Efficient Hardware for VVC Residual Syntax Elements Generation

Gabriel Bitencourt Cardoso*, Jiovana Sousa Gomes†, Sergio Bampi†, Fábio Luís Livi Ramos*
{gabrielbc.aluno, fabioramos}@unipampa.edu.br, {jsgomes, bampi}@inf.ufrgs.br
*Federal University of Pampa, Brazil
† Informatics Institute, Federal University of Rio Grande do Sul, Brazil

***Abstract*—Video processing is something ubiquitous these days (every year more and more video data is available e.g., streaming services like YouTube, Netflix, etc.). The task of tackling its storage and playback is handled by video codecs. One recent alternative is the Versatile Video Coding (VVC) standard, being the state-of-the-art in terms of video coding capabilities. As a drawback of its coding efficiency, the processing time has also severely increased. Thus, a viable solution is to use dedicated hardware accelerators for bottleneck tasks of the flow. One of these steps is the residual syntax elements processing, historically being the major contributor to the entropy encoding input. Hence, this work introduces a novel efficient hardware design for a partial VVC residual syntax elements generation.**

***Index Terms*—Video Compression, Hardware Design, VVC, Residual Coding.**

I. Introduction

In today's world, digital video applications have become integral to people's daily lives, finding utility in various contexts such as security, entertainment, work, and education. The significance of video is evident from projections by Ericsson [1], which indicate that video content will constitute a staggering 80% of global mobile network traffic by 2028.

Given the exponential growth in video consumption, there is a pressing need for effective video encoding solutions to facilitate the transmission and storage of this vast amount of data across our everyday devices. One noteworthy solution that has emerged recently is the AV1 format [2], developed by the AOMedia consortium. This format aims to be both royalty-free and competitive against traditional standards, offering an alternative with high performance. Conversely, Versatile Video Coding (VVC), also referred to as H.266, represents cutting-edge technology in compression efficiency as the latest standard from the Joint Video Experts Team [3]. As the state-of-the-art in video compression, VVC sets new benchmarks in reducing file sizes while maintaining high visual quality.

By leveraging advanced techniques, VVC addresses the challenges posed by the ever-increasing demand for high-resolution video content. Nevertheless, the advancements in compression efficiency that it achieves come at the expense of increased computational complexity when compared to previous solutions. This heightened complexity poses a hurdle in terms of processing power and efficiency, particularly for general-purpose computers that may struggle to handle the demanding computations involved.

To surmount this challenge, one viable solution is the utilization of dedicated hardware accelerators. By offloading the intensive computational tasks required by VVC onto specialized hardware accelerators, the overall processing time can be greatly reduced, enabling real-time video encoding and decoding even for high-resolution content.

Therefore, being the residual coding a bulky step to provide data for the Entropy Encoding stage (the last step of the encoding flow) for previous and current codecs [4], [5], a hardware design seems an adequate choice. The goal is to provide CABAC (Context-Adaptive Binary Arithmetic Encoder) [6] - the arithmetic encoding algorithm of VVC entropy stage - with enough input (called bins) to avoid it starving.

Hence, this work introduces a novel efficient hardware design for the partial VVC Residual Syntax Elements (RSE) generation. To the best of the authors' knowledge, this is the first-ever proposed architecture for the mentioned goal found in the literature.

II. Residual Coding in VVC

In previous standards, such as HEVC [4], and also for VVC [5] residual coding has a tendency to comprise the majority of input data for the entropy encoding. Thus, one might assume that it is a worthy part to be improved via hardware accelerators.

