CABAC accelerator architectures for video compression in future multimedida : a survey

DOI:
10.1007/978-3-642-03138-0_4

Document status and date:
Published: 01/01/2009

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

CABAC Accelerator Architectures for Video Compression in Future Multimedia: A Survey

Yahya Jan and Lech Jozwiak

