MOTION ESTIMATION WITH DIAMOND SEARCH ALGORITHM FOR THE H.264 STANDARD ENCODER

Victor Hugo Caldas Marinho[1], Alba Sandyra Bezerra Lopes[1], Ivan Saraiva Silva[2], Edgard de Faria Correa[1]

{victorhugo,alba}@lasic.ufrn.br; edgard@dimap.ufrn.br; ivan@ufpi.edu.br

[1]Federal University of Rio Grande do Norte

[2]Federal University of Piaui

ABSTRACT

This paper presents an architecture for H.264 Motion Estimation. The algorithm used in the implementation is the fast block matching algorithm Diamond Search. This algorithm reduces the complexity needed to precess video sequences on the encoder module. The architecture was implemented in VHDL using Quartus 10.0 and synthesized using a Stratix III. Results of design cost and performance are presented.

1. INTRODUCTION

The H.264/AVC is the most advanced video compression standard with archive its goal reducing in almost 50% the number of bits needed to represent de information. This gain is consequence of high level increase in computational complexity of its modules, mainly the ME (Motion Estimation) [4].

The ME is the most compute intensive module of a video encoder, This module takes advantage of temporal redundancy existent in a video sequence One way to identify this redundancy is through the similarities existent between two neighboring frames. Many was developed to perform this process. These algorithms should generate efficient results in limited time.

Full Search is considered the only existing optimal algorithm, since it seeks the best match possible in all regions of search area. Through extensive testing of the lower value of distortion is found, however this requires a high number of operations to perform the process. And the number of operations increases as much as the size of search area increases too. [3]

To overcome this disadvantage many algorithms has been created aiming to reduce this complexity. One of the fast block-matching algorithm proposed is called Diamond Search [2]

This work is part of the H.264 Network, that aims develop Brazilian technology for the SBTVD (Brazilian Digital Television System).

2. THE DIAMOND SEARCH ALGORITHM (DS)

The DS algorithm employs two search patterns, the Large Diamond Search Pattern (LDSP) as showed on Figure 1 and the Small Diamond Search Pattern(SDSP) [5] as showed on, as shown in figures 1.a and 1.b. The pattern LDSP comprises nine points, eight of which circulate a central point, to compose the diamond. The pattern SDSP consisting of five points to form a small diamond, for refinement.

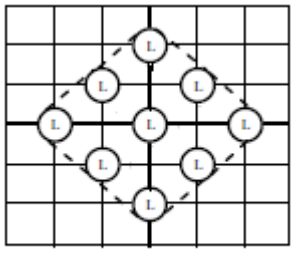

Figure 1. SDSP Search Pattern

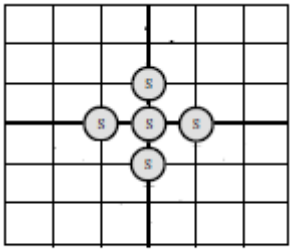

Figure 2. LDSP Search Pattern

When the algorithm starts the execution, the center of diamond is equivalent to the center area of research. When the lowest value of distortions is found in one vertex of the diamond, is necessary calculate more five block to form a new diamond Figure 3. If the lowest value was found in an edge, three new blocks are calculated.

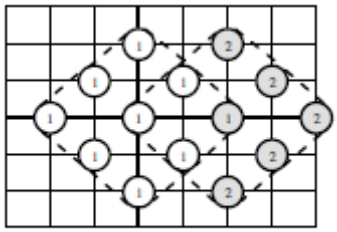

Figure 3. Vertex of the Diamond

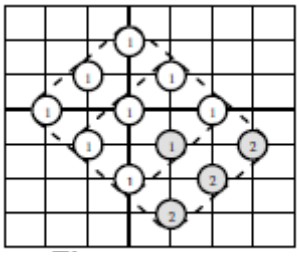

Figure 4. Edge of the Diamond

The LDSP is repeated as many times is necessary until find the lowest distortion in the center of the diamond. At this point, the SDSP is started for refinement. Among the five values of SDSP, the lowest distortion is found, is generated a motion vector of this position in the search area.

