Survey of advanced CABAC accelarator architectures for future multimedia.

Survey of Advanced CABAC Accelerator Architectures for Future Multimedia

Yahya Jan and Lech Jozwiak

Faculty of Electrical Engineering
Eindhoven University of Technology, The Netherlands
{Y.Jan,L.Jozwiak}@tue.nl

Abstract. The future high quality multimedia systems require efficient video coding algorithms and corresponding adaptive high-performance computational platforms. In this paper, we survey the hardware accelerator architectures for Context-based Adaptive Binary Arithmetic Coding (CABAC) of H.264/AVC. The purpose of the survey is to deliver a critical insight in the proposed solutions, and this way facilitate further research on accelerator architectures, architecture development methods and supporting EDA tools. The architectures are analyzed, classified and compared based on the core hardware acceleration concepts, algorithmic characteristics, video resolution support and performance parameters, and some promising design directions are discussed.

Keywords: RC hardware architectures, accelerators, multimedia processing, UHDTV, video compression, H.264/AVC, CABAC.

1 Introduction

The real-time performance requirement of modern multimedia applications, like: video conferencing and telephony, medical imaging, and especially High Definition Television (HDTV) and new emerging Ultra HDTV (UHDTV) require highly efficient computational platforms. The problem is amplified by demands of higher and higher quality, particularly in the video broadcast domain, what results in a huge amount of data processing for the new standards of digital TV, like UHDTV that requires a resolution of (7680x4320)∼33Megapixel with a data rate of 24Gbps. Additionally, the latest standards video coding algorithms are much more complex. The computational platforms for multimedia are also required to be (re-)configurable, to enable their adaptation to the various domains, accessing networks, standards and work modes. Hardware accelerators constitute the kernel of such (re-)configurable high-performance platforms.

The H.264/AVC [1] is the latest multi-domain video coding standard that provides the coding efficiency of almost 50% higher than former standards at the cost of almost four times increase in computational complexity. Context-based Adaptive Binary Arithmetic Coding (CABAC) [2], an entropy coding technique, covers Main and High profiles of H.264/AVC for high-end applications. Its purely software based implementation results in an unsatisfactory performance for High

J. Becker et al. (Eds.): ARC 2009, LNCS 5453, pp. 342–348, 2009.

Definition (HD) video (e.g. 30-40 cycles are required on average for a single bin decoding on DSP [3]). CABAC is a bottleneck in the overall codec performance. Consequently, a sophisticated hardware accelerator for CABAC is an absolute necessity. However, the bitwise serial processing nature of CABAC, the strong dependencies among the different partial computations, a substantial number of memory accesses, and variable number of cycles per bin processing put a huge challenge on the design of such an effective and efficient hardware accelerator.

This paper surveys several most advanced recently proposed hardware accelerator architectures for CABAC. Its main purpose is to deliver a critical insight in the proposed solutions, and this way facilitate further research on accelerator architectures, development methods and supporting electronic design automation (EDA) tools. The architectures are analyzed, classified and compared based on the core hardware acceleration concepts, algorithmic characteristics, video resolution support and performance parameters in the hardware accelerator domain, like throughput, frequency, resource utilization and power consumption. Based on the architecture comparison some promising design directions are discussed in view of the requirements of current and future digital multimedia applications.

The rest of the paper is organized as follows. Section 2 introduces CABAC. Section 3 covers the main hardware accelerator concepts, implementation difficulties in CABAC and presents a critical review of advanced hardware accelerator architectures for CABAC in detail. Section 4 concludes the paper.

2 Introduction to CABAC

CABAC utilizes three elementary processes to encode a syntax element (SE), i.e. an element of data (motion data, quantized transform coefficients data, control data) represented in the bitstream to be encoded. The processes are: binarization, context modeling and binary arithmetic coding, as shown in Figure 1.

The *binarization* maps a non-binary valued SE to a unique binary representation referred to as bin string. Each bit of this binary representation is called a bin. The *context modeling* process estimates the probabilities of the bins in the form of context models, before they are encoded arithmetically. CABAC defines

