International Conference on Information Engineering, Management and Security
2015 [ICIEMS 2015]

ISBN	978-81-929742-7-9		**VOL**	01
Website	www.iciems.in		**eMail**	iciems@asdf.res.in
Received	10 - July - 2015		**Accepted**	31- July - 2015
Article ID	ICIEMS033		**eAID**	ICIEMS.2015.033

Optimization of the critical loop in Renormalization CABAC decoder

Karthikeyan.C[1], Dr.Rangachar[2]
[1]Assistant Professor, ECE Department, MNM Jain Engg.College, Chennai
[1]Research scholar, Hindustan University, Chennai-603103
[2]Senior Professor, Dean for school of electrical science, Hindustan University, Chennai, INDIA

Abstract: *Context-based adaptive binary arithmetic coding (CABAC) is needed in the present days for high speed H.264/AVC decoder. The high speed is achieved by decoding one symbol per clock cycle using parallelism and pipelining techniques. In this paper we present an innovative hardware implementation of the renormalization which is a part of CABAC binary arithmetic decoder. The renormalization of range and value is specified as a sequential loop process that shifts only one bit per cycle until the range and value are renormalized. To speed up this process, a special hardware technique is used. The hardware will take one clock cycle to shift n bit data. The proposed hardware is coded using HDL language and synthesized using Xilinx CAD tool.*

Keywords: CABAC, renormalization, H.264, AVC, MPEG2 etc

I. INTRODUCTION

For multimedia coding applications, ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group (MPEG) jointly developed the latest video standard H.264/AVC (ITU-T Recommendation H.264:2003). Compared with existing video coding standards this provides more than twice the compression ratio while maintaining video coding quality. The higher throughput is due to the adoption of many new techniques, such as multiple reference frames, weighted prediction, deblocking filtering and context-based adaptive entropy coding. There are two approaches available for context- based adaptive entropy coding namely context-based adaptive variable length coding (CAVLC) and context-based adaptive binary arithmetic coding (CABAC). The CABAC coding achieves better compression efficiency better than CAVLC, but it brings higher computation complexity during decoding.

The compression efficiency is up to 50% over a wide range of bit rates and video resolutions compared to previous standards (e.g. MPEG2 or H.263). The downside is that the decoder complexity also increased; it is about four times higher [2]. Using a DSP processor to decode a single bin, it takes 30 to 40 cycles. In order to improve the video decoding, the throughput of a video coder using CABAC reaches almost 150 Mbin/s, which makes it difficult to implement in a programmable processor. Therefore, an efficient hardware decoder [3] is important for low-power and real-time H.264 codec applications. The decoding process of CABAC is bit-serial and has strong data dependency because the next bin process is depended on the previous bit decoding result. This data dependency makes the designer to exploit parallelism during decoding is difficult. The context models [5] of the current syntax element (SE) are closely related to the results of its neighboring macro blocks (MBs) or blocks, which leads to frequent memory access. The researchers are addressing these issues for exploring the parallelism and optimize memory access.

Cite this article as: Karthikeyan C, Dr. Rangachar. “Optimization of the critical loop in Renormalization CABAC decoder.” *International Conference on Information Engineering, Management and Security (2015)*: 199-203. Print.

Figure 1 shows H.264/AVC's basic coding structure for encoding one macro block, a sub block of a frame of the video stream. The decoder is used inside the encoder to obtain best perceptual quality at the decoder side. To reduce block artifacts an adaptive deblocking filter is used in the motion compensation loop. This combined with multiple reference frames and sub-pixel inter and intra mode motion compensation gives very strong compression results.

