Evaluation of HEVC hardware IME using a SoC Platform

Author1 and
Author2 and
Author3
University
Department

Email: xxx@yyy.es

author4
University
Department

Email: xxx@yyy.es

***Abstract*—In the HEVC standard, motion estimation is one of the most complex task of the video encoder, requiring a great percentage of the encoding time mainly due to (a) a large set of Coding Tree Unit partitioning modes, (b) the presence of multiple reference frames, and (c) the varying size of Coding Units in comparison with its predecessor H264/AVC. In addition, HEVC adopts Variable Block Size Motion Estimation to obtain advanced coding efficiency.**

In this work, we evaluate a hardware IME design when applied to an System-On-Chip platform. In this evaluation we will measure the impact in the Rate/Distortion performance of applying different CTU sizes and reference search areas. Furthermore, we will evaluate the effect of the DMA transfers required in the computational performance. This architecture has been synthesized and implemented on the Xilinx SoC, Zynq-7 Mini-ITX Motherboard XC7Z100 (xc7z100ffg900-2). Our hardware IME architecture can be configured to work with different CTU sizes and reference search ranges. The results show there are negligible differences in Rate/Distortion between different configurations, but there is a great impact in the encoder complexity as both the CTU size and search range increase.

We have evaluated our hardware IME design using different CTU size configurations, search area sizes and DMA burst sizes in order to determine the maximum speed gain respect to the HEVC reference software. Results show that the encoding time could be reduced 588 times. Besides, this evaluation shows that the DMA transfer is the bottleneck of the system.

I. Introduction

The High Efficiency Video Coding (HEVC) standard [1] has been launched on January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC). This new standard replaces the current H.264/AVC [2] standard in order to deal with nowadays and future multimedia market trends like 4K and 8K definition video content and high quality color depth at 10 bit. HEVC greatly improved the coding efficiency over its predecessor (H.264/AVC) by a factor of almost twice while maintaining an equivalent visual quality [3].

Regarding complexity, HEVC decoder does not appear to be significantly different from the H.264/AVC one [4]. However, HEVC encoder is expected to be several times more complex than H.264/AVC encoder [5] and will be a hot research topic in years to come. e.g., encoding one second of a 1080p60 HD (High Definition) video with the reference software encoder can take longer than one hour running in an off-the-shelf desktop computer.

As in previous video standards, Motion Estimation (ME) is by far the most complex task of encoder, requiring more than 90% of the encoding time [6]. For HEVC standard, the complexity is even more critical due to several issues such as (a) a large set of Coding Tree Unit (CTU) partitioning modes, (b) the presence of multiple reference frames, and (c) the varying size of Coding Units (CU) in comparison with its predecessor H264/AVC. In addition, HEVC adopts Variable Block Size Motion Estimation (VBSME) to obtain advanced coding efficiency, which comes at the expense of a huge increase of computational complexity.

Several hardware architectures to speed up the HEVC ME module have been proposed with the aim to reduce the overall encoder complexity as much as possible. The Integer-pel Motion Estimation (IME) block is in charge of motion estimation. In most of the state-of-the-art proposals, the IME hardware architectures are only focused on the motion search algorithm since it takes most of the computational time of the IME block. Generally, the most popular motion search algorithm in hardware implementations is the Full Search (FS) algorithm. It follows greedy behavior by searching at all points of the established search area of a reference frame, and, as a consequence, it is able to provide an optimal result (i.e., a motion vector that minimizes the residual error of the actual CTU).

The architecture proposals in [6]–[10] present an IME hardware block using FS strategy. In [6], a Sum of Absolute Differences (SAD) unit on a Field-Programmable Gate Array (FPGA) is proposed that is able to test all partition modes of a CTU except the set of asymmetric partition modes. Authors fixed a search area size lower than the one established by the HEVC standard, being able to run as fast as 30 fps at 2k video resolutions. In [8], the maximum CTU size is reduced to 32x32 with a search area size of ± 23 pixels. This architecture is implemented on an FPGA device and achieves 30 fps at 1080p video resolutions. Different configurable search areas are studied in [9], achieving a maximum frame rate of 57 fps for a 720p video resolution.

In addition, in [11] and [12], different implementations of suboptimal motion search strategies called fast ME algorithms, such as new Diamond Search (DS) or new Three Step Search (TSS), are shown. Similar hardware ME architectures have also been studied for the previous H264/AVC standard in [13]–[17], which are of interest for our work due to the high similarity of the IME block architecture in both standards.

