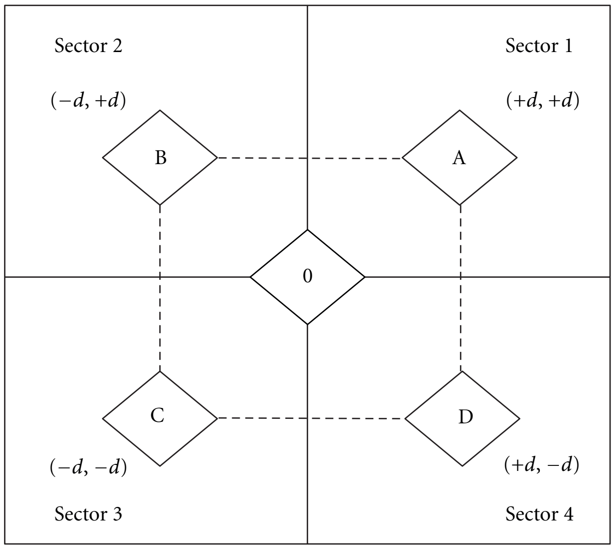

Figure 5: MPDS search in five regions of the search area.

Figure 5 describes the search positions of MPDS algorithm. Each initial search point is defined by its coordinates inside the sector. The point $(0, 0)$ is the central position and it will obtain the same vector as the DS algorithm. The search in the sectors A, B, C and D will be done according the distance parameter d. The d parameter is the distance (number of samples in X- and Y-axis) from the central point (0, 0). The sector A, B, C and D starts searching, respectively, at positions (d, d), $(-d, d)$, $(-d, -d)$ and $(d, -d)$. When the search ends the MPDS algorithm selects the best result from the five applied diamonds. Pseudocode 1 describes the MPDS algorithm.

The d parameter has a high impact on the results obtained by the MPDS algorithm. If a low motion activity video has been encoded, the MPDS algorithm would achieve good results if low values of d are used since the motion would be so small that there would be good candidate blocks near the origin. On the other hand, high motion activity videos would achieve better results using the opposite (high values of d). Figure 6 shows the PSNR curves for MPDS algorithm with each one of the ten used test video sequences cited before. The curves were generated considering the variation of the d parameter value, from zero to 40, considering blocks of 8×8 pixels. One can notice that when

```
(1) Define $d$
(2) Frame <= 0
(3) Block <= 0
(4) Repeat
(5)   Repeat
(6)     SAD_zero <= Execute_DS (0, 0)
(7)     SAD_A <= Execute_DS ($d$, $d$)
(8)     SAD_B <= Execute_DS ($-d$, $d$)
(9)     SAD_C <= Execute_DS ($-d$, $-d$)
(10)    SAD_D <= Execute_DS ($d$, $-d$)
(11)    Lowest_SAD <= Min (zero, A, B, C, D)
(12)    Generate_Vector (Lowest_SAD)
(13)    Block ++
(14)  While (Block<Max_Blocks)
(15)  Frame ++
(16) While (Frame<Max_Frames)
```

Pseudocode 1

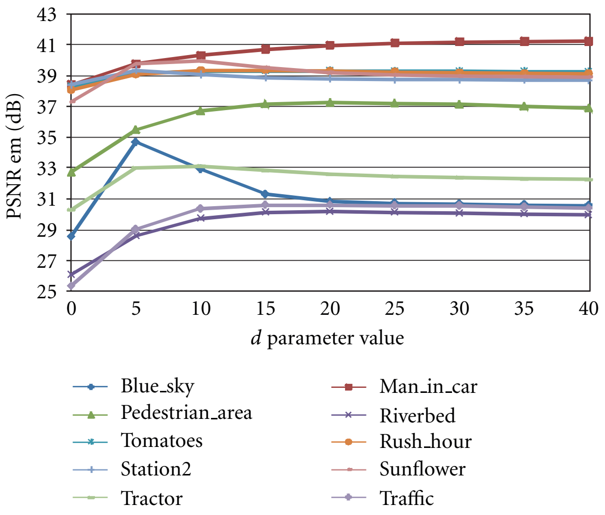

Figure 6: Variation of d parameter for MPDS algorithm.

$d = 0$ the MPDS applies all DS in the same position (i.e., in the correlated block) and the MPDS will obtain the same result obtained by a single DS.