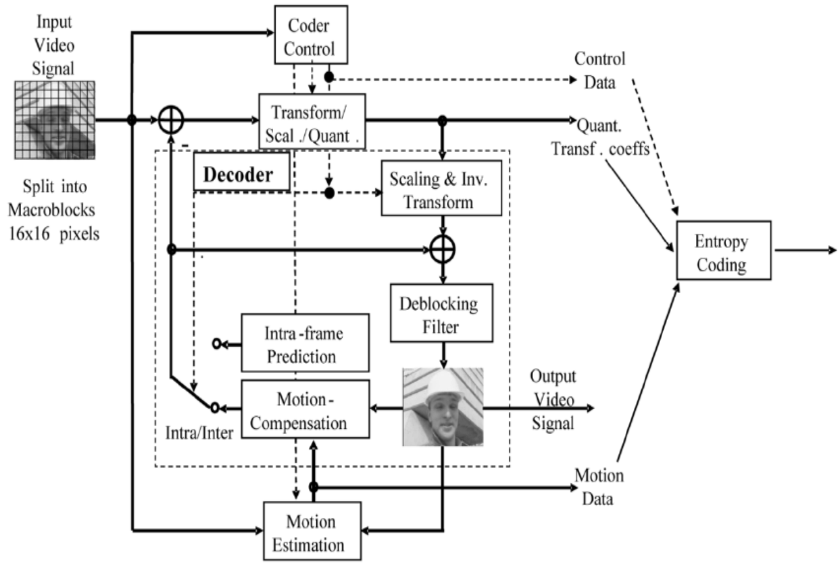

Figure. 1. H.264/AVC macro block encoder with functional blocks and data flows.

The decoder is a central part of the encoder. In section II, we introduce the primary steps of CABAC encoding and decoding process. In Section III, we describe the basic scheme of our CABAC decoder architecture. We present an overview of the framework of our renormalization hardware architecture. In this section IV, we focus on the simulation and synthesize of the proposed architecture. In Section V, we summarize the conclusions and future work.

II. CABAC ENCODER AND DECODER

In this section the basic principles of CABAC encoding and decoding process are discussed. The CABAC encoding and decoding process consists of three elementary steps.

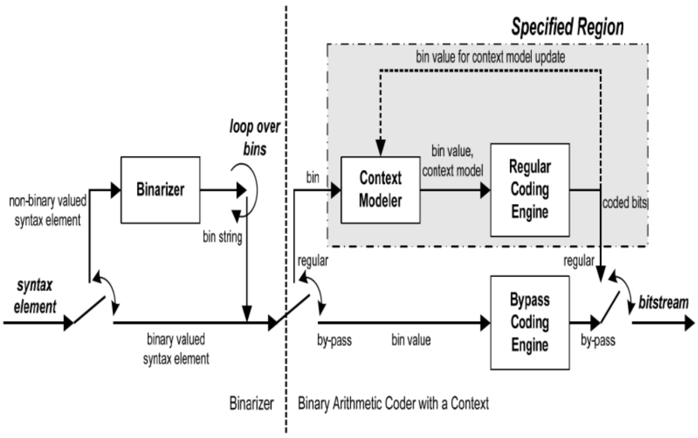

Figure 2 CABAC Encoder diagram

Figure 2 shows the encoding procedure of CABAC [9]. In the first step a given binary valued syntax element is uniquely mapped to a binary sequence, called bin string by the binarizer unit. When the input itself is in binary format this initial step is bypassed. For each element of the bin string or for each binary valued syntax element, one or two subsequent steps may follow depending on the coding mode. In the regular coding mode, prior to the actual arithmetic coding process the given binary decision which, in the sequel, referred to as a bin, enters the context modeling stage, where a probability model is selected such that the corresponding choice may depend on previously encoded syntax elements or bins. After the assignment of a context model the bin value along with its associated model is passed to the regular coding engine, where the final stage of arithmetic encoding together with a subsequent model updating

Cite this article as: Karthikeyan C, Dr. Rangachar. "Optimization of the critical loop in Renormalization CABAC decoder." *International Conference on Information Engineering, Management and Security (2015)*: 199-203. Print.

takes place. Bypass coding mode is chosen for selected bins in order to allow a speedup of the whole encoding process by means of simplified coding engine without the usage of an explicitly assigned model.

