International Journal of Soft Computing 9 (3): 122-130, 2014
ISSN: 1816-9503

Power Efficient Cross Diamond Search Architecture for Low Bit Rate Mobile Applications

[1]D. Rukmani Devi, [2]P. Rangarajan and [3]Raja Paul Perinbam
[1]Department of Electronics and Communication,
R.M.K. Engineering College, Anna University, Chennai, India
[2]Department of Electrical and Electronics,
R.M.D. Engineering College, Anna University, Chennai, India
[3]Department of Electronics and Communication,
Karpaga Vinayaka College of Engineering and Technology, Anna University, Chennai, India

Abstract: Cross Diamond Search (CDS) is one of the fast search Motion Estimation (ME) algorithm that caters to the requirement of video coding with relatively minimal limitations. It exploits the cross centre-biased motion vector characteristics in real-world video sequences. The performance of the CDS algorithm is relatively better, compared to that other of the fast search ME algorithms which use a number of search points and similar picture quality as parameters. The objective is to build a new architecture with high performance CDS which requires nine Processing Elements (PEs) to obtain the minimum SAD. The proposed architecture was designed to take advantage of the properties of the CDS algorithm. The study has shown that the CDS algorithm with an efficient architecture with minimum number of gates suitable for the ME algorithms in video coding standards could achieve the desired video output. The results observed after implementing comparative performance analysis have shown that the CDS architecture occupies 23% of logic elements, 1% of dedicated logic registers and consumes as low as 84.63 mW power when it is implemented using Altera Quartus II FPGA device.

Key words: Cross diamond search, FPGA, hardware implementation, motion estimation, power

INTRODUCTION

Motion estimation is the most computationally intensive part of video compression and video enhancement systems. Reducing bit rate in video compression standards (Tasdizen *et al*., 2009; Wang *et al*., 2002; Morrison, 1992; Kuhn, 1999) using motion estimation concepts by exploiting the temporal redundancy between successive frames and to enhance the quality of displayed images in video enhancement systems by extracting the true motion information. Block Matching Algorithm (BMA) is widely adopted for ME in real video coding systems because of its simplicity and overhead. In a typical BMA each frame of a video sequence is divided into a fixed number of non-overlapping square blocks. For each block in the current frame, the search for the best matching block is made in the search area of previous frame under a certain criterion: the best matching is usually conducted through calculating the Sum of Absolute Differences (SAD) between the corresponding pixels in the blocks as described in Eq. 1:

$$SAD(m, n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[CB(i, j)\text{-}RB(i+m, j+n)\right] \quad (1)$$

Where:

N×N	=	The block size
CB (i, j)	=	The pixel intensity at the location (i, j) in the current block
RB (i, j)	=	The pixel intensity at the location (i, j) in the reference block
SAD (m, n)	=	Represents SAD at search position (m, n)
(m, n)	=	The displacement vector of the MB

SAD is the most preferred block matching criterion because of its suitability for hardware implementation. Among the BM algorithms, Full Search (FS) algorithm achieves the best performance since it searches all search locations in a given search range. However, the computational complexity of FS algorithm is very high, especially for the recently available high resolution and high frame rate video formats. Several fast search ME algorithms have been developed for low bit-rate applications which use small frame sizes and require small

Corresponding Author: D. Rukmani Devi, Department of Electronics and Communication, R.M.K. Engineering College, Anna University, Chennai, India

search ranges. These algorithms try to approach the PSNR of FS algorithm by computing the SAD values for fewer search locations in a given search range. The most successful fast search algorithms are New Three Step Search (NTSS) (Li *et al*., 1994), hierarchical BMA (Paul and Viscito, 1994), Diamond Search (DS), (Zhu and Ma, 2000), Hexagon-Based Search (HEXBS) (Zhu *et al*., 2002), simplex search (Rehan *et al*., 1998; (Al-Mualla *et al*., 2001) Flexible Triangle Search (FTS). (Rehan *et al*., 2005) and Cross Diamond Search (CDS) (Hao *et al*., 2012; Cheung and Po, 2002). Fast search ME algorithms perform very well for low bit-rate applications such as video phone and video conferencing because fast and complex movements are seldom in these applications. Analyzing and comparing in various aspects all these fast ME algorithms has concluded that CDS algorithm performs less number of search points, 40% speed up without affecting picture quality. In these fast algorithms, only selected subsets of search positions are evaluated using SAD. As a result, these algorithms usually produce sub-optimal solutions but the computational saving over FS is significant when it comes to hardware implementation. On the other hand, the number of SAD calculations is not the only criterion for the choice of motion estimation algorithm. Other criteria such as algorithm regularity, suitability for pipelining and parallelism, computational complexity and number of gates which directly affect power consumption and cost of hardware are also very important. Due to these reasons, there are several hardware implementations for the Full Search algorithm (Mohammadzadeh *et al*., 2005) and Hierarchal Search algorithm (Wang *et al*., 2004) which is very regular data array architecture. Hardware implementations for fast search algorithm such as three-step search (Jung and Lee, 2004; Seth *et al*., 2004; Sarma *et al*., 2003), Flexible Triangle search (Rehan *et al*., 2006) and diamond search (Chao *et al*., 2002; Chatterjee and Chakrarbarti, 2010).

