High Throughput Hardware Architecture for Motion Estimation with 4:1 Pel Subsampling Targeting Digital Television Applications

Marcelo Porto[1], Luciano Agostini[2], Leandro Rosa[2], Altamiro Susin[1], and Sergio Bampi[1]

[1] Microeletronics Groups (GME), UFRGS – Porto Alegre, RS, Brazil
{msporto,bampi}@inf.ufrgs.br br, altamiro.susin@ufrgs.br
[2] Group of Architectures and Integrated Circuits (GACI),UFPel – Pelotas, RS, Brazil
{agostini, lrosa.ifm}@ufpel.edu.br

Abstract. Motion estimation is the most important and complex operation in video coding. This paper presents an architecture for motion estimation using Full Search algorithm with 4:1 Pel Subsampling, combined with SAD distortion criterion. This work is part of the investigations to define the future Brazilian system of digital television broadcast. The quality of the algorithm used was compared with Full Search through software implementations. The quality of 4:1 Pel Subsampling results was considered satisfactory, once it presents a SAD result with an impact inferior to 4.5% when compared with Full Search results. The designed hardware considered a search range of [-25, +24], with blocks of 16x16 pixels. The architecture was described in VHDL and mapped to a Xilinx Virtex-II Pro VP70 FPGA. Synthesis results indicate that it is able to run at 123,4MHz, reaching a processing rate of 35 SDTV frames (720x480 pixels) per second.

Keywords: Motion estimation, hardware architecture, FPGA design.

1 Introduction

Nowadays, the compression of digital videos is a very important task. The industry has a very high interest in digital video codecs because digital videos are present in many current applications, such as: cell-phones, digital television, DVD players, digital cameras and a lot of other applications. This important position of video coding in the current technology development has boosted the creation of various standards for video coding. Without the use of video coding, processing digital videos is almost impossible, due to the very high amount of resources which are necessary to store and transmit these videos. Currently, the most used video coding standard is MPEG-2 [1] and the latest and more efficient standard is H.264/AVC [2]. These standards reduce drastically the amount of data necessary to represent digital videos.

A current video coder is composed by eight main operations, as shown in Fig. 1: motion estimation, motion compensation, intra-frame prediction, forward and inverse transforms (T and T^{-1}), forward and inverse quantization (Q and Q^{-1}) and entropy coding. This work focuses on the motion estimation, which is highlighted in Fig. 1.

D. Mery and L. Rueda (Eds.): PSIVT 2007, LNCS 4872, pp. 36 – 47, 2007.

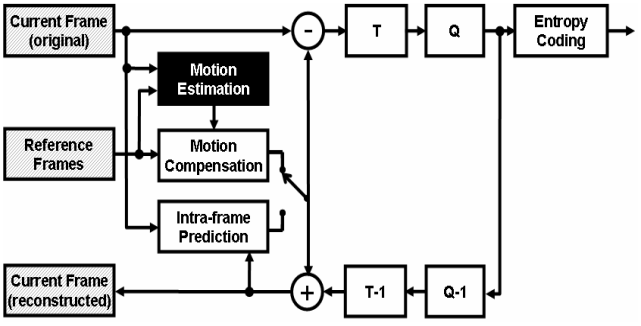

Fig. 1. Block diagram of a modern video coder

Motion estimation (ME) operation tries to reduce the temporal redundancy between neighboring frames [3]. One or more frames that were already processed are used as reference frames. The current frame and the reference frame are divided in blocks to allow the motion estimation. The idea is to replace each block of the current frame with one block of the reference frame, reducing the temporal redundancy. The best similarity between each block of the current frame and the blocks of the reference frame is selected. This selection is done through a search algorithm and the similarity is defined through some distortion criterion [3]. The search is restricted to a specific area in the reference frame which is called search area. When the best similarity is found, then a motion vector (MV) is generated to indicate the position of this block inside the reference frame. These steps are repeated for every block of the current frame.

Motion compensation operation reconstructs the current frame using the reference frames and the motion vectors generated by the motion estimation. The difference between the original and the reconstructed frame (called residue) is sent to the transforms and quantization calculation.