In a previous work [**?**], authors presented a new hardware architecture that performs IME computation using FPGA technology. they presented two innovative techniques: (a) a new SAD adder tree structure, and (b) a new memory scan order; achieving encoding frame rates up to 116 fps and 30 fps at 2K and 4K video formats, respectively. The SAD adder tree structure performs the additions at the first level of the tree, starting from the maximum size of the CTU, and halving the amount of additions at the next tree levels. This approach is different from the rest of state-of-the-art works, which divide a CTU into smaller blocks for consecutive accumulations, keeping the same additions in each step and thus requiring a higher number of steps to acquire all SADs. With that proposal, authors took advantage of the resources provided by the FPGA, obtaining the minimum possible latency when calculating SADs of all levels and partitions for a CTU. In this way, SADs corresponding to asymmetric partitions are obtained in a fast and efficient way. Regarding the new memory scan order, a series of reconfigurable shift registers and processing elements are responsible for storing the necessary pixels of both reference and current frames, keeping them always available for calculating the SADs and Motion Vectors (MVs) of a CTU, avoiding external memory accesses since available data are highly reused by reconfiguring the displacement in a more efficient way.

In this work, we evaluate the IME design presented in [**?**] when applied to an evaluation board. In that evaluation we will measure the impact in the Rate/Distortion (R/D) performance of applying different CTU sizes and search areas. Furthermore, we will evaluate the effect of the DMA transfers required in the computational performance.

The rest of the paper is organized as follows. Section II presents a brief overview of the architecture design while in Section III, numerical software and hardware experiments analyzing the results of our hardware design over an evaluation board are presented. Finally, in Section IV some conclusions and future work are drawn.

II. Hardware Architecture Description

In this section, we present a brief overview of a complete IME design in a System-On-Chip (SoC) platform using the SAD HEVC module proposed in [**?**]. SoC consist of two well-defined parts, a Processing System (PS) based on an ARM processor and several hard peripherals like Ethernet, USB, etc., and a Programmable Logic (PL) been made up of a FPGA. Our architecture has been modeled in VHDL, and it has been synthesized, simulated, implemented, and tested on the Xilinx SoC, Zynq-7 Mini-ITX Motherboard XC7Z100 (xc7z100ffg900-2). The correctness of our design was tested and verified with the HEVC HM 14 reference model [18].

In the proposed architecture, ARM processor manages the transfer between the IME SAD module and a Double Data Rate (DDR) memory which stores both reference and current frames, by a Direct Memory Access (DMA) module. The system is shown in Figure 1.

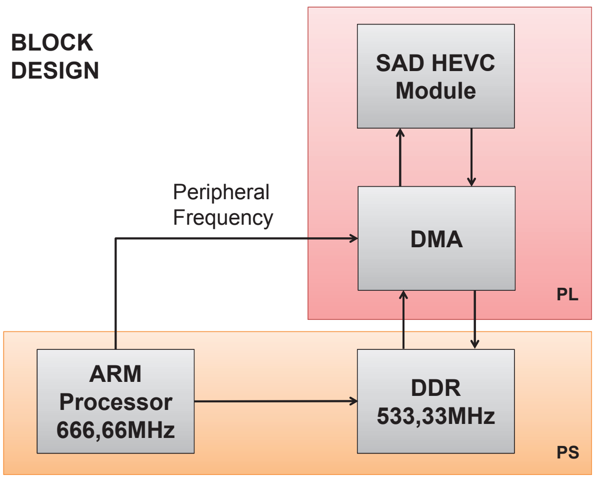


Fig. 1. Top-level hardware architecture

ARM processor works at 666.66 MHz, and the DDR at 533.33 MHz, whereas the clock frequency of the PL is restricted by the maximum frequency of the SAD HEVC module which is the responsible for the IME calculation.

