Fast Motion Estimation Algorithm with Efficient Memory Access for HEVC Hardware Encoders

Farhad Pakdaman[1,2], Moncef Gabbouj[2], Mahmoud Reza Hashemi[1], and Mohammad Ghanbari[1,3]

[1] School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[2] Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland
[3] School of Computer Science and Electronic Engineering, University of Essex, UK
e-mail: {farhad.pakdaman, moncef.gabbouj}@tut.fi, {farhad.pakdaman, rhashemi, ghan}@ut.ac.ir, ghan@essex.ac.uk

***Abstract*— The encoding process in the HEVC standard is several times more complex than the previous standards. Since motion estimation is responsible for most of this complexity, the new Test Zone (TZ) search is usually adopted as the fast search algorithm, to alleviate the complexity. However, the TZ search requires a high rate of access to the off-chip memory, which contributes heavily to the total consumed encoding power. In this paper we demonstrate that the process of finding the best starting search point in this algorithm, does not allow effective reduction of memory access in hardware encoders. As a solution, a new fast motion estimation algorithm is proposed which estimates a proper single starting search point, in addition to an adaptively reduced search range, based on available information from the coded neighboring blocks. The experimental results show that this algorithm on average can reduce the required memory access for ME by ~78% and reduce the integer ME time by ~70%, with only 1.1% Bjontegaard Delta (BD) Rate.**

Keywords*— *Video coding, HEVC, Memory access reduction, Fast motion estimation

I. INTRODUCTION

The High Efficiency Video Coding (HEVC) standard [1] can offer a coding efficiency of up to twice the previous standards, but this efficiency comes at the price of much higher computational complexity and power consumption. Similar to the previous standards, the major part of this complexity comes from the Motion Estimation (ME), which is repeated on block sizes from 64×64 pixels to 4×4 pixels for each Coding Tree Unit (CTU), and also with larger search windows, as the main aim of introducing HEVC is for high resolution video content

As the full search is extremely slow and power consuming, the Test Zone (TZ) search [2] is usually used as the main fast integer ME (IME) algorithm in HEVC. The TZ search offers a fast ME as well as R-D performance almost equal to the full search. The process of ME in this algorithm consists of two main steps. In the first step, the algorithm tests six candidates to find the best starting point for the search. These candidates include motion vectors (MV) of the neighboring blocks, zero motion, the median predictor, and the MV of certain parent block sizes. In the second step, the search is continued with iterative diamond patterns around the best starting point and a refinement process is performed if needed.

Although the TZ search is much faster than the full search, it is still considered slow and heavy, especially for hand-held devices with limited processing power. Furthermore, to cover the large motion intensity, especially in high resolution scenes, TZ search requires a wide search range to perform well. Fetching the huge data associated with the search range, it requires a high rate of access to the off-chip memory, which consumes about half of the processing power of the encoder [3]. The state of the art hardware encoders usually adopt a large on chip SRAM to accommodate the whole search window for a CTU in the memory, and prevent multiple off-chip accesses for sub-blocks [4]. However, fetching this wide search window still imposes a large rate of communications between memory and the encoder. This approach has two drawbacks: first it consumes a considerable amount of power, and second, the high rate of access to memory, can limit the performance of other processing elements in Multiprocessor Systems on Chip (MPSoC) environments, where the memory is shared between processing elements. While several research papers propose algorithm-level methods to speed-up the TZ search, they ignore the importance of off-chip communications, and thus do not succeed in reducing the access to the off-chip memory.

To mitigate this issue, this paper analyzes the TZ search, from a memory access point-of-view, and concludes the source of the problem to be the multiple initial predictors of the TZ search. Then an effective method is introduced to estimate a single starting search point, and an adaptively reduced search range, using the information from the previously coded neighboring blocks. The proposed method is presented at the algorithm level to maximize the gain and also skip design-specific differences of hardware encoders.

The rest of the paper is organized as follows: in section II we summarize the related published methods, section III analyzes the memory access of the TZ search and explains why the conventional methods fail to solve it. Our proposed method is presented in section IV, and experimental results are provided in section V. Finally, we conclude the remarks in section VI.

II. RELATED WORK

Several researchers have tried different methods to reduce the complexity load of ME algorithms. Hu and Yang [5] employ statistical inference on each search point, to decide if it should be considered in TZ search. Yang, Yang, and Jiang [6] proposed to start the search with the more probable horizontal and vertical motions. Researchers in [7] propose a GPU-based

2018 7th European Workshop on Visual Information Processing (EUVIP)
26-28 November, 2018, Tampere, Finland
978-1-5386-6897-9/18/$31.00 ©2018 IEEE

solution to exploit the parallelism for speeding-up ME. As quad-tree structure of the HEVC forces ME on several block sizes, several researches [8][9] investigated fast block partitioning to avoid ME on unnecessary block sizes.

