International Journal of Engineering & Technology, 7 (4.10) (2018) 928-934

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Review on Architectures of Motion Estimation for Video Coding Standards

Prayline Rajabai C[1], Sivanantham S[2]*

School of Electronics Engineering, Vellore Institute of Technology
Vellore, Tamil Nadu-632014, India.
**Corresponding author E-mail: ssivanantham@vit.ac.in*

Abstract

Various video coding standards like H.264 and H.265 are used for video compression and decompression. These coding standards use multiple modules to perform video compression. Motion Estimation (ME) is one of the critical blocks in the video codec which requires extensive computation. Hence it is computationally complex, it critically consumes a massive amount of time to process the video data. Motion Estimation is the process which improves the compression efficiency of these coding standards by determining the minimum distortion between the current frame and the reference frame. For the past two decades, various Motion Estimation algorithms are implemented in hardware and research is still going on for realizing an optimized hardware solution for this critical module. Efficient implementation of ME in hardware is essential for high-resolution video applications such as HDTV to increase the decoding throughput and to achieve high compression ratio. A review and analysis of various hardware architectures of ME used for H.264 and H.265 coding standards is presented in this paper.

Keywords: *H.264/AVC; H.265/MPEG; Hardware architecture; Motion Estimation; video coding*

1. Introduction

Tremendous advancements in video technology and consumer electronics for the past few decades led to the requirement of processing video data with lesser complexity and optimized performance. Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T) Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) introduced many video coding standards [33]. The latest coding standard H.265, widely known as High-Efficiency Video Coding (HEVC) is the successor of H.264/MPEG-4 AVC. These coding standards are used for various applications such as the broadcast of HD TV signals through satellite, Internet and mobile network video, cam-coders, Multi-view video, scalable video for streaming applications, security applications, real-time conversations such as video chatting, video conferencing, etc. In any video processing application, either the video data is either stored or transmitted over the network. Real-time video data is of enormous size which increases the traffic in the communication network significantly. Hence video data has to be compressed/encoded by exploiting the spatial and temporal redundancies. Video coding systems, therefore should provide higher coding efficiency; high throughput and lower energy consumption as well, because most of the video are captured using battery operated devices.

Among various video coding standards in the literature, nowadays H.264 and H.265 are widely used for various applications. Perhaps, H.265 is computationally complex compared to H.264 with the trade-off between high throughput and optimized performance. The efficiency of H.265 is double than that of H.264 [33]. Most of the features of various modules like Transform, quantization, motion estimation, motion compensation, intra-prediction and deblocking filter (DBF) are common for both the coding standards.

Discrete Cosine Transform (DCT) and Inverse Discrete Cosine Transform (IDCT) used in H.264 and H.265 are the same, whereas H.264 uses 4×4 and 8×8 Transform Unit (TU) sizes and H.265 uses 4×4, 8×8, 16×16, 32×32 TU sizes for DCT and IDCT. Transform and inverse transform are performed on blocks of pixels and hence the colour transitions at the edges of the block are not smooth. The sharp colour transitions at the edge of each block will affect the visual quality of the decoded video. This degradation in the visual quality is compensated by a module called DBF at the decoder. It eliminates the blocking artifacts on the block boundaries between two blocks and thus smoothens the image pixels on both sides of the boundary. Smoothening of colour transitions at the edges of each block improves the visual quality of the reconstructed video frames. The output of DBF is used as a reference frame for subsequent video coding called inter prediction. Inter prediction comprises of Motion Estimation (ME) and Motion Compensation (MC).

Motion estimation is a technique which is used to remove the temporal redundancy between the video frames and thus providing high compression ratio [11]. In motion estimation, both the current frame and the reference frame are divided into non-overlapping blocks of size N×N. Comparision is performed for each block in the current frame with a candidate block within the defined search range in the reference frame. The displacement between a block in the current frame and the best match in the reference frame is called a Motion Vector (MV). This MV describes the position of a block in the current frame and this information alone has to be coded. Thus the compression efficiency of a codec is improved using ME. Motion Compensation (MC) reconstructs the current frame based on the MV obtained from ME.

ME algorithms are generally classified as full search algorithms and fast search algorithms. Fast search algorithms are again categorized as lossy and lossless algorithms. Lossy algorithms are classified into five types based on i) reduction in search positions, ii) simplification of the matching criterion, iii) bit-width reduction, iv) predictive search and v) hierarchical search. Full search and fast full search algorithms belong to the lossless category [11]. Hardware architectures for motion estimation algorithms are required for real-time video coding applications with low area and high throughput [27]. This paper presents a brief review and analysis of the hardware architectures of Motion estimation implemented for H.264 and H.265 coding standards. This paper is organized as follows. Motion estimation and compensation techniques used for H.264 and H.265 coding standards are discussed in Section 2. Section 3 presents about the various hardware architectures of motion estimation, section 4 presents the comparison of various hardware architectures of motion estimation, section 5 presents the design challenges of motion estimation and the paper is concluded in section 6.

