A Power-Efficient SAD Architecture to Compressed Block Processing in Motion Estimation

Vítor Costa, Murilo Perleberg, Luciano Agostini, Marcelo Porto
Video Technology Research Group (ViTech)
Graduate Program in Computing (PPGC)
Federal University of Pelotas (UFPel) - Pelotas, RS, Brazil
{vscosta, mrperleberg, agostini, porto}@inf.ufpel.edu.br

***Abstract*—Video coding has become essential nowadays, enabling the transfer of video streams across various physical and online platforms. However, the video encoding process is extremely demanding in terms of computational power, requiring dedicated hardware for efficient real-time applications. Dedicated hardware design for video coding must handle various physical limitations, such as power dissipation, demanded area in chip, and memory related issues (size, accesses, and power). The Motion Estimation (ME) is the most important encoding step of current video encoders, but also the most time, computational, and memory demanding. To tackle this issue, this paper proposes the use of compressed blocks in the ME, enabling efficient Sum of Absolute Differences (SAD) calculation over a reduced amount of data, reducing internal memory demand and saving power dissipation in both internal memory and SAD processing. A dedicated hardware architecture for a block compressor and a configurable SAD architecture, which works over the compressed blocks data, were proposed. The ASIC synthesis results demonstrated that the use of compressed blocks can reduce the internal ME memory size and its power dissipation by up to 75%. Moreover, the proposed efficient SAD architecture can achieve up to 58.6% of power dissipation reduction in the SAD processing.**

Keywords—Motion Estimation, Block Compression, SAD architecture, Hardware Design, Video Coding.

I. INTRODUCTION

Currently, video content traffic on the internet has achieved a new purpose. In the past, video media were primarily used for entertainment purposes, to connect distant friends, or to record unique moments. Nowadays, video content has become highly diversified: streaming companies are preferred over traditional television by audiences [1], digital marketing and social networks increasingly explore the creation of short videos that quickly capture the viewer's attention [2], and meeting platforms no longer serve just to connect two people but are also used for business and learning [3]. All these trends highlight the necessity of having an efficient way to handle video content.

However, the video encoding process is a highly resource-intensive task, demanding many limitations and optimizations in the encoding standards to enable its implementation in an efficient dedicated hardware architecture that aligns with physical constraints such as power and heat dissipation, circuit area, operating frequency, and memory access, among others. Among all the stages in modern video encoders such as High Efficiency Video Coding (HEVC) [4], Versatile Video Coding (VVC) [5], and AOMedia Video 1 (AV1) [6], the Motion Estimation (ME) stands out due to its enormous complexity and computational demand, given the numerous computations the encoder needs to perform. The ME step consumes up to 60.7% of the total encoding time [7] and at least 29.5% of the memory accesses in the HEVC encoder [8].

The intensive memory accesses from ME results in high power dissipation demand from the memory in ME systems. So, to reduce memory accesses block compression algorithm is proposed in [9], which compresses the Candidate Blocks (CB) and the Prediction Unit (PU) data before the internal ME memory storage [9]. The block compressor algorithm presented in [9] compresses a 4x4 samples (8 bits) block into 32 bits, providing a reduction of 75% in internal ME memory with a BD-Rate drawback of only 0.2% [9]. The block compressor works with 16 compression modes, which can be signalized with a four-bit mode prefix. However, the ME process in [10] requires the full block decompression before the SAD operations, taking no advantages of the possible compressed-data reuse.

This paper proposes the use of compressed blocks for the efficient design of a dedicated and configurable SAD architecture, reducing internal ME memory size/power and also reducing the power dissipation of the SAD operation. This paper also introduces a new approach, by proposing a configurable SAD architecture to work over the compressed blocks data, which can be dynamically configured to work over a reduces amount of data according to the compression mode used at the block compressor. Two dedicated hardware architectures are presented, the Block Compressor Unit (BCU), which implements the block compressor algorithm presented in [9], and the Configurable SAD Tree Calculator (CSTC), which features a Mode Analyzer (MA) unit that can dynamically deactivate unnecessary operators according to the combination of compression modes applied over the compressed blocks, thereby reducing power dissipation of the SAD processing unit. The ASIC synthesis results of BCU show area and power overhead of only 52.97 Kgates and 85.02mW, respectively. The CSTC synthesis results show power dissipation reductions up to 58.6%, in comparison with the original approach.

