Low Area Reconfigurable Architecture for 3D-HEVC DMMs Decoder Targeting 1080p Videos

[1,2]Gustavo Sanchez, [1]Ramon Fernandes, [1]Rodrigo Cataldo, [3]Luciano Agostini, [1]César Marcon
[1] Pontifical Catholic University of Rio Grande do Sul – Porto Alegre, Brazil
[2] IF Farroupilha – Alegrete, Brazil
[3] Federal University of Pelotas – Pelotas, Brazil

***Abstract*— The 3D High Efficiency Video Coding (3D-HEVC) needs specialized tools, such as the Depth Modeling Mode (DMM), for dealing with texture and depth map information. This work proposes a low area reconfigurable hardware for sharing a DMM-1 and DMM-4 decoding implementation. When synthesizing for a 65nm ST technology, the designed architecture uses only 5,165 gates and dissipates 1.57 mW, considering the frequency to decode the worst-case scenario. Compared to the state-of-the-art, it is notable that our proposal requires only a few extra hardware to enable a DMM-1 and DMM-4 sharing architecture.**

Keywords—3D-HEVC; Intra-Frame Prediction; Depth Maps Decoding; DMMs; Hardware Design

I. INTRODUCTION

Based on the well-known High Efficiency Video Coding (HEVC) [1], its 3D extension (3D-HEVC) [2] has provided several benefits for 3D video coding. To reduce the bandwidth necessary for 3D video transmission, 3D-HEVC adopts a model based on Multiview Video plus Depth (MVD) [3], which associates a depth map to each texture view. The coding process consists of interlaced coding the texture views and their associated depth map at the encoder; while, at the decoder side, view synthesis techniques interpolate the texture views based on depth map information, making a dense set of high-quality virtual texture views placed among the original views [4].

Unlike the texture view, which contains smooth transitions between samples, depth maps have large homogeneous areas within objects, with sharp edges on object borders.

The traditional HEVC algorithms demonstrate limited efficiency for encoding the homogeneous regions and sharp edges of depth maps, and are, therefore, subject to reduced encoding efficiency and prone to generating undesired artifacts in the synthesized views and consequently a mistaken interpretation of background/foreground samples [5]. The adoption of different encoding tools became necessary to address these limitations, such as Depth Modeling Modes (DMMs) [6], Segment-Wise Direct Component Coding (SDC) [7], and Depth Intra Skip (DIS) [8]. Combined, these tools and the original texture encoding tools can maintain high-quality virtual views while achieving a significant reduction in bitrate.

Currently exists only a superficial exploration of dedicated hardware for depth maps coding, which is expected since 3D-HEVC is a relatively new extension. In our previous work [9], we proposed a hardware architecture for DMM-1 decoding in 3D-HEVC based on a lossless compression technique for wedgelet memory. The DMM-1 decoder is capable of processing 1080p videos at 30 frames per second while minimizing power consumption and memory requirements. To the best of our knowledge, no other work presents a hardware approach for decoding these depth map tools. Meanwhile, the few existing works target the encoder side [10]-[12].

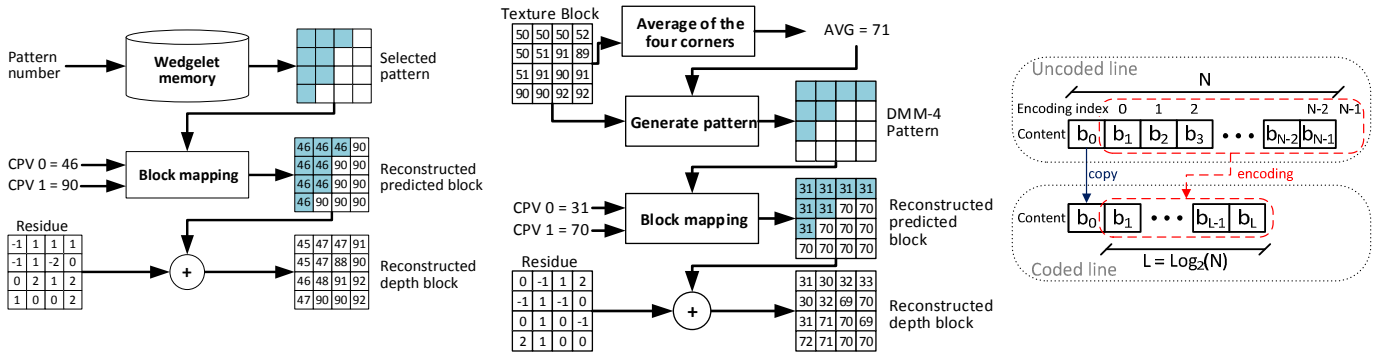
In this work, we present a reconfigurable DMMs decoding hardware design based on our previous work [9], which only implements DMM-1. Since DMM-1 and DMM-4 contain several common operations in decoding, the main contribution of our reconfigurable architecture is sharing hardware resources between modes, thus being capable of using a low area and low power design. To the best of our knowledge, this is the first decoder architecture capable of decoding a DMM-4 block.