Motion estimation is the video coder operation that provides the highest gain in terms of compression rates. However, motion estimation has a very high degree of computational complexity and software implementations could not reach real time (24-30 frames per second) when high resolution videos are being processed.

This paper presents an FPGA based architecture dedicated to the motion estimation operation. This architecture used the Full Search with 4:1 Pel Subsampling (also called Pel Decimation) [3] as search algorithm, and the Sum of Absolute Differences (SAD) [3] as distortion criterion. ME design considered a search area with 64x64 pixels and blocks with 16x16 pixels. This implies a search range of [-25, +24]. The architecture was described in VHDL and mapped to Xilinx Virtex-II Pro FPGAs.

This work was developed within the framework in an effort to develop intellectual property and to carry out an evaluation for the future Brazilian system of digital television broadcast, the SBTVD [4]. The presented architecture was specifically designed to reach real time when processing standard definition television frames (720x480 pixels).

Section 2 of this paper presents the 4:1 Pel Subsampling search algorithm and the SAD criterion. Section 3 presents a software evaluation of the search algorithms used.

Section 4 presents the designed architecture, detailing its main modules. Section 5 presents the designed architecture for SAD calculation. Section 6 presents the synthesis results and comparison with related works. Finally, section 8 presents the conclusions.

2 Description of the Used Algorithm

This section presents some details about the Full Search with 4:1 Pel Subsampling search algorithm and about the SAD distortion criterion. The architectural design presented in this paper was based in these two algorithms.

2.1 Full Search with 4:1 Pel Subsampling Algorithm

Full Search with 4:1 Pel Subsampling algorithm is based on the traditional Full Search algorithm; however, the distortion criterion is not calculated for all samples. In 4:1 Pel Subsampling, for each pixels calculated, three pixels are discarded [3]. With this algorithm only a quarter of the block samples are calculated, increasing the performance and decreasing the complexity of the motion estimation operation. Fig. 2 shows the 4:1 Pel Subsampling relation for a block with 8x8 samples. In Fig. 2, the black dots are the samples which are used in the SAD calculation and the white dots are the samples which are discarded.

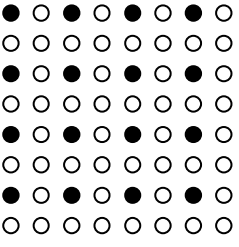

Fig. 2. 4:1 Pel Subsampling in an 8x8 block

2.2 SAD Criterion

Distortion criterion defines how the differences between the regions are evaluated. Many distortion criteria were proposed [3]; however, the most used for hardware design is the Sum of Absolute Differences (SAD). Equation (1) shows the SAD criterion, were **SAD(x, y)** is the SAD value for **(x, y)** position, **R** is the reference sample, **P** is the search area sample and **N** is the block size.

$$SAD(x,y) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|R_{i,j} - P_{i+x,j+y}\right| \quad (1)$$

3 Quality Evaluation

The search algorithm defines how the search for the best match will be done in the search area. The search algorithm choice has a direct impact in the motion vector quality and in the motion estimator performance.

There are lots of algorithms to define the search method; however, Full Search algorithm [5] is the most used for hardware implementations. Full Search algorithm is the only one that presents the optimal results in terms of best matching. All the others are fast algorithms which were designed to reduce the computational complexity of the motion estimation process. These algorithms produce sub-optimal results, because many positions are not compared. A good strategy for ME hardware implementation is to use Full Search algorithm with pixel subsampling (also called Pel Subsampling) because this can reduce the number of pixel comparisons keeping good quality results.

A software analysis was developed to evaluate and compare the quality of Full Search and Full Search with 4:1 Pel Subsampling algorithms. The main results are shown in Table 1. The search algorithms were developed in C and the results for quality and computational cost were generated. The search area used was 64x64 pixels with 16x16 pixels block size. The algorithms were applied to 10 real video sequences with a resolution of 720x480 pixels and the average results are presented in Table 1. The quality results were evaluated through the percentage of error reduction and the PSNR [3]. The percentage of error reduction is measured comparing the results generated by the motion estimation process with the results generated by the simple subtraction between the reference and current frame. Table 1 also presents the number of SAD operations used by each algorithm.