An effective approach to reduce the complexity of the TZ search is to adaptively reduce the required search range, for each block. Dai *et al.* [10] proposed a method that decides the search range based on the distribution of the MV difference in previously coded blocks. In [11] the standard deviation of the motion vector predictors is used alongside block size-dependent parameters to decide the search range. Researchers in [12] however, obtain an Adaptive Search Range (ASR) through a weighted average of MVs from neighboring blocks, such that blocks with higher similarity in depth intensity receive higher weights. The method presented in [13] also uses MV from the collocated block in previously coded block to set the search range. Proposed method in [14] reduces the number of search directions after the second round of diamond search in TZ search, because statistics indicate that after the second round, the best direction barely changes between two consecutive rounds.

Although all the above mentioned methods reduce the complexity and power consumption of the HEVC encoding, it should be noted that in modern hardware systems, the power consumption for *communications* is gaining on *computations*. As a result, accessing the off-chip memory, consumes about half of the total encoding power [3]. While the above-mentioned methods [10]-[14] reduce the search range which potentially can reduce the access to the memory as well, in section III we explain why using multiple initial predictors, prevents them from effectively reducing the access to off-chip memory.

A direct approach to reduce the off-chip communication in the encoder, is to compress the reconstructed frames before writing them into the off-chip memory [3][15]. This way, lower access rate is required for obtaining the reference frames. However this approach has two main shortcomings: first, lossless compression has a limited compression ratio, and second, random access to the reference frames will be complicated, because of the serial nature of lossless coding. A hardware approach is to avoid repeated access to the shared parts of the reference window between the neighboring blocks. To do so, researchers proposed data reuse schemes [16][17] which can greatly reduce the off-chip memory access. However this approach requires a considerable increase of on-chip SRAM, which is expensive and not available for all implementations [4].

A scheduling approach is presented in [18] to reduce the memory access, where authors propose to accumulate the requests to nearby memory locations, such that the number of total access to the memory is reduced. Moreover, Sinangil *et al.* investigated the trade-off between memory access and coding efficiency, through analyzing the memory requirements of different partitioning schemes [19]. They show that memory access and on-chip area can be saved significantly, if certain block sizes are ignored in hardware. However, their approach results in considerable loss of coding efficiency.

Most of the presented approaches either ignore the importance of memory access and concentrate on the computational complexity, or provide solutions at the hardware-level that have limited potential in mitigating the problem. In what follows, we present our algorithm-level solution, which tackles the problem from the source, and hence provides better memory access reduction.

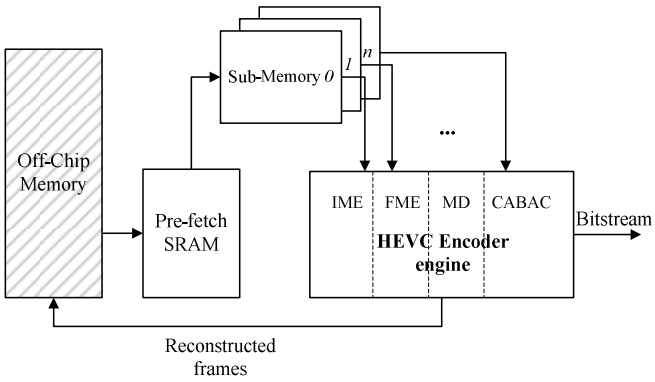


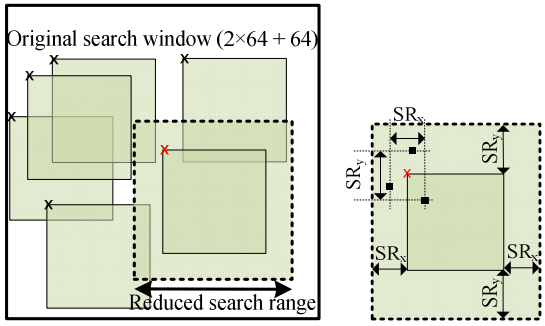
Fig. 1. Abstract architecture of a nominal hardware HEVC encoder

