Proceedings of the Eighth International Conference on
Parallel, Distributed, GPU and Cloud Computing for Engineering
Edited by: P. Iványi, J. Kruis and B.H.V. Topping
Civil-Comp Conferences, Volume 12, Paper 2.7
Civil-Comp Press, Edinburgh, United Kingdom, 2025
ISSN: 2753-3239, doi: 10.4203/ccc.12.2.7

Performance Constraints in IME-HEVC Software Integration

O. M. López-Granado[1], M. Martínez-Rach[1], H. Migallón[1], R. Gutierrez Mazon[2] and M. Perez Malumbres[1]

[1] Computer Engineering Department, Miguel Hernández University, Elche, Spain
[2] Communications Engineering Department, Miguel Hernández University, Elche, Spain

Abstract

High-Efficiency Video Coding (HEVC) improves compression efficiency, compared to H.264/AVC, but motion estimation remains a major computational bottleneck, especially for high-resolution video sequences. To accelerate encoding, we implemented a hardware-based motion estimation module, achieving significant speed-ups. However, integrating this hardware with software introduces constraints that may limit its impact on overall encoding performance. This paper analyzes these integration challenges, identifying key bottlenecks in the software/hardware encoding process. Based on this analysis, we propose a refined integration approach combining the hardware module with a slice-based parallel HEVC encoder. This optimized version achieves a significant speed-up of up to 146.90 times compared to the sequential HEVC encoder using full-search motion estimation.

Keywords: video codecs, HEVC, FPGA, IME, temporal prediction, Sum of Absolute Difference architecture.

1 Introduction

The Joint Collaborative Team on Video Coding (JCT-VC) developed the High Efficiency Video Coding (HEVC) standard [1] to replace H.264/AVC [2], improving compression efficiency by delivering the same video quality at half the bit rate [3]. However, HEVC encoding is significantly more complex than its predecessor [4], with motion estimation (ME) being the most computationally demanding task, accounting for about 90% of the total encoding time [5]. This complexity is due to the increased number of coding tree unit (CTU) partitioning modes and reference frames, making motion estimation a computationally intensive process that requires acceleration for practical applications.

To address this, both hardware and software acceleration approaches have been explored. Hardware accelerators focus on integer motion estimation (IME), often using the full search (FS) algorithm due to its structured processing patterns. Several works [5–12] propose FPGA and ASIC-based IME accelerators, achieving significant speed-ups for HD and UHD formats. Alternative fast ME algorithms, such as Diamond Search and TZSearch, have also been implemented in hardware [13] [14].

Software-based acceleration, such as slice-based parallel encoding [15–20], exploits spatial parallelism by encoding video frames in separate slices using multiple threads. Previous work [18] demonstrated speed-ups of up to 9.3× on a 12-core platform.

This paper integrates both hardware and software acceleration, deploying multiple FPGA-based IME units alongside slice-based parallel encoding. We evaluate its performance to determine the optimal configuration for HEVC acceleration.

2 Proposed Software-Hardware Video Acceletaror Architecture

This section describes the proposed HEVC integrated accelerator, detailing both hardware and software accelerators, their advantages, and limitations. We also present the integration framework using OpenCL [21], which allows the HEVC software to initialise the hardware platform (Xilinx FPGA) and call the hardware kernel for integer motion estimation (IME). Based on the performance analysis, we determine the best configuration for the integrated accelerator.

2.1 IME hardware module

HEVC introduces a quadtree-based partitioning structure called CTU [3], which significantly increases the complexity of motion estimation. To address this, we have designed an IME hardware module based on the Full Search (FS) algorithm, which

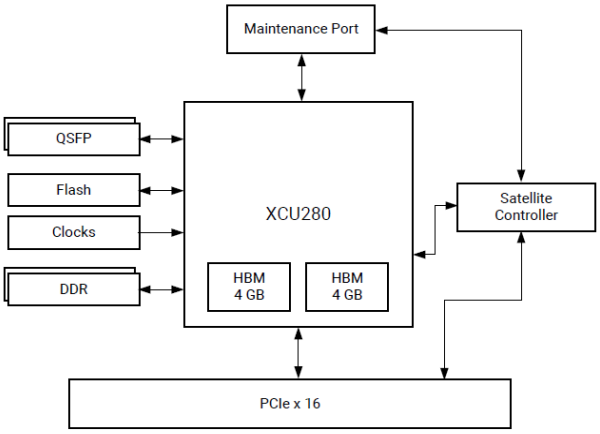

Figure 1: U280 block diagram.

evaluates all possible partitions to find the best rate-distortion (R/D) cost. The module includes memory areas for storing CTU and search area pixels, a sum of absolute differences (SAD) adder tree, and a comparator unit [12]. The IME module processes 64×64 CTUs with a 192×192 pixel search area.