2. Motion estimation algorithm

In a video, often two successive frames are similar unless there is any movement of the object captured by the camera, change in the camera position or changes in illumination [25]. So in a video data, successive frames with respect to time can contain same pixel data which are known as temporally redundant data. This temporal redundancy is exploited to encode the video data by measuring the amount of displacement or the variation of a block between two successive frames. Motion estimation (ME) is the technique to measure and estimates the amount of the displacement of a block or an object between the successive images/frames in a video and provides the motion vector which is used to encode the video data by removing the temporal redundancy.

Various ME algorithms are proposed and implemented to estimate the MV. These algorithms follow two different approaches for motion estimation; i) pixel-based ME and ii) block based ME [4]. Pixel-based motion estimation also called as pel-recursive algorithm calculates the motion vector for every pixel in the image. Hence it is time-consuming it makes unsuitable for real-time video processing applications [25]. The Block-based motion estimation also called as block matching algorithm is faster than the pixel-based approach.

The motion in a video can be classified into i) translational motion and ii) rotational motion [30]. In block-based ME technique, the image frame is partitioned into non-overlapping blocks of size 16×16, 8×8 or 4×4 and the motion vector is calculated for each block in the image. Finally, a single motion vector is calculated for the whole image frame and this determines the translational motion of the video. This technique does not hold good for real-time video sequences as there can be rotational motion also. However, block-based ME technique is utilized in most of the video coding standards due to its effectiveness in compression [30]. The block matching algorithm for ME varies on the various parameters like block distortion measure, block size and search range. Block-based ME techniques are again classified into full search algorithm and fast search algorithm [11]. In full search algorithm, the motion vector is calculated for all search candidates within the search window and the search candidate with minimum Sum of Absolute Difference (SAD) is taken as the best match. Even though this technique provides optimum results, since all search candidates are analyzed, it is computationally expensive and time-consuming which makes the video processing critical in real-world applications. Fast search ME algorithms are proposed to reduce the computational cost and time [15, 31]. Few fast search ME algorithms seen in literature [16, 18, 19, 25, 29, 32, 34, 42] are

i. Two-Dimensional Logarithmic Search Algorithm (TDL)
ii. Three-Step Search Algorithm (TSS)
iii. New Three-Step Search Algorithm (NTSS)
iv. Four Step Search Algorithm (4SS)
v. Cross Search Algorithm (CSA)
vi. One-at-a-Time Search Algorithm (OTA)
vii. New One-at-a-Time Search Algorithm (NOTA)
viii. Modified Three-Step Search Algorithm (MTSS)
ix. Diamond Search Algorithm (DS)
x. New diamond search algorithm (NDS)
xi. New cross-diamond search algorithm (NCDS)
xii. Hexagonal Search Algorithm (HS)

Though it provides less optimal results compared to full search algorithms, it reduces the computational overhead to a great extent. Based on the algorithm characteristics, the fast search ME algorithm is classified into three types [4]. They are i) Search candidate reduction, ii) Simplification of matching criteria instead of the classical SAD and iii) Predictive search. Figure 1 shows the picture representation of ME. The block with minimum distortion or SAD is considered as the best match and the resultant motion vector is then used for the compression of the video data.

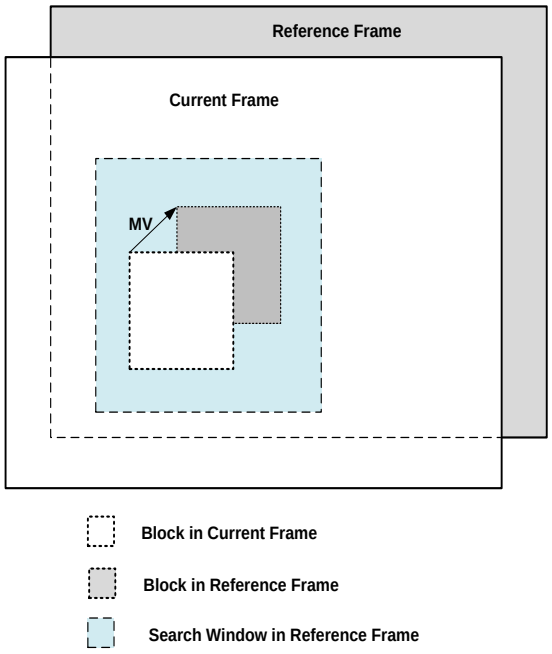

Fig. 1: Motion Estimation

$$SAD(i,j) = \sum_{k=0}^{N-1}\sum_{l-0}^{N-1} |C(i,j) - R(i+k, j+l)| \quad (1)$$

$$SAD_{min} = \min\left(SAD(i,j)\right) \quad (2)$$

Where, *C(i,j)* represents the pixel value in the current frame and *R(i+k, j+l)* represents the same pixel value in the reference frame with the displacement of the pixel as 'k' in the horizontal direction and 'l' in the vertical direction. The video coding standards do not impose any restriction on using different algorithms or different hardware architectures for ME [25]. It can vary based on the targeted application. It is seen from the literature that the computational complexity for ME is more than 50% of the overall video coding [9]. Though various ME algorithms and hardware architectures are implemented for different coding standards, extensive works on optimization of these motion estimation algorithms and hardware architectures are still going on due to the fact of reducing the computational complexity, hardware resources and the power consumption of this block. Different coding standards use different block sizes for ME. Table 1 shows the different block sizes used for ME in various coding standards. HEVC video coding standard supports three different Motion Vector Prediction (MVP) modes for predicting the motion vectors known as inter mode, skip mode and merge mode [20] to improve the compression efficiency [17, 41].