From Figure 6 it is easy to notice that the MPDS algorithm obtains different results based on the d parameter value. The optimum value can significantly change according to the video motion characteristics. Low motion activity videos like blue_sky can reach best results using $d = 5$, with more than 6 dB gain in comparison to the DS algorithm. The video pedestrian_area has a lot of motion activities and reaches the best results with a $d = 20$, obtaining a quality gain superior to 4.5 dB in relation to DS algorithm.

The same experiment was made for 16×16 block sizes to identify the d value that maximize the average results of MPDS algorithm (considering the ten sequences). Using $d = 10$ the MPDS algorithm can achieve the best average gain in comparison to the DS algorithm (1.69 dB). This was the used value for d parameter on MPDS algorithm.

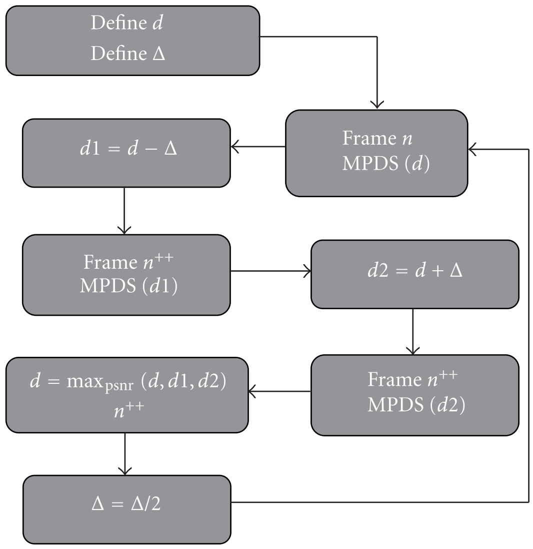

Figure 7: Dynamic control of the d parameter in the DMPDS algorithm.

3.3. Dynamic Multipoint Diamond Search Algorithm. With the analyses about the d parameter it was possible to notice that a fixed d parameter would not achieve the optimum quality result for all video sequences. The best d parameter value for the blue_sky sequence does not generate the best result for the traffic sequence, for example. It happens because these two sequences have different motion activity scenarios, which has significant influence in ME quality results. In this context, the DMPDS algorithm was developed, inserting a dynamic control of the d parameter of the MPDS algorithm. This makes the DMPDS algorithm more robust to handle videos with different motion activity scenarios.

The DMPDS defines the d parameter value dynamically. The variations of the d parameter are influenced by the characteristic of the current scene. For a low motion activity scene, the d parameter value can be dynamically reduced, resulting in better quality results. If the characteristic changes, and the motion activity is increased, the d parameter value can be dynamically increased. This dynamic adjustment enhances the robustness of the algorithm to deal with any kind of video sequences.

The algorithm to define the d parameter is presented in Figure 7. Firstly, an initial d value and a dynamic variation (Δ) are set. The first frame is processed with an original d parameter, the second frame with $d1 = d - \Delta$, and the third one with $d2 = d + \Delta$. The d used in the frame that obtains the lowest SAD becomes the new d and the dynamic variation (Δ) is divided by 2. This process is repeated until the dynamic variation reach the value 1, when the oscillation becomes 1 until the algorithm is restarted. This technique tries to approximate the optimal d for each frame using a low complexity heuristic. The start value of the dynamic parameter Δ is 5 and the start value of d is 10.

It is important to notice that the DMPDS algorithm could be executed in a parallel or in a sequential fashion. Each

Table 1: Quality and computational cost results for the DMPDS Algorithm.

Video	DS		MPDS		DMPDS		FS	
	PSNR (dB)	Number of ECB ($\times 10^9$)	PSNR (dB)	Number of ECB ($\times 10^9$)	PSNR (dB)	Number of ECB ($\times 10^9$)	PSNR (dB)	Number of ECB ($\times 10^9$)
Blue_sky	30.38	0.04	33.23	0.27	33.73	0.24	34.51	14.66
Man_in_car	38.15	0.03	39.41	0.22	39.60	0.24	40.34	14.66
Pedestrian_area	32.56	0.05	35.05	0.33	35.25	0.34	36.15	14.66
Riverbed	24.61	0.06	26.48	0.35	26.86	0.36	27.88	14.66
Rolling_tomatoes	37.76	0.03	38.27	0.25	38.32	0.28	38.65	14.66
Rush_hour	36.70	0.03	37.29	0.31	37.28	0.36	37.60	14.66
Station2	38.00	0.04	38.39	0.26	38.50	0.22	38.80	14.66
Sunflower	37.31	0.05	38.68	0.37	38.53	0.43	39.11	14.66
Traffic	25.10	0.06	29.05	0.38	28.81	0.39	33.38	14.66
Tractor	29.65	0.07	31.69	0.35	31.85	0.33	32.54	14.66
Average	33.02	**0.05**	**34.75**	**0.31**	**34.87**	**0.32**	35.90	**14.66**

sector can execute its search independently, and in parallel with the other sectors.