The new proposal has been implemented on a Xilinx Alveo U280 FPGA (see Figure 1), and it supports up to four IME units working in parallel via independent memory banks and DMA channels. The board features 8 GB HBM2, 32 GB of DDR4, and a high bandwidth interconnect system. The Ultrascale+ XCU280 FPGA comprises three super logic regions (SLRs), with the bottom SLR (SLR0) containing a high bandwidth memory (HBM) controller to interface with the HBM2 subsystem via 32 pseudo-channels (PCs), each with direct access to 256 MB of memory (8 GB in total). There is a limitation imposed by the FPGA board design, which divides the total implemented silicon area into two different SLRs (see Figure 2), so, we have to map up to two HW units in SLR0, so the last two HW units should be placed in SLR1, although inter-SLR data transfer introduces minor performance penalties. In Table 1 we show the FPGA resources required by the implementation of four different Intellectual Property cores (IPs) that will map one, two, three, or four HW units.

Table 1: Resource utilization of implemented hardware unit.

Resources	1 HW Unit	2 HW Units	3 HW Units	4 HW Units
CLB LUT	175K(15%)	350K (30%)	525K (46%)	701K (61%)
CLB register	174K	350K	524K	699K
Block-RAM36	48 (2.7%)	96 (5.4%)	144 (8.1%)	192 (10.8%)
Freq. Max (MHz)	284	252	241	210

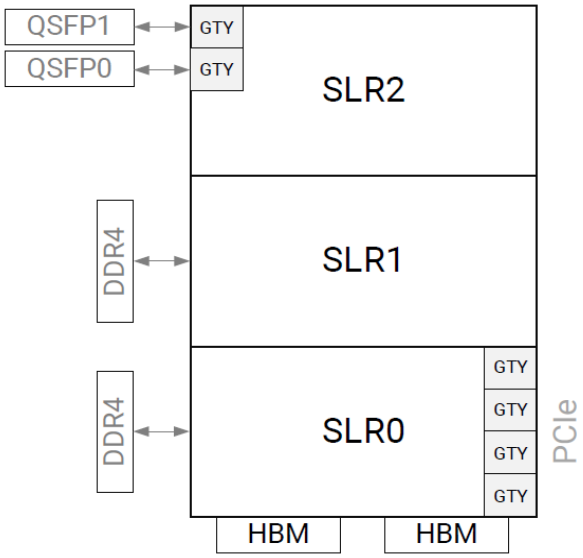

Figure 2: Floorplan of the XCU280 device.

2.2 Slice-based HEVC encoder

To take advantage of multi-core processing, in a previous work we developed a parallel slice-based HEVC version, where each frame is divided into independent slices containing consecutive CTUs. Each slice is encoded in parallel, reducing computation time. Our approach dynamically allocates slices based on available threads, ensuring load balancing, achieving speed-ups of up to 9.3× for 12 cores using this method [22].

2.3 OpenCL integration

We integrate the IME hardware with the slice-based HEVC encoder using OpenCL [21], allowing efficient CPU-FPGA communication via DMA. Each encoding thread offloads motion estimation to the FPGA's IME module, which processes CTUs and stores the results in a lookup table. The IME module supports up to four hardware units, allowing multiple parallel operations. Each unit includes input buffers for CTU and SA pixels, an output buffer for storing SADs and MVs, and communicates with the CPU via the Xilinx DMA (XDMA) subsystem over PCIe for high-speed data transfers.

To integrate the IME unit with OpenCL, we modified the HEVC encoder to handle initialization, data transfer, and execution. Each IME unit has dedicated input/output buffers, allowing multiple operations to run simultaneously. When only one IME unit is available, OpenCL serializes requests from multiple encoding threads, while four units enable full parallelization.

