Fast Center Search Algorithm with Hardware implementation for Motion Estimation in HEVC Encoder

Ahmed Medhat, Ahmed Shalaby, Mohammed S. Sayed, Maha Elsabrouty
Egypt-Japan University of Science and Technology
P.O.Box 179, New Borg El-Arab City,
Alexandria 21934, Egypt
{ahmed.abdelsalam, ahmed.shalaby, mohammed.sayed, maha.elsabrouty}@ejust.edu.eg

Farhad Mehdipour
E-JUST Center, Kyushu University
3-8-33 Momochihama, Sawara-ku,
Fukuoka 814-0001, Japan
farhad@ejust.kyushu-u.ac.jp

***Abstract*—This paper presents a Fast Center Search Algorithm (FCSA) and its hardware implementation design of integer Motion Estimation for High Efficiency Video Coding (HEVC). FCSA achieves average time saving ratio up to 40% for HD video sequences with respect to full search, with insignificant loss in terms of PSNR performance and bit rate. The proposed hardware implementation shows that it meets the requirements of 30 4K frame per second with ±16 search window at 550 MHz. The prototyped architecture utilizes 8% of the LUTs and 4% of the slice registers in Xilinx Virtex-6 XC6VLX-550T FPGA.**

Keywords—HEVC, motion estimation, inter prediction, sum of absolute difference (SAD) architecture, block matching

I. INTRODUCTION

The new high efficiency video coding (HEVC) standard was introduced targeting to double the compression efficiency with respect to the previous standard H.264. It can achieve 50% bit rate saving compared to H.264 for the same video quality [1]. Quad tree structure is the basic feature that distinguishes HEVC from MPEG-4 AVC. Code tree unit (CTU) is the basic unit of HEVC instead of the macroblock in H.264/MPEG-4 AVC. CTU size is chosen by the encoder. The size is variable, and can be larger or smaller than the traditional fixed size macroblock [1]. Each CTU is split into one or more code units (CUs). Each CU has a related partitioning into prediction units (PUs) and transform units (TUs).

Motion estimation (ME) is used for inter frame prediction where the motion in the current frame is predicted from one or more reference frames. Motion Estimation consumes more than 90% of encoding time. Full search algorithm is the superior technique to perform motion compensation in terms of compression ratio (CR), bit rate (BR) and peak signal to noise ratio (PSNR) by evaluating the prediction error at every possible location within certain search window. On the other hand, it is the most computational consuming algorithm especially for HD videos. Many fast algorithms were proposed to reduce the computational complexity of the full search. Nevertheless, most of these algorithms are not suitable for hardware implementation [2].

Coding block (CB) in HEVC has variable size. For each CU, the size of its luma CB can be defined as $2N \times 2N$, where $N \in \{4, 8, 16, 32\}$. CBs can be further divided into one, two or four prediction blocks (PB). Therefore, there are more than 600 blocks with different sizes at every candidate in a search window for every CTB. In full search algorithm, all possible candidates are checked to find the best matched block in terms of Lagrangian cost function [3].

Aiming to provide an efficient solution for both time and hardware saving, we propose fast center search algorithm (FCSA). Our proposed algorithm along with its hardware implementation achieves fewer search points than full search and some other fast searches. In addition, it is more suitable for hardware implementation. This paper is organized as follows; related work is explained in section II. In section III, the detailed description of the proposed FCSA is described. The proposed SAD unit is explained in section IV. Section V shows the simulation results and discussion. Finally, section VI concludes the paper.

II. RELATED WORK

In full search algorithm, all possible candidates are checked to find the best matched block in terms of Lagrangian cost function [3]. Profiling shows that as ME unit reaches up to 96% of the encoding time [4]. Therefore, many fast search algorithms are proposed to reduce the computational time while preserving the same BR and PSNR. These fast algorithms can be classified into three main groups. Algorithms, in group one, add an early termination condition to the full search. For example, in [5] Hashad et al. proposed a fixed range of PSNR as an early termination condition. This idea reduces the computations of full search, but, the drawback is that, this PSNR range is fixed and some blocks may not reach it even if full search is used. While in [6], Bo et al. used the previous frame SAD value for the same block location as a stop criterion to speed-up the full search.

However, this comes at the expense of the memory utilization as large memory is required to store all of these SAD values. This impacts the overall hardware area, power consumption and memory bandwidth requirements. Moreover, SAD value as a stop criterion may be misleading, as the cost function depends on other parameters not only on SAD value [3].