II. MOTIVATION AND A CASE STUDY

The ME aims to estimate the displacement of a PU, a block of samples from a frame being encoded, with the most similar Candidate Block (CB) from an already encoded reference frame [11]. While it provides expressive compression capabilities, the ME is by far the most resource intensive step in current video

This research was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. It was also financed in part by the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul – Brasil (FAPERGS), and by the Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq).

coders, since it must calculate the SAD similarity criterion to all CBs inside a delimited Search Area (SA) to find the best CB to a current PU.

The search for the best CB can prove quite a challenge computationally especially for High (HD) and Ultra-High Definition (UHD) video resolutions. To evaluate a 64x64 PU, up to 329 CBs may be evaluated [10], thus requesting 84,224 4x4 CBs from the memory, demanding up to 2.6MB of internal memory for the ME when considering 8-bit samples from both PU and CB. Considering an UHD 4K video (3840x2160 pixels) with bit depth of 8 and a framerate of 60, each frame of the video displays 2,040 64x64 PUs, accounting for a memory bandwidth of up to 307 Gigabytes per second, a prohibitive number for current memory technologies.

Data compression over both the PU and CBs can be performed as strategy to reduce memory requirements in dedicated ME design. In [9], a block compressor algorithm is proposed to compress 4x4 PU/CBs, reducing the necessary 128 bits (4 x 4 x 8 bits) into only 32 bits. Considering the HEVC standard as an example, the necessary memory to store a 64x64 PU and its 192x192 SA may be reduced from 40kB to 10kB, a reduction of 75%. This compression also causes a reduction in the number of bytes necessary to perform the evaluations required by each 64x64 PU of the ME step, from 2.6MB to 658kB. Furthermore, up to 85.9% of the power dissipation of a dedicated ME hardware for the HEVC comes from the memory unit [10]. For that reason, by compressing the blocks it is possible to not just reduce the memory size necessary but it also greatly reduce the power related to the internal ME memory in at least 75%.

Besides the benefits related to the memory, another approach can be also exploited by the use of compressed blocks during the ME process. The algorithm proposed in [9], which will be detailed in Section III, for the block compression comes with the benefit of enabling data reutilization during the SAD calculations. If less values are utilized in order to represent a whole 4x4 block, some SAD calculations become redundant. The algorithm defines 16 modes of compression, and according to the combination of those selected compression modes, some calculations can be bypassed during the SAD process. By monitoring the compression modes of each block, it is possible to dynamically deactivate some operation units in the SAD tree, further reducing the power from the processing unit of the ME.

III. Proposed Architectures

This paper proposes two different hardware architectures, called: Block Compressor Unit (BCU) and Configurable SAD Tree Calculator (CSTC).

A. Block Compressor Unit (BCU)

The BCU is responsible to compress 4x4 blocks (PUs and CBs) of 8-bit samples into a vector with only 32 bits. The compression algorithm employed by the BCU was proposed in [9], where the authors define 16 compression modes to be evaluated over the 4x4 input block. Each mode uses only four of the 16 sample values available at the input block, reconstructing an entire 4x4 block by reusing those four selected sample values. The modes can be characterized in vertical, horizontal, and quadrant, according to which samples are selected to represent the input block, as shown in Fig. 1. Vertical modes such as modes 0–3, and mode 13, uses the values from one column or the average value from each column [9]. Horizontal modes, such as modes 4–7, and mode 14, uses the values from one row or the average value from each row [9]. Quadrant modes such as modes 8–12, and mode 15, uses a different sample value from each 2x2 quadrant or the average value from each quadrant [9].