II. DMMS DECODING ALGORITHMS

Fig. 1 exemplifies the decoding of two 4×4 DMM-1 and DMM-4 encoded depth blocks (Fig. 1 (a) and (b), respectively). For a given block size, the input of the DMM-1 decoding algorithm requires the number of the selected pattern, the Constant Partition Value (CPV) of each region, and the residual block. A DMM-1 decoder requires accessing the wedgelets memory for reading the wedgelet pattern. Then, the CPV of each region is mapped into this DMM-1 pattern to reconstruct the predicted block. Finally, the decoder adds the residues to the predicted block, generating the reconstructed depth block.

In the DMM-4, the decoder requires the CPV, the residual block, and the correlated texture block. The main difference between DMM-4 and DMM-1 decoding is that DMM-1 obtains its pattern by accessing a memory with a pre-defined set of patterns, while DMM-4 dynamically rebuilds the pattern doing the same operation performed in the DMM-4 encoder. To rebuild the pattern, DMM-4 firstly computes the average (AVG_TEXT) of the four corner texture samples. Based on this average value, DMM-4 compares each texture sample with the AVG_TEXT and sets samples above the average to one region and samples below the average to the other region. After building the pattern, the process is similar to DMM-1 for reconstructing the block. It maps the CPVs according to the DMM-4 pattern for reconstructing the predicted block and adds the residual block to it, obtaining the final reconstructed block.

978-1-5386-9562-3/18/$31.00 ©2018 IEEE



(a) DMM-1 Decoding example (b) DMM-4 Decoding Example (c) FB&C Model TABLE I.

Fig. 1. Decoding examples and FB&C Model.

When comparing DMM-1 decoding to DMM-4, it is clear that DMM-4 requires more operations, because it rebuilds the pattern dynamically, while DMM-1 obtains it by performing one read in the wedgelet memory.

Another important aspect that requires consideration in the decoding process occurs before decoding a block in either DMM-1 or DMM-4. The encoder does not transmit the CPV to avoid transmitting additional information. Therefore, the 3D-HEVC encoder performs a CPV prediction to enhance its encoding efficiency, sending this information in the bitstream. As such, before decoding in DMM-1 and DMM-4, it is necessary to reconstruct the CPV by computing the CPV prediction and adding to the delta CPV prediction, which is present in the bitstream. The architecture designed in this work is capable of reconstructing the CPV information before decoding the DMMs. [2] presents more details pertaining to the delta CPV computation.

A. Efficient Storage for DMM-1 Patterns

One of the major problems in the DMM-1 hardware design relates to the memory for storing its wedgelets [13]. Some works already proposed solutions for better storage of DMM-1 patterns such as [9], [13], and [14]. However, [9] and [14] can only be applied to the encoder side since they reduce the number of possible wedgelets.

In our previous work [13], First Bit and Change (FB&C) algorithm was proposed for better storage of DMM-1 patterns with the possibility to be used in both encoder and decoder, as it allows the obtaining all wedgelets patterns. Therefore, we have chosen this algorithm for integration with the designed decoder (as done in our previous work [9]).

FB&C considers the DMM-1 characteristic that divides each block into two and only two regions. Then, interlaced ones and zeros in a single DMM-1 pattern row do not have a representation, leading to an inefficient storage model if N×N bits are stored per pattern.

Fig. 1(c) illustrates the FB&C model, where the encoding of each row copies the first bit content of the original line to the coded one. The remaining of the encoded line contains the position where there is a bit change. Using this solution allows savings of 63.2% in DMM-1 memory requirements.

III. DESIGNED ARCHITECTURE

Fig. 2 displays the high-level block diagram of the designed DMMs decoder architecture. The design contains six modules: (i) CPV Reconstruction, (ii) Register bank, (iii) Block Reconstruction, (iv) Wedgelet Decision, (v) Wedgelet Memories, and (vi) Control. This high-level diagram is similar than the architecture designed in our previous work, with significant differences inside the Control and the Block reconstruction units. A Small difference in registers was required to compare with our previous architecture and no differences in CPV Reconstruction, Wedgelet Decision and Wedgelet Memories were required since the CPV reconstruction is the same for both algorithms and DMM-4 does not need access the wedgelets memory and its decisions.