The CABAC encoder consists of three elementary steps: binarization, context modeling and binary arithmetic coding [4]. These incoming data are the coefficients from the transformations in Figure 1 together with some context information. In the second step a fitting probability model, based on the context, is selected for each binary symbol. This model drives the arithmetic coder (step three) by providing an estimate of the probability density function (PDF) of the symbol that will be encoded. The better this estimate, the better the compression. CABAC uses in total 399 models to model the PDFs of each syntax element such as macro block type, motion vector data, texture data, etc. The models are kept 'up to date' during encoding through the use of an *adaptive* coder [6] which estimates the PDF based on previously coded syntax elements.

There are three major data dependencies are extracted as follows: Renormalization is dependent on range update.

- Probability transition is dependent on bin decision
- Context switching is dependent on decoded bin

These three data dependency relations lead to three recursive computation loops, which can hardly be sped up by pipelining [7],[10], and thus largely limit the system performance. The following table I illustrates the frequency and the necessary operation to the internal variables. If the decoded symbol is the least probable symbol (LPS), it takes more cycles to evaluate the next coding range and coding offset required for the next symbol decoding. The coding range should always be modified and the offset should also be decremented. To find the shift amount n, we also need to count the leading zeros of the codeword. On the contrary, the consequent operations are much simpler when the decoded symbol is the most probable symbol

Table I Update variable after one symbol decoding

Sl.No	Case	MPS decoding	LPS decoding	Inference
1	Frequency	Frequent	None	No renormalization
	Range	R_{MPS}	-	
	offset	No change	-	
2	Frequency	Rare	Always	Renormalization
	Shift amount	1	Arbitrary	
	Coding range	R_{MPS}<<1	R_{LPS} << n	
	Coding offset	Offset<<1	(Offset-R_{MPS}) << n	

The following is the renormalization process in the arithmetic decoding engine

1. It accepts bit inputs from slice data and the variables codIRange and codIOffset.
2. After the renormalization process it outputs the updated variables codIRange and codIOffset.
3. A flowchart of the renormalization is shown in Figure 4. The current value of codIRange is first compared to 0x0100:
   - If codIRange is larger than or equal to 0x0100, no renormalization is needed and the RenormD process is finished.
   - Otherwise (codIRange is less than 0x0100), the renormalization loop is entered. Within this loop, the value of codIRange is doubled, i.e., left-shifted by 1 and a single bit is shifted into codIOffset by using read_bits(1).

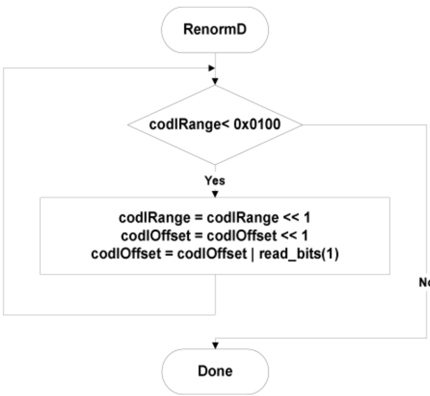

Figure 4 Flowchart of renormalization

Cite this article as: Karthikeyan C, Dr. Rangachar. "Optimization of the critical loop in Renormalization CABAC decoder." *International Conference on Information Engineering, Management and Security (2015)*: 199-203. Print.

III HARDWARE IMPLEMENTATION OF RENORMALIZATION

Re-normalization engine based on a head-one detector The last step of the decode decision engine flow is renormalization. To keep the precision of the whole decoding process, the refined codIOffset and codIRange have to be renormalized to ensure that the codIRange is not less than 256. For example, if the refined codIRange is 9'b000001010, the codIRange should be shifted five bits while the codIOffset reads five bits from the bit stream during the renormalization process. Based on the principle of renormalization, we find that if we locate the first appearing '1' inside the codIRange, we can successfully decide the number of bits of the codIRange to shift and of the codIOffset to read. Moreover, the renormalization process is part of the critical timing path in CABAC hardware decoder implementation.