4. Software Evaluation

The DMPDS algorithm was described in C language and evaluated in a framework containing the ME and MC (Motion Compensation) processes. The framework just contained the ME and MC because the ME algorithm can be performed in many different standards like MPGE-2, H.264/AVC, or even in the emerging HEVC. For this reason no other features were added in this framework so the ME is able to be evaluated without a specific video coding standard, since no characteristics of a standard like fractional ME [12], entropy coding tools, and others are going to be evaluated. The multi point technique is useful to reduce the local minima falls and then it will reduce the residual information improving the overall encoder efficiency.

The ten previously presented 1080 p video sequences were used in this evaluation, with a 16×16 block size and a search range of $[-64, +64]$. The results for quality and computational cost of DS, MPDS, DMPDS, and FS algorithms are presented in Table 1. The computational cost is measured in a number of Evaluated Candidate Blocks (ECB) and the quality results are presented in PSNR.

The best DMPDS result in comparison to MPDS is achieved in the *blue_sky* video, where the DMPDS obtains a PSNR gain of about 0.50 dB. The DMPDS also presents a lower number of comparisons in this video and this is a significant result since the quality result can be improved with a reduction of the computational cost.

In some videos the MPDS achieves better results than the DMPDS, for example, in the rush_hour video. It happens because the number of frames encoded in this evaluation was only 200 and the DMPDS could not converge to the optimum point for the video sequence.

In the average results, the DMPDS achieves a PSNR 0.12 dB higher than MPDS with a minimum increase of complexity (about 3% more comparisons). However, the DMPDS could achieve a higher gain for real videos because there would be more frames in the same sequence (the 200 frames in the evaluation represent less than 7 seconds of a scene).

Table 2: Average results for DMPDS and DS with iteration restrictions and subsampling.

Algorithm	PSNR (dB)	Number of ECB ($\times 10^9$)	Number of Comp. ($\times 10^9$)
DMPDS	34.87	0.32	81.55
DS	33.06	0.05	12.33
DMPDS 4 : 1	34.65	0.30	19.29
DS 4 : 1	32.76	0.05	2.97
DMPDS 4 : 1 and 5 iterations	33.67	0.24	15.54
DS 4 : 1 and 5 iterations	31.42	0.04	2.64

Comparing to FS, the DMPDS achieves 1.03 dB lower PSNR; however, it is possible to reduce the number of ECBs in more than 45 times. One can notice that a good tradeoff between computational complexity and digital video quality was reached using the proposed DMPDS algorithm.

The DS algorithm does not have any limitation in the number of iterations. This is a problem if we want to design a real-time system. The proposed hardware solution should be able to guarantee that the real-time restriction should be reached, that is, at least 30 frames per second. Since the number of DS iterations is variable, and consequently the number of clock cycles are nondeterministic, an iteration limit should be set. Results targeting hardware implementation, considering the restriction in the number of iterations, are presented in Table 2. In this case, the evaluation considers a restriction of five LDSP iterations and only the DMPDS and the DS algorithms are considered with this evaluation. This restriction of five LDSP iterations was used in the hardware architecture design presented in the next section. Table 2 also presents the evaluation considering the use of 4 : 1 pixel subsampling. This is a useful technique to reduce the number of calculations with low impact in the quality results. The use