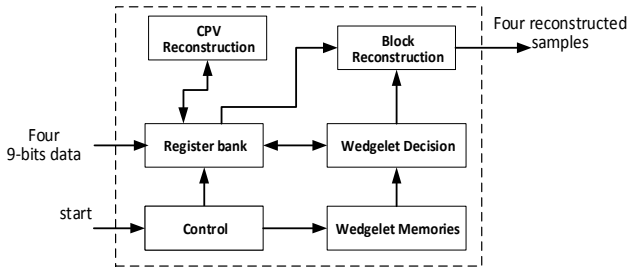


Fig. 2. High-level block diagram of the DMMs decoder.

The architecture was designed requiring at most 36 bits per cycle as input (four 9-bits data) to create a low area and low energy consumption decoder. Therefore, this design helps to reduce the communication with the main decoder memory, increasing the global decoder efficiency.

Fig. 3 displays a diagram with the architecture decoding flow for both DMM-1 and DMM-4 decoders. Initially, when the global decoder requires decoding a DMM block, it signalizes the architecture to start processing. In the next cycle, the decoder requires receiving five data among the input information, namely: DeltaCPV_0, DeltaCPV_1, Block_size, Mode, and Pattern_num. Notice that the input of our architecture can receive four chunks of 9-bits data. However, Block_size and Mode do not require a significant amount of bits, and as such, a chunk of 9-bits input data can integrate both.

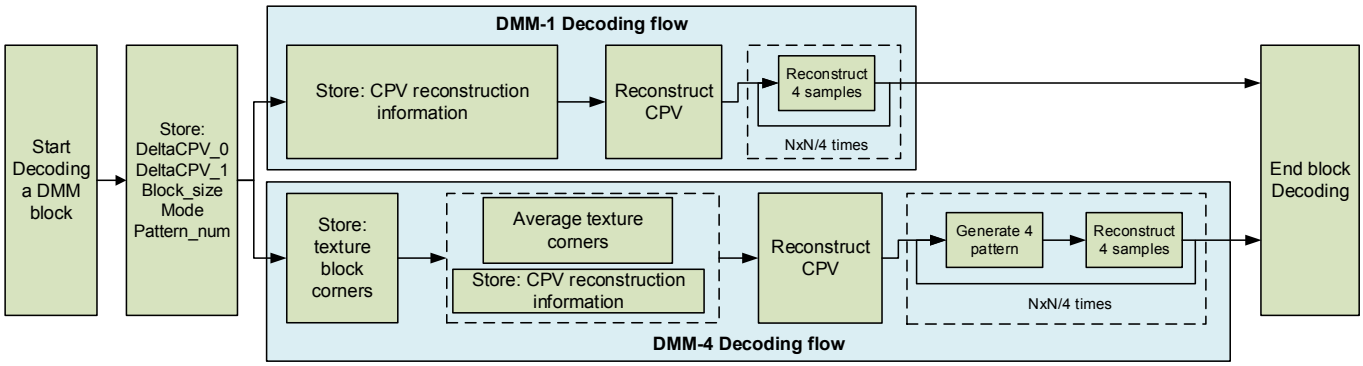


Fig. 3. Diagram of the architecture decoding flow for each mode.

Besides, Pattern_num is only used in case of DMM-1 being selected and discarded when the DMM-4 is selected, without being stored in the Register Bank. According to the Mode, the Control selects between two flows: (i) DMM-1 Decoding, and (ii) DMM-4 Decoding.

The DMM-1 decoding flow works exactly as in our previous work [9]. In the next two cycles, it stores neighbor blocks information required to reconstruct the CPV, as described in Section 2. Also, in these two cycles, the wedgelets memory is accessed, reading the corners information of the DMM-1 pattern since it is required to reconstruct the CPV. In the next cycle, the CPV Reconstruction module reconstructs the original CPV, generating all information required to decode the DMM-1 block. Afterward, the next cycles are used to reconstruct the encoding block, four samples per cycle using the Block reconstruction module, requiring N×N/4 cycles to reconstruct the entire block. Finally, after completing the block reconstruction, the architecture signalizes that the DMM-1 decoding is finished.