Table 1: Different block sizes used in motion estimation [4, 9]

Video Coding Standard	Block size	Number of Motion Vector
MPEG-2	16×16	1
MPEG-4	16×16, 8×8	5
AVS	16×16, 16×8, 8×16, 8×8	9
H.264/AVC	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4	41
H.265	64×64, 32×32, 16×16, 8×8, 4×4 and its sub-blocks of all block sizes	1360

3. Hardware architectures for motion estimation

Several hardware architectures have been proposed to implement the ME algorithms to optimize the coding throughput, reducing the hardware complexity, improving the performance of the hardware design. These hardware architectures can be broadly classified into four categories such as architectures i) without parallelism ii) with parallelism iii) with pipelining and parallelism and iv) re-configurable architectures. Reconfigurable architectures can be configured for multiple coding standards and for various applications. High-performance hardware architecture to perform motion estimation is necessary to achieve the high throughput requirement in real-time video codecs [6, 23]. Low power and high-speed architectures can be realized by implementing a dedicated hardwired design for fast search ME algorithm [26]. By implementing the ME algorithm in hardware, we have the advantages like i) reduction of computation complexity by employing the sub-partition SAD reusing ii) increase in the throughput by employing parallelism, i.e., by scheduling the Processing Element (PE) to work in parallel iii) Regularizing and simplification of the memory access and the control logic. These advantages are to be viewed as the principles while designing the hardwired architecture for ME [22]. Resource sharing, pipelining and parallelism can be implemented in hardware to increase the throughput by exploiting the ME algorithm. In hardware, ME algorithm is basically performed in three stages. The absolute difference for each pixel of the current MB is calculated in the first stage then the SAD is calculated at the second stage and in the final stage MV is identified by finding the minimum SAD [22]. There are many Fractional Motion Estimation (FME) architectures proposed to implement multi-iteration algorithms which limit the design throughput and increase the latency. FME architectures [21, 38] implementing single iteration algorithms achieve high throughput with the degradation in performance. In general, the hardware architectures of motion estimation has three main blocks i) the absolute difference unit ii) the adder tree and i) the compare-select unit [2].

3.1. Motion estimation architecture without parallelism

Motion Estimation without parallelism is implemented by Liu *et al*., in [22] as propagate partial SAD hardware architecture. In Non-parallel architectures, the ME algorithm is processed in three stages as i) Calculation of the absolute difference on each pixel of current macroblock ii) Calculation of the SAD for all pixels in a current macroblock for every search position iii) Decision of the final MV as the search candidate with minimum SAD value. Performing the MV calculation in three stages without employing parallelism critically consume a massive amount of time and hence non-parallel architectures are suitable only for low-resolution video applications and when the search range is small [22]. Nowadays the users of electronic gadgets expect high-resolution video and hence the architecture without parallelism could not support the principles of the hardwired ME to a great extent. For high resolution and higher complexity applications, architecture with parallelism outperforms the architecture without parallelism. Figure 2 shows a typical SAD Architecture without parallelism [28].

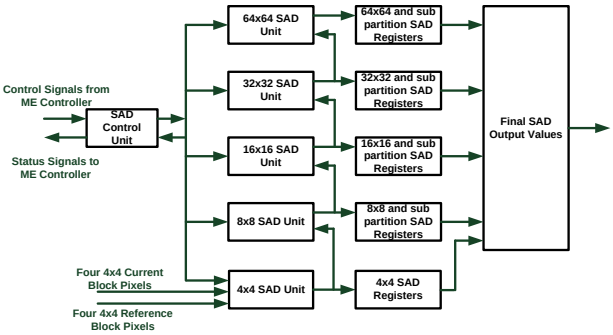

Fig. 2: SAD architecture without parallelism

3.2. Motion estimation architecture with parallelism

The hardware architecture for ME using parallelism was implemented in [10, 13, 14, 22, 28, 36, 37]. Nalluri P *et al*., [28] proposed a low complexity SAD architecture for variable block size motion estimation for HEVC video coding. Due to the asymmetric motion partitioning in HEVC, the motion estimation task becomes very complex [28]. This architecture employs parallelism at different levels to achieve the optimized results. The SAD architecture with parallelism is shown in Figure 3. The levels of parallelism are limited by the data bus width size from the memory to the SAD block. For a non-parallel architecture, data read from memory are 128 bits for the current block and 128 bits for the reference block for the pixel size of 8 bits and the block size of 4×4. Hence the total bus width size is 256 bits. For a parallel version of depth 0, the number of bits for the current block and the reference block will increase to 512 bits and hence the total bus width size is 1024. The bus width size for any parallel version of the SAD architecture is given as