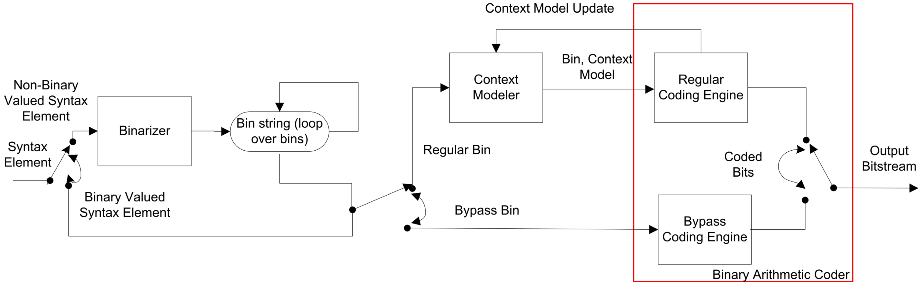

Fig. 1. Block Diagram of CABAC Encoder

460 unique context models, each of which corresponds to a certain bin or several bins of a SE, and are updated after bin encoding. Context model comprises of the probability state index (pStateIdx) and the most probable symbol (MPS) value of the bin. The *binary arithmetic coding engine* consists of two sub-engines: regular and bypass. The regular engine utilizes adaptive context models, but the bypass engine assumes a uniform context model to speed up encoding. To encode a bin, the regular coding engine requires the context model and the interval range (width) R and base (lower bound) L of the current code interval. The interval is then divided into two subintervals (R_{LPS}, R_{MPS}) according to the probability estimate (ρ_{LPS}) of the least probable symbol (LPS) [2]. Then one of the subintervals is chosen as the new interval based on whether the bin is equal to LPS or MPS. The context model is then updated, and the renormalization takes place to keep R and L within their legal ranges. The process repeats for the next bin.

3 Overview of Hardware Accelerators for CABAC

The main concepts of hardware acceleration can be summarized as follows: parallelism exploitation for execution of a particular computation instance due to availability of multiple application-specific operational resources working in parallel; parallelism exploitation for execution of several different computation instances at the same time due to pipelining; application-specific processing units with tailored processing and data granularity. More specifically these concepts can be oriented towards the data parallelism, functional parallelism and their mixture. In the past a number of different basic architecture types for hardware acceleration were proposed: parallel; pipeline; parallel pipeline; general purpose processor augmented by loosely coupled hardware accelerator; extensible/customizable application specific instruction set processor (ASIP) with basic accelerators in the form of instruction set extensions (ISE). These basic architectures will be used to categorize the CABAC accelerators.

Before considering the accelerators, we have to give a brief overview of the main implementation issues in CABAC. Five memory operations are involved in the en-/decoding of a single bin and two blocking dependencies that hamper the parallel and pipeline approaches. The first dependency is relevant to the context model update. Unless the context model is not updated for the current bin, the next bin processing cannot be started, because the same context model may be used to en-/decode the next bin. Other dependency involves the interval range R and base L update. Unless both are not renormalized in the renormalization stage, which involves multiple branches, the next bin processing cannot be initiated, because the probability estimation of the next bin depends on the current interval range R. These strong dependencies are some of the main challenges in the accelerator design, and a number of solutions are proposed.

The ***straightforward datapath/controller*** approach relies on the data flows in the algorithm of the software based solution. This accelerates the computations to some degree, but does not exploit the true (parallel) nature of the application algorithms. In CABAC accelerators, it takes as many as 14 cycles to process a single bin [4]. After further optimizations throughput of 0.2 bin/cycle is achieved in [4] for en-/decoding, and 0.33∼0.50 bin/cycle in [5] for decoding.

The inefficiency of the straightforward approach for HD video motivated the research community to propose ***parallel accelerators*** to process more than 1 bin/cycle. However, in en-/decoding of even a single bin complex interdependencies have to be resolved as discussed before, and consequently, the algorithm can not be parallelized in its true basic nature. Utilizing the static and dynamic characteristics of the SEs that can be discovered through CABAC analysis for real video, the parallelism can be achieved up to some level for some SEs, what can result in processing of more than 1 bin/cycle. However, in parallel en-/decoding of two or more regular bins the context models have to be supplied to the coding engines. Due to the blocking dependencies, this cannot be performed in parallel.

In the first parallel architecture for CABAC decoding [3] the parallelism is achieved through a cascade of the arithmetic decoding engines: two regular ones and two bypass. This enables the decoding of 1 Regular Bin (1RB), 1RB with 1 Bypass Bin (1BB), 2RB with 1BB, and 2BB bins in parallel for frequently occurring SEs, like residual data and results in the throughput of 1∼3 bins/cycle. The architectures [6][7][8][9] are based on the same concept, but after specific extensions are capable to en-/decode HD video. In [10] five different architectures for CABAC encoder were proposed. Two RB with BB architectures perform better for HD video than the others. A predictive approach is employed in [11]. Unlike [3][6][7][8][9], in which there is a latency due to the cascaded arithmetic engines, this architecture initiates decoding of two bins simultaneously by prediction.