3. IMPLEMENTATION

The implementation described in this work was based on [1], with some differences as the use of 8x8 blocks, instead of 16x16 pixel with subsample, and fixed the search area 32x32. The distortion criterion used is the Sum of Absolute Differences (SAD). The architecture has nine processing units (PU), where each PU can calculate eight samples in parallel, ie, an entire row of an 8x8 block. Then eight accumulations are needed to generate the SAD of each block. The block diagram of the architecture is showed Figure 5, where CB is the candidated block, SDSP are the building blocks of this pattern for the current frame and the blocksare LDSP this standard to the same frame .

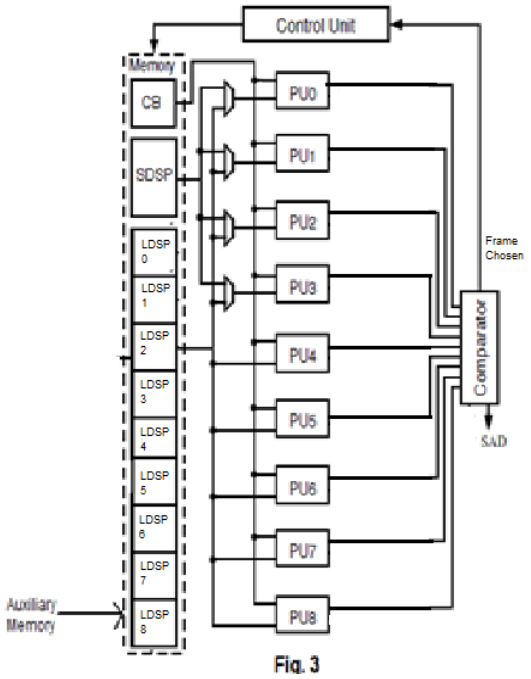

Figure 5. Block diagram of the implemented architecture

The startup DS values are loaded from the research area selected, nine of the starting blocks LDSP, the four values of the block and SDSP candidate is investigated in fifteen auxiliary memories. It also starts at [0.0] the motion vector, which is the value of the central framework.

The nine blocks LDSP are calculated in parallel, and the results is sent to a comparator (Figure 6). Each block has an associated index, to identify the position of the block in LDSP. The result of the comparator is the lowest value of SAD and it sends to the control unit of the index block. The control unit analyzes this index, and decides the next step of the algorithm and updates the position of the central block and motion vector. If the chosen block is zero, the lowest SAD was found in the center of the diamond, and the process of refinement (SDSP) should begin. Since the blocks are stored in the SDS buffer memory, to calculate the SDSP it is simply just change the switch of multiplexers that are at the entrance of pus.

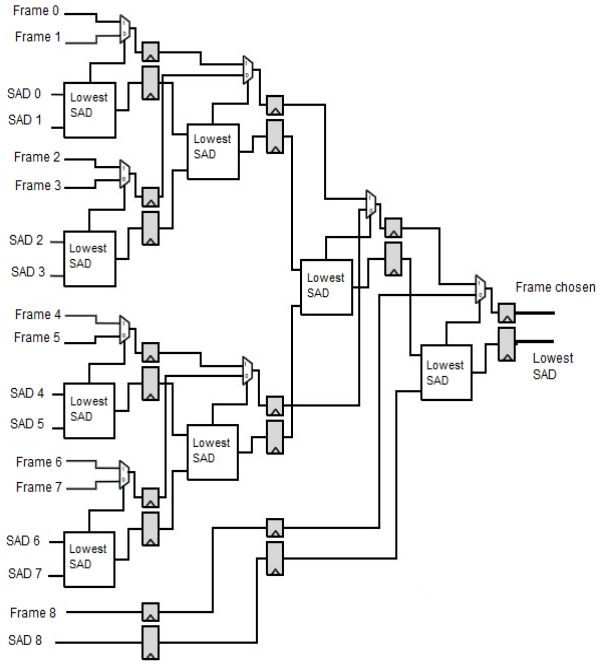

Figure 6. Comparator Module

When the index was one, three, five or seven, the control unity start up the LDSP again assigning the block found by calculating the center and five new blocks. If the index is two, four, six or eight, the control unit starts again the LDSP again, giving the block found by calculating the center and three new blocks. As previously mentioned, this step is repeated as many times as necessary.

The comparator receives the nine values and uses a pipeline to compare them two by two. In the first stage compares the values of the blocks in the second the results of the first stage, resulting in a total of five stages.

The control unit uses a state machine and the index of the comparator define which values are written into the memories, when the SAD computation starts and the standard should be initiated.