The selection of the best mode is performed exhaustively, where all 16 modes are evaluated, and the one with the smallest SAD compared to the original block is chosen. After obtaining the best mode, the four selected sample values are truncated to 7-bit values, discarding their least significant bit. This allows the storage of the four sample values in 28 bits (4 x 7 = 28), with the four most significant bits of the 32-bit vector indicating which one of the 16 compression modes was used [9]. It is important to mention that, besides a complete ME hardware design, including the block compressor, is presented in [9], the paper only describes the compression algorithm, providing no details about the block compressor hardware implementation.

The proposed BCU architecture can be seen in Fig. 2, and it is divided into four stages. The first stage is responsible for generating the 4x4 reconstructed blocks reutilizing four sample values of the input block, for each of 16 modes. After that, the second stage computes the SAD values from each mode by using 16 SAD trees. The third stage performs the selection of the best mode of compression based on the smallest SAD. Using a tree composed by 31 multiplexer (MUX) selection, the binary value (0000 to 1111) of each mode is gated through the MUXes, and the SAD values of each mode in a single MUX are subtracted with the smallest one been used as selector. Finally, the fourth stage elaborates the 32-bit vector based on the best mode analyzed (the one with smallest SAD value), alongside concatenating the 4-bit prefix, regarding the best compression mode, with the truncated values of the four selected samples.

By using the BCU to compress a 128-bits block in a 32-bit word, it is possible to achieve a reduction in internal ME memory size of 75%. Moreover, the use of the compressed PU and CB data to perform the ME can further reduce the internal memory bandwidth and SAD computations. It is worth to mention that, considering the results in [9], the use of proposed block compressor in the ME of the HEVC introduced low coding efficiency drawback of only 0.2% in the BD-Rate.

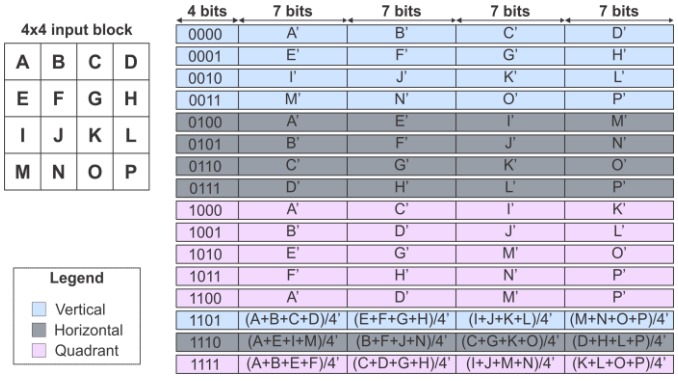

Fig. 1. Exemplification of compression modes and linear 32-bit vector containing four generic sample values.

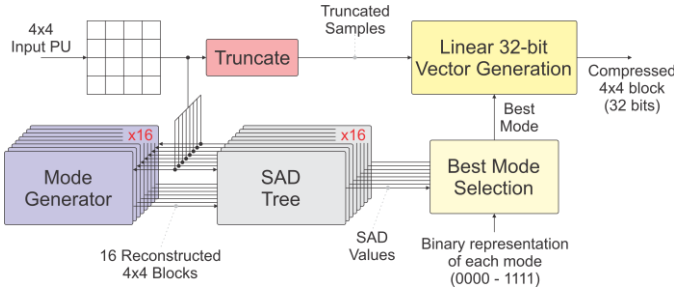

Fig. 2. High-level diagram of the BCU architecture.

B. *Configurable SAD Tree Calculator (CSTC)*

As previously stated, the use of compressed data for the PUs and the CBs can reduce the amount of computation during the SAD calculation. To explore the benefits of the compressed block this paper proposes the Configurable SAD Tree Calculator (CSTC). The CSTC implementation is shown in Fig. 3 and Fig. 4. It consists of a typical 4x4 block SAD tree with 31 operators (adders and subtractors), but with a control unit called Mode Analyzer (MA). The MA evaluates which operators must be used in order to obtain the SAD, based on the four most significant bits (compression mode) of each of the two compressed input blocks.