The CDS algorithm is one of the fast search ME algorithm preferred for video conferencing applications; it uses a small frame size and requires a small search range, especially in consumer electronic products (Cheung and Po, 2002; Jia and Zhang, 2004) and (Zhu *et al*., 2009). The CDS algorithm generates quality results close to those of the FS by computing the SAD values for fewer search locations in a given search range. They appear to be an attractive solution for developing the hardware architecture to implement the CDS algorithm.

Many hardware architectures have been developed for the full search algorithm (Saha and Ghosh, 2007; Tasdizen *et al*., 2009; Yap and McCanny, 2004; Warrington *et al*., 2009; Wang *et al*., 2002; Elhaji *et al*., 2013) which has a very regular data array. At the same time, only a small number of hardware architectures were proposed for fast search ME algorithms (Chao *et al*., 2002; Jung and Lee, 2004; Rehan *et al*., 2006; Chatterjee and Chakrarbarti, 2010; Tseng *et al*., 2012). To the best of the knowledge, no hardware architecture is presented for the CDS ME algorithm.

Therefore, in this study, a high performance ME hardware architecture is proposed for efficiently implementing CDS algorithm. The proposed architecture was designed to take advantage of the properties of the CDS algorithm so that the FPGA implementation requires a low number of cycles per macro block search and low number of gates, leading to a fast and efficient implementation.

CROSS DIAMOND SEARCH ALGORITHM

The CDS is the first algorithm exploiting the characteristics of the cross centre-biased Motion Vector Probability (MVP) distribution.

Centre-biased MVP distribution: A search pattern with a certain shape and size has a significant impact on the efficiency and the effectiveness of the Search algorithm. Therefore, the search pattern is important and must be designed to fit the characteristics of MVP distribution. In fact, every discovery of the new characteristic of MVP distribution is followed by the upgrading of the search pattern and the improvement of the search algorithm's performance.

The search pattern used in the initial fast BMAs, such as Three Step Search (TSS) is square-shaped and the search points are distributed equally in the search window which is based on the idea that the motion vectors are distributed with the same probability on each point in the search window. In TSS, nine search points are uniformly distributed over the current search area of each step. A further study of MVP shows that the uniform MVP Distribution Model can not describe the MVP distribution well and truly. Fast search algorithms such as the NTSS, Four Step Search (FSS), DS, HEXBS and Block-Based Gradient Descent Search (BBGDS) were proposed by exploiting the characteristics of centre-based MVP distribution.

In centre-biased MVP distribution, >80% of the blocks are stationary or quasi-stationary. Most of the motion vectors are enclosed in the central 5×5 area for the search range of ±15 as depicted in Fig. 1. Based on