The residual coding derives from the residues after the prediction step, and the transform coefficients after the prediction. For short, they will be referred to here simply as coefficients or **coefs**. Generally, they are organized in a square-shaped organization, as one may see in Fig. 1. The extraction of the information from the residual coefficients will generate

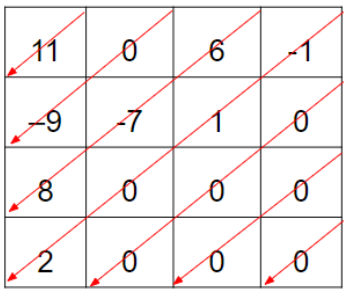

Fig. 1. Example of 4x4 Residual Coding Coefficients.

last sig coef x	3									
last sig coef y	0									
sig	1	1	0	1	1	1	1	0	1	1
gt1	0	0		1	1	1	1		1	1
par	1	1		0	0	1	0		1	1
gt3				0	1	1	1		1	1
sign	1	0		0	0	1	0		1	0
rem					1	1	2		2	3

Fig. 2. Example of Residual Syntax Elements Generation

the related syntax elements, following the red arrows order as depicted in Fig. 1, starting at the bottom-right value up to the top-left coefficient. Hence, the referred example will by the guide to better explain each one of them next.

The first step is to, inside the transform matrix, discover where is the first significant (i.e., non-zero) coefficient in reverse scan order position. The position where it is found out is signaled by the **last sig coef x** and **last sig coef y**. Thus, it is assumed that all previous read coefficients are non-significant (i.e., have the value zero).

For the blocks (or the remaining of the blocks) where there are significant coefficients, six basic RSE may occur, namely: (i) **sig** - indicates if the coefficient is significant i.e. if the coefficient has a value other than zero; (ii) **sign** - indicates the signal of the number, in case it has a value other than zero; (iii) **par** (from parity) - says the parity of the coefficient; (iv) **gt1** (i.e., greater than one) - shows if the absolute value is greater than one, in case it is significant; (v) **gt3** (i.e., greater than three) - indicates if the absolute value of the coefficient is greater than three; and (vi) **rem** (from remaining) - which make a manipulation with the coefficients greater than or equal to four - basically it subtracts the absolute value by four and divides the rest by two. Fig. 2 shows how they are produced based on the example of Fig. 1, as long as where, in the 4x4 block, the last position RSE occurs (i.e., the **last sig coef** RSE).

III. Hardware Design for Residual Coding in VVC

The hardware design for the RSE is depicted in Fig. 3. Basically, combinational logic is used to correctly generate all the elements presented in the previous section. Since all elements are generated at once for a given coefficient, appearing as Coef in Fig. 3, there is no need to revisit it after the first usage, saving memory access when one considers a complete encoder design.

To shed some light on the design, here are the explanations of each RSE generation:

SIG - a simple comparison is performed to verify if the coefficient is zero is or not.

GT1 - another comparison is made, but now to verify if the coefficient is greater than one in absolute value

PAR - the least significant bit o the coefficient corresponds to the parity of the value.

GT3 - a comparison is made to verify if the coefficient is greater than three, considering again its module.

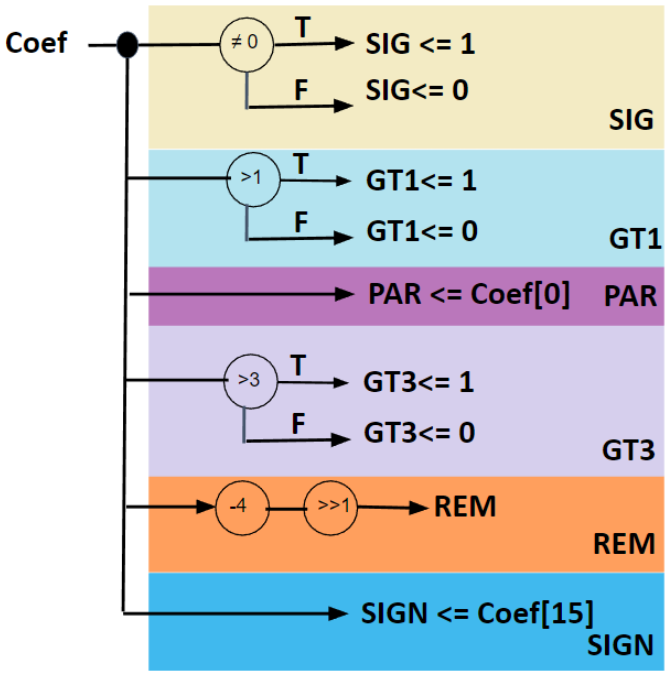

Fig. 3. Architecture for the VVC Residual Syntax Elements Generation.