Regarding the IME process, each video frame is subdivided and partitioned into basic coding units called CUs. The coding structure in HEVC consists of CUs with a maximum size of 64x64 pixels, as large as that of CTUs (Coding Tree Units), which can be recursively divided in picture squares until achieving a block size of 8x8 pixels. Each coding unit (CU) consists of Prediction Units (PUs) whose size can vary from the maximum size of the CU to 4x8 or 8x4 for Inter prediction, supporting 8 partitioning modes. In our proposal, the SAD HEVC module responsible for IME calculation can be configured to work with CTU sizes of 64x64 and 32x32. In the case of 64x64 CTU size, the PL can work at 220 MHz whereas with a 32x32 CTU size the PL clock frequency is fixed as maximum in the evaluation board used, 250 MHz. However, the module could work at a maximum frequency of 333 MHz. Therefore the maximum frequency is limited by the maximum PL clock frequency, which for the evaluation platform used is 250MHz.

Our SAD HEVC module consists of (a) internal memory areas to allocate pixels of the CU of current frame and the pixels belonging to the search area in the reference frame, (b) a distortion block where pixels belonging to both frames are subtracted, (c) a Sum of Absolute Difference (SAD) adder tree block, and (e) an accumulative comparator block that saves the minimum SAD values and its corresponding MVs for all CU partitions, as shown in Figure 2. For more details of this module, see authors' previous work [**?**].

In Table I, we show the resources used to implement our SAD HEVC module for maximum CTU sizes of 64x64 and 32x32, respectively, on a Zynq-7 Mini-ITX Motherboard XC7Z100 FPGA. As shown, our SAD HEVC module requires a 63% and 16% of the total used area for 64x64 and 32x32 CTU size, respectively.

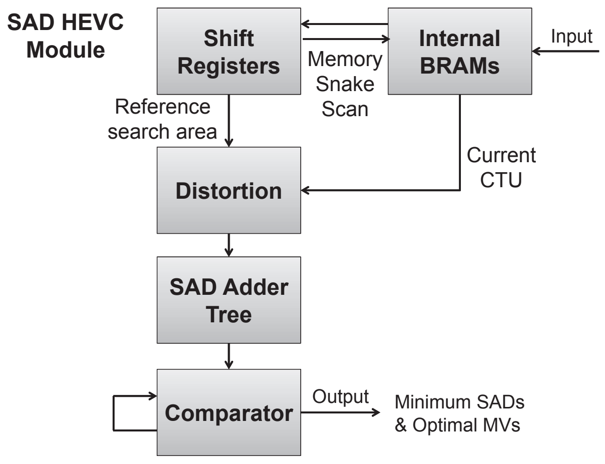


Fig. 2. Hardware SAD HEVC module

TABLE I. UTILIZATION RESOURCES ON MINI-ITX

Resources	CTU 64x64	CTU 32x32
LUTs	166383	40911
Flip-flops	190159	50028
Block-RAMs	32	16

III. NUMERICAL EXPERIMENTS

ME is a task integrated in the HEVC Inter Prediction. For this reason, we have considered the Low Delay B (LB) coding structure as configuration mode. In LB, the first picture is encoded as an intra-picture. Other pictures are encoded as generalized bidirectional pictures (inter). This coding structure is the most popular for video conferencing and streaming, designed for interactive real-time communication.

Given the previous configuration, we have performed several experiments of the HEVC IME in order to observe how both parameters, the CTU size and search area size impact on the R/D performance and coding complexity of the HEVC encoder. We have chosen CTU sizes of 64x64 and 32x32, and their corresponding Search Range (SR) sizes as 100% of CTU, 80% of CTU, and 50% of CTU. In addition, two video sequences from the HEVC common conditions video set were selected: ParkScene at 1920x1080 resolution (24 fps) and Traffic 2560x1600 (30 fps). To perform these tests, we have used the HEVC HM 14 reference model [18]. The HEVC reference software was compiled with Visual Studio 2010 and run over a PC platform with an Intel Core i7-3770 CPU 3.40GHz with 8GB RAM.

We have measured the time of the IME software module using a FS algorithm when a video sequence is encoded. We have focused our work towards the design and hardware implementation of an FS algorithm that is able to significantly speed up the motion estimation process of the HEVC encoder without losing R/D performance, since the most accurate strategy is the FS algorithm which exhaustively searches motion for all PUs at every single point of the established search area. Therefore, due to computational regularity and excellent video quality, FS motion estimation is commonly employed in hardware implementations [19].