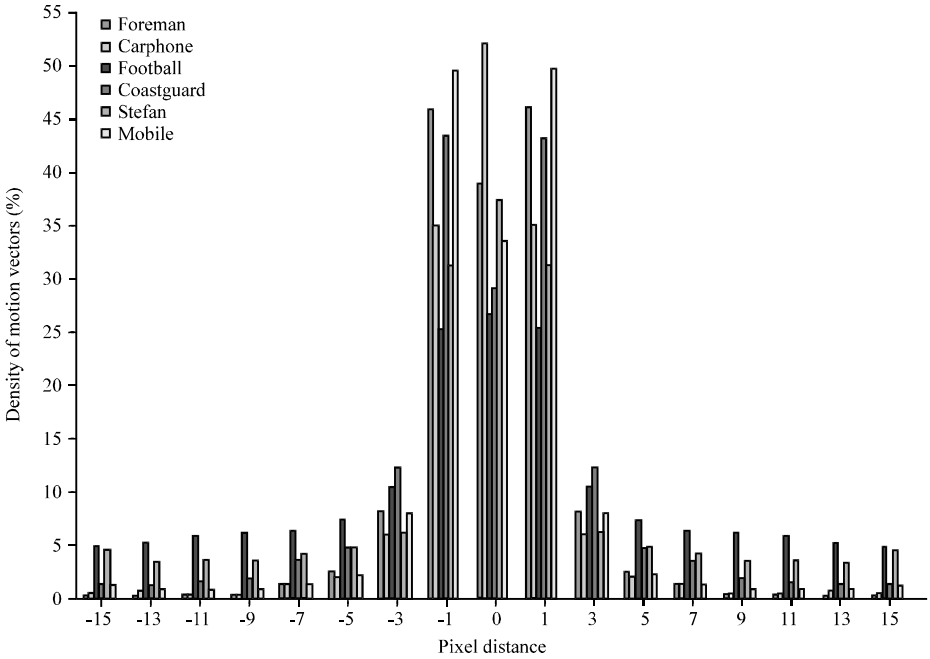

Fig. 1: Average MVP distribution of different video sequences with a search range of ±15

this phenomenon in real-world sequences, the NTSS checks eight more proximate points around the centre in the first step and employs the half-way stop technology to accelerate the search process the 4SS and BBGDS use the smaller square search patterns and half-way technology to speed up the process of searching for the Motion Vector (MV) near the centre. The DS and HEXBS employ the diamond and hexagonal patters to find the minimum MV without affecting picture quality.

Several more accurate MVP distribution models are built upon the detailed statistical data and the analysis of the MV in the DS, HEXBS and CDS. In the DS, the statistical data indicate that >50% of the motion vectors are enclosed in a circular support with a radius of 2 pixels and centered on the position of zero motion. Moreover, 98% of the stationary sequence is analyzed based on the statistical data (Seth *et al*., 2004). In the CDS, the analyses confirm that the MVs concentrate in some special area around the centre with different ratios; about 81.80% of the MVs are found located in the central 5×5 square area and 77.52 and 74.76% are found in the central 5×5 diamond area and cross area, respectively. The Cross Centre-Biased (CCB) Model describes the characteristics of the MVP distribution more exactly so the CDS algorithm based on it outperforms the other fast BMAs.

The CDS algorithm: The DS algorithm (Zhu and Ma, 2000) uses a Large Diamond Shaped Pattern (LDSP) and a Small Diamond Shaped Pattern (SDSP) as shown in Fig. 2a. As the MVP possesses over 96% CCB characteristics in the central 5×5 pixel area, an initial

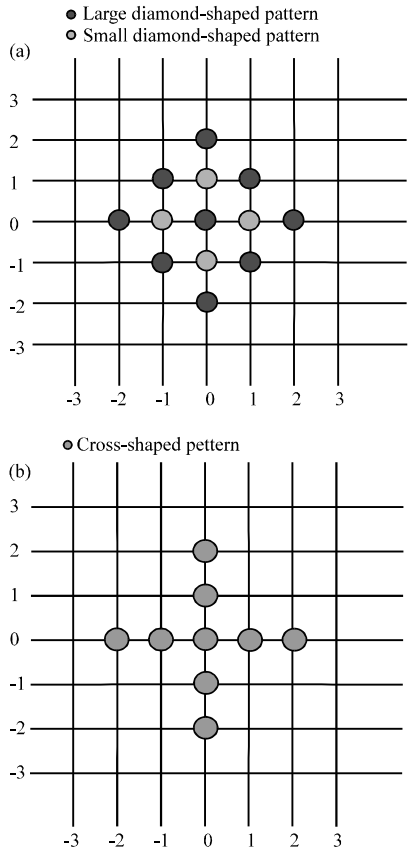

Fig. 2: Search patterns used in the CDS algorithm; a) LDSP and SDSP and b) CSP

Cross-shaped Pattern (CSP) as shown in Fig. 2b is considered as the initial step of the CDS algorithm.

The CDS differs from the DS by performing a CCB CSP and employing a halfway-stop technique for quasi-stationary or stationary candidate blocks. The steps involved in the implementation of the CDS algorithm may be described as follows:

Step 1 (Starting): From the nine search points of the CSP located at the centre of the search window, a minimum SAD is found. If the minimum SAD point occurs at the centre of the CSP, the search ends. Otherwise, it proceeds to step 2.

Step 2 (Half-diamond searching): Two additional search points of the central LDSP closest to the current minimum of the central CSP are checked, i.e., two of the four candidate points are located at (±1, ±1). If the minimum SAD found in the previous step is located in the middle wing of the CSP, i.e., (±1, 0) or (0, ±1) and the new minimum SAD found in this step still coincides with this point, the search stops. Otherwise, it proceeds to step 3.

Step 3 (searching): A new LDSP is formed by repositioning the minimum SAD found in the previous step at the centre of the LDSP. If the new minimum SAD point is still at the centre of the newly formed LDSP then go to step 4. Otherwise, step 3 is repeated.