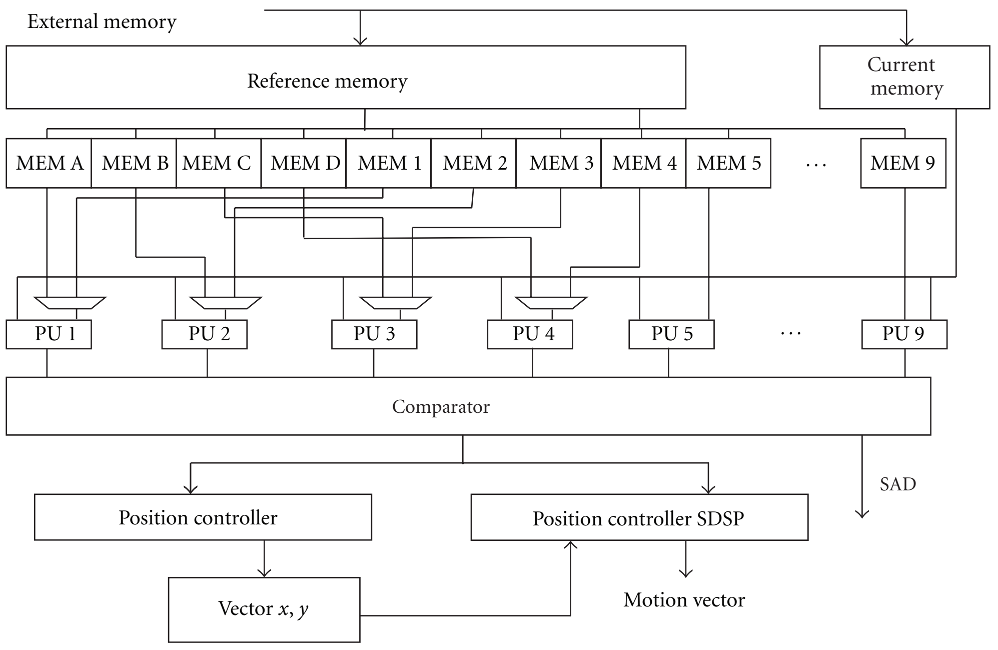

Figure 8: Block diagram of the DS core architecture.

of pixel subsampling is an interesting strategy especially for hardware implementations, since the internal memory and processing unities can be reduced. It helps to increase the processing rate without an impressive impact in the video quality.

The use of pixel subsampling in a 4 : 1 level together with the DMPDS algorithm reduces the number of ECBs and comparison operations about four times, as presented in Table 2. In this case, the quality losses are very small (less than 0.13 dB) and the number of comparisons is only 56.4% higher than DS without pixel subsampling, with an average PSNR gain of 1.59 dB.

The PSNR gain of the DMPDS algorithm over DS increases when the restriction in the number of iterations and the subsampling are considered. The DMPDS PSNR gain can reach 2.25 dB over DS considering the 4 : 1 subsampling and the restriction of five iterations. Another positive point is that the difference in the number of comparisons between DMPDS and DS is reduced in this scenario. In this case, the number of comparisons used by the DMPDS algorithm is six-times higher than the DS. If no iteration restriction or subsampling is considered, the DMPDS PSNR gain is of 1.81 dB and the number of comparisons is 6.4-times higher than DS.

Considering the good quality and complexity results achieved by the DMPDS algorithm with subsampling and the restriction of LDSP iterations, the designed hardware architecture considered these two simplifications, intending to reduce the hardware cost and increase its performance.

5. DMPDS Architecture

The DMPDS architecture works with a block size of 16×16 samples, and it uses a 4 : 1 pixel subsampling rate. Each one of the five DS cores is restricted to five iterations for the reasons presented in the last section. The DMPDS architecture performance is focused on real-time processing for QFHD videos, with low consumption of the hardware resources.

The DMPDS architecture is formed by five DS cores and some additional logic to dynamically control the d parameter, to define which is the best vector among the five cores and to group these DS cores. Figure 8 presents the block diagram of the DS architecture, which is strongly based on the DS architecture presented in [13] for this module. To start the ME process, it is necessary to fill the reference memory and the current memory. The reference memory is a 34×34 bytes memory, which can store all data for five iterations and the final refinement. The current memory has 8×8 bytes and it contains the block being processed.

This first memory reading process takes 34 cycles to fill the DS reference memory. However, when this memory is being filled, this memory is also being read by the local memories (8×8 bytes), which are responsible to store the data of each possible candidate in the current iteration. During the reference memory writing process, the processing units (PUs) also start to perform the SAD computation of the available blocks. Each PU calculates the SAD between the current block and a candidate block. The PU architecture was implemented in a pipeline with five stages, as presented in Figure 9.

When the PUs finish to compute the SADs, these SADs are compared and the position is updated. The process is repeated until the comparator finds the lowest SAD in the center of the DS (in PU 5). In this case the data for the final refinement is already stored in Local Memories A–D and in Local Memory 5, then the refinement can be computed without any additional access to the external memory.