The second group of fast searches adopts adaptive search strategy with variable size search window. They minimize the necessary computations for full search by reducing the number of possible candidates to be checked. For instance, in [7] Paul et al. proposed an adaptive search window with three different sizes. The most suitable search window for each block is determined basis on average SAD value and the average motion vector (MV) obtained from the previous frame for the current block area. In the third group of algorithms, fast BMAs is used with different strategies, sizes and shapes to impact the search speed. For example, in [9] Belghith et al. proposed a combination of small diamond and horizontal diamond search pattern for motion estimation stage in HEVC. In [8], several methods are reviewed based on combinations of cross diamond and hexagon search pattern. Each fast algorithm speeds up the encoder by different rates, changes bit rate and PSNR by different rates [8]. Despite all of the above efforts to develop efficient search techniques, full search is still the best efficient symmetric algorithm in terms of hardware and parallelism [2].

III. PROPOSED FAST CENTER SEARCH ALGORITHM

In video sequences, motion of prediction blocks can be divided into three different categories, the first category is stationary motion i.e. no motion at all. In this case, the best matching candidate in the search window is the center point of the search window. Stationary motion represents about 40% – 60% at every frame. Quasi-stationary motion, which is the second category, represents 30% – 40% where the best matching candidate is around the center point by ± 2 pixels. The third category is rapid motion [8]. This indicates that more than 70% of prediction blocks have motion vectors around the center point of their search windows within a small range of pixels. Therefore, it is preferable to start BMA from the center point in search window. Thus, FCSA is proposed to reduce the computational complexity of full search, and to improve the encoder time.

A. Starting BMA search from center point

Full search checks all possible candidates in search window with the search order shown in Fig. 1(a). FCSA has the possibility to check all candidates in search window, but, with a different order of search as shown in Fig. 1(b). FCSA starts BMA from the center point at every row of search window, then, it takes, consecutively, one candidate left to the center point and one candidate to the right till it reaches search window's edges. Once it finishes the center row of search window, it takes one row down to the center row, thereafter one row up till it reaches search window's edges. The proposed FCSA finds the best matched candidate faster than traditional full search. In addition, FCSA uses an early termination criterion to stop search once the termination criterion is satisfied.

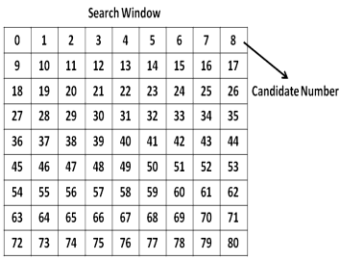

Fig. 1(a) Full Search Order

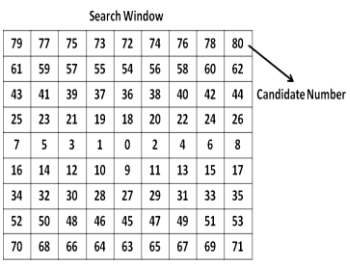

Fig. 1(b) FCSA Search Order

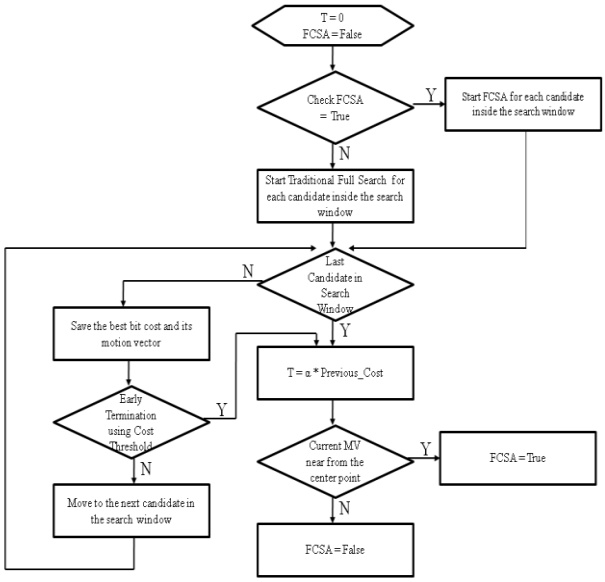

Fig. 2. Fast Center Search Algorithm (FCSA) flow chart

TABLE I. COSTS OF BLOCKS IN 28TH FRAME FROM CONTAINER USING QUANTIZATION PARAMETER = 28, LAGRANGIAN PARAMETER= 35

762	862	935	…	564	785	1286
2449	2517	2600	…	2697	2312	2665
.	.	.	…	.	.	.
.	.	.	…	.	.	.
.	.	.	…	.	.	.
792	909	874	…	1057	1108	817
1622	1724	169	…	1993	1730	1959