The slice-based HEVC encoder assigns one thread per slice, and the IME module processes as many operations as there are hardware units available.

3 Experimental Results

This section presents the experimental tests designed to evaluate the HW/SW integration. The tests use the HEVC Random Access (RA) coding mode, specially suited for broadcast, where the first frame is encoded as an intra (I) frame, and subsequent frames in each Group of Picture (GOP) are encoded as inter (B) frames using past and future references. The search area (SA) is fixed at 128×128 pixels, as specified in the HEVC standard, and the evaluations were performed on the HEVC common conditions video set. Testing was performed using the HEVC HM 16.3 reference software [23] on a dual Intel Xeon Gold 6230 system (2.10 GHz, 20 cores per CPU, 256 GB RAM) running CentOS Stream 8. The setup included gcc 8.5.0, OpenMP 4.5, and Xilinx Runtime 2.15.225, with the datacenter-based FPGA being a Xilinx A-U280-A32G-DEV-G.

To assess the performance impact of our integrated HW/SW accelerator, we analyse the effect of using multiple hardware units alongside varying numbers of software threads. Table 2 shows the encoding results for the first 200 frames of the Cactus video sequence under RA encoding mode with a Quantization Parameter (QP) value of 32, testing configurations with 1 to 4 HW units and 1 to 12 software threads/slices. The table shows (a) the number of hardware units and their operating frequency, (b) total HEVC encoding time, (c) average time for a single CTU_SAD calculation, (d) average time for an IME operation on a single SA location, (e) speed-up relative to single-threaded HEVC encoding (SW_SpUp), (f) speed-up relative to single HW unit encoding (HW_SpUp), and (g) speed-up relative to CTU_SAD time with one HW unit (ΔCTU_SAD).

The results show that all integrated configurations achieve strong SW acceleration (SW_SpUp), indicating that the slice-based parallel encoder performs consistently regardless of the number of HW units. However, increasing the number of HW units does not proportionally reduce encoding time (HW_SpUp), an unexpected result. Two key factors are likely to explain this: (a) hardware units are rarely used simultaneously, so parallelism is only fully exploited for the first CTU of each frame, and (b) due to FPGA board constraints, the first two HW units are in SLR0 (with direct HBM access), while the third and fourth are in SLR1, requiring extra cycles for memory access.

Table 2 also shows the average CTU_SAD time across different IME configurations, with ΔCTU_SAD showing minimal impact when using two HW units but significant increases when using three or four. This further supports the effect of FPGA memory access constraints when allocating HW units in SLR1.

Finally, we have performed a complete evaluation of the IME SW/HW integrated version by comparing it to the FS algorithm of the HEVC reference software with several video sequences of the HEVC common test conditions. In Table 3 we show the computing time, PSNR and bitrate of both the HEVC using FS ME and our integrated SW/HW parallel version using four threads/slices per frame and one HW IME module. As can be seen, speed-ups of up to 146.90x are obtained with a negligible PSNR loss

Table 2: Time profile using different Threads and HW Units (Cactus video sequence).

HW Unit	Threads	Time(s)	CTU_SAD(μs)	IME (μs)	SW_SpUp	HW_SpUp	ΔCTU_SAD
	1	3,379.5	344,035	0,108	1.00x	-	-
	2	1,773.0	359,766	0,136	1.91x	-	-
	4	962.8	388,388	0,185	3.51x	-	-
HW1 (249 MHz)	8	547.8	443,075	0,239	6.17x	-	-
	12	413.6	530,569	0,286	8.17x	-	-
	1	3,373.7	343,617	0,106	1.00x	-0.17%	-0.12%
	2	1,763.5	360,313	0,144	1.91x	-0.53%	0.15%
	4	961.3	384,845	0,187	3.51x	-0.15%	-0.91%
HW2 (247 MHz)	8	548.9	442,305	0,236	6.15x	0.20%	-0.17%
	12	414.1	532,898	0,282	8.15x	0.12%	0.44%
	1	3,411.0	412,166	0,109	1.00x	0.93%	19.80%
	2	1,778.2	437,193	0,148	1.92x	0.29%	21.52%
	4	972.7	471,405	0,183	3.51x	1.03%	21.37%
HW3 (192 MHz)	8	554.9	560,534	0,226	6.15x	1.29%	26.51%
	12	416.6	680,632	0,285	8.19x	0.73%	28.28%
	1	3,415.7	437,171	0,108	1.00x	1.07%	27.07%
	2	1,782.3	464,603	0,136	1.92x	0.53%	29.14%
	4	973.8	503,222	0,182	3.51x	1.15%	29.57%
HW4 (176 MHz)	8	554.8	604,964	0,232	6.16x	1.27%	36.54%
	12	419.7	741,479	0,277	8.14x	1.48%	39.75%