The first LDSP takes 34 cycles to be processed and the next iterations can be performed in 27 cycles each because

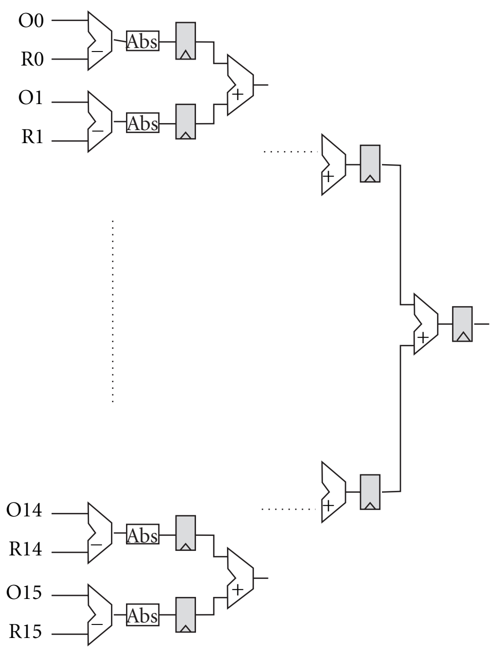

Figure 9: PU architecture.

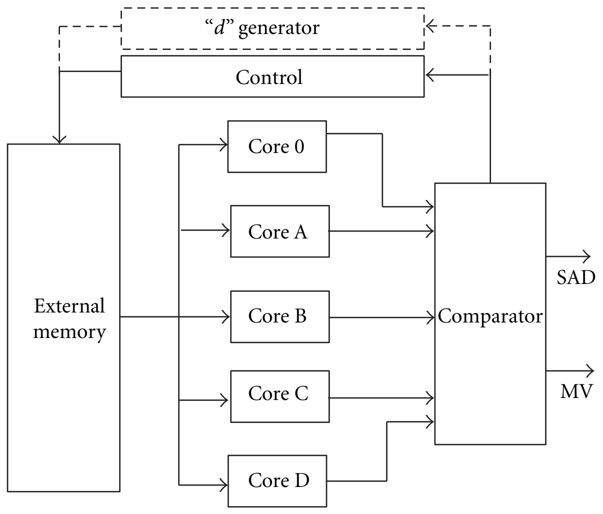

Figure 10: Block diagram of the DMPDS architecture.

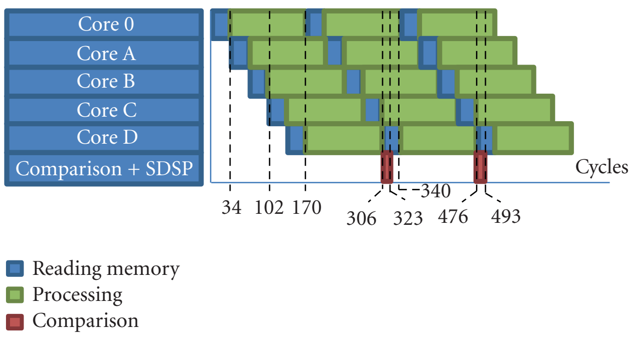

Figure 11: Clock cycles diagram.

there is no need to fill the reference memory again. In the worst case scenario, the DS is computed in 169 cycles (five LDSP iterations plus one SDSP refinement).

The block diagram of the DMPDS architecture is presented in Figure 10. Each core in Figure 10 is composed by a DS architecture presented in Figure 8. The comparator in Figure 10 is responsible to find the best SAD among the SADs of the five cores. The DMPDS architecture basically controls where to write the data received from the external memory and it informs the external memory which data is necessary. In advance to start the ME process, the Core O (in the center) starts to receive data from the off-chip memory. It is necessary for 34 cock cycles to fill the internal memories of this core. When these memories are completely filled, this core does not read additional data from the external memory until this block is completely computed and the next one is required.

This filling process is repeated for the memories of the next cores until the memories of core D are completely filled. In this moment, it is possible to start to fill again the memories of the O core with data for the next motion vector generation. After core D computes its SAD it is possible to define the best SAD among the five cores and the final motion vector.