The cascaded processing engines of the parallel accelerators increase the critical path delay and hardware resources. In addition, it accelerates only the processing of certain frequent SEs, and the number of bin(s)/cycle varies. Therefore, ***pipeline accelerators*** were proposed with the prime goal of achieving the real-time performance for HD video. However, the pipeline hazards appear as a byproduct of pipelining, due to the tight dependencies in the CABAC algorithm. There are two pipeline hazards in CABAC: data and structural. A data hazard occurs when the same context model is used for the next bin as for the current bin (read after write). A structural hazard occurs when the context memory is accessed at the same time due to the context model write for the current bin and context model read for the next bin. These hazards cause the pipeline stalls that decrease the throughput from the maximum of 1 bin/cycle to a lower value.

Zheng *et al.* [12] proposed a two-stage pipeline decoding architecture for residual data only. The stalls in the pipeline are eliminated using standard look ahead (SLA) technique, to determine the context model for the next bin using both possible values of the current bin. This SLA approach is also used in [13][14]. Yi *et al.* [15] proposed a two-stage pipeline decoding architecture, to reduce the pipeline latency and to increase the throughput. The data hazards are removed using the forwarding approach, and the structural hazards by using a context model reservoir. However, the stalls due to SE switching limit the throughput to an average of 0.25 bin/cycle. This problem is solved in [16] by using a SE predictor, that increases the throughput to 0.82 bin/cycle. Li *et al.* [17] proposed a three-stage dynamic pipeline codec architecture. The pipeline is dynamic as the pipeline latency varies between one and two cycles depending on the bin type. For data hazards removal a pipeline bypass scheme is used and for structural hazards a dual-port SRAM. Tian *et al.* [18] proposed a three-stage pipeline encoding architecture. Two pipeline buffers are introduced to resolve the pipeline hazards and the latency in [17]. This results in the throughput of exactly 1 bin/cycle.

The ***parallel pipeline accelerators*** combine the acceleration features of both approaches, what often results in a super fast accelerator. We could benefit from this approach, if we would be able to process multiple bins in a pipeline fashion without any stall. Although we can not fully utilize this approach, because it will make the accelerator architecture very complex or may even be impossible to design, its limited application is possible by utilizing the characteristics of SEs, like the processing of a single RB with one or more BB in parallel pipeline fashion. This approach drastically improves the throughput which is the requirement of future multimedia systems. Shi *et al.* [14] proposed a parallel pipeline approach for the real-time decoding of HD video with 4-stages that can decode 1RB or 2BB bin(s)/cycle without any stall. Due to the processing of multiple bypass bins in pipeline average throughput of 1.27 bins/cycle is achieved. The decoding rate of 254Mbins/s of this approach is much higher compared to ∼45Mbins/s required for HD1080i video. Structural hazards are solved using two dual-port SRAMs and data hazards using forwarding technique and redundant circuitry.

The configurability and extensibility makes ASIP interesting option for the high-end adaptive applications. ***ASIP-based accelerators*** for CABAC were

Table 1. Comparison of Different Hardware Accelerator Architectures

Design	**Freq.**	**Throughput**	**VLSI Tech.**	**Circuit**	**Resolution**
Approach	**MHz**	**Bin(s)/Cycle**	**TSMC(μm)**	**Area (gates)**	**Support**
Datapath/Control					
[4] Codec	30	0.2	-	80,000(Inc.)*	SD480i
[5] Decoder	200	0.33∼0.5	0.13	138,226(Inc.)	CIF
Parallel					
[3] Decoder	149	1∼3	0.18	0.3mm^2+32x105reg	SD
[9] Encoder	186	1.9∼2.3	0.35^{AMS}	19,426(Exc.)	CIF, HD
[11] Decoder	303	0.41	0.18	-	SD480i
Pipeline					
[12] Decoder	160	1	0.18	46.4K(Inc.)	HD1080i
[15] Decoder	225	0.25/0.82[16]	0.18	81,162+12.18KB	HD1080p
[17] Codec	230	$0.60^{Enc}/0.50^{Dec}$	0.18	0.496mm^2(Inc.)	HD1080i
[18] Encoder	186	1	0.35^{AMS}	19.1K(Exc.)	-
Parallel pipeline					
[14] Decoder	200	1.27	0.18	28,956+10.81KB	HD1080i
ASIP/ISE					
[20] Decoder	120	0.021/0.028**	-	-	-

**Context Memory included in the area calculation **LPS/MPS bins.*

proposed in [19][20], but they do not satisfy the real-time requirements, e.g. in [20] 36 and 48 cycles are consumed in MPS and LPS bins decoding, respectively.