Considering the 16 modes of compression supported by the BCU, it is possible to determine some correlation, illustrated in Fig. 5, between vertical, horizontal and quadrant modes. The best-case scenario occurs if both selected modes (PU and CB) are the same, i.e., both are vertical, horizontal or quadrant. In this case, only four *Multiplexed Diff* units are required to obtain the SAD. This happens since there is only four different subtractions performed by the SAD tree, so it is only necessary to sum the absolute results of those four subtractions and shift two bits to the left to multiply the obtained result by four. A medium-case scenario occurs when one of the modes is quadrant and the other one is either vertical or horizontal. In in this case there are exactly eight different subtractions to be performed, requiring only eight enabled *Multiplexed Diff* units to fully obtain the SAD. The worst-case scenario happens when one of the modes is vertical and the other one is horizontal. In this situation, it is mandatory to use all 16 *Multiplexed Diff* unit available in the CSTC architecture.

The MA part of the architecture implements the logic presented in Fig. 4, generating a control word. Each bit of the control word goes into each of the 16 *Multiplexed Diff* units of the architecture seen in Fig. 3. Each *Multiplexed Diff* unit is composed by two 2:1 MUXes followed by a subtractor. When the control word is "0", it forces the value "0" to the subtractor inputs, consequently, data-gating the following unit inside the SAD tree. This strategy allows for a reduction in the architecture switching activity, thus decreasing its dynamic power dissipation.

IV. Synthesis Results

The proposed architectures were described in VHDL and synthesized using the RTL Compiler tool [12] to a TSMC 40nm technology standard-cells library [13]. The operational frequency of 370MHz was used for both architectures, which corresponds to the maximum frequency supported by the BCU architecture, thus having the longest critical path between the

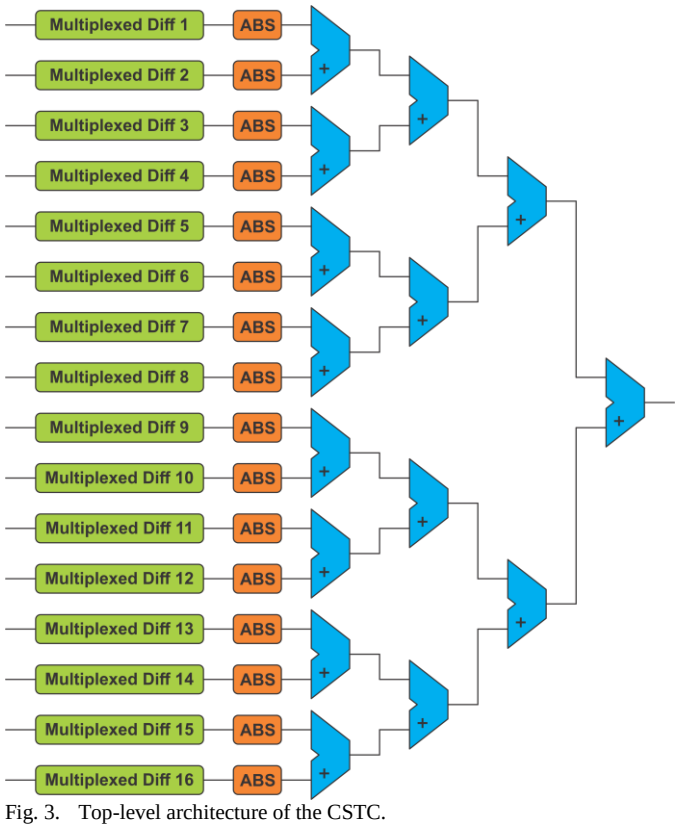

Fig. 3. Top-level architecture of the CSTC.