and an average bitrate increment of 6.1%.

Table 3: Evaluation of SW FS version and the HW/SW proposal with four threads/slices.

Video Sequence	SW/HW FS			SW FS					
	Total Time (s)	PSNR (dB)	Bitrate (Mb)	Total Time (s)	PSNR (dB)	Bitrate (Mb)	Speed-up	PSNR	Bitrate
NebutaFestival	4261.77	29.44	19.643	400876.0	29.43	19.442	94.06x	-0,02%	1.03%
Traffic	1342.69	36.52	1.632	183217.0	36.55	1.584	136.45x	0.09%	3.04%
BQTerrace	2778.43	33.81	3.687	365374.1	33.83	3.546	131.50x	0.06%	3.96%
BasketballDrive	2885.97	35.58	3.885	313520.6	35.63	3.583	108.64x	0.14%	8.42%
Cactus	2386.68	34.88	3.678	308749.8	34.94	3.579	129.36x	0.16%	2.78%
ParkScene	1159.24	34.84	1.836	147030.7	34.88	1.805	126.83x	0.11%	1.73%
BQSquare	164.64	31.92	0.480	16701.3	31.99	0.434	101.44x	0.19%	10.73%
BasketballPass	175.43	33.47	0.529	14737.6	33.56	0.471	84.01x	0.28%	12.21%
RaceHorses	113.48	32.19	0.380	9251.5	32.30	0.354	81.52x	0.34%	7.23%
Johnny	1087.28	39.56	0.477	159718.1	39.60	0.439	146.90x	0.11%	8.63%
FourPeople	1193.12	38.15	0.916	161274.3	38.20	0.868	135.17x	0.13%	5.49%

4 Conclusions

This work presents a fully integrated HW/SW version of the HEVC encoder, combining a hardware-based IME module with a slice-based parallel HEVC encoder. During the integration process, we identified several factors limiting overall performance,

including DMA transfer overhead, software-related inefficiencies from the OpenCL API, and the concurrent execution behaviour of the slice-based encoder.

Although the speed-up achieved is very significant, the designed platform has room for improvement, so our future work will focus on identifying the reasons why we cannot achieve even better speed increases and, if possible, develop new strategies to overcome these drawbacks.

Acknowledgements

This research was supported by Grant PID2021-123627OB-C55, funded by MCIN/ AEI/ 10.13039/ 501100011033 and by 'ERDF A way of making Europe'.

References

[1] B. Bross, W.J. Han, J.R. Ohm, G.J. Sullivan, Y-K Wang, and T. Wiegand. High Efficiency Video Coding (HEVC) Text Specification Draft 10. *Document JCTVC-L1003 of JCT-VC, Geneva*, January 2013.

[2] ITU-T and ISO/IEC JTC 1. Advanced Video Coding for Generic Audiovisual Services. *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16, 2012*, 2012.

[3] G.J. Sullivan, J.R. Ohm, W.J. Han, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. *Circuits and systems for Video Technology, IEEE Transactions on*, 22(12):1648 –1667, December 2012.

[4] F. Bossen, B. Bross, K. Suhring, and D. Flynn. HEVC Complexity and Implementation Analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1685–1696, 2012.

[5] Ahmed Medhat, Ahmed Shalaby, Mohammed S Sayed, and Maha Elsabrouty. A highly parallel sad architecture for motion estimation in hevc encoder. In *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'14)*, pages 280–283, Ishigaki, November 2014.