Figure 11 presents the clock cycles diagram of the MPDS architecture. The DMPDS architecture has a latency of 170 cycles to fill all cores' internal memories. In the worst case scenario, the core D needs more 135 cycles to finish the SAD computation and more 17 cycles are necessary to perform the refinement of Core D; the comparator decides which are the best SADs and to generate the final motion vector. After the first motion vector is generated, only more 170 cycles necessary to generate the next one. It happens because when the first vector finishes it is processing, the cores O, A, B and C are already processing the next block. The DMPDS architecture efficiently explores the used hardware maintaining the cores operating during 169 of 170 cycles after the first iteration (considering the worst case), that is, the architecture was designed to be active during 99.4% of the time. The external memory bandwidth necessary for the DMPDS architecture is 34 bytes per cycle.

The control block is responsible to implement the state machine to model the control flow presented in Figure 7 (Section 2). This step is important to define the d parameter for the next frame. Since this operation is simple, it does not imply extra clock cycles and it can be done in parallel with other operations. The control block is responsible to correctly select which core is being fed from the external memory and to inform the external memory the value of d in the current iteration. Firstly, the architecture processes one frame with $d = 10$. The second frame will be processed with $d = 5$, and the third one will be processed with $d = 15$. Then the d is updated to the value that generated the lowest SAD among these three first frames and the dynamic parameter Δ becomes 2. This process is repeated with the new d and the new dynamic parameter until $\Delta = 1$.

6. Synthesis Results and Comparisons

The DMPDS architecture was described in VHDL and synthesized to an EP4S40G2F40I2 Altera Stratix 4 FPGA and to an XC5VLX30-3FF324 Xilinx Virtex 5 FPGA. The synthesis results are presented in Table 3. The performance results considered the worst case scenario, when all cores use five iterations. The DMPDS architecture synthesis results show that it can work with an operational frequency of 187.58 MHz (Stratix 4) and 294 MHz (Virtex 5).

The high operation frequency reached in both syntheses is function of the well-balanced pipelined designed in all architectural stages. The Stratix 4 synthesis used dedicated RAM blocks and also registers. The registers were used to implement the pipeline and the block RAMs were used to implement the memory hierarchy. The Virtex 5 synthesis used only registers to implement both the memory hierarchy and the pipeline.

Considering the reached operation frequency and since the designed architecture consumes 170 clock cycles to process one 16×16 block, then the DMPDS architecture is able to process 213.2 1080 p frames per second or 53.3 QHDTV frames per second.

The comparative results for the DMPDS architecture and some related works are presented in Table 4. The DMPDS architecture was compared with the results of related works [6, 14–19]. Table 4 presents some synthesis results, like the reached operations frequency, the use of hardware resources, and the use of memory bits. The used technology for each solution is also presented.

Table 4 also shows the number of cycles necessary to process one block and the reached processing rate in frames per second considering 1080 p (1920×1080 pixels) and QFHD (2160×3840 pixels) resolutions.

Since the related works are focused in different technologies and using different algorithms, a fair comparison is not easy. But the possible comparisons are presented in the next paragraphs.

The work [6] presents a previous work about the MPDS architecture. Comparing to [6], the area and memory usage are the same and the main difference is the reached operational frequency. However, the DMPDS architecture presents an average PSNR improvement of 0.12 dB when compared to [6].

The work [14] proposes a tree-engine architecture for fractional motion estimation (FME) with a variable block size (VBSME). The work [14] cannot process QFHD videos in real-time. Our solution uses a bigger and well-organized memory hierarchy to reach a better processing rate. DMPDS architecture also uses less hardware resources, but this was expected since it does not support FME and VBSME.

The work [15] performs the Dynamic Variable Step Search (DVSS) fast ME. The architecture in [15] uses the same block size of our work and it needs 467 cycles to process a block, which is much more than our work. Comparing the area results, the architecture presented in [15] uses less internal memory. However, our solution can process QFHD videos in real-time.

Table 3: DMPDS synthesis results.

Device	Frequency (MHz)	Area	Memory bits (K)	Register (K)
Stratix 4	187.58	34.5 K ALUTS	46.2	44.5
Virtex 5	294.00	56.3 K LUTS	—	110.0

The architecture presented in [16] is a configurable ME architecture and it performs the following algorithms: (1) Hexagon-Based-Search (HEXBS), (2) Block Based Gradient Descent Search (BBGDS), (3) Three Step Search (TSS) algorithms. It needs 390, 437, and 680 cycles to process each block with each algorithm, respectively. The DMPDS needs fewer cycles to process a block than any configuration of [16]. Our work uses more hardware resources comparing to [16], however, we are able to process QFHD videos while [16] can only process 1080 p videos in real-time.