III. MOTIVATION

To have a better perspective of the off-chip memory access, Fig. 1 presents an abstract architecture of a nominal hardware HEVC encoder, which is common among different implementations [4][14]. To encode each block of a frame, that block, and its associated search window in the reference frame, should be fetched from the off-chip memory, into the *pre-fetch* SRAM. This operation usually takes place before the encoding process of the previous block is finished, to cover the latency of accessing the off-chip memory. Then through different stages of the encoding pipeline, this data will be fed into the on-chip SRAMs, as parts of the memory sub-system, to serve as module specific memories. This way considerable off-chip traffic and bandwidth is saved, through the fast and power-efficient on-chip communication. After the encoding process is done, the reconstructed blocks are stored off-chip again, to serve as the reference for the upcoming frames.

As mentioned in related works section above, several previous works proposed adaptive search range reduction techniques to reduce the computational complexity of TZ search. However, they all fail to take advantage of the reduced search range to effectively reduce the required memory transactions. The underlying cause for this inefficiency is that, the first step of the TZ search needs to test six initial motion vector predictors, to decide the best starting point. This operation requires fetching six blocks of 64×64 pixels into the encoder. Only after the best starting point is set, the above mentioned methods can limit the range of the search process around the starting point, and thus partially reduce memory transactions.

Fig. 2 (a) illustrates this phenomenon by example, where an original search range of 64 pixels is set for a CTU of 64×64 pixels, with six starting points (as in common test conditions [20]). Since the initial predictors can be spread anywhere in the original search window, the hardware encoders fetch the whole search range beforehand (to pre-fetch SRAM in Fig. 1) to avoid design complications [4][14]. Hence, no memory access



(a) (b)

Fig. 2. (a) Memory access for testing initial predictors, plus refinement search window around the best starting point. All shadowed boxes are fetched seperately (b) Example of proposed memory access reduction based on three MVs from neighboring blocks

reduction can be achieved. Even if a hypothetical ideal hardware design could access only these six blocks (associated with the six initial predictors) instead of the whole search window, the amount of memory transactions to fetch these blocks would be 67% of the whole search window, for the values of this example (6×64×64 pixels, out of 192×192 pixels). Please note that this is in addition to the reduced refinement search window, which the ideal encoder needs to fetch after deciding the best starting point. Furthermore, it is important to note that the SRAM memory of the encoder chip does not function as a cache memory in a general purpose processor, and thus the locality of accessed addresses cannot be exploited for memory access reduction.

Considering another 64×64 pixels for fetching the current CTU, the required memory transactions of the ASR methods [10]-[14], for each CTU will be $(6+1)\times 64^2 + (64+2RSR)^2$, where *RSR* represents the Reduced refinement Search Range. Hence, even when $RSR \rightarrow 0$, 8×64^2 pixels should be accessed. Comparing this with the case of the common test conditions, including 192×192 pixels for search window and 64×64 pixels for the current block, 20% will be the upper bound of memory access reduction for the ASR methods, in an ideal hardware encoder.

This example clearly demonstrates that to be able to reduce the access to the off-chip memory, reducing the search range is not enough and having a single starting point is vital to benefit from the search range reduction.

Investigations of methods [10]-[14] reveal that in homogeneous motion regions, which include the major area of a frame, MVs of the neighboring blocks have a high correlation with the motion of the current block. However, in complex motion areas they tend to be more random and uncorrelated. Based on the same observation, we propose to use the MV of three neighboring blocks, to estimate a single starting point for the TZ search. To do so, instead of accessing the picture block associated with these MVs to find *the best* starting point (as in baseline TZ search), we analyze their coordinates to *estimate* a single starting point. Considering the presented example in Fig. 2 (a) with this new method, only the reduced search range (dotted rectangle) will be required for ME.

IV. PROPOSED METHOD

As explained above, the test for the best starting point is a major source of memory access in TZ search. To alleviate this, MVs from neighboring blocks are used to estimate an early starting point for the search. To do so, MVs of blocks: left (*Pred_A*), above (*Pred_B*) and above-right (*Pred_C*) of the current block are used, which are highly correlated to the motion in current block. Equation (1) estimates an starting search point (*SP*) for the current block, where *MaxMVx* denotes the maximum value between MVs of the three neighboring blocks, in x direction and the opposite for *MinMVx*.