Table 1. Software evaluation of Full Search and 4:1 Pel Subsampling

Search Algorithm	Error reduction (%)	PSNR (db)	# of SAD operations (Goperations)
Full Search	54.66	28.48	82.98
Full Search with 4:1 Pel Subsampling	50.20	27.25	20.74

Full Search algorithm presents the optimal results for quality, generating the highest error reduction and the highest PSNR. However, it uses four times more SAD operations than the Full Search with 4:1 Pel Subsampling. The quality losses generated with the use of 4:1 Pel Subsampling are small. These losses are of only 4.46% in the error reduction and only 1.23dB in the PSNR.

It is important to notice that the Full Search with 4:1 Pel Subsampling algorithm can reduce significantly the computational costs of the motion estimation process with small losses in the quality results.

Full Search based algorithms (including its version with 4:1 Pel Subsampling) are regular algorithms and they do not present data dependencies. These features are

important when a hardware design is considered. The regularity is important to allow a reuse of the basic modules designed and the absence of data dependencies allow a free exploration of parallelism. Other important characteristic of Full Search based algorithms is that this type of algorithm is deterministic in terms of the clock cycles used to generate a new motion vector. This characteristic is important to allow an easy integration and synchronization of this module with other encoder modules.

The parallelism exploration is important to generate a solution tuned with the application requirements. Considering these features and the good results for quality, the Full Search with 4:1 Pel Subsampling algorithm was chosen to be designed in hardware. Using 4:1 Pel Subsampling it is possible to simplify the architecture, keeping the desired high performance.

4 Designed Architecture

There are many hardware architectures proposed in the literature that are based in the Full Search algorithm, such as [6] and [7]. These solutions are able to find the optimal results in terms of blocks matching. However, this type of architecture uses a very high amount of hardware resources. Full Search complexity can be reduced with little losses in the results quality, using the subsampling technique. This complexity reduction implies an important reduction in the hardware resources cost.

The architecture designed in this paper used Full Search with 4:1 Pel Subsampling algorithm with SAD distortion criterion. The block diagram of the proposed architecture is presented in Fig. 3. This architecture was designed to operate considering blocks with 16x16 samples and considering a search range of [-25, +24] samples.

The internal memory is organized in 5 different memories, as presented in Fig. 3. One memory is used to store the current frame block and the other four memories are used to store the search area.

The current block memory has 8 words and each word has 8 samples with 8 bits, in a total of 64 bits per memory word. This memory stores 64 samples (8x8) instead of 256 samples (16x16) because of the 4:1 subsampling relation.

The four memories used to store the search area have 32 words and each word has 32 samples with 8 bits, in a total of 256 bits per memory word. The data from the search area were divided in four memories, considering the frame as a bi-dimensional matrix of samples: samples from even lines and even columns, samples from even lines and odd columns, samples from odd lines and even columns and samples from odd lines and odd columns. Each word of each search area memory stores half of a line of the search area. Then, the memory that stores the samples from even lines and even columns stores the samples (0, 2, 4, … , 62), while the memory that stores the samples from even lines and odd columns stores the samples (1, 3, 5, … , 63). This division was made to allow a more efficient 4:1 Pel Subsampling processing.

The architecture presented in Fig. 3 was designed to explore the parallelism and to minimize the local memory access. The data read from memories are reused and each stored sample is read only once from the memories. The search area was divided in exactly four different memories to allow the data reuse and the minimization of the number of local memory accesses. The data read from the search area memories are