REM - a subtraction by four is performed on the absolute value of the coefficient, proceeded by a shift-right by one position (i.e., an integer division by two).

SIGN - the most significant bit of the coefficient corresponds to its signal (considering a two's complement representation).

One may notice again that all RSE are generated at once, even if they are not valid for a given coefficient (for instance, if the absolute value of the Coef is one, it does not generate PAR, nor GT3, etc.). Therefore, the validity of each RSE depends on the generation of another RSE, as can be seen in Table I. In our design, we are assuming that the valid RSE are the ones to be registered for further usage in VVC's CABAC [3]. Thus, the enable signal of the specific register is the valid flag for each one of them.

IV. Synthesis Results

The design was described in VHDL and synthesized for ST 65 nm using the Cadence Genus tool. The frequency,

TABLE I
REQUIRED CONDITIONS TO GENERATE RESIDUAL SEs

Syntax Elements	**SIG**	**GT1**	**PAR**	**GT3**	**SIGN**	**REM**
Value	0, if coef. = 0 1, otherwise	1, if coef. >1 0 otherwise	0 when even, 1 when odd	1, if coef. >3, 0 otherwise	0 when negative, 1 when positive	(coef. - 4) /2
Valid Condition	-	SIG = 1	GT1 = 1	GT = 1	SIG = 1	GT3 = 1

TABLE II
SYNTHESIS RESULTS

Max freq:	800 MHZ
NAND2 Area:	682
Leak power:	1.6 uW
Internal power:	408 uW
Net power:	1896 uW

area, and power results are shown in Table II. These results where generated considering only logic synthesis standard setup (e.g., switching activity) at this moment. We considered it to be sufficient that only one transformed coefficient is processed per cycle since a hardware design for VVC CABAC has not yet been found in the literature. Thus, one may not infer how much of these RSE the entropy encoder is able to handle at once without starving. Moreover, since the VVC CABAC (differently from HEVC and AVC) has the obligation to perform its operation using a multiplier with non-constant inputs [3], it is predicted that this will have a throughput penalty when compared to previous standards (at the advantage of having more coding efficiency).

To the best of the authors' knowledge, there is no similar work in the literature for VVC (there is only for HEVC, such as [4]. Hence, no direct comparison could be made against other similar hardware designs.

V. CONCLUSION

This work has introduced an efficient hardware design for a partial VVC Residual Syntax Elements generation, being the first ever found in the literature for that purpose. The goal was to generate six of the basic RSE at once for every transformed coefficient. Synthesis results showed the overall performance of the architecture. As future work, a complete design for all RSE is intended, and the processing of more than one transformed coefficient is also on the radar.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento e Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and also by CNPq and FAPERGS Brazilian research support agencies.

REFERENCES

[1] Ericsson, "Ericsson Mobility Report," Tech. Rep., 2022. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report

[2] P. de Rivaz and J. Haughton, "AV1 bitstream & decoding process specification," *The Alliance for Open Media*, p. 182, 2019. [Online]. Available: https://aomediacodec.github.io/av1-spec/av1-spec.pdf

[3] B. Bross, J. Chen, and S. Liu, "Versatile video coding (VVC) draft 6," *Document JVET-M1001*, 2019.

[4] F. L. L. Ramos, A. V. P. Saggiorato, B. Zatt, M. Porto, and S. Bampi, "Residual Syntax Elements Analysis and Design Targeting High-Throughput HEVC CABAC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 2, pp. 475–488, 2020.

[5] H. Schwarz, M. Coban, M. Karczewicz, T.-D. Chuang, F. Bossen, A. Alshin, J. Lainema, C. R. Helmrich, and T. Wiegand, "Quantization and Entropy Coding in the Versatile Video Coding (VVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3891–3906, 2021.

[6] D. Marpe, Schwarz. H, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 13, no. 7, pp. 620–636, 7 2003.