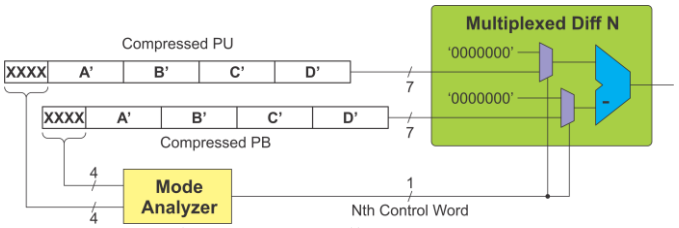

Fig. 4. Overview of the multiplexed diff architecture and mode analyzer

two architectures. In order to validate the architectures and generate the power results, a Value Change Dump (VCD) file containing ten thousand random-generated inputs were used.

The power overhead and cell area consumption imposed by the BCU is provided by Table I, and the synthesis results of the CSTC architecture are presented in Table II, where the power dissipation was evaluated for the different scenarios the CSTC can work. As can be seen in Table II, the area results are the same (21.67 Kgates), since the hardware remains the same for all scenarios. The only difference lies in the power estimation results, where each scenario provides the proper block compression modes, as presented in Fig. 5, to correctly disable the *Multiplexed Diff* units. Table II also presents the synthesis results for a Default SAD tree (approach in [9]), which has the same behavior of the CSTC in the worst-case scenario. The Default SAD tree architecture is very similar to the one presented in Fig. 3, but without the Mode Analyzer (MA) unit and the MUXes inside the *Multiplexed Diff* units.

Considering the Default SAD tree as the baseline result for comparisons, it is possible to see that CSTC requires more area (about 14.7%) due to the additional control hardware. Taking

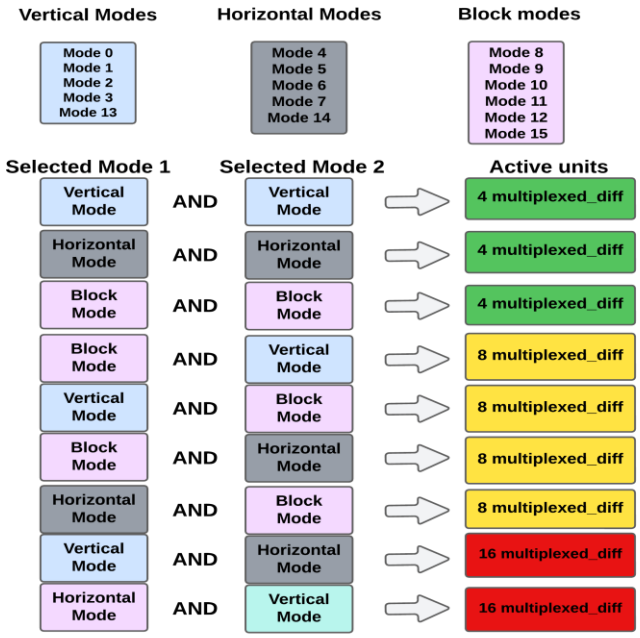

Fig. 5. Correlation between compression modes and its effects on SSTC

the power results, the worst-case scenario of CSTC presents higher power dissipation (about 6.8%), as expected. However, medium-case and best-case scenarios present power dissipation reductions of 35.2% and 58.6%, respectively. To find the power reductions of the CSTC in a realistic scenario, the occurrences of the selected compression modes of each PU and CB during the video encoding process must be discovered. However, considering a balanced distribution of the tree scenarios (33.33% each), the CSTC average power reduction is 29% when compared to the Default SAD Tree.

V. Conclusions

This paper proposed a power-efficient approach to the SAD calculation over compressed block at the ME which can be applied to any current video encoding standard. Two dedicated hardware architectures were presented, the first one is a block compressor responsible to compress the block samples from the current and the reference frames. The second architecture is a configurable SAD tree that works over the compressed data, aiming to dynamically disable unnecessary operation units. The proposed architectures can produce a reduction of 75% in both internal ME memory size and its power dissipation, and also a power reduction up to 58.6% in SAD processing unit.