To improve clock frequency, this path must be kept as short as possible. Thus, a parallel 'head-one detector' re-normalization architecture is proposed in the figure 5. Nine bits of the codIRange are split into three parts (3-bit vector), each of which determines whether there is a '1' among three input bits.

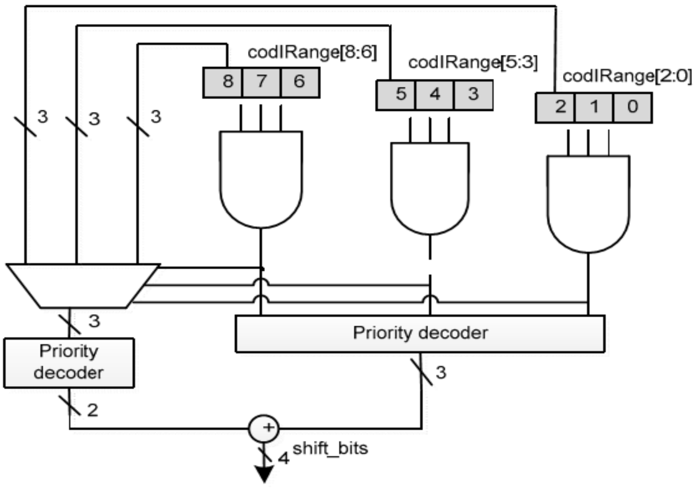

Figure. 5 Re-normalization engine based on a head-one detector

IV RESULT AND DISCUSSION

The proposed architecture is coded using HDL language. We have used structural level implementation and the simulation result of renormalization of given data is shown in the figure 6

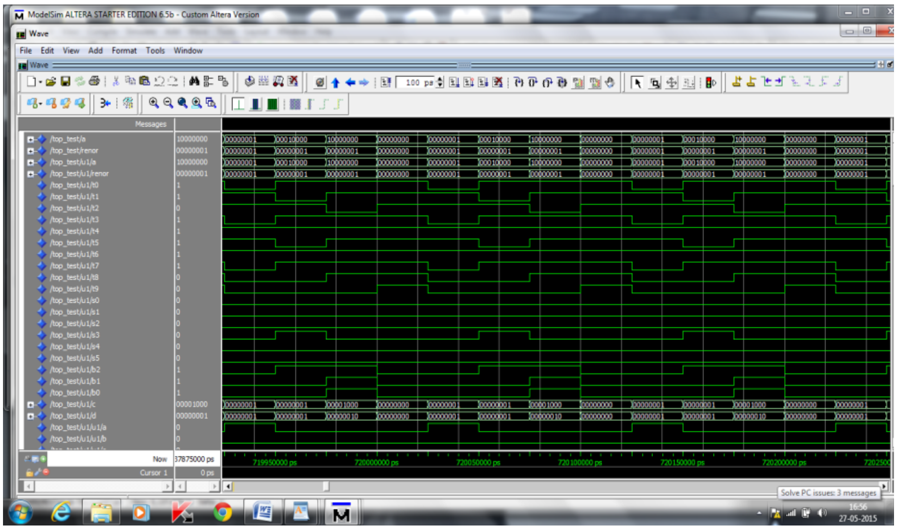

Figure 6 simulation result of renormalization process

The above code is further synthesized using Xilinx EDA tools. The device used for synthesize is vertex 4 200k FPGA. The RTL diagram is shown in the figure 7. The device utilization summery is shown in the table II.

Table II Device utilization summery

Cite this article as: Karthikeyan C, Dr. Rangachar. "Optimization of the critical loop in Renormalization CABAC decoder." *International Conference on Information Engineering, Management and Security (2015)*: 199-203. Print.