$$SP_x = (MaxMV_x + MinMV_x)/2\ ,\ SP_y = (MaxMV_y + MinMV_y)/2 \quad (1)$$

In homogenous motion regions where neighboring blocks have similar MVs, a small search range is enough to find the best vector, while in complex motion, a wider search range is required to compensate for the inaccurate starting point. Since the variation of the motion in a certain region can indicate the motion complexity, we define the search range based on the maximum difference between MVs of the neighboring blocks, as depicted in (2). This equation considers a rectangular search window with dimensions according to the size of maximum difference between MVs in that dimension.

$$SR_x = MaxMV_x - MinMV_x,\ \ SR_y = MaxMV_y - MinMV_y \quad (2)$$

$$Thr_x = \mu_{|MV|,x} + \beta\sigma_{|MV|,x}\ ,\ \ Thr_y = \mu_{|MV|,y} + \beta\sigma_{|MV|,y} \quad (3)$$

A flowchart of the proposed method is given in Fig. 3. For each CTU, first the search range is decided using (2). Then an adaptive threshold is used to decide whether the motion in the region is homogenous, or complex. This threshold is calculated using (3), where the average motion intensity ($\mu_{|MV|,x}$) and the standard deviation of the motion intensity ($\sigma_{|MV|,x}$) from the previously coded frames, are used to calculate the threshold in each direction. Also β (β=1 in this paper) is a parameter to regulate the trade-off between coding efficiency and encoding performance. If the obtained search ranges in both horizontal and vertical directions are smaller than the thresholds, the motion is considered homogenous. In this case, (1) is used to set the starting point for search. Then a search window with size of (SR_x, SR_y), around the CTU-sized block, located at the (SP_x,SP_y), is fetched into the encoder. The rest of the ME for every sub-block is followed as the second step of the TZ search.

Fig. 2 (b) visualizes this process where three MVs from neighboring blocks (Bold dots in the figure) are used to decide the starting point and the search range. However, if the search range in at least one direction is larger than the threshold, it indicates a complex motion region. In this case, information from the neighboring blocks is not enough for predicting the location of the motion, and requires a more intensive refinement steps and extra complexity. Thus, the encoder fetches the original search range and starts the baseline TZ search by testing the initial predictors. However, our

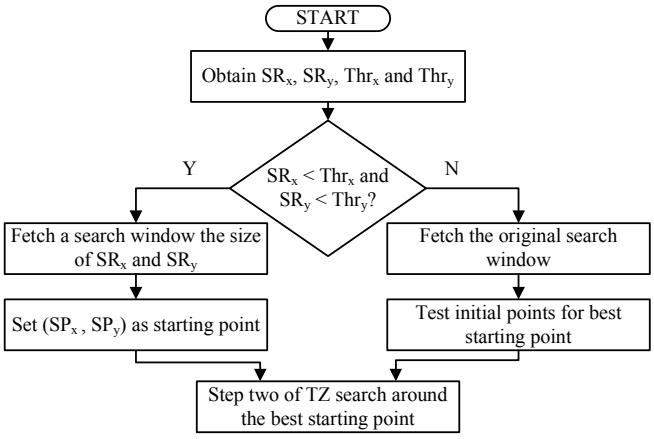


Fig. 3. The proposed fast ME algorithm with memory access reduction

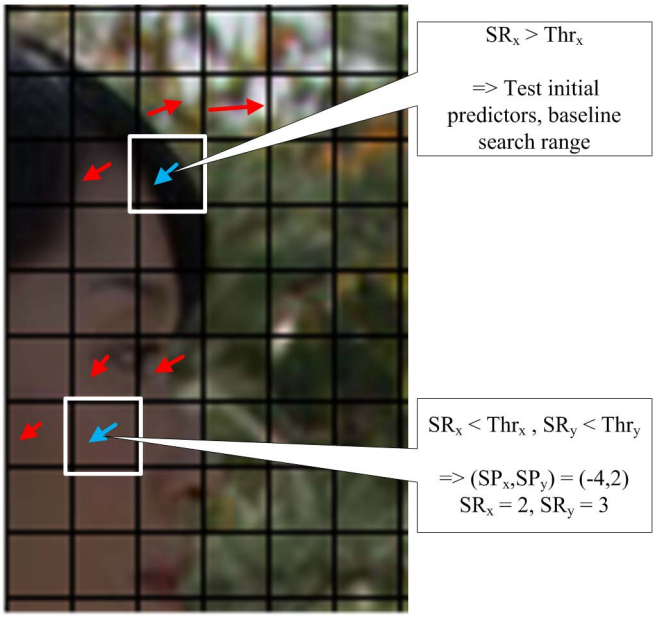


Fig. 4. A visual example of the proposed algorithm, on two different blocks of Kimono1

observations show that this condition occurs only for ~10-18% of CTUs.