Step 4 (Ending): With the minimum SAD point in the previous step as the centre, a new SDSP is formed. The new minimum SAD point is identified from the four new candidate points which is the final solution of the motion vector. The flowchart of the CDS algorithm is shown in Fig. 3.

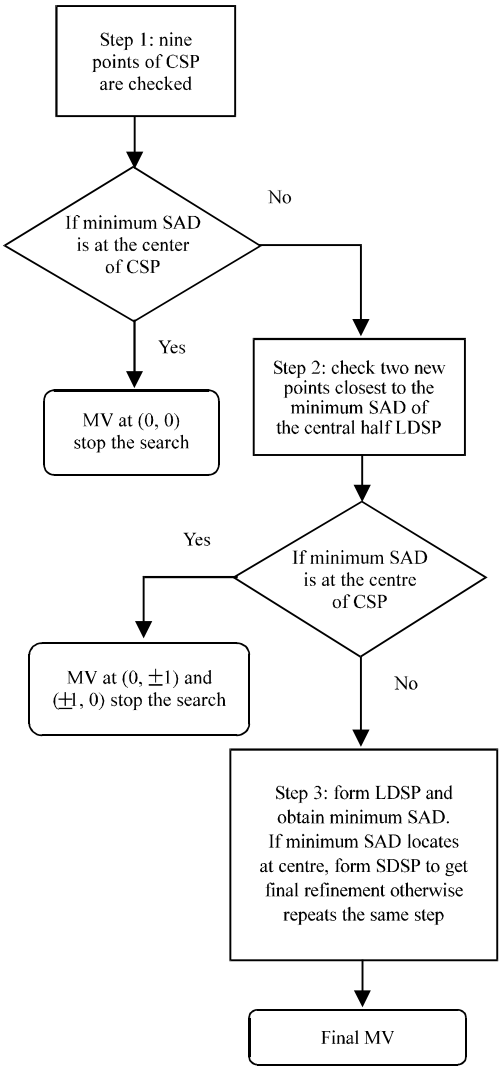

Fig. 3: Flowchart of the CDS algorithm

Analysis of the CDS algorithm: The CDS algorithm checks only nine (first step stop) and eleven (2nd step stop) search points and it leads to a theoretical speedup of 25 and 20.5 times as compared to the 225 checking points used in the FS algorithm. With the introduction of the CCB motion vector distribution characteristics, it can facilitate the optimization of the DS to CDS and reduce the computations significantly, especially for low bitrate video applications. By tackling the image sequences with gentle or no motion such as background information that is reasonably handled by first-step and small motion, in which a more accurate estimation is accomplished by the Secind step. On the other hand, the CDS algorithm performs similar to the DS algorithm when the minimum SAD point falls neither in the centre nor at any points on the middle wing of the CSP. The CDS usually keeps advancing between successive LDSPs by three or five new points as in the DS algorithm, after it proceeds to the third or fourth step, if necessary.

The CDS algorithm consumes less number of search points with slightly higher values in SAD compared to other algorithms. For video conferencing sequences MissAmerica the CDS achieves 40% searching speed and 3% gain in video quality than the DS. For a relatively higher degree of motion sequences like Coastguard and Football, the CDS achieves 25% searching speed with a slight decrease in the video quality. Figure 4 depict the performance comparison of the CDS algorithm with other existing algorithms in terms of the Average Number of Search Points (ANSP). Figure display the performance comparison of the CDS algorithm with other existing algorithms in terms of the Peak Signal to Noise Ratio (PSNR).

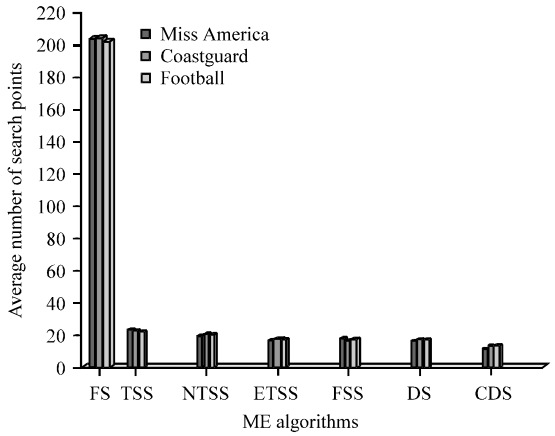

Fig. 4: Performance comparison of the CDS with other algorithms in terms of the ANSP