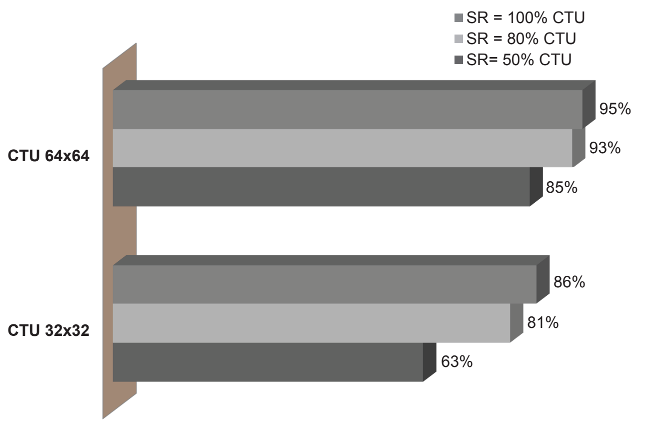


Fig. 3. Percentage of SW reference encoding time required for SAD module with a Full-Search strategy.

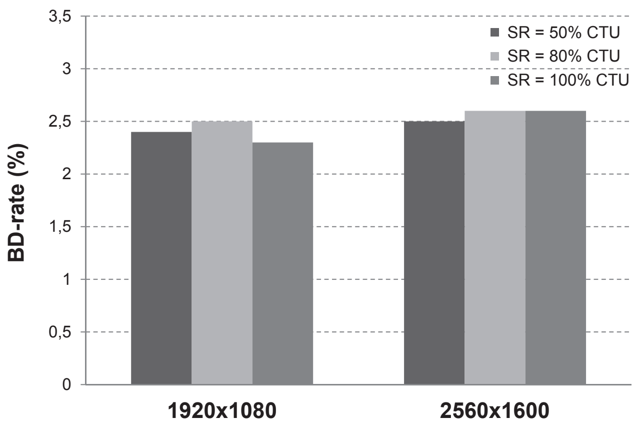
In Figure 3, we show the percentage of IME software time spent by encoder using a FS algorithm, without the R/D optimization procedure. This percentage of time is similar for all video sequences tested. As can be seen, this time depends on the CTU size, and the SR size, as shown in Figure 3. Generally, the encoder spends more time in the IME module when both the CTU size and SR are higher, as expected. The time spent by HEVC encoder to perform the ME ranges between 63% to 95% of the time required to encode the whole video sequence. Therefore, a hardware design performing the IME computation in a fast and accurate way makes sense in order to reduce both the IME complexity and the overall encoder time as much as possible.

In addition, we have analyzed the impact of the previous parameters on the R/D performance using the Bjontegaard metric calculation (BD-rate). Bjontegaard's metric allows computing the average per cent saving in bitrate between two rate-distortion curves. The R/D curves have been obtained for the compression levels (QP values): 22, 27, 32, and 37, taking into account a reference curve with typical configuration given by the HEVC software model, HM-14, with a CTU size of 64x64 and a search range of 64 (128x128 search area size).

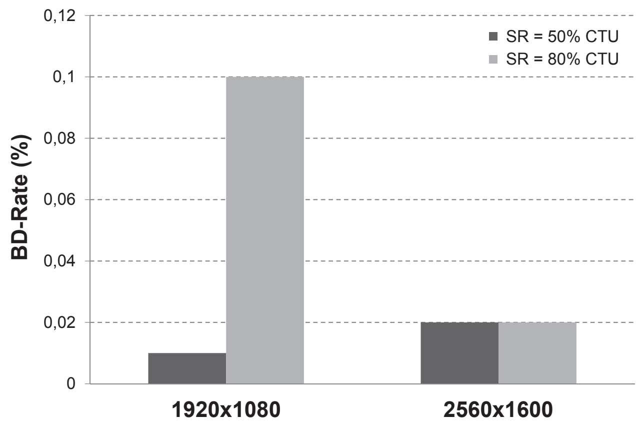
In Figure 4, we can observe the percentage of BD-rate obtained with different CTU and SR sizes for 1920x1080 and 2560x1600 video resolutions. As can be seen, there are slight differences between the SR sizes for a given CTU size, specially for a CTU size of 64x64 where the difference is around 0.1% as a maximum (see Figure 4(b)). Anyway, differences between all configurations set are negligible, being the maximum around 2.7% of the bitrate increased for a given PSNR, when the CTU size is 32x32 (see Figure 4(a)). Although BD-rate differences may depend on the video content, similar results were obtained with other video sequences.