[6] J. Byun, Y. Jung, and J. Kim. Design of integer motion estimator of HEVC for asymmetric motion-partitioning mode and 4K-UHD. *Electronics Letters*, 49(18):1142–1143, 2013.

[7] Xu Yuan, Liu Jinsong, Gong Liwei, Zhang Zhi, and Robert K.F. Teng. A high performance VLSI architecture for integer motion estimation in HEVC. In *IEEE 10th International Conference on ASIC (ASICON'13)*, pages 1–4, Shenzhen, October 2013.

[8] Thomas D'huys. Reconfigurable data flow engine for HEVC motion estimation. In *IEEE International Conference on Image Processing (ICIP'14)*, pages 1223–1227, Paris, October 2014.

[9] Purnachand Nalluri, Luis Nero Alves, and Antonio Navarro. High speed SAD architectures for variable block size motion estimation in HEVC video coding. In *IEEE International Conference on Image Processing (ICIP'14)*, pages 1233–1237, Paris, October 2014.

[10] Belal Mohamed, Ahmed Shalaby, and Mohammed S Sayed. High-level synthesis hardware accelerators of integer-pixel motion estimation of HEVC on SoC FPGA platform. In *2nd Europe - Middle East - North African Regional Conference of the International Telecommunications Society (ITS): Leveraging Technologies For Growth*, February 2019.

[11] S. Gogoi and R. A Peesapati. A hybrid hardware oriented motion estimation algorithm for HEVC/H.265. *Journal of Real Time Image Processing*, 18, January 2021.

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

[13] Randa Khemiri, Hassan Kibeya, Hassen Loukil, Fatma Ezahra Sayadi, Mohamed Atri, and Nouri Masmoudi. Real-time motion estimation diamond search algorithm for the new high efficiency video coding on FPGA. *Analog Integrated Circuits and Signal Processing*, 94(2):259–276, Aug 2018.

[14] R. Haddar, A. Chaari, H. Kibeya, M. A. Ben Ayed, and N. Masmoudi. Fpga-based implementation of tzsearch algorithm for h.265/hevc standard. In *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 605–610, Dec 2017.

[15] S. Radicke, J. Hahn, C. Grecos, and Qi Wang. A multi-threaded full-feature hevc encoder based on wavefront parallel processing. In *2014 International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, pages 90–98, Aug 2014.

[16] E. Ryu, J. Nam, S. Lee, H. Jo, and D.Sim. Sample adaptive offset parallelism in hevc. *Multimedia and Ubiquitous Engineering. Lecture Notes in Electrical Engineering*, 240, 2013.

[17] H. Migallón, V. Galiano, P. Piñol, O. López-Granado, and M. P. Malumbres. Distributed Memory Parallel Approaches for HEVC Encoder. *The Journal of Supercomputing*, pages 1–12, 2016.

[18] P. Piñol, H. Migallón, O. López-Granado, and M.P. Malumbres. Slice-based parallel approach for hevc encoder. *J Supercomputing*, 71:1882–1892, 2015.

[19] Héctor Migallón, Otoniel López-Granado, Vicente Galiano, Pablo Piñol, and Manuel P. Malumbres. *Shared Memory Tile-Based vs Hybrid Memory GOP-Based Parallel Algorithms for HEVC Encoder*, pages 521–528. Springer International Publishing, Cham, 2016.

[20] Gabriel Cebrián-Márquez, José Luis Martínez, and Pedro Cuenca. Inter and intra pre-analysis algorithm for HEVC. *Journal of Supercomputing*, 73:414–432, 2019.

[21] Rjoji Tsuchiyama, Takashi Nakamura, Takuro Iizuka, Akihiro Asahara, and Satoshi Miki. *The OpenCL Programming Book*. Fixstars Corporation, 2009.

[22] P. Piñol, H. Migallón, O. López-Granado, and M. P. Malumbres. Slice-based parallel approach for hevc encoder. *The Journal of Supercomputing*, 71(5):1882–1892, May 2015.

[23] Fraunhofer-HHI. HEVC Reference Software (HM-16.3), 2015. available at: `http://hevc.hhi.fraunhofer.de/svn/ svn\_HEVCSoftware/ tags/HM-16.3/`.