Sl.No	Description	Utilized	Available	% of utilization
1	Slices	15	89088	0%
2	4 input LUTs	26	178176	0%
3	Bounded IOBs	16	960	1%
4	Maximum combinational path delay 8.48 ns			

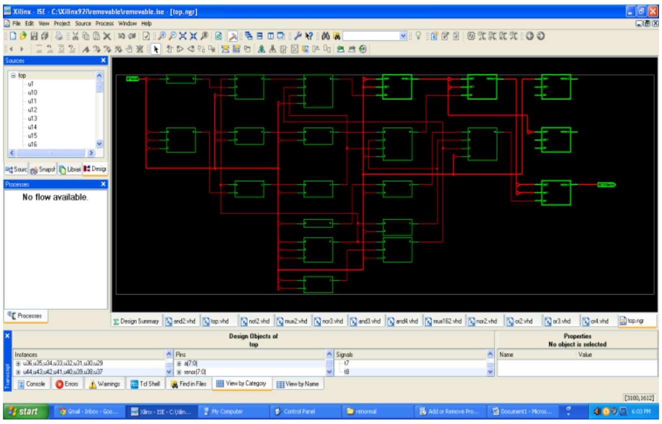

Figure 7 RTL View of renormalization

CONCLUSION

In this work we have presented a novel FPGA-design for renormalization engine which is present in CABAC decoder. CABAC decoder uses leading one detector for the renormalization. We have proposed a hardware which will have one clock cycle to find the leading one in the given bit stream. The proposed hardware is simulated and synthesized using CAD tools. The maximum frequency of operation is 117 MHz.

REFERENCES

[1] T. Wiegand, G. J. Sullivan, and G. B. A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560–576, July 2003.

[2] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, “Video coding with H.264/AVC: tools, performance, and complexity,” IEEE Circuits and Systems Magazine, vol. 4, no. 1, pp. 7–28, 2004.

[3] Wei Yu, Yun He, “A high performance cabac decoding architecture”, IEEE Trans. Consum. Electron., vol. 51, no. 4, pp. 1352-1359, Nov. 2005.

[4] D. Marpe, H. Schwarz, G. Bl¨attermann, G. Heising, and T. Wiegand, “Context-based adaptive binary arithmetic coding in JVT/H.26L,” Proc. IEEE International Conference on Image Processing (ICIP’02), vol. 2, pp. 513–516, September 2002.

[5] D. Marpe, H. Schwarz, 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, vol. 13, no. 7, pp. 620–636, July 2003.

[6] R. R. Osorio, J. D. Bruguera, “A New Architecture for fast Arithmetic Coding in H.264 Advanced Video Coder”, IEEE Proc. Euromicro Conference on Digital System Design, pp. 298-305, Sept. 2005.

[7] Yuan, T.C., “A Novel Pipeline Architecture for H.264/AVC CABAC Decoder”, IEEE Asia Pacific Conf. on Circuit and Systems, p.208-311, 2008

[8] Kuo, M.Y., Li, Y., Lee, C.Y., “An Area-Efficient High-Accuracy Prediction-Based CABAC Decoder Architecture for H.264/AVC”, IEEE Int. Symp. on Circuit and Systems, p.160-163, 2011.

[9] Liao, Y.H., Li, G.L., Chang, T.S., “A highly efficient VLSI architecture for H.264/AVC level 5.1 CABAC decoder”, IEEE Trans. Circ. Syst. Video Technol., 22(2): 272-281, 2012.

[10] Shi, B., Zheng, W., Lee, H.S., Li, D.X., Zhang, M., Pipelined Architecture Design of H.264/AVC CABAC Real-Time Decoding. 4th IEEE Int. Conf. on Circuits and Systems for Communications, p.492-496, 2008.

Cite this article as: Karthikeyan C, Dr. Rangachar. “Optimization of the critical loop in Renormalization CABAC decoder.” *International Conference on Information Engineering, Management and Security (2015)*: 199-203. Print.