$$w = 256 * 4^{p} \tag{3}$$

where, 'w' represents the data bus width size and 'p' represents the number of parallel stages. It is inferred from the above equation that the bus width size increases as the number of parallel stage increases which results in the increase in the resource utilization. It is also known that increase in the memory data bus width decreases the speed of operation and thus this architecture operates at a very low frequency of 30-60MHz less than the partial propagate SAD architecture proposed by Liu *et al*., [22].

The SAD tree architecture implemented in [22] is a highly parallel architecture. It is a two-stage SAD tree architecture where the absolute difference and the carry of one 4×4 PE is calculated and given to a 4:2 compressor based Carry Select Adder (CSA) which computes the carry and the sum of a 4×4 SAD module. The carry and sum of sixteen 4×4 SAD blocks are stored in buffers and the variable block size adder tree calculates the SAD's. This architecture improves the processing speed by employing two pipeline stages and achieves a frequency of 204.8MHz at the price of 88.5k gates. The architecture implemented by Tseng *et al*., [37] reduces the number of clock cycles required for processing and hence achieves high speed at the cost of high resource utilization.

Video applications targeted by most of the existing works are for the resolutions up to HD or 4k. For ultra-HD or 8k (7680 × 4320) applications which need more throughput, more efficient hardware architecture is proposed in [9] by Gang *et al*., They have implemented Fractional Motion Estimation (FME) architecture which employs high degree of parallelism and pipelining. A novel Bilinear Quarter pixel Approximation (BQA) technique proposed in this architecture reduces the complexity of the interpolation filter. In H.265, interpolation filters of 7 tap and 8 tap are used to improve the coding efficiency. The inclusion of interpolation filter

provides 21.7% of bitrate reduction in H.265 compared to H.264 [39] but the computation complexity increases due to the complex interpolation filter and factional search process. Pipelining and parallelism are employed by exploiting the neighboring pixel correlations. Memory organization is complicated to achieve pipelining. Even though it supports ultra HD encoding applications, there is degradation in the average PSNR and also it utilizes more hardware resource as well as power.

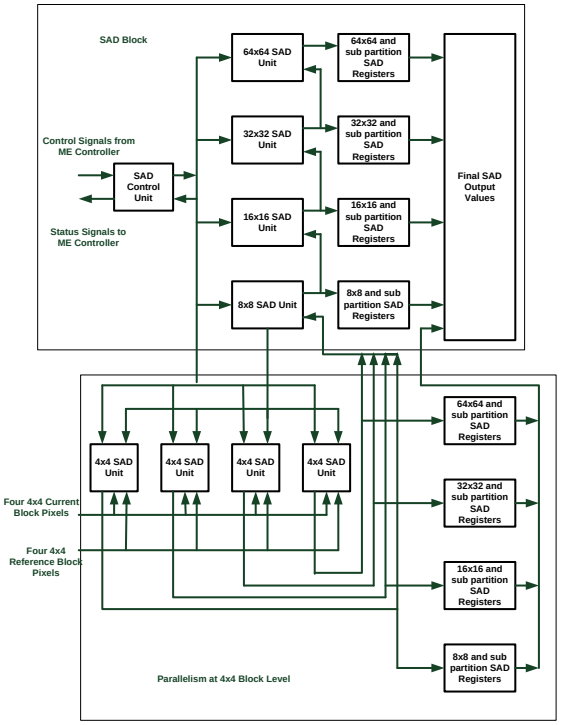

Fig. 3: SAD architecture with parallelism

A low complexity hardware architecture for motion estimation implemented using modified xor function in [2] The modified xor function replaces the conventional SAD architecture and it simplifies the calculation of SAD. Parallelism and reuse of the partial SAD values of the smaller blocks is employed which increases the throughput of the architecture. Though it claims that is architecture is simple with good video quality there is a slight decrease in the video quality. A fully pipelined and parallel three-step search architecture is implemented in [12]. In this architecture, nine Processing Elements (PE) are used in parallel to compute 9 SAD's in each step and each step requires 256 clock cycles to calculate the SAD's.

3.3. Reconfigurable architecture for motion estimation

The reconfigurable architecture allows different configurations that can be customized to suit our application [3]. Reconfigurability achieves higher performance and achieves higher flexibility [7]. Reconfigurable architectures can be static or dynamic. Dynamically reconfigurable architectures are more suitable for implementing multimedia applications [3].