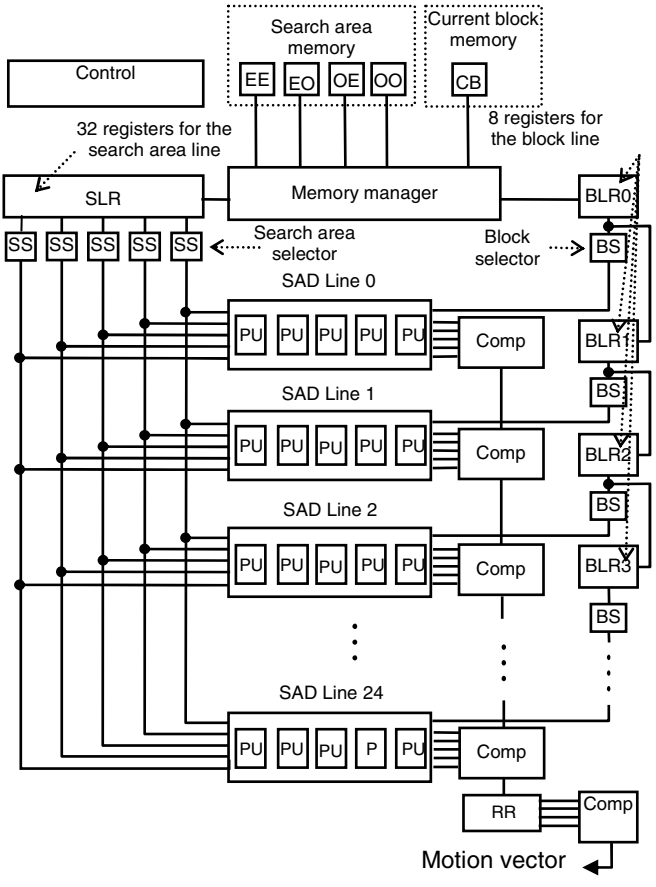

Fig. 3. Motion Estimation Architecture

sent to all SAD lines (see Fig. 3) which use this data when necessary. The data read from the current block memory is shifted through the SAD lines (using BLR registers in Fig. 3) and they are used to generate the SADs.

When ME architecture starts, the memory manager reads half of a line of the search area (one word of one search area memory), and one line of the current block (one word from the current block memory). With these data, half of the line of the search area and one line of the current block are available to be processed. These lines are stored in the search line register (SLR in Fig. 3) and in the block line register (BLR in Fig. 3). The processing unit (PU in fig. 3) calculates the distortion between two samples of the current block and two samples of a candidate block from the search area. Five PUs are used to form a SAD line. A set of 25 SAD lines forms a SAD matrix, as presented in Fig. 3. The control manages all these modules.

Four iterations over the 25 SAD lines are necessary to process a complete search area. One iteration is used for each search area memory. The result memory register (RMR in Fig. 3) stores the best match from each iteration. The final best match is

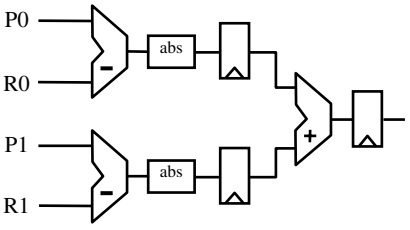

Fig. 4. PU architecture

generated after the comparison between the four results stored in RMR and then the motion vector is generated. All the operations necessary to generate one motion vector (MV) uses 2615 clock cycles.

5 SAD Calculation Architecture

SAD calculation architecture was hierarchically designed. The highest hierarchical level instance is the SAD matrix which is formed by 25 SAD lines. Each SAD line is formed by five processing units (PUs), as presented in Fig. 3. Fig. 4 shows the PU architecture.

When the 4:1 Pel Subsampling algorithm is used, the number of SAD calculations per line of the current block decreases to a half in comparison with Full Search algorithm, once the block was sub-sampled. This reduction of the number of calculations allows a reduction of the parallelism level of each PU without a reduction of the global ME performance. The PU architecture designed in this work is able to process a quarter of line of each candidate block (two samples) per cycle.

The subsampling reduces the size of the current block from 16x16 to 8x8 samples. The search area division in four sub-areas (stored in four different memories) implies in sub-areas with 32x32 samples, once the complete area has 64x64 samples. Then it is possible to conclude that there are 25 candidate blocks starting in each line of the search sub-area, because there are 32 samples per search line and 8 samples per block line.

The partial SAD of the candidate block (a quarter of line) must be stored and added to the SADs of the other parts of the same block to generate the total SAD for the block. The total SAD is formed by 8 lines with 8 samples in each line (64 SADs must be accumulated).