In the DMM-4 decoding flow, in the next decoding cycle, the four corners samples of the associated texture block are stored. In the following two cycles, while the architecture receives the neighbor samples required to reconstruct the CPVs, the texture corner samples are averaged using the block reconstruction module, since it already contains adders and therefore it is was necessary to insert only one more additional adder instead of three adders (see Fig. 4, which is later next). Additionally, the CPV reconstruction module also generates the binary pattern of the corner samples required for reconstructing the CPV value. Then, in the next cycle, the architecture reconstructs the original CPV value in the CPV reconstruction followed by a two-cycle operation that generates the four binary patterns, reconstructing the four samples. This two-cycle operation is repeated N×N/4 times for reconstructing the entire block. Finally, the decoder signalizes that the DMM-4 decoding is completed.

Notice that several operations performed in DMM-1 and DMM-4 decoding are similar, enabling a hardware design with lower area requirements than implementing two independent architectures for each mode.

Fig. 4 presents the block reconstruction module for the first two reconstruction samples (REC_0 and REC_1). Notice that the generation of REC_2 is similar to REC_0 and the generation of REC_3 is similar to REC_1, so the diagram omits them. The INPUT value is a texture sample or the depth residue according to the cycle. In the texture average computation, the four INPUT information are the texture corner samples and they are added and shifted right 2 bits (i.e., divided by 4), generating the TEXT_AVERAGE_OUT information that is stored in the register bank. When generating the four DMM-4 binary pattern information, the INPUT receives four texture samples, which are subtracted from TEXT_AVERAGE_IN (that comes from the register bank). The subtraction is stored in the register at the right side of the architecture and the information is loopback for the next cycle to select if CPV_REC0 or CPV_REC1 should be added with the INPUT value that contains a residue, outputting the reconstructed sample.

In DMM-1, the process is simpler than in DMM-4, as the PATTERN is read from the wedgelet memory and decoded in the wedgelet decision (because it was stored using FB&C). Then, the pattern is used to decide which CPV_REC should be selected to be added to the residue that is delivered to this module in the INPUT variable.

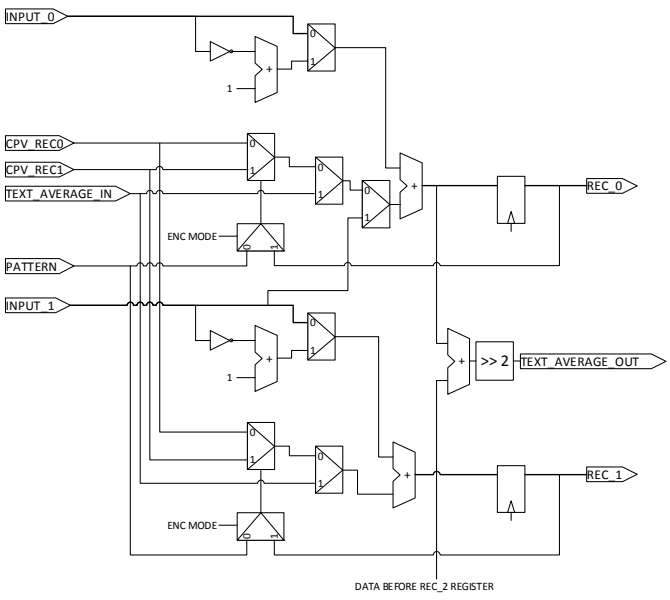


Fig. 4. Diagram of the block reconstruction for the first two samples.

IV. SYNTHESIS RESULTS AND COMPARISONS

The designed architecture was described in VHDL and synthesized for standard cells 65nm ST technology. The frequency used for synthesis considered the worst case for the proposed architecture, i.e., that all blocks were selected with size 4×4 and using the DMM-4 mode, aiming to decode 1080p videos at 30 frames per second (fps). However, this assumption overestimates the required area, processing rate, and power dissipation of the designed architecture. Since a real implementation would require decoding higher block sizes, which would require less frequency and because the mode selection is shared with others modes such as DMM-1 (that could allow reducing the operational frequency), or others modules such as HEVC intra-frame prediction, which would allow maintaining this architecture idle with a significant reduction in the power consumption. TABLE I. presents the synthesis results. Even with this overestimated assumption, the required area of the proposed architecture is of only 5,165 gates with a power dissipation of 1.57 mW.

TABLE I. also displays a comparison with a DMM-1 decoder architecture designed in our previous work [9]. Notice that our new architecture requires 27.6% higher area and 65.2% higher power consumption than that of the DMM-1 architecture. However, our new architecture contains two modes that can be decoded, and its synthesis results considered the worst case when DMM-4 was always selected. As previously described in Section II, decoding DMM-4 requires a higher computational complexity than DMM-1 because DMM-4 requires several computations for rebuilding its pattern, unlike the memory read operation employed in DMM-1. In addition, the designed architecture shares resources between DMM-1 and DMM-4 due to the common operations of both algorithms. As such, independent DMM-1 and DMM-4 architectures would probably require a higher area than our proposal.