Thomas *et al*., [35] proposed a reconfigurable data flow engine for HEVC Motion Estimation and Lu *et al*., [24] proposed a reconfigurable on-chip motion estimation architecture which supports multiple video coding standards. ME architecture implemented in [35] and [40] supports Variable Block Size Full Search Motion Estimation (VBS-FSME), unlike the algorithms that belong to the fast search category. A typical reconfigurable architecture for VBS-FSME implemented in [35] is shown in Figure 4 is reconfigurable with respect to the number of PE's and operates at a frequency of 125MHz. It has three functional units, i)SAD generator block which has an array of Processing Elements (PE), ii) SAD Comparator block and iii) memory. Figure 5 shows the internal architecture of each functional units of the reconfigurable VBS-FSME architecture.

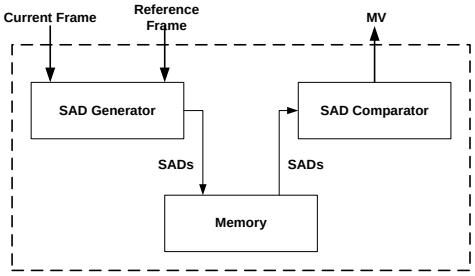

Fig. 4: Architecture of VBS-FSME

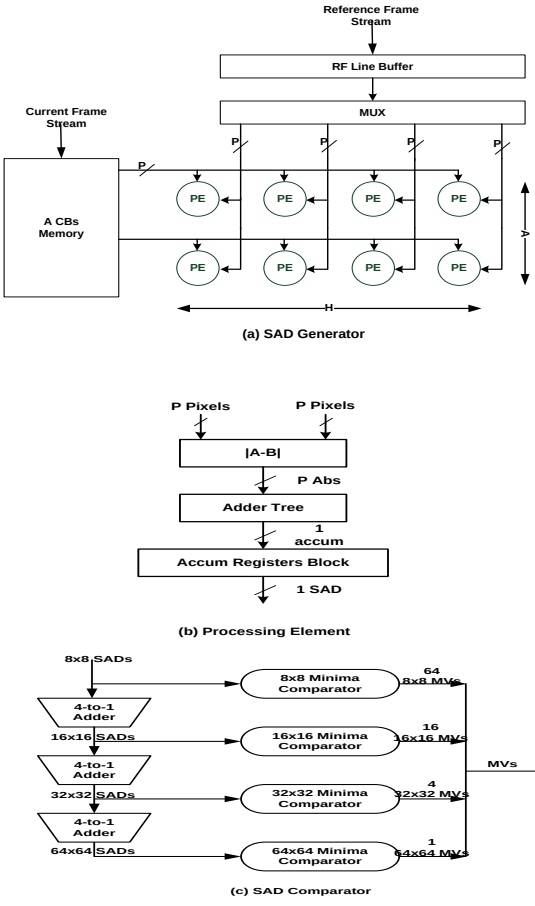

Fig. 5: Functional units of the reconfigurable VBS-FSME architecture

The PE's elements used in the SAD generator is fixed to 16×16 in most of the ME architectures [5, 8] whereas in this architecture, the PE's are scalable in two dimensions both horizontally and vertically to achieve optimized performance. The minima comparators used in the SAD comparator block identify the minimum distortion using the computed SAD and then give out the corresponding MV. This architecture can process around 27 fps of 1080p video using a reference frame and with the search window of 64×64 pixels and is validated in Xilinx Virtex 5 FPGA. The architecture implemented in [40] reduces the memory usage and reconfiguration is supported for various search ranges to have a trade-off between the area and the performance.

Though the PE array is reconfigurable only in the horizontal dimension, the time consumed to perform ME is less compared to other architectures and hence it can support real-time encoding of UHD video at a frequency of 282 MHz. Even though the ME architecture implemented in [1] achieves high throughput compared to [40], the area overhead is huge and hence not suitable for

handheld or battery operated application which requires very less power consumption.
ME architecture implemented in [24] reuses the partial 4x4 SAD blocks and schedules of the PE blocks appropriately to achieve run-time optimization. Simple control logic allows the programmability and the reconfigurability of the architecture. The programmability feature makes the search range flexible and thereby enables the usage of this architecture for multiple video coding standards.

4. Comparison of various hardware architectures for motion estimation

Table 2 shows the comparison of the results of different motion estimation architectures proposed by multiple authors for H.264 and H.265 coding standards.

Various ME algorithms for H.264 coding standard is implemented using Xilinx FPGA in [4] and it is seen that the FBS-DS algorithm can support a maximum resolution of HD video at a low frequency of 129 MHz. DS Algorithm for H.264 is implemented in [26] achieves a frequency of 308 MHz which supports only CIF resolution videos. Only Fast Search algorithms are implemented for H.265 coding standards and the maximum frequency achieved is 188MHz. From Table 2, it is also seen that the area occupancy for the implementation of ME algorithms of H.265 coding standard which supports of very less resolution itself is more than twice than that of H.264. It clearly states that the implementation of ME algorithms for H.265 is computationally more complex compared to H.264 coding standard even for low resolution videos. Also, the hardware implementation for the fast search ME algorithm in [9] achieves optimized results in terms of area and frequency for H.265 coding standard.