A visual example of this algorithm can be found in Fig. 4. Red arrows in this figure represent the MVs for neighboring blocks of two certain blocks in Kimono1 video sequence. It can be observed that for the upper instance, the neighboring blocks contain different motion directions due to the person's motion and camera panning. Consequently the obtained search range exceeds the Thr_x, meaning that this block goes through the baseline ME steps. The neighboring blocks of the lower instance though suggest uniform motion region. Hence, this block will benefit from the proposed search range and starting point.

V. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed method, it was implemented on top of the HM reference software [2] and tested with Low-Delay P (LD) configuration as suggested in the common test conditions [20]. All the tests were repeated for QP values of 22, 27, 32, and 37, and the Bjontegaard Delta Rate (BD-Rate) and BD-PSNR were used to evaluate bitrate and quality [21]. The platform for tests includes an Intel Core i7-930 processor with 8 GBs of memory and Windows 8.1 as the operating system. Table I introduces the seven different test sequences which were used to evaluate the performance of our method for different motion characteristics.

TABLE I. TEST SEQUENCES FOR EXPERIMENTAL RESULTS

Abr	Class	Test Sequence	Resolution
TS1	A	PeopleOnStreet	2560×1600
TS2	B	Kimono1	1920×1080
TS3	B	ParkScene	1920×1080
TS4	E	KristenAndSara	1280×720
TS5	E	Johnny	1280×720
TS6	C	BQMall	832×480
TS7	D	BasketballPass	416×240

TABLE II. EXPERIMENTAL RESULTS COMPARED TO THE BASELINE TZ SEARCH [2]. IMET DENOTES THE IME TIME SAVING AND MAR DENOTES MEMORY ACCESS REDUCTION.

Test Sequence	BD-Rate (%)	BD-PSNR (db)	IMET(%)	MA (%)
TS1	2.07	-0.09	-67.5	-74.26
TS2	0.6	-0.02	-48.66	-56.07
TS3	1.08	-0.03	-69.93	-77.71
TS4	0.72	-0.01	-78.89	-87.66
TS5	-0.01	0	-79.56	-88.4
TS6	0.8	-0.02	-73.44	-83.49
TS7	2.41	-0.08	-69.69	-77.16
AVG	1.1	-0.04	-69.67	-77.82

The experimental results in Table II, compares the performance of our method with the baseline TZ search, where IMET denotes the percentage of IME time saving, and MA denotes the percentage of reduction in access to the off-chip memory, which was collected through software simulations. As the table shows, the proposed method can reduce the time of TZ search by ~70% on average, while the BD-Rate increases only by 1.1% compared to the baseline TZ search. As suggested by the algorithm, this time saving depends on the motion activity. For instance, Kimono1 has a complicated motion activity, thus many blocks need to go through the baseline TZ search process. On the contrast, Johnny has a very simple motion in most of the image area. Consequently the IMET for Kimono is comparatively smaller and for Johnny is larger.

Also based on Table II, this technique leads to an average of ~78% access reduction to the off-chip memory, which leads to considerable reduction in power consumption. As explained earlier, this access reduction is due to simultaneously reducing of search range and having a single starting point approach.

In Table III, we compare the results of our method with the ASR method in [11]. The proposed method gains ~20% more time saving compared to the ASR method, however the ASR method achieves 0.38% better BD-Rate. This is because the ASR method only reduces the search range and benefits from all the six starting points of the TZ search.

From the off-chip memory access point of view, as discussed earlier, the proposed method gains a ~78% access

TABLE III. THE EXPERIMENTAL RESULTS COMPARED TO THE METHOD PROPOSED IN [11]

Test Sequence	BD-Rate (%)	BD-PSNR(db)	IMET(%)
TS1	1.01	-0.04	-9
TS2	0.28	-0.01	-10.48
TS3	0.49	-0.01	-23.6
TS4	0.11	0	-28.01
TS5	-0.01	-0.01	-29.14
AVG	0.38	-0.01	-20.05

reduction in hardware encoders, compared to the baseline TZ search. For the ASR method [11] and based on analysis provided in section III, this amount will be 0%, considering the nominal hardware encoders, and below ~20% for the case of ideal hardware design.