4 Conclusion

In this paper, we reviewed numerous approaches for the hardware accelerator architectures for CABAC from the viewpoint of the hardware acceleration concepts and performances. The straightforward architecture usually en-/decode from 0.2 to 0.5 bin/cycle, as shown in Table 1. Since the SEs are processed in a sequential manner, no substantial speed up is achieved. In the parallel approach the number of bins/cycle depends on the type of SE and fluctuates mostly between 1 and 4 bin(s)/cycle. In the purely pipeline approach the throughput never goes above 1 bin/cycle, but independent of SEs it remains at 1 or close to 1 bin/cycle. In the parallel pipeline approach, some extra performance is obtained from the characteristics of SEs, that enables to process some bins in parallel, and results in average throughput of more than 1 bin/cycle for HD video. This can be further improved, if the processing of one or more regular bin(s) and/or one or more bypass bin(s) is performed in parallel, but with steady and balanced pipeline, simple control and minimal area. From the analysis and comparison it follows that the parallel pipeline accelerator approach seems to be the most promising. However, the computational requirements of the current and future multimedia systems are increasing and require further research on accelerator architectures.

References

1. ITU-T: Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H. 264— ISO/IEC 14496-10 AVC) (May 2003)
2. Marpe, D.a.: Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. IEEE Transactions on CSVT, 620–636 (July 2003)
3. Yu, W., et al.: A high performance cabac decoding architecture. IEEE Transactions on Consumer Electronics, 1352–1359 (November 2005)
4. Ha, V., et al.: Real-time mpeg-4 avc/h.264 cabac entropy coder. In: 2005 Digest of Technical Papers. International Conference on ICCE, pp. 255–256 (January 2005)
5. Chen, J., et al.: A hardware accelerator for context-based adaptive binary arithmetic decoding in H. 264/AVC. In: ISCAS 2005, pp. 4525–4528 (2005)
6. Mei-hua, et al.: Optimizing design and fpga implementation for cabac decoder. In: International Symposium on HDP 2007, pp. 1–5 (June 2007)
7. Bingbo, L., et al.: A high-performance vlsi architecture for cabac decoding in h.264/avc. In: 7th International Conference on ASICON 2007, pp. 790–793 (October 2007)
8. Deprá, D.A., et al.: A novel hardware architecture design for binary arithmetic decoder engines based on bitstream flow analysis. In: SBCCI 2008, pp. 239–244 (2008)
9. Osorio, R.R., et al.: High-throughput architecture for h.264/avc cabac compression system. IEEE Transactions on CSVT, 1376–1384 (November 2006)
10. Pastuszak, G.: A high-performance architecture of the double-mode binary coder for h.264.avc. IEEE Transactions on CSVT, 949–960 (July 2008)

11. Kim, C., et al.: High speed decoding of context-based adaptive binary arithmetic codes using most probable symbol prediction. In: ISCAS 2006, p. 4 (2006)
12. Zheng, J., Wu, D., Xie, D., Gao, W.: A novel pipeline design for h.264 cabac decoding. In: Ip, H.H.-S., Au, O.C., Leung, H., Sun, M.-T., Ma, W.-Y., Hu, S.-M. (eds.) PCM 2007. LNCS, vol. 4810, pp. 559–568. Springer, Heidelberg (2007)
13. Eeckhaut, H., et al.: Optimizing the critical loop in the h.264/avc cabac decoder. In: IEEE International Conference on FPT 2006, pp. 113–118 (December 2006)
14. Shi, B., et al.: Pipelined architecture design of h.264/avc cabac real-time decoding. In: 4th IEEE International Conference on ICCSC 2008, pp. 492–496 (May 2008)
15. Yi, Y., et al.: High-speed h.264/avc cabac decoding. IEEE CSVT, 490–494 (2007)
16. Son, W., et al.: Prediction-based real-time cabac decoder for high definition h.264/avc. In: IEEE International Symposium on ISCAS 2008, pp. 33–36 (May 2008)
17. Li, L., et al.: A hardware architecture of cabac encoding and decoding with dynamic pipeline for h.264/avc. J. Signal Process. Syst., 81–95 (2008)
18. Tian, X.a.: Implementation strategies for statistical codec designs in h.264/avc standard. In: 19th IEEE International Symposium on RSP 2008, pp. 151–157 (June 2008)
19. Flordal, O., et al.: Accelerating cabac encoding for multi-standard media with configurability. In: 20th International IPDPS 2006, p. 8 (April 2006)
20. Rouvinen, J., et al.: Context adaptive binary arithmetic decoding on transport triggered architectures. In: SPIE Conference Series (March 2008)