Therefore, in order to reduce the complexity, allowing faster versions with reduced consumption, the CTU size and the SR could be reduced as much as possible, due to the slight impact of those parameters in the previous BD-rate result.

Regarding the hardware IME proposal, we have measured the CUs per second which are able to be processed depending on CTU size, SR, and the DMA burst size. In our complete IME design described in Section II, the DMA data width is

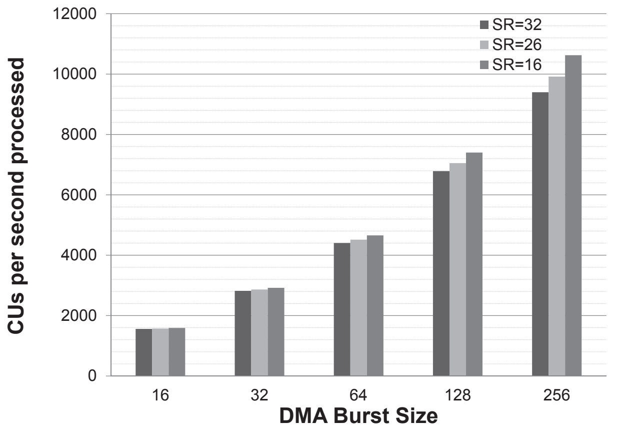


(a) BD-Rate of RD curves with a CTU size of 32 and several SR sizes

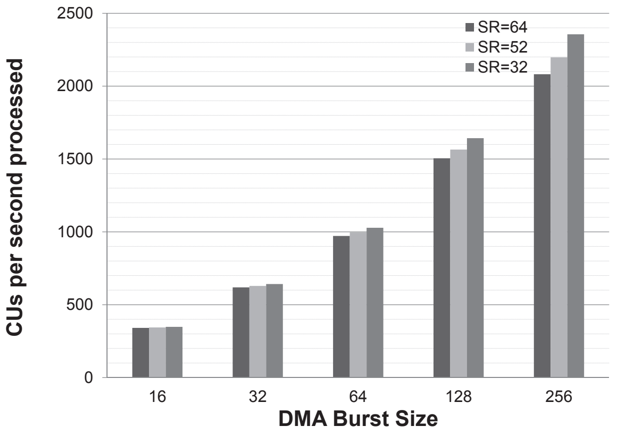


(b) BD-Rate of RD curves with a CTU size of 64 and several SR sizes

Fig. 4. Percentage of BD-Rate respect to a reference RD curve with CTU and SR sizes of 64



(a) CUs per second processed for a CTU size of 32



(b) CUs per second processed for a CTU size of 64

Fig. 5. CUs per second processed for each DMA burst size with our proposed SAD HEVC module

defined as 32 bits and the burst size can be configured from 16 to 256. For example, in the case of a CTU size of 32x32 pixels where the operation frequency is 250 MHz and with a DMA burst size of 256, the proposed system can transfer with a rate of 2Mbps.

In Figure 5 we show the number of CUs per second processed for each DMA burst size. Figures 5(a) and 5(b) show how the SR size impact in the CUs per second when the CTU size is 32x32 and 64x64, respectively. As can be seen, the number of CUs per second processed increases as the DMA burst size does, being 10626 and 2356 the maximum number of processed CUs per second for CTU sizes of 32x32 and 64x64, respectively. That increment is exponential because the time required for the DMA initialization is constant and independent on the DMA burst size. Therefore, the maximum burst size of 256 and the CTU size of 32x32 and a SR size of 16 is the most accurate configuration in order to achieve the maximum CUs per second using our hardware system.

However, the number of CUs per second achieved previously depend on the processing time of our entire hardware system, which consist of the DMA transfers and the SAD HEVC module. So, in Figure 6 we show DMA time, SAD HEVC module time, and total time to process a CU for each DMA burst size when using a configuration set of 32x32 CTU size and a SR of 50% CTU (±16). As expected, the maximum DMA burst size provides the best results, requiring the least time to process a CU. Furthermore, depending on the DMA burst size chosen, the differences between DMA transfers and SAD HEVC module processing time varies, being the SAD HEVC module 21x faster for a burst size of 256 and 146x for a burst size of 16.