The SAD lines, presented in Fig. 5, groups five PUs and they make the accumulation to generate the final value of the candidate block SAD. Each PU is responsible for SAD calculation of five different candidate blocks, in distinct times. Then, a line of SADs calculates the SAD of 25 different candidate blocks.

Fig. 5 presents the five PUs (highlighted in gray) of one SAD line and it also presents the accumulators used to keep the partial and final values of the SAD calculations for each block. As each PU processes in parallel the SAD calculation of two samples, then each PU generates 32 partial results of SAD for each processed block. These 32 partial results must be added to generate the final block SAD. A simple structure of adder and accumulation is enough to generate this result.

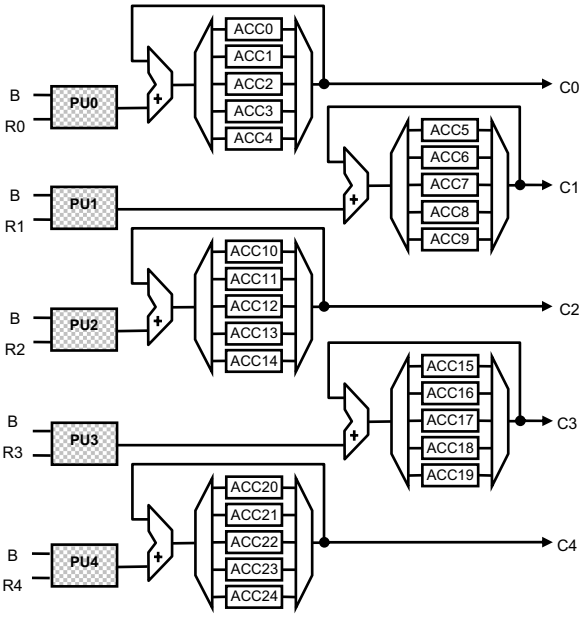

Fig. 5. Block Diagram of a SAD Line

Each PU calculates the SAD of 25 blocks, then a register is used to store the SAD of each block (ACC0 to ACC24 in Fig. 5) and a pair of demux/mux is necessary to control the correct access to the registers. When a SAD line concludes its calculations, the registers ACC0 to ACC24 will contain the final SADs of the 25 candidate blocks from one specific line of the search sub-area.

Search area selector (SS in Fig. 3) and block selector (BS in Fig.3) choose the correct data for each PU in a SAD line. A new and valid data is available to the PUs at each clock cycle.

Comparator (Comp modules in Fig. 3) receives the output from the SAD lines, and it can make five comparisons of SADs in parallel, in a pipeline with five stages. This module is responsible to compare 25 SADs from one SAD line (5 SADs per clock cycle) and to compare the best SAD (lowest value) of this SAD line with the best SAD of the previous SAD line, as shown in Fig. 3.

The result of each comparator consists of the best SAD among all SADs previously processed and a motion vector indicating the position of the block which generates this best SAD. This result is sent to the next comparator level (see Fig. 3).

The five SAD lines outputs (C0 the C4 in Fig. 5) generate five values of SAD in each clock cycle. In five clock cycles, all the 25 values of SAD from the SAD line are ready and these values are used in the comparator. The comparator architecture is showed in Fig 6.

A motion vector generator is associated to each SAD line to generate de motion vector for each candidate block. These motion vectors are sent to the comparator with the corresponding SAD result.