VI. CONCLUSION

In this paper, a fast ME algorithm for hardware implemented HEVC encoders is presented. The paper points out to the high memory access in conventional ME schemes and demonstrates that the search for the best starting point in TZ search highly limits the opportunity for memory access reduction. To alleviate this problem, the proposed method estimates the best starting search point earlier, in addition to an adaptive search range, based on MVs of the neighboring blocks. Experimental results show that this method can lead to ~70% reduction in IME time with negligible loss of coding efficiency. Moreover, since the proposed method provides a more targeted memory access, it gains ~78% reduction in memory transactions which significantly reduces the power consumption of hardware implemented video encoders.

REFERENCES

[1] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding," *Ieee Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[2] C. Rosewarne, B. Bross, M. Naccari, K. Sharman., and G. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Encoder Description," *JCTVC-V1002*, 2015.

[3] X. Lian, Z. Liu, W. Zhou, and Z. Duan, "Lossless Frame Memory Compression Using Pixel-Grain Prediction and Dynamic Order Entropy Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8215, no. c, pp. 1–1, 2015.

[4] S. Tsai, C. Li, H. Chen, P. Tsung, K. Chen, and L. Chen, "A 1062Mpixels/s 8192x4320p High Efficiency Video Coding (H.265) Encoder Chip," in *Symposium on VLSI Circuits*, 2013, pp. 4–5.

[5] N. Hu and E. Yang, "Fast Motion Estimation Based on Confidence Interval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1310–1322, 2014.

[6] S.-H. Yang, H.-J. Yang, and J.-Z. Jiang, "Fast motion estimation for HEVC with directional search," *Electron. Lett.*, vol. 50, no. 9, pp. 673–675, Apr. 2014.

[7] S. Radicke, J. Hahn, C. Grecos, and Q. Wang, "Highly-parallel HVEC motion estimation with CUDA," in *European Workshop on Visual Information Processing (EUVIP)*, 2013, pp. 148–153.

[8] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Trans. Broadcast.*, vol. 63, no. 3, pp. 547–561, 2017.

[9] S. Ahn, B. Lee, and M. Kim, "A novel fast CU encoding scheme based on spatiotemporal encoding parameters for HEVC inter coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 3, pp. 422–435, 2015.

[10] W. Dai, O. C. Au, S. Li, L. Sun, and R. Zou, "Adaptive search range algorithm based on Cauchy distribution," in *IEEE Visual Communications and Image Processing*, 2012, pp. 1–5.

[11] L. Du, Z. Liu, T. Ikenaga, and D. Wang, "Linear adaptive search range model for uni-prediction and motion analysis for bi-prediction in HEVC," in *2014 IEEE International Conference on Image Processing, ICIP 2014*, 2014, pp. 3671–3675.

[12] T. Lee, Y. Chan, and W. Siu, "Adaptive search range by neighbouring depth intensity weighted sum for HEVC texture coding," *Electron. Lett.*, pp. 3–4, 2016.

[13] S. Kim, D. K. Lee, C. B. Sohn, and S. J. Oh, "Fast motion estimation for HEVC with adaptive search range decision on CPU and GPU," in *2014 IEEE China Summit and International Conference on Signal and Information Processing, IEEE ChinaSIP 2014 - Proceedings*, 2014, pp. 349–353.

[14] S.-Y. Jou, S.-J. Chang, and T.-S. Chang, "Fast Motion Estimation Algorithm and Design for Real Time QFHD High Efficiency Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 9, pp. 1533–1544, 2015.

[15] D. Silveira, G. Povala, L. Amaral, B. Zatt, L. Agostini, and M. Porto, "Memory bandwidth reduction for H.264 and HEVC encoders using lossless reference frame coding," in *Proceedings - IEEE International Symposium on Circuits and Systems*, 2014, pp. 2624–2627.

[16] C. Y. Chen, C. T. Huang, Y. H. Chen, and L. G. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, 2006.

[17] J. C. Tuan, T. S. Chang, and C. W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, 2002.

[18] C.-C. Ju *et al.*, "A 0.5 nJ/Pixel 4 K H.265/HEVC Codec LSI for Multi-Format Smartphone Applications," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 56–67, Jan. 2016.

[19] M. E. Sinangil, A. P. Chandrakasan, V. Sze, and M. Zhou, "Memory cost vs. coding efficiency trade-offs for HEVC motion estimation engine," in *International Conference on Image Processing*, 2012, pp. 1533–1536.

[20] F. Bossen, "Common Test Conditions and Software Reference Configurations," *JCTVC-H1100*, 2012.

[21] G. Bjontegaard, "Calculation of Average PSNR Differences Between RD Curves," *13th Video Coding Expert. Gr. Meet.*, 2001.