Therefore, the bottleneck of total hardware time is the DMA transfer, as can be also seen in Figure 7. Having chosen a burst size of 256, the DMA transfer represents the 95% of the entire processing time of a CU, whereas the SAD HEVC module only requires the 5% of time. This bottleneck can be overcome in future works, for instance, transferring only the parts of the reference frames which differ from the ones already stored in internal BRAMS. In this way, the DMA transfers will be reduced and consequently, the total time required.

After performing the whole analysis, it can be determined that the HW configuration which better adapts to the application requirements (low power consumption, encoding time, compressed video quality) is the one that established a CTU size of 32x32, a SR of 16, and a DMA burst size of 256.

Although the Full Search algorithm is the most widely used in hardware ME implementations, the HEVC reference software uses by default the Diamond Search algorithm. So,

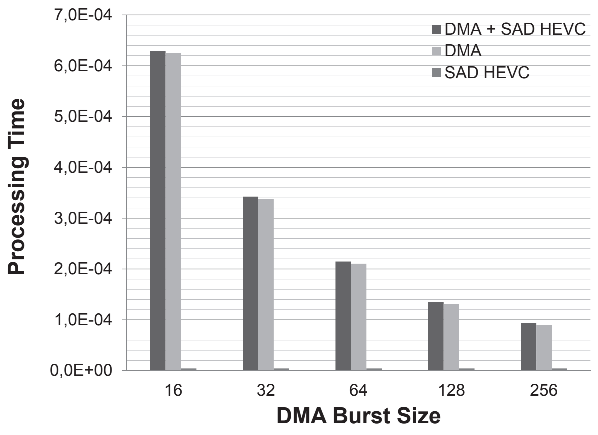


Fig. 6. Hardware Processing Time for different DMA burst sizes

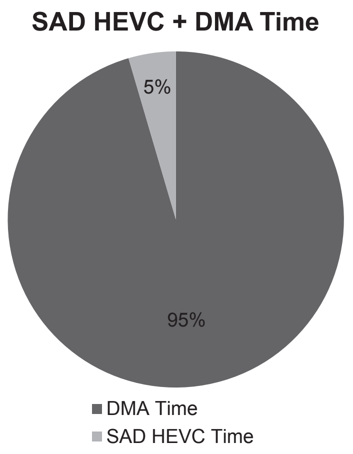


Fig. 7. Percentage of both DMA time and SAD module time for a burst size of 256.

in Figure 8 we show the gain achieved using our IME HW proposal with the previous configuration set in relation to the IME SW using both FS and DS strategies. So, looking at the information provided in Figure 8, for a 2560x1600 video resolution, the inclusion of our IME hardware module will speed up the IME computation 55 times and 588 times for diamond-like search algorithm and full-like search algorithm, respectively.

IV. Conclusion

In this work, we have presented a full hardware IME architecture which also includes the DMA. We have analyzed how the CTU size and the search range area in the IME module impact on the HEVC performance in terms of Rate/Distortion and processing time. As shown, differences in R/D are negligible for all configurations, being 2.7% the maximum BD-rate increment for a CTU size of 32x32 and 0.1% for a CTU size of 64x64. Regarding complexity, both the CTU size and the SR impact over the total encoding time, being the faster configuration a CTU size of 32x32 and a SR of 50% of the CTU size. Remark, that the ME module requires between 63% to 95% of the total encoding time.

Respect to the evaluation of the hardware IME proposal,

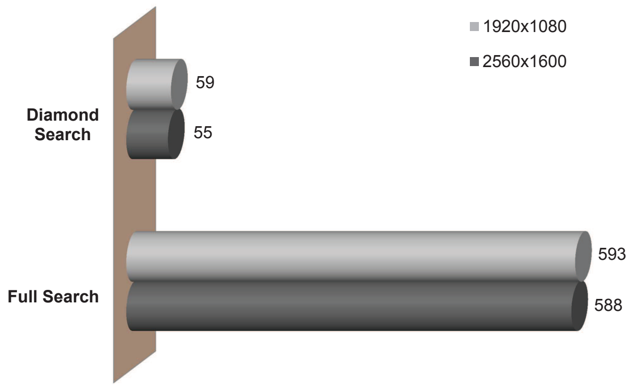


Fig. 8. HW gain (x times) facing SW Full Search and SW Diamond Search strategies

we have measured the maximum CUs per second that can be processed as a function of the search range, the CTU size and the DMA burst size. The results show that the number of CUs processed increases as the burst size does, being 10626 the maximum number of CUs per second processed for a CTU size of 32x32, a SR of 50% of the CTU size and a DMA burst size of 256. Looking at the results, we can assess, that the bottleneck of the system resides in the DMA transfer, requiring DMA transfer the 95% of the total processing time.