B. Early termination Criterion

Statistics for various video sequences show that there is a strong correlation between SADs of the collocated prediction blocks in the consecutive frames [6]. In HEVC, bit cost is the criterion to decide the best matched block [3]. Therefore, the proposed early termination algorithm depends on the cost function of the previous processed prediction block in the same frame. From Tables I it can be recognized how strong the correlation in cost function among the adjacent prediction blocks in the same frame. Therefore, the proposed early termination algorithm checks if cost function of the current candidate is less than *T*, it means that the current candidate is accurate enough to stop search. *T* is calculated according to:

$$T = \alpha * previous\_cost \quad (1)$$

As α represents matching parameter between the cost of the previous block and the current one. It is set to 1.4 in all our simulations, and, *previous_cost* is the bit cost of the previous block in the same frame.

C. Fast Center Search Algorithm

FCSA is based on starting BMA search from the center point of the search window with an early termination criterion to achieve high performance with fewer number of search points on average. The FCSA flow chart shown in Fig. 2 is summarized in steps as follows:

Step 1. Traditional full search for first block at every video frame without any early termination algorithm is used. At the start, the bit cost threshold *T* is equal to zero to ensure that all candidates will be checked. In addition, FCSA is disabled for first prediction block at every frame.
Step 2. *T* value is used to stop search if the cost of the current candidate is equal or less than *T*. This means that achieved cost is good enough to stop BMA search.
Step 3. Once current prediction block processing is finished, *T* value is updated by the cost value of the current block according to (1).
Step 4. The motion vector of the current block is tested to determine its position, near or far from center point. If it is near, FCSA is enabled to start next search from center point. Vice versa, if it is far, FCSA is disabled to use traditional full search.
Step 5. Steps from 2-4 are repeated for all prediction blocks in the same frame. For every new frame in a video sequence, FCSA starts from step 1.

D. Software Analysis

A software analysis was performed to evaluate the time saving, BR increase and PSNR decrease using x265 open-source code project for HEVC [10]. Testing results are compared for different 2K resolution video sequences as given in Table II. The largest CU is fixed to 64x64 with a maximum depth level is equal to four, resulting in a minimum CU size of 8x8. The maximum search range used is set to 64, the used entropy coder is CABAC, the number of frames taken in each sequence is 250 and the quantization parameter is 28. All the simulations were carried on Windows 7 OS platform with Intel Xeon X5690 at 3.46 GHz CPU and 96 GB RAM.

Table II shows that our proposed FCSA achieves around 40.74% time saving with respect to full search, in addition, FCSA saves the same PSNR quality of the full search and needs just 0.92% extra bit rate. Table II shows that FCSA exploits its idea about center search and saves large amount of time in video sequences where there is no rapid motion. On the other hand, the time saving is less in video sequences that have rapid motions. Table III compares our proposed FCSA with previous fast searches introduced for HEVC in [9]-[7]. FCSA achieves better time saving than [9] with better PSNR and BR. While compared with [7], FCSA does not achieve the same time saving, but, achieves better PSNR and BR with notable values. Generally, FCSA shows that it does not affect PSNR and bit rate and saves the same quality of full search. In addition, FCSA is more compatible with hardware and exploits parallelism in hardware unlike other fast searches.

TABLE II. RESULTS OF FCSA VERSUS FULL SEARCH ALGORITHM

Sequence (Resolution)	ΔTime (%)	ΔBR (%)	ΔPSNR (db)
Duck (1280x720)	-54.04	0.88	0
Ice (1280x720)	-46.03	0.84	-0.012
Blue_Sky (1280x720)	-31.27	0.85	-0.004
Shields (1280x720)	-38.88	0.92	-0.019
Stockholm (1280x720)	-28.96	0.97	0
Park_Joy (1280x720)	-45.28	1.04	0
Average	-40.74	0.92	-0.006