Table 2: Comparison of hardware architectures for motion estimation

Author	Coding Standard	Algorithm	Area	Technology	Max supported resolution	Frequency MHz
[4]	H.264	FTSS[1]	78154µm^2	Xilinx FPGA	-	-
		SEA[2]	3302274µm^2	Xilinx FPGA	QCIF–176×144	100
		VBS[3] – DS[4]	1103 LUTs	FPGA	-	135
		FBS[5] - DS	1576 LUTs	FPGA	HD-1920×1080	129
		F2SS[6]	457.5	90nm	CIF @30fps	129
[9]	H.265	FS[7]	199.2k	0.18µm	7680×4320 @30fps	188
[20]	H.265	FS	291.27k	65nm Xilinx Virtex 5 FPGA	NA	171.9
[26]	H.264	DS	3.5k	Xilinx Virtex 5	CIF @128fps	308
[22]	H.264	FS-parallel	88.5k	0.18µm	NA	204.8
[22]	H.264	FS-Non-parallel	84.1k	0.18µm	NA	231.6
[2]	H.264	-	6157 LUT	FPGA Cyclone IV	CIF @30fps	293
[24]	H.264	NA	116.3k	0.18µm	NA	345
[8]	H.264	-	210k	0.18µm	720×576 @30fps	260
[27]	-	ASA[8]	38.2k	Xilinx Virtex 5	1280×720	243

[1] FTSS - Fast Three-Step Search Algorithm
[2] SEA Successive Elimination Algorithm
[3] VBS Variable Block Size
[4] DS Diamond Search
[5] FBS Fixed Block Size
[6] F2SS Fast Two Stage Search Algorithm
[7] FS Fast search
[8] ASA - Adaptive Search Algorithm

5. Design challenges of motion estimation

Motion Estimation is a computationally complex module and various ME algorithms are proposed to achieve optimized results. Most of these algorithms are implemented in software and few in hardware. The hardware architectures are designed to reduce the area, power and to achieve high speed. However, due to high data dependencies of the algorithm and high computational requirement, implementation of hardware architecture with low power, reduced area is very complex and challenging for real-time applications [8, 40]. Optimization with respect to area can be achieved by reusing the PEs. Novel memory access and address generation schemes can be thought for power optimization and to reduce the computational complexity. Power optimization can be achieved by designing ME architectures with systolic arrays. Novel scheduling methodologies for processing in a PE can help to achieve high speed. Apart from area reduction, power optimization and high processing speed, the implemented architecture should also maintain a high compression ratio and achieve good video quality. High compression ratio and excellent video quality can be achieved by tweaking the essential aspects of ME such as the block size, search area and distortion metric. Usage of the variable block size, provision of search area flexibility and distortion metric correction, however increases the computational complexity and hence the area overhead.

Hardware architectures of ME employ the PE array of various sizes. In some architectures the size of the PE array is fixed whereas in few architectures PE arrays are reconfigurable based on the trade-off between various criteria of the targeted applications such as area requirement, throughput requirement and the number of cycles to compute the SAD to have minimum latency in ME. The non-reconfigurable architectures achieves high throughput compared to the reconfigurable architectures as the time required to calculate the MV is high in reconfigurable architectures. Whereas the area occupancy of the reconfigurable architectures are much less compared to the non-reconfigurable architectures. Also, reconfigurable architectures provide the flexibility in configuring the search area, the data dependency limits the throughput of the architectures for real-time applications. Real-time applications require less processing time to support high-resolution videos with excellent quality. Thus, designing a flexible reconfigurable ME hardware architecture to support real-time applications with high throughput, optimized for area and power is a great challenge.

6. Conclusion

This paper presents the fundamentals of Motion Estimation algorithms for various coding standards like H.264 and H.265. It also presents a brief review of the hardware architectures of motion estimation which is used to improve the coding efficiency in H.264 and H.265 coding standards. Various design challenges in

the implementation of ME architectures for these coding standards are also discussed. It is seen that for UHD applications the area overhead of the ME architecture is around 200k and the frequency achieved is 188MHz which is very less. Hence research is still going on to optimize the architecture with novel memory access schemes, parallelism and pipelining. It is also seen that the reconfigurable SoC fabric with pipelining and parallelism achieves the maximum flexibility with optimized run-time results for various applications.

References

[1] Alcocer, E., Gutierrez, R., Lopez-Granado, O., Malumbres, M.P., "Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder", Journal of Real-Time Image Processing, (2016), pp. 1-11.

[2] AlQaralleh, E.A., Abu-Sharkh, O.M., "Low-complexity motion estimation design using modifed xor function", Multimedia Tools and Applications, Vol.75, No.24, (2016), pp.16809-16834.