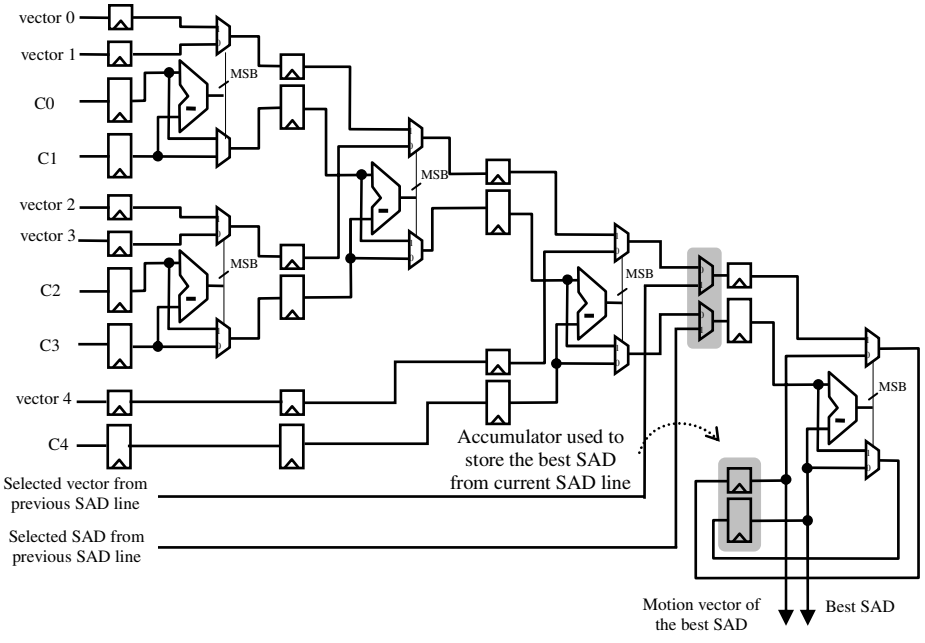

Fig. 6. Comparator Block Diagram

6 Synthesis Results

The synthesis results of the proposed architecture are summarized in Table 1. The synthesis was targeted to a Xilinx Virtex-II Pro VP70 FPGA and the ISE synthesis tool was used [8].

The synthesis results indicated that the designed architecture used 30,948 LUTs (46% of total device resources), using 19,194 slices (58% of total device resources) and 4 BRAMs (1% of device resources). This architecture is able to run at 123.4 MHz and a new motion vector is generated at each 2615 clock cycles.

The synthesis results show that the designed architecture can reach a processing rate of 35 SDTV frames (720x480 pixels) per second. This processing rate is enough to process SDTV frames in real time. The performance could be better if the parallelism level was increased or if a faster target device was used.

Some related works, using the Full Search algorithm with 4:1 Pel Subsampling, can be found in the literature, such as [9], [10] and [11]. However, these works target a standard cell technology and a comparison between our FPGA results is not easily made.

Other related works to Full Search algorithms targeting FPGA implementations as [12], [13] and [14] were also found.

The published solutions consider a search range of [-16, +15] while our solution considers a search range of [-25, +24]. The higher search range was defined to allow better quality results when processing high resolution videos. We did not find any published solution based on Full Search algorithm with a search range larger or equal to [25, +24].

We calculated the number of cycles that our solution needs to generate a motion vector considering the range [-16, +15], to allow a comparison of our architecture with the published solutions. This calculation results in 634 clock cycles to generate a motion vector. The operation frequency was estimated in the same 123.4MHz, once the architecture would be reduced to work in the [-16, +15] range.

Table 2. Synthesis results for [-25, +24] search range

ME Module	Frequency (MHz)	CLB Slices	LUTs
Global Control	269.2	91	164
Processing Unity	341.5	38	67
SAD line	341.5	489	918
Comparator	224.6	317	235
Vector Generator	552.7	6	10
Memory Manager	291.4	311	613
Search Area Selector	508.3	343	596
Block Selector	541.4	33	58
SAD Matrix	143.7	19,083	30,513
Motion Estimator	123.4	19,194	30,948
Device: Virtex-II Pro VP70			

The comparison with these related works, including Full Search and Full Search with 4:1 Pel Subsampling algorithms, is presented in Table 3. Table 3 presents the Pel Subsampling rate, the used technology, the operation frequency and the throughput. The throughput considers the number of HDTV 720p frames processed per second.

Our architecture presents the second higher operation frequency, just less than [13]; however, our throughput is about 120% higher than [13]. This is the highest throughput among the FPGA based architectures. The architecture presented in [11] can reach a higher throughput than ours; however, this result was expected once this architecture was designed in 0.18um standard cell technology.