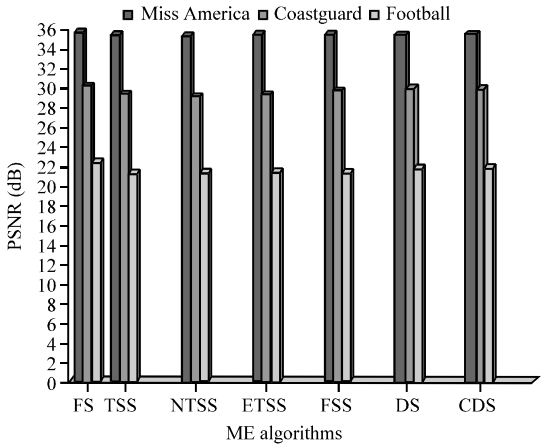

Fig. 5: Performance comparisons of the CDS with other algorithms in terms of the PSNR in dB

PROPOSED CDS ARCHITECTURE

Due to irregular search positions, there is no adequate architecture that supports the conventional DS and CDS algorithm. The DS and CDS algorithm achieves lesser number of search points than the Full Search (FS) and the same PSNR as the FS. This is the main objective, to propose an architecture that is suitable for the CDS algorithm, to embark upon irregular data-flow by using an interleaved memory organization supporting random access depending upon the search positions. The general block diagram of proposed CDS architecture is shown in Fig. 6.

It consists of a PE array, SAD comparator, current Macro Block (MB) buffer, search area buffer, address generation unit and state machine controller. The hardware finds a MV for a 16×16 MB based on the minimum SAD criteria in a search range of pixels (-7, +7)

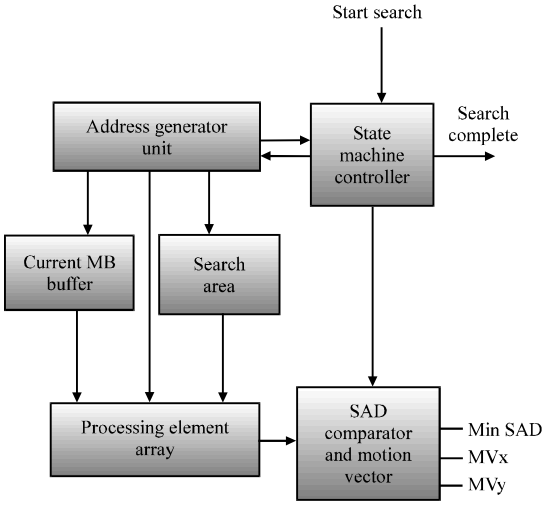

Fig. 6: Proposed CDS architecture

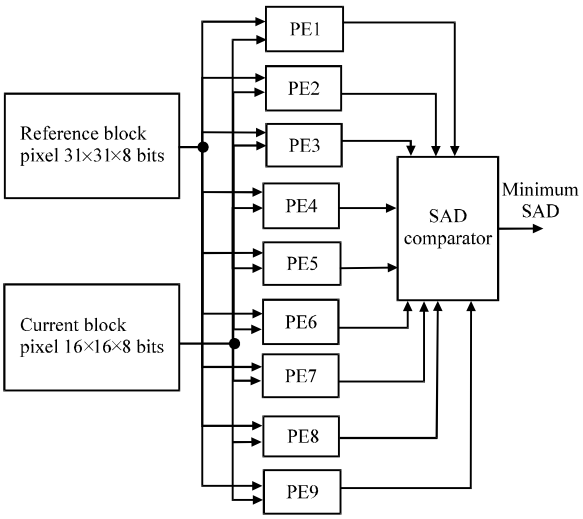

Fig. 7: Internal structure of PE array

using the luminance data. The architecture has nine processing elements, to calculate the SAD between the current and reference pixels. On-chip memories are used to store the Current Block (CB) data pixels and the Reference Block (RB) data pixels. A set of memory modules are used to store the reference block data, to avoid pipeline bubbles due to an irregular data-flow (Chao *et al*., 2002; Tseng *et al*., 2012). The nine search points of the CSP are calculated in parallel and the results are sent to the comparator as depicted in Fig. 7. Each search point receives an index, to identify the position of this block in the CSP. The comparator finds the lowest SAD and sends the index of the chosen search points to the control unit. If the best result was found at the centre of the CSP, the search terminates.