Future works include the use of pipeline in order to increase the operating frequency of both architectures, a statistical analysis over the occurrences of the compression modes in a specific video encoder, to define a "realistic scenario" for the CSTC architecture, and the evaluation of the coding efficiency losses introduced by the use of the block compressor in the VVC and AV1 encoders.

TABLE I. BUC synthesis results using TSMC 40nm

Architecture	Cell area (Kgates)	Leakage power (mW)	Dynamic power (mW)	Total power (mW)
PUDC	52.97	0.15	84.88	85.02

TABLE II. CSTC synthesis results for each operation scenario using TSMC 40nm

Architecture	Cell area (Kgates)	Dynamic power (mW)	Total power (mw)	Total power delta (%)
Default SAD Tree	18.89	1.26	1.26	-
CSTC (best-case)	21.67	0.52	0.52	-58.6%
CSTC (medium-case)	21.67	0.81	0.82	-35.2%
CSTC (worst-case)	21.67	1.34	1.35	+6.8%

Acknowledgment

I dedicate this work to my grandmother, who passed away on 22 March 2024.

References

[1] T. Evens, A. Henderickx and P. Conradie, "Technological affordances of video streaming platforms: Why people prefer video streaming platforms over television," 2023 European Journal of Communication, 39, pp. 1-19, doi: 10.1177/02673231231155731.

[2] Dash. "Video marketing statistics for your 2024 campaings". Retrieved May 30, 2024: https://www.dash.app/blog/video-marketing-statistics

[3] Clockwise. "The rise and rise of virtual meetings". Retrieved May 30, 2024: https://work-clockwise.com/articles/the-rise-and-rise-of-virtual-meetings/

[4] ITU. "H.265: High Efficiency Video Coding". Retrieved May 30, 2024: https://www.itu.int/rec/T-REC-H.265-202309-I/en

[5] ITU. "H.266: Versatile Video Coding". Retrieved May 30, 2024: https://www.itu.int/rec/T-REC-H.266-202309-I/en

[6] Alliance for Open Media. "AV1 Video Codec". Retrieved May 30, 2024: https://aomedia.org/av1/

[7] M. Grellert, S. Bampi and B. Zatt, "Complexity-scalable HEVC encoding," 2016 Picture Coding Symposium (PCS), Nuremberg, Germany, 2016, pp. 1-5, doi: 10.1109/PCS.2016.7906356.

[8] A. Mativi, E. Monteiro and S. Bampi, "Memory access profiling for HEVC encoders," 2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS), Florianopolis, Brazil, 2016, pp. 243-246, doi: 10.1109/LASCAS.2016.7451055.

[9] Y. Fan, L. Huang, B. Hao and X. Zeng, "A Hardware-Oriented IME Algorithm for HEVC and Its Hardware Implementation," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 8, pp. 2048-2057, Aug. 2018, doi: 10.1109/TCSVT.2017.2702194.

[10] M. Perleberg, V. Afonso, L. Agostini, B. Zatt and M. Porto, "Memory-Centered Motion Estimation System With CTB-Based Full-Splitting Algorithm," in IEEE Transactions on Consumer Electronics, doi: 10.1109/TCE.2024.3399123.

[11] V. Sze, M. Budagavi, and G. J. Sullivan, "High Efficiency Video Coding (HEVC)," *Springer International Publishing*, 2014. doi: 10.1007/978-3-319-06895-4

[12] Cadence. "Cadence RTL Compiler". Retrieved May 30, 2024: https://www.cadence.com/en_US/home/training/all-courses.html

[13] TSMC. "40nm Technology". Retrieved May 30, 2024: https://www.tsmc.com/english/dedicatedFoundry/technology/logic/l_40nm