4. SYNTHESIS AND VALIDATION RESULTS

The proposed architecture was described in VHDL and Quartus Web Edition 10.0 was used to synthesize the total architecture on the Stratix III EP3SL70F484C2 device. For simulations we used the ModelSim Altera Starter Edition 6.5e .[6]

Words of 128-bits are used to load the search area. Once the architecture uses 32x32 as search area size, it

takes 64 clock cycles to load an entire search are. To fill the memories correspondent to 8x8 blocks, 8 clock cycles are needed. In the best case to fill the memory with al needed data are needed 35 clock cycles. In the worst case ever tested, LDSP being repeated ten times, 224 clock cycles are needed, and the average case, LDSP repeated five times, 119 clock cycles to produce a result. Table 2 shows the number of frames per second for some video standards.

Table 1 presents the results of architecture synthesis on the chosen device. According to the Table 1, the total architecture uses 18% of the ALUTs available and less then 1% of the total memory bits available.

Table 1. Diamond Search Synthesis Results

	Total Used	Total Available
Total combination functions (ALUTS)	6.749	54000
Dedicated logic registers	5.198	54.0000
Total block memory bits	16.384	2.267.136

To be perfectly possible to see a video, the DS must process at least 30 frames per second, using the number of clock cycles and the dimensions of the video is possible to evaluate this requirement. The Table 2 presents the performance results of the architecture. It's is possible realize that in the best case, the architecture can process 78 frames per second of a HDTV video, and even in the worst case, the architecture can also process real time because the number of frames achieved is 40.

Table 2. Performance Results

Video Resolution	Frames per Second		
	Best Case	Average Case	Worst Case
SDTV (720x480)	209	148	108
HDTV (10280x720)	78	55	40

The complete architecture was validated using real video sequences and the VHDL were compared with the implementation of the same Diamond Search algorithm in C. The validation process proved that the results are valid and consistent

5. COMPARISON WITH RELATED WORK

In this section is made a comparison of the implemented Diamond Search architecture with an implementation of the Full Search algorithm, described on [3].

Table 3 presents the synthesis results of the Full Search architecture in the same device as the one used in this work.

Table 3. Full Search Synthesis Results

	Total Used
Total combination functions (ALUTS)	8.655
Dedicated logic registers	6.486
Total block memory bits	10.420

When comparing the results of synthesis between the DS and FS, can be observe that the DS takes up less space and the FS registers and uses morememory blocks. However, this does not cause a significant increase in the size of the architecture, making the DS a better choice than the FS. Table 4 presents the frequency achieved for both architectures. The frequency of the DS architecture is higher than the FS architecture. Once this architecture uses less SAD calculation (as showed on Table 5), the number of frames per second that it is possible to compute is sufficient to aim real time for HDTV digital video.

Table 4. Frequency results

Architecture	Frequency
Diamond Search	229.62
Full Search	197.51

Table 5. Number of SAD calcutations

Architecture	Number of SAD calculaitions
Diamond Search (Best Case)	13
Diamond Search (Worst Case)	49
Diamond Search (Average Case)	94
Full Search	625

6. CONCLUSIONS

This paper presented the algorithm in hardware Diamond Search. Through comparisons with the FS was found that the DS dramatically reduces the number of SAD operations, with a small degradation of the results, occupying less space and with a higher frequency.

7. REFERENCES

[1]Porto, Marcelo Schivalon. Arquiteturas de Alto Desempenho e Baixo Custo em Hardware para Estimação de Movimento em Vídeos Digitais. 2008. Dissertação(Mestre em Ciência da Computação) – Instituto de Informática – Universidade Federal do Rio Grande do Sul(UFRGS).

[2] Zhu, Shan and Ma, Kai-Kuang, A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation.2000. IEEE transactions on image processing, vol. 9, no. 2.

[3]Lopes, Alba Sandyra Bezerra, Arquitetura com Elevada Taxa de Processamento e Reduzida Largura de Banda de Memória para a Estimação de Movimento de Vídeos Digitais. 2011. Dissertação(Mestre em Ciência da Computação) – Departamento de Informática e Matemática Aplicada – Universidade Federal do Rio Grande do Norte(UFRN).

[4]Richardson, I. 2003. H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia. John Wiley&Sons, Chichester.

[5]Kuhn, P. Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwe Academic Plubisher, Boston, 1999.

[6]http://www.altera.com