Table 1: Portrayal of the FSM controller

States	Description of states
01→01	Start search with initial nine points located at the centre of the CSP. If the minimum SAD is at the centre of the CSP, stop the search
01→10	The search enters the next state by adding new search points in the location of (±1, ±1) depending on the previous state. If the minimum SAD is either (0, ±1) or (±1, 0), stop the search. Otherwise, the state changes
01→11	If SAD is located at any one of the locations, i.e. (±1, ±1) and (0, ±2), the search continues with LDSP (2 or 3 iterations). If the minimum SAD is located at the centre retain the state 11
11→01	SDSP is formed based on the minimum SAD centre obtained from the previous state. If the minimum SAD is at the centre of the SDSP, stop the search. The minimum SAD and position of the MV are stored in the memory for the purpose of the next block search

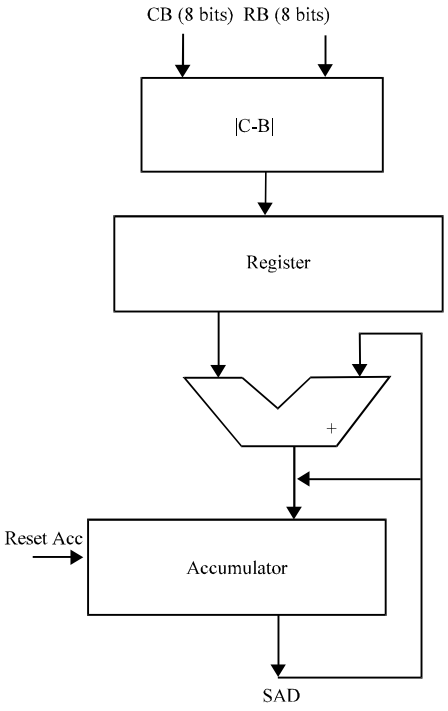

Fig. 8: Processing element

Otherwise, the control unit analyses the block index and decides on half diamond searching, pointing the indices (±1, ±1). The lowest SAD is identified at the CSP; the control unit generates the corresponding motion vector and position of the MV for this block. If the minimum SAD is any one of the corner points, the control unit analyses the block index and decides the next step (forming LDSP) of the algorithm. If the position of the candidate point is at the centre then the lowest SAD was found in the centre of the LDSP. So, the control unit starts the final refinement with the SDSP. In the SDSP calculation, four more candidate blocks are evaluated. The lowest SAD is identified and the control unit generates the corresponding motion vector and position of the motion vector to this block.

Processing unit: The PE is the primary element in the SAD calculation and consists of the absolute difference, adder, accumulator and register as illustrated in Fig. 8.

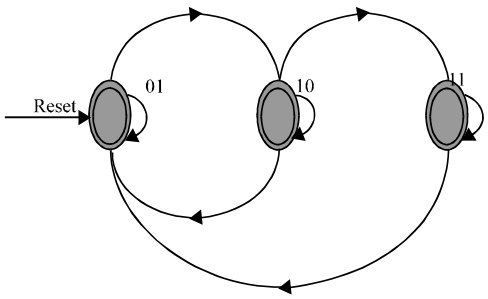

Fig. 9: FSM controller of the CDS architecrture

State machine controller: The state machine controller is also known as Finite State Machine (FSM) is responsible for getting data from the reference and current memory and executing the CDS algorithm effectively. The state machine controller diagram is shown in Fig. 9. The detailed depiction of a state machine controller is given in Table 1.

PERFORMANCE EVALUATION

The CDS algorithm stops the search when the best match is found at the centre of the CSP. It is the best case when only nine candidate blocks are evaluated. This architecture uses 47 clock cycles to fill the memories and to start the SAD calculation. The PEs needs 11 cycles to calculate the SAD of one block. The comparator uses five cycles to choose the lowest SAD. So, in the best case, this architecture can generate a motion vector in 16 cycles. In the worst case, 12 clock cycles are necessary to process the LDSP and SDSP needs only 4 cycles for the PES and one more cycles to the comparator. The same 24 cycles are used by the PEs and the comparator. Both edge and vertex search use the same 34 cycles.

Synthesis result: The proposed CDS architecture is implemented in Verilog HDL, verified simulation using Altera modelsim 6.c, synthesized with synplicity synplify Pro 8.9 and mapped to an Altera Quartus II with device family Cyclone III. The proposed hardware research at 360 MHz and achieves 3555 Logic Elements (LEs) which is 23% of all the LEs of an EP3C16F484C6 and <1% of the Dedicated Logic Registers (DLRs) and the power

Table 2: Comparison of the existing ME hardware architectures with the proposed CDS architecture

Architecture	Algorithm	FPGA	MB size	No. of PEs	Search range	LUTs, DLRs	Frequency (MHz)
Ben Atitallah *et al.* (2012)	LDPS	Stratix III	16×16	-	[-9,9] and [-7, 7]	48491571	136.00
			8×16	-			
			16×8	-			
			8×8	-			
Sanchez *et al.* (2012)	DMPDS	Stratix 4	16×16	45	[-64, +64]	34.5K	187.58
Sanchez *et al.* (2012)	DMPDS	Virtex 5	16×16	45	[-64, +64]	56.3K	294.00
Porto *et al.* (2008)	SDS	Virtex 4	16×16	9	[-64, +64]	3541	185.70
El-Ashry *et al.* (2001)	FTS	Spartan 3	16×16	16	[-16, +16]	6320	55.80
Proposed	CDS	Cyclone III	16×16	9	[-7, 7]	3555 Les114DLRs	360.00