So, taking into account both the R/D and the encoding complexity, the best configuration parameters set for our hardware IME architecture is the one conformed by a CTU size of 32x32, a SR of 50% of the CTU size (± 16) and a DMA burst size of 256. To sum up, with the inclusion of our hardware IME proposal, we can speed-up the encoding time 558 times.

As a future work, we intend to reduce the DMA transfer time, reusing part of the reference frame of the search area for the IME calculation in contiguous CUs, only transferring the new reference pixels required at every moment. Furthermore, as we only use a 16% of the total board resources, we could add more SAD HEVC modules to our architecture so as to speed-up even more the ME computation.

Acknowledgment

This research was supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2015-66972-C5-4-R co-financed by FEDER funds.

References

[1] B. Bross, W. Han, J. Ohm, G. 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. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” *Circuits and systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1648 – 1667, December 2012.

[4] J. Ohm, G. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, “Comparison of the coding efficiency of video coding standards - including high efficiency video coding (hevc),” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1669–1684, 2012.

[5] F. Bossen, B. Bross, K. Suhring, and D. Flynn, “HEVC complexity and implementation analysis,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1685–1696, 2012.

[6] A. Medhat, A. Shalaby, M. S. Sayed, and M. Elsabrouty, “A highly parallel sad architecture for motion estimation in hevc encoder,” in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS’14)*, Ishigaki, Nov. 2014, pp. 280–283.

[7] J. Byun, Y. Jung, and J. Kim, “Design of integer motion estimator of hevc for asymmetric motion-partitioning mode and 4k-uhd,” *Electronics Letters*, vol. 49, no. 18, pp. 1142–1143, 2013.

[8] X. Yuan, L. Jinsong, G. Liwei, Z. Zhi, and R. K. Teng, “A high performance vlsi architecture for integer motion estimation in hevc,” in *IEEE 10th International Conference on ASIC (ASICON’13)*, Shenzhen, Oct. 2013, pp. 1–4.

[9] T. D’huys, “Reconfigurable data flow engine for hevc motion estimation,” in *IEEE International Conference on Image Processing (ICIP’14)*, Paris, Oct. 2014, pp. 1223–1227.

[10] A. N. Purnachand Nalluri, Luis Nero Alves, “High speed sad architectures for variable block size motion estimation in hevc video coding,” in *IEEE International Conference on Image Processing (ICIP’14)*, Paris, Oct. 2014, pp. 1233–1237.

[11] V. VG, D. Yagain, and G. R. K, “Motion estimation block for hevc encoder on fpga,” in *IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE’14)*, Jaipur, India, May 2014, pp. 1–5.

[12] P. Davis and M. Sangeetha, “Implementation of motion estimation algorithm for h.265/hevc,” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, no. 3, pp. 122–126, 2014.

[13] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, “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, pp. 578–593, 2006.

[14] W. Elhamzi, J. Dubois, and J. Miteran, “An efficient low-cost fpga implementation of a configurable motion estiation for h.264 video coding,” *Springer Journal of Real-Time Processing*, vol. 9, no. 1, pp. 19–30, 2014.

[15] T. Moorthy and A. Ye, “A scalable architecture for variable block size motion estimation on field-programmable gate arrays,” in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE’08)*, Niagara Falls, May 2008, pp. 1303–1308.

[16] M. Kthiri, P. Kadionik, H. Levi, H. Loukil, B. Atitallah, and N. Masmoudi, “An fpga implementation of motion estimation algorithm for h.264/avc,” in *IEEE 5th Internation Symposium on I/V Communications and Mobile Network (ISVC’10)*, Rabat, Sep. 2010, pp. 1–4.

[17] G. Pastuszak and M. Jakubowski, “Adaptive computationally scalable motion estimation for the hardware h.264/avc encoder,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 5, pp. 802–812, 2013.

[18] HEVC software repository HM–14.0 reference model, https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-14.0.

[19] Y.-L. S. Lin, C.-Y. Kao, H.-C. Kuo, and J.-W. hen, *VLSI Design for Video Coding - H.264/AVC Encoding from Standard Specification to Chip*. New York, NY: Springer, 2010.