The work [17] presents the hardware architecture for the Fast Top-Winners Search Algorithm. Our work uses almost the double of the memory used by [17]. However, our solution reaches a much better processing rate and it is able to process QFHD videos in real-time, while [17] cannot even reach real-time for 1080 p videos.

The work [18] presents an architecture for the Multiresolution ME Algorithm (MMEA). This solution can process only 28 1080 p frames per second while our DMPDS architecture can process 34 or 53.3 QFHD frames per second, depending on the target FPGA. But, is important to notice that this solution considers two reference frames in the ME process, which improves the ME quality.

The work [19] presents the Adaptive True Motion Estimation Algorithm (ATME) and it also uses techniques for Frame Rate UpConversion (FRUP). It also evaluates the algorithm for HD 1080 p videos. The hardware designed in [19] processes a 16×16 block in 104 cycles, which is less cycles than our architecture. However, our architecture reaches a much better operating frequency, which causes a final processing rate near to two-times higher than [19].

It is difficult to compare the quality results among the related works, since almost none of them (except [6, 19]) evaluate their algorithms and architectures for HD 1080 p videos. Also [14–18] do not mention any alternatives to avoid local minima falls, so probably these architectures will process high resolution videos with a worst quality if compared with our work.

Finally it is important to emphasize that only our work and the previous version of this work [6] are able to reach a frame rate of 30 QFHD fps among all related works.

7. Conclusions

This paper presented a new ME search algorithm and its hardware design. The Dynamic Multipoint Diamond Search (DMPDS) algorithm was developed focusing on high resolution videos.

An investigation about the traditional fast ME search approach shows that this type of approach loses efficiency

Table 4: Synthesis results and comparisons.

Architecture	Technology	Frequency (MHz)	Area	Memory (K bits)	Cycles per block	1080 p fps	QFHD fps
Porto et al. [6]	Stratix 4	199.2	34.5 KALUTS	46.2	170	144	36
Kao et al. [14]	180 nm	154	321 KGates	9.72	631	30	7.5
Tasdizen et al. [15]	Virtex 5	130	2282 KCLBs	0.51	467	34	8.5
Vanne et al. [16]	130 nm	200	14 KGates	2.5	390/437/680	63/56/36	15.75/14/9
Lai et al. [17]	180 nm	83.3	26 KGates	28.7	1282	8	2
Yin et al. [18]	180 nm	200	260 KGates	11.3	872	28	7
Cetin and Hamzaoglu [19]	90 nm FPGA	63	33 KLUTS	8 dual-port block RAM	104 (average)	74.7	18.7
DMPDS	**Stratix 4**	**187.58**	**34.5 KALUTs**	**46.2**	**170**	**136**	**34**
DMPDS	**Virtex 5**	**294**	**56.3 KLUTs**	—	**170**	**213.2**	**53.3**

when the resolution is increased. The results of this investigation is presented in this paper and shows that traditional search algorithms like Diamond Search may loses more than 3 dB in PSNR when compared to Full Search if high resolution videos are considered. The explanation about why this expressive quality loss occurs is also presented in this paper where it is possible to conclude that the losses are function of a fragility in the traditional search approach which are willing to fall in a local minimum near to the search area center without finding the global minimum.

The DMPDS algorithm was developed focusing on high quality when encoding high definition videos. The DMPDS algorithm uses a multipoint approach to reduce the number of local minima choices, in comparison with traditional fast ME algorithms. To reach this objective, five DS instances are started in different positions of the same search area. The DMPDS is an evolution of a previous algorithm developed in our group called MPDS. The main new idea of the DMPDS is to dynamically adapt the distance of each DS core inside a search area. This solution caused a PSNR gain of 0.12 dB with a negligible impact in the number of calculations. The DMPDS algorithm can reach a PSNR gain of 1.85 dB in comparison with the DS algorithm and DMPDS reaches a PSNR only 1.03 dB lower than that reached by FS. The DMPDS algorithm is also able to reduce in more than 45 times the number of calculations when compared to FS, but DMPDS used six-times more calculations than DS.