[3] Cervero, T., Lopez, S., Callico, G., Tobajas, F., De Armas, V., Lopez, J., Sarmiento, R., "Survey of reconfgurable architectures for multimedia applications", In Proc. of SPIE: VLSI Circuits and Systems IV, Vol.7363, (2009), pp. 736303/1 – 736303/12.

[4] Chakrabarti, I., Batta, K.N.S., Chatterjee, S.K., "Motion estimation for video coding- efficient algorithms and architectures", Springer Book Series: Studies in Computational Intelligence, Vol. 590, (2015), pp. 85-108.

[5] Chen, C.Y., Chien, S.Y., Huang, Y.W., Chen, T.C., Wang, T.C., Chen, L.G., "Analysis and architecture design of variable block-size motion estimation for h. 264/avc", IEEE Transactions on Circuits and Systems I: Regular Papers, Vol.53, No.3, (2006), pp.578-593.

[6] Chen, T.C., Chien, S.Y., Huang, Y.W., Tsai, C.H., Chen, C.Y., Chen, T.W., Chen, L.G., "Analysis and architecture design of an hdtv720p 30 frames/s h.264/avc encoder", IEEE Transactions on Circuits and Systems for video technology, Vol.16, No.6, (2006), pp.673-688.

[7] Compton, K., Hauck, S., "Reconfigurable computing: a survey of systems and software", ACM Computing Surveys, Vol.34, No.2, (2002), pp.171-210.

[8] Deng, L., Gao, W., Hu, M.Z., Ji, Z.Z., "An efficient hardware implementation for motion estimation of avc standard", IEEE Transactions on Consumer Electronics, Vol.51, No.4, (2005), pp.1360-1366.

[9] He, G., Zhou, D., Li, Y., Chen, Z., Zhang, T., Goto, S., "High-throughput power-efficient vlsi architecture of fractional motion estimation for ultra-hd hevc video encoding", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.23, No.12, (2015), pp.3138-3142.

[10] Hsia, S.C., Hong, P.Y., "Very large scale integration (vlsi) implementation of low- complexity variable block size motion estimation for h. 264/avc coding", IET Circuits, Devices & Systems, Vol.4, No.5, (2010), pp.414-424.

[11] Huang, Y.W., Chen, C.Y., Tsai, C.H., Shen, C.F., Chen, L.G., "Survey on block matching motion estimation algorithms and architectures with new results", Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, Vol.42, No.3, (2006), pp.297-320.

[12] Jong, H.M., Chiueh, T.D., et al., "Parallel architectures for 3-step hierarchical search block-matching algorithm", IEEE Transactions on Circuits and Systems for Video Technology, Vol.4, No.4, (1994), pp.407-416.

[13] Jou, S.Y., Chang, S.J., Chang, T.S., "Fast motion estimation algorithm and design for real time qfhd high efficiency video coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol.25, No.9, (2015), pp.1533-1544.

[14] Kao, C.Y., Lin, Y.L., "A memory-efficient and highly parallel architecture for variable block size integer motion estimation in h. 264/avc", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.18, No.6, (2010), pp.866-874.

[15] Kerfa, D., Belbachir, M.F., "Star diamond: an efficient algorithm for fast block matching motion estimation in h264/avc video codec", Multimedia Tools and Applications, Vol.75, No.6, (2016), pp.3161-3175.

[16] Koga, T., "Motion compensated interframe coding for video-conferencing", In Proc. of National Conference on Telecommunication, (1981), pp. G5.3.1-5.

[17] Kudo, S., Kitahara, M., Shimizu, A., "Motion vector prediction methods considering prediction continuity in hevc", In Proc. of IEEE Picture Coding Symposium, (2016), pp. 1-5.

[18] Lam, C.W., Po, L.M., Cheung, C.H., "A new cross-diamond search algorithm for fast block matching motion estimation", In Proc. of IEEE International Conference on Neural Networks and Signal Processing, (2003)., Vol.2, pp.1262-1265.

[19] Li, R., Zeng, B., Liou, M.L., "A new three-step search algorithm for block motion estimation", IEEE Transactions on Circuits and Systems for Video Technology, Vol.4, No.4, (1994), pp.438-442.

[20] Lin, J.L., Chen, Y.W., Huang, Y.W., Lei, S.M., "Motion vector coding in the hevc standard", IEEE Journal of Selected Topics in Signal Processing, Vol.7, No.6, (2013), pp.957-968.

[21] Lin, Y.K., Lin, C.C., Kuo, T.Y., Chang, T.S., "A hardware efficient h. 264/avc motion-estimation design for high-definition video", IEEE Transactions on Circuits and Systems I: Regular Papers, Vol.55, No.6, (2008), pp.1526-1535.

[22] Liu, Z., Goto, S., Ikenaga, T., "Optimization of propagate partial sad and sad tree motion estimation hardwired engine for h. 264", In Proc. of IEEE International Conference on Computer Design, (2008), pp. 328-333.