Table 3: Performance analysis of the CDS architecture synthesized using synopsis design compiler

Algorithm	CDS
Number of PEs	9
Search range	31×31
Block size	16×16
Total area (μm^2)	6529
Total dynamic power	108.4692 μW

consumption is 84.63 mW. The CDS FPGA results were compared with those of the existing architectures and are given in Table 2. The CDS is the fastest one among the compared designs and it uses less hardware resources than the other solutions, reaching the highest throughput. The CDS architecture synthesized by the Synopsis Design Compiler occupies a combinational area of 3730.27 μm^2, a non-combinational area of 2798.8 μm^2, making up a total area of 6520.12 μm^2 and it consumes a total dynamic power of 108.4692 μW as shown in Table 3.

CONCLUSION

The CDS hardware architecture has been proposed, due to its property of centre-biased motion vector distribution. The software analyses show that this algorithm can drastically reduce the number of SAD operations with a very minor degradation in the video quality. The synthesis results showed that the designed architecture can achieve a high throughput with low hardware resources. The presented architecture is able to run at 360 MHz in an Altera Quartus II with device family Cyclone III. The proposed CDS architecture is implemented using 3555 LEs (23%) and 1% of the DLRs only. Consequently, the proposed hardware can support higher frame rates compared with the other designs. In addition, the hardware requirements are comparable with those of other algorithms which make the proposed design more efficient in terms of power consumption. The proposed CDS architecture is also synthesized synopsis design compiler and it consumes 108.4692 μW total dynamic power.

REFERENCES

Al-Mualla, M.E., C.N. Canagarajah and D.R. Bull, 2001. A simplex minimization for single and multiple-reference motion estimation. IEEE Trans. Circuits Syst. Video Technol., 11: 1209-1220.

Ben Atitallah, A., S. Arous, H. Loukil and N. Masmoudi, 2012. Hardware implementation and validation of the fast variable block size motion estimation architecture for H.264/AVC. Int. J. Electronics Commun., 66: 701-710.

Chao, W.M., C.W. Hsu, Y.C. Chang and L.G. Chen, 2002. A novel hybrid motion estimator supporting diamond search and fast full search. Proceedings of the IEEE International Symposium on Circuits and Systems, Volume 2, May 26-29, 2002, Phoenix-Scottsdale, AZ., pp: 492-495.

Chatterjee, S.K. and I. Chakrarbarti, 2010. Low power VLSI architectures for one bit transformation based fast motion estimation. IEEE Trans. Consumer Electronics, 56: 2652-2660.

Cheung, C.H. and L.M. Po, 2002. A novel cross-diamond search algorithm for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol., 12: 1168-1177.

El-Ashry, R., M. Rehan, H. El-Kamchouchi and F. Gabeli, 2001. Performance-Optimized FPGA implementation for the flexible triangle search block-based motion estimation algorithm. Proceedings of the IEEE 24th Canadian Conference on Electrical and Computer Engineering, May 8-11, 2001, Niagara Falls, ON., pp: 640-643.

Elhaji, M., A. Zitouni, S. Meftali, J.L. Dekeyser and R. Tourki, 2013. A low-power oriented architecture for H.264 variable block size motion estimation based on a resource sharing scheme. Integration VLSI J., 46: 404-412.

Hao, W.J., L.C. Zhang and Y.N. Wang, 2012. Cross-diamond search algorithm for motion estimation based on projection. Adv. Mater. Res., 433-440: 3713-3717.

Jia, H. and L. Zhang, 2004. A new cross diamond search rithm for block motion estimation. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Volume 3, May 17-21, 2004, Beijing, China, pp: 357-360.

Jung, S.T. and S.S. Lee, 2004. A 4-way pipelined processing architecture for three-step search block-matching motion estimation. IEEE Trans. Consumer Electr., 50: 674-681.

Kuhn, P., 1999. Algorithms Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation. Kluwer Academic Publishers, Boston.

Li, R., B. Zeng and M.L. Liou, 1994. A new three-step search algorithm for block motion estimation. IEEE Trans. Circuits Syst. Video Technol., 4: 438-442.

Mohammadzadeh, M., M. Eshghi and M.M. Azdfar, 2005. An optimized systolic array architecture for full-search block matching algorithm and its-implementation on FPGA chips. Proceedings of the 3rd International IEEE-NEWCAS Conference, June 19-22, 2005, Tehran, Iran, pp: 327-330.