TABLE III. RESULTS OF FCSA VERSUS PREVIOUS FAST SEARCH ALGORITHMS

Sequence (Resolution)	ΔTime (%)	ΔBR (%)	ΔPSNR (db)
Belgith et al. [9]	-40.68	1.10	-0.120
Anand Paul [7]	-77.43	3.44	-0.183
Proposed FCSA	-40.74	0.92	-0.006

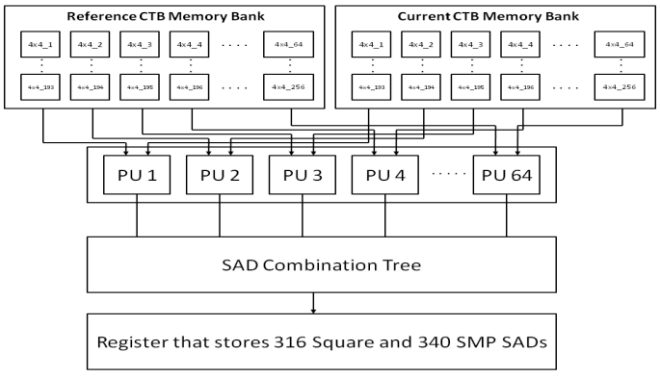

Fig. 3. Architecture of highly parallel SAD unit

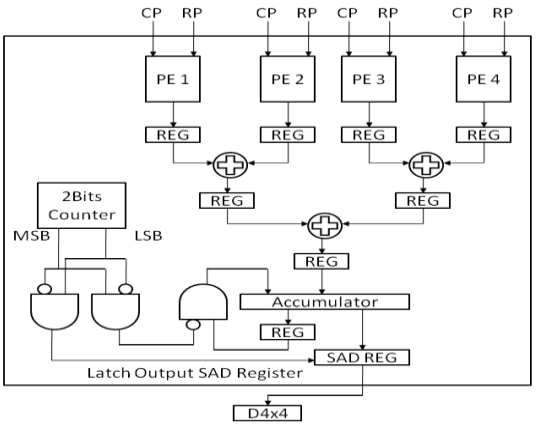

Fig. 4. Processor Unit (PU) Architecture

IV. PROPOSED PARALLEL SAD ARCHITECTURE

In this section, we present high-performance low cost SAD unit architecture for ME unit in HEVC that provides calculations for various square and symmetric partition SAD sizes from 4x4 up to 64x64. In addition, it exploits parallelism as a main feature supported by HEVC. The abstraction levels of our implementation for fast SAD unit are shown in Fig. 3. The architecture is composed of 1) current block memory, 2) reference block memory, 3) processing unit (PU), 4) an SAD

of combination tree and 5) registers for storing all SAD combinations.

Concerning PUs, there are 64 PUs each contains four processor elements (PEs). A PU, shown in Fig. 4, calculates 4x4 blocks SAD every four cycles where PE, calculates the absolute difference between two pixels, one pixel from the current CTB and the other from the candidate CTB at the reference frame. Fig. 4 shows the PU, group of four PEs, that calculates one row of 4x4 SADs every cycle. Hence, these values are stored in registers and accumulatively added to calculate the SAD of the whole row. Next, the SAD of the whole row is fed to the accumulator to be added incrementally with the SAD value of the next three rows to calculate the SAD value of a 4x4 block. Therefore, the output 4x4 SAD value is latched every four cycles using the 2-bit counter shown in Fig. 4. An extra logic gates shown in Fig. 4, are used to reset the accumulator every four cycles.

PE calculates the absolute difference between two pixels. We use 8-bit comparator based on the 2-bit comparator architecture which is proposed in [11]. The SAD combination tree combines the sixty-four 4x4 SADs from the PU to return all the combinations from 4x4 up to 32x32 every four cycles. Hence, this array of PUs are used four sequential times to calculate the remaining larger partitions (64x64, 64x32, 32x64). Consequently, it takes 16 cycles to calculate all possible partitions from 4x4 up to 64x64. Since the PU is used four sequential times, the SADs output from the PU at every four cycles have to be stored at different registers.

V. Prototyping Results and Duscussion

The proposed architecture was implemented in Verilog HDL, and synthesized using Xilinx ISE. The target platform is Xilinx Virtex-6 XC6VLX-550T FPGA. In addition, simulation and function verification of our architecture were done using Xilinx ISE simulator. Our synthesis results are shown in Table IV and compared to results proposed in [12]. It can be observed that our architecture can achieve a maximum frequency of 550 MHz where it utilizes 15,042 LUT which represent 8% of LUTs resources and 32,012 slice register bits which represent 4% of slice register resources.