Table 3. Comparative results for search range [-16, +15]

Solution	Pel Subsam-pling	Technology	Freq. (MHz)	HDTV 720p (fps)
[9]	4:1	0.35 um	50.0	8.75
[10]	4:1	0.35 um	50.0	22.56
[11]	4:1	0.18 um	83.3	63.58
[12]	No	Altera Stratix	103.8	5.15
[13]	No	Xilinx Virtex-II	191.0	13.75
[14]	No	Xilinx XCV3200e	76.1	20.98
Our	4:1	Xilinx Virtex-II Pro	123.4	54.10

6 Conclusions

This paper presented a FPGA based hardware architecture for motion estimation using the Full Search algorithm with 4:1 Pel Subsampling and using SAD as distortion criterion. This architecture considers blocks with 16x16 samples and it uses a search area with 64x64 samples, or a search range of [-25, + 24]. This solution was specifically designed to meet the requirements of standard definition television (SDTV) with 720x480 pixels per frame and it was designed focusing the solutions for the future Brazilian system of digital television broadcast.

The synthesis results indicated that the motion estimation architecture designed in this paper used 30,948 LUTs of the target FPGA and that this solution is able to operate at a maximum operation frequency of 123.4 MHz. This operation frequency allows the processing rate of 35 SDTV frames per second.

Comparisons with related works were also presented and our architecture had the highest throughput among the FPGA based solutions and the second highest throughput among all solutions.

References

1. International Telecommunication Union. ITU-T Recommendation H.262 (11/94): generic coding of moving pictures and associated audio information - part 2: video. [S.l.] (1994)
2. Joint Video Team of ITU-T and ISO/IEC JTC 1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 or ISO/IEC 14496-10 AVC) (2003)
3. Kuhn, P.: Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation. Kluwer Academic Publishers, Dordrecht (1999)
4. Brazilian Communication Ministry, Brazilian digital TV system (2006), Available at: http://sbtvd.cpqd.com.br/
5. Lin, C., Leou, J.: An Adaptative Fast Full Search Motion Estimation Algorithm for H.264. In: IEEE International Symposium Circuits and Systems, ISCAS 2005, Kobe, Japan, pp. 1493–1496 (2005)
6. Zandonai, D., Bampi, S., Bergerman, M.: ME64 - A highly scalable hardware parallel architecture motion estimation in FPGA. In: 16th Symposium on Integrated Circuits and Systems Design, São Paulo, Brazil, pp. 93–98 (2003)
7. Fanucci, L., et al.: High-throughput, low complexity, parametrizable VLSI architecture for full search block matching algorithm for advanced multimedia applications. In: International Conference on Electronics, Circuits and Systems, ICECS 1999, Pafos, Cyprus, vol. 3, pp. 1479–1482 (1999)
8. Xilinx INC. Xilinx: The Programmable Logic Company. Disponível em (2006), www.xilinx.com
9. Huang, Y., et al.: An efficient and low power architecture design for motion estimation using global elimination algorithm. In: International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2002, Orlando, Florida, vol. 3, pp. 3120–3123 (2002)
10. Lee, K., et al.: QME: An efficient subsampling-based block matching algorithm for motion estimation. In: International Symposium on Circuits and Systems, ISCAS 2004, Vancouver, Canada, vol. 2, pp. 305–308 (2004)

11. Chin, H., et al.: A bandwidth efficient subsampling-based block matching architecture for motion estimation. In: Asia and South Pacific Design Automation Conference, ASPDAC 2005, Shanghai, China, vol. 2, pp. D/7–D/8 (2005)
12. Loukil, H., et al.: Hardware implementation of block matching algorithm with FPGA technology. In: 16th International Conference on Microelectronics, ICM 2004, Tunis, Tunisia, pp. 542–546 (2004)
13. Mohammadzadeh, M., Eshghi, M., Azadfar, M.: Parameterizable implementation of full search block matching algorithm using FPGA for real-time applications. In: Fifth International Caracas Conference on Devices, Circuits and Systems, ICCDCS 2004, Punta Cana, Dominican Republic, pp. 200–203 (2004)
14. Roma, N., Dias, T., Sousa, L.: Customisable core-based architectures for real-time motion estimation on FPGAs. In: Cheung, P.Y.K., Constantinides, G.A. (eds.) FPL 2003. LNCS, vol. 2778, pp. 745–754. Springer, Heidelberg (2003)