Morrison, G., 1992. Video coding standards for multimedia: JPEG, H.261, MPEG. Proceedings of the IEE Colloquium on Technology Support of Multimedia, April 13, 1992, London, pp: 2-4.

Paul, B.B. and E. Viscito, 1994. Hierarchical motion estimation with 2-scaletilings. Proceedings of the IEEE International Conference on Image Processing, Volume 3, November 13-16, 1994, Austin, TX., pp: 260-264.

Porto, M., L. Agostini, S. Bampi and A. Susin, 2008. A high throughput and low cost diamond search architecture for HDTV motion estimation. Proceedings of the IEEE International Conference on Multimedia and Expo, June 23-April 26, 2008, Hannover, pp: 1033-1036.

Rehan, M., A. Antoniou and P. Agathoklis, 1998. A new fast block matching algorithm using the simplex technique. Proceedings of the IEEE Symposium on Advances in Digital Filtering and Signal Processing, Jun 5-6, 1998, Victoria, BC., pp: 30-33.

Rehan, M., M.W. El-Kharashi, P. Agathoklis and F. Gebali, 2006. An FPGA implementation of the flexible triangle search algorithm for block based motion estimation. Proceeding of the International Symposium on Circuits and Systems, May 21-24, 2006, Island, Kos.

Rehan, M., P. Agathoklis and A. Antoniou, 2005. Block-based motion estimation Using an Enhanced Flexible triangle search algorithm. Proceedings of the Canadian Conference on Electrical and Computer Engineering, May 1-4, 2005, Saskatoon, Sask, pp: 259-262.

Saha, A. and S. Ghosh, 2007. A Speed-area optimization of full search block matching hardware with applications in high definition TVs (HDTV). Proceedings of the 14th International Conference on High Performance Computing-HiPC, December 18-21, 2007, Goa, India, pp: 83-94.

Sanchez, G., F. Sampaio, M. Porto, S. Bampi and L. Agostini, 2012. DMPDS: A fast motion estimation algorithm targeting high resolution videos and its FPGA implementation. Int. J. Reconfigurable Comput., Vol. 2012. 10.1155/2012/186057.

Sarma, M., D. Samanta and A. Sundar, 2003. VLSI architecture for multi resolution three step search algorithm. Proceedings of the 5th International Conference on ASIC, Volume 2, October 21-24, 2003, Itanagar, India, pp: 918-921.

Seth, K., P. Rangarajan, S. Srinivasan, V. Kamakoti and V. Balakuteshwar, 2004. A parallel architectural implementation of the new three-step search algorithm for block motion estimation. Proceedings of the 17th International Conference on VLSI Design, Jan. 05-09, IEEE Computer Society, Washington, DC, USA., pp: 1071-1076.

Tasdizen, O., H. Kukner, A. Akin and I. Hamzaoglu, 2009. A high performance reconfigurable motion estimation hardware architecture. Proceedings of the IEEE Date Design Automation and Test in Europe Conference and Exhibition, April 20-24, 2009, Nice, France, pp: 882-885.

Tseng, C.F., Y.T. Lai and M.J. Lee, 2012. A VLSI architecture for three-step search with variable block size motion vector. Proceedings of the 1st IEEE Global Conference on Consumer Electronics, October 2-5, 2012, Tokyo, pp: 628-631.

Wang, C.N., S.W. Yang, C.M. Liu and T. Chiang, 2004. A hierarchical N-Queen decimation lattice and hardware architecture for motion estimation. IEEE Trans. Circuits Syst. Video Technol., 14: 429-449.

Wang, Y., J. Ostermann and Y. Zhang, 2002. Video Processing and Communications. Prentice Hall, Boston.

Warrington, S., W.Y. Chan and S. Sudharsanan, 2009. Scalable high-throughput architecture for H.264/Avc variable block size motion estimation. Proceedings of the IEEE International Symposium on Circuits and Systems, May 21-24, 2006, Island of Kos.

Yap, S.Y. and J.V. McCanny, 2004. A VLSI architecture for variable block size video motion estimation. IEEE Trans. Circuits Syst. II: Express Briefs, 51: 384-389.

Zhu, C., X. Lin and L.P. Chau, 2002. Hexagon-based search pattern for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol., 12: 349-355.

Zhu, S. and K.K. Ma, 2000. A new diamond search algorithm for fast block-matching motion estimation. IEEE Trans. Image Process., 9: 287-290.

Zhu, S., J. Tian, X. Shen and K. Belloulata, 2009. A new cross-diamond search algorithm for fast block motion estimation. Proceedings of the 16th IEEE International Conference on Image Processing, November 7-10, 2009, Cairo, Egypt, pp: 1581-1584.