[23] Liu, Z., Song, Y., Shao, M., Li, S., Li, L., Ishiwata, S., Nakagawa, M., Goto, S., Ikenaga, T., "A 1.41 w h. 264/avc real-time encoder soc for hdtv1080p", In Proc. of IEEE Symposium on VLSI Circuits, (2007), pp.12-13.

[24] Lu, L., McCanny, J.V., Sezer, S., "Reconfigurable system-on-a-chip motion estimation architecture for multi-standard video coding", IET Computers & Digital Techniques, Vol.4, No.5, (2010), pp.349-364.

[25] Metkar, S., Talbar, S., "Motion estimation techniques for digital video coding", Springer Briefs in Applied Sciences and Technology, (2013), pp. 33-45.

[26] Mukherjee, R., Mahajan, V., Dhar, A.S., Chakrabarti, I., "High performance visi design of diamond search algorithm for fast motion estimation", Journal of Circuits, Systems and Computers, Vol.25, No.09, (2016), pp.16501-16514.

[27] Mukherjee, R., Saha, P., Chakrabarti, I., Dutta, P.K., Ray, A.K., "Fast adaptive motion estimation algorithm and its efficient vlsi system for high definition videos" Expert Systems with Applications, Vol.101, (2018), pp.159-175.

[28] Nalluri, P., Alves, L.N., Navarro, A., "A novel SAD architecture for variable block size motion estimation in hevc video coding", In Proc. of IEEE International Symposium on System on Chip, (2013), pp.1-4.

[29] Ndili, O., Ogunfunmi, T., "Algorithm and architecture co-design of hardware-oriented, modified diamond search for fast motion estimation in h. 264/avc", IEEE Transactions on Circuits and Systems for Video Technology, Vol.21, No.9, (2011), pp.1214-1227.

[30] Ng, K.H., Po, L.M., Cheung, K.W., Wong, K.M., "Block-matching translational and rotational motion compensated prediction using interpolated reference frame", EURASIP Journal on Advances in Signal Processing, Vol.2010, (2010), pp.1-9.

[31] Paramkusam, A.V., Reddy, V., "A novel fast search motion estimation boosted by multilayer concept", Multimedia Tools and Applications, Vol.75, No.4, (2016), pp.2169-2188.

[32] Po, L.M., Ma, W.C., "A novel four-step search algorithm for fast block motion estimation", IEEE Transactions on Circuits and Systems for Video Technology, Vol.6, No.3, (1996), pp.313-317.

[33] Sullivan, G.J., Ohm, J., Han, W.J., Wiegand, T., "Overview of the high efficiency video coding (hevc) standard", IEEE Transactions on Circuits and Systems for Video Technology, Vol.22, No.12, (2012), pp.1649-1668.

[34] Tham, J.Y., Ranganath, S., Ranganath, M., Kassim, A.A., "A novel unrestricted center-biased diamond search algorithm for block motion estimation", IEEE transactions on Circuits and Systems for Video Technology, Vol.8, No.4, (1998), pp.369-377.

[35] Thomas, D., Momcilovic, S., Pratas, F., Sousa, L., "Reconfigurable data flow engine for hevc motion estimation", In Proc. of IEEE International Conference on Image Processing, (2014.), pp.1223-1227.

[36] Tsai, A.C., Bharanitharan, K., Wang, J.F., Lee, K.I., "Effective search point reduction algorithm and its vlsi design for hdtv h. 264/avc variable block size motion estimation", IEEE Transactions on Circuits and Systems for Video Technology, Vol.22, No.7, (2012), pp.981-988.

[37] Tseng, C.F., Lai, Y.T., Lee, M.J., "A vlsi architecture for three-step search with variable block size motion vector", In Proc. of IEEE 1st Global Conference on Consumer Electronics, (2012), pp. 628-631.
[38] Tsung, P.K., Chen, W.Y., Ding, L.F., Tsai, C.Y., Chuang, T.D., Chen, L.G., "Single-iteration full-search fractional motion estimation for quad full HD H.264/AVC encoding", In Proc. of IEEE International Conference on Multimedia and Expo, (2009), pp.9-12.
[39] Ugur, K., Alshin, A., Alshina, E., Bossen, F., Han, W.J., Park, J.H., Lainema, J., "Motion compensated prediction and interpolation filter design in h. 265/hevc", IEEE Journal of Selected Topics in Signal Processing, Vol.7, No.6, (2013), pp.946-956.
[40] Vayalil, N.C., Kong, Y., "VLSI architecture of full-search variable-block-size motion estimation for HEVC video encoding", IET Circuits, Devices & Systems, Vol.11, No.6, (2017), pp.543-548.
[41] Wang, C.C., Li, G.L., "Hardware-friendly advanced motion vector prediction method and its architecture design for high effciency video coding", Multimedia Tools and Applications, Vol.76, No.23, (2017), pp.25285-25296.
[42] Zhu, S., Ma, K.K., "A new diamond search algorithm for fast block-matching motion estimation", IEEE Transactions on Image Processing, Vol.9, No.2, (2000), pp.287-290.