Some simplifications were inserted in the DMPDS algorithm intending to reduce the cost of the hardware design. These simplifications were the restriction in the number of iterations and the use of subsampling. These simplifications were evaluated and showed good results, with the DMPDS algorithm increasing the PSNR gains and reducing the complexity loses when compared to DS. These results are also presented in this paper.

The DMPDS hardware design was focused on-high throughput for high definition videos. To reach this performance goal, five instances of a DS architecture were grouped to work in parallel. Each DS architecture was designed using an efficient memory hierarchy and a well-balanced pipeline, intending to avoid unnecessary external memory accesses and to provide a high throughput. The control efficiently schedules the external memory accesses to allow a maximum use of the five DS cores, allowing a very high processing rate (similar to that obtained with a single instance of the DS architecture). In the worst case, the DMPDS architecture is able to process one block in only 170 clock cycles. The DMPDS architecture was synthesized targeting an Altera Stratix 4 and a Xilinx Virtex 5 FPGAs and the synthesis results shows that the DMPDS architecture was able to process up to 53.3 QFHD frames per second. This is the highest processing rates among all compared works and it is the function of the low number of cycles used to process each block and also to the high operation frequency reached.

References

[1] Y. S. Cheng, Z. Y. Chen, and P. C. Chang, "An H.264 spatio-temporal hierarchical fast motion estimation algorithm for high-definition video," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '09)*, pp. 880–883, May 2009.

[2] I. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*, Wiley, 2002.

[3] ITU-T e ISO/IEC JTC1, "Generic coding of moving pictures and associated audio information—Part 2: Video," ITU-T Rec. H.262 and ISO/IEC, 13818-2 (MPEG-2), 1994.

[4] T. Wiegand, G. Sullivan, and A. Luthra, Eds., Draft ITU-T Recommendation and final draft international standard of joint video specification (ITU-T Rec.H.264—ISO/IEC, 14496-10 AVC), 2003.

[5] JCT, Working Draft 3 of High-Efficiency Video Coding, JCTVC-E603, 2011.

[6] M. Porto, G. Sanchez, D. Noble, S. Bampi, and L. Agostini, "An efficient ME architecture for high definition videos using the new MPDS algorithm," in *Proceedings of the 24th symposium on Integrated circuits and systems design (ACM SBCCI '11)*, pp. 119–124, 2011.

[7] G. Sanchez, M. Porto, S. Bampi, and Agostini, "Real time QFHD motion estimation architecture for DMPDS algorithm," in *IEEE Southern Programmable Logic Conference*, 2012.

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

[9] Altera Corporation, "Altera: The Programmable Solutions Company," http://www.altera.com/.

[10] Xilinx, http://www.xilinx.com/.

[11] Xiph.org: Test media, 2011, http://media.xiph.org/video/derf/.

[12] M. M. Corrêa, M. T. Schoenknecht, R. S. Dornelles, and L. V. Agostini, "A high-throughput hardware architecture for the H.264/AVC half-pixel motion estimation targeting high-definition videos," *International Journal of Reconfigurable Computing*, vol. 2011, Article ID 254730, 9 pages, 2011.

[13] G. Sanchez, D. Noble, M. Porto, and L. Agostini, "High efficient motion estimation architecture with integrated motion compensation and FME support," in *Proceedings of the IEEE 2nd Latin American Symposium on Circuits and Systems (LASCAS '11)*, pp. 1–4, February 2011.

[14] C. Y. Kao, C. L. Wu, and Y. L. Lin, "A high-performance three-engine architecture for H.264/AVC fractional motion estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, pp. 662–666, 2010.

[15] O. Tasdizen, A. Akin, H. Kukner, and I. Hamzaoglu, "Dynamically variable step search motion estimation algorithm and a dynamically reconfigurable hardware for its implementation," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1645–1653, 2009.

[16] J. Vanne, E. Aho, K. Kuusilinna, and T. D. Hämäläinen, "A configurable motion estimation architecture for block-matching algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 4, pp. 466–476, 2009.

[17] Y. K. Lai, L. F. Chen, and S. Y. Huang, "Hybrid parallel motion estimation architecture based on fast top-winners search algorithm," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1837–1842, 2010.

[18] H. Yin, H. Jia, H. Qi, X. Ji, X. Xie, and W. Gao, "A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 9, pp. 1242–1254, 2010.

[19] M. Cetin and I. Hamzaoglu, "An adaptive true motion estimation algorithm for frame rate conversion of high definition video and its hardware implementations," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 923–931, 2011.