V. CONCLUSIONS

This paper presented a low area hardware design for DMMs decoding in 3D-HEVC. The architecture was inspired by our previous work targeting DMM-1 decoding. However, the architecture shown in this work was designed by sharing resources between both modes for achieving a small area overhead. When synthesized for a 65nm ST technology, the architecture required an area of 5,165 gates and a power dissipation of 1.57 mW, considering the worst-case scenario, i.e., all decoding blocks were DMM-4 with block size 4×4. The architecture required a small increment in the area and power dissipation compared to our previous work, considering that it introduced the DMM-4 module, which decoding process requires higher computational effort than DMM-1 decoding.

TABLE I. SYNTHESIS RESULTS AND COMPARISONS

Solution	This	DMM-1 [9]
Implemented algorithms	DMM-1 and DMM-4	DMM-1
Area (gates)	5,165	4,047
Decoding block sizes	All	All
Frequency (MHz)	58.3	38.9
Cycles per block – DMM-1	10/22/70/262	10/22/70/262
Cycles per block – DMM-4	15/39/135/519	-
Processing rate HD 1080p fps	30.0	30.0
Power (mW)	1.57	0.95

ACKNOWLEDGMENT

This paper was achieved in cooperation with Hewlett-Packard Brazil Ltda. using incentives of Brazilian Informatics Law (Law nº 8.248 of 1991). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] G. J. Sullivan, et al "Overview of the high efficiency video coding (HEVC) standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[2] G. Tech, Y. Chen, K. Muller, J. R. Ohm, A. Vetro, and Y. K. Wang, "Overview of the multiview and 3d extensions of high efficiency video coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, pp. 35–49, Jan 2016.

[3] K. Muller, P. Merkle, and T. Wiegand, "3-D video representation using depth maps," Proceedings of the IEEE, vol. 99, no. 4, pp. 643–656, April 2011.

[4] P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger, "Depth map creation and image-based rendering for advanced 3dtv services providing interoperability and scalability," Image Commun., vol. 22, no. 2, pp. 217–234, Feb. 2007.

[5] X. Zhao, Y. Chen, L. Zhang, and M. Karczewicz, "Texture mode dependent depth coding in 3d-hevc," in 2013 IEEE International Conference on Image Processing, Sept 2013, pp. 1738–1741.

[6] P. Merkle, K. Muller, D. Marpe, T. Wiegand, "Depth Intra Coding for 3D Video based on Geometric Primitives," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), v. 26, n. 3, pp. 570-582, Feb. 2015.

[7] H. Liu and Y. Chen, "Generic segment-wise dc for 3D-HEVC depth intra coding," in 2014 IEEE International Conference on Image Processing (ICIP), Oct 2014, pp. 3219–3222.

[8] J. Lee, M. Park, and C. Kim, "3D-CEI: Depth intra skip (DIS) mode," document JCT3V-K0033, Geneva, Switzerland, Feb. 2015.

[9] G. Sanchez, L. Agostini, and C. Marcon, "High efficient architecture for 3D-HEVC DMM-1 decoder targeting 1080p videos," IEEE International Symposium on Circuits and Systems (ISCAS), 2018, p. 5.

[10] A. Farouk and E.B. Bourennane, "An efficient hardware solution for 3d-HEVC intra-prediction," Journal of Real Time Image Processing, 2017.

[11] G. Sanchez, C. Marcon, and L. Agostini, "Real-time scalable hardware architecture for 3D-HEVC bipartition modes," Journal of Real-Time Image Processing, vol. 13, no. 1, pp. 71–83, Mar. 2017.

[12] G. Sanchez, B. Zatt, M. Porto, and L. Agostini, "A real-time 5-views HD 1080p architecture for 3D-HEVC depth modeling mode 4," in Symposium on Integrated Circuits and Systems Design (SBCCI), Sept 2014, pp. 1–6.

[13] G. Sanchez, C. Marcon, L. Agostini. "Energy-aware Light-weight DMM-1 Patterns Decoders with Efficiently Storage in 3D-HEVC," Proceedings of the Symposium on Integrated Circuits and Systems (SBCCI), pp. 1-6, 2016.

[14] M. Shuying, Y. Wang, C. Zhu, Y. Lin, J. Zheng. "Reducing Wedgelet Lookup Table Size with Down-sampling for Depth Map Coding in 3D-HEVC," Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP), pp. 1-5, Oct. 2015.