From simulation results, our architecture takes 16 cycles to obtain all SAD combinations from 4x4 up to 64x64 using 64 PUs. This array of PUs takes four cycles to calculate its SADs combinations from 4x4 up to 32x32. Hence, it is used four times to calculate the larger SADs combinations up to 64x64. Since, our proposed FCSA saves about 40% of search points, so, our proposed architecture using FCSA can process up to 30 4K resolution fps with ±16 pixels search range.

TABLE IV. Synthesis results of the proposed Architectue

	Yuan et al. [12]	**Proposed Hardware**
No. of slice registers	19744(2.9%)	32012(4%)
No. of slice LUTs	55346(16%)	15042(8%)
Maximum frequency	110 MHz	550 MHz
Available SAD partitions	8x8 up to 32x32	4x4 up to 32x32
No. of processed 64x64 CTBs every second	220M CTBs	458M CTBs

VI. Conclusion

Fast center search algorithm (FCSA) has been presented in this paper. The proposed FCSA starts searching from the center point of the search window. In addition, it uses stop criteria to terminate the search once it finds an acceptable bit cost. FCSA showed better performances in terms of time saving, BR and PSNR compared to full search and other fast search algorithms. FCSA speeds up ME process by 40% with respect to full search with a negligible degradation of PSNR and BR. In addition, hardware architecture has been presented that implements the proposed FCSA algorithm. The architecture was prototyped, simulated and synthesized on Xilinx Virtex-6 XC6VLX-550T FPGA. This hardware uses 64 PUs to meet the requirements of 30 4K fps with ±16 search window video coding with an operating frequency of 550 MHz. Totally, 16 clock cycles for every 64x64 CTB are needed to obtain SAD values of all combinations.

Acknowledgment

We would like to thank Egypt-Japan University of Science and Technology (E-JUST) for the continuous support and the Egyptian Ministry of Higher Education for funding this work.

References

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, December 2012.

[2] M. E. Sinangil , V. Sze , M. Zhou and A. P. Chandrakasan "Hardware-aware motion estimation search algorithm developmentfor high-efficiency video coding (HEVC) standard," Proc.IEEE Int. Conf. Image Process. (ICIP), pp.1529 -1532, October 2012.

[3] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards including High Efficiency Video Coding (HEVC)," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1668-1683, December 2012.

[4] F. Sampaio, S. Bampi, M. Grellert, L. Agostini and J. Mattos, "Motion vectors merging: low complexity prediction unit decision heuristic for the inter-prediction of HEVC encoders," IEEE International Conference on Multimedia and Expo (ICME), pp. 657-662, July 2012.

[5] A. Hashad, R. Sadek, and S. Mandour, "A novel reduced diamond search algorithm with early termination for fast motion estimation," Int. Journal of Video & Image Process. and Network Security (IJVIPNS), vol. 10, no. 04, Agust 2010.

[6] L. Bo, L. Wei, and T. Ming, "A fast block matching algorithm using smooth motion vector field adaptive search technique," J. Comput. Sci & Technol., vol. 18, no. 1, pp. 14-21, January 2003.

[7] A. Paul, "Adaptive search window for high efficiency video coding," Springer J. Sign. Process. Syst, Septemper 2013.

[8] C. Madhuvappan, and J. Ramesh, "Video compression motion estimation algorithms - A survey," Int. J. SCi. Engineering Research, vol. 5, no. 2, Febraury 2014.

[9] F. Belghith, H. Kibeya, H. Loukil, M. Ayed, and N. Masmoudi, "A new fast motion estimation algorithm using fast mode decision of high-efficiency video coding standard," J. R. -Time Img. Process., Febr 2014.

[10] X265 team of ISO/IES Moving Picture Experts Group (MPEG), and ITU-T Video Coding Experts Group (VCEG) through JCT-VC reference software, http://x265.org/index.html

[11] L. Yufei, F. Xiubo and W. Qin, "A high-performance low cost SAD architecture for video coding," Consumer Electronics IEEE Transactions, vol. 53, no. 2, pp. 535-541, May 2007.

[12] X. Yuan, L. Jinsong, G. Liwei, Z. Zhi and R. Teng, "A high performance VLSI architecture for integer motion estimation in HEVC," IEEE 10th Int. Conf. on ASIC (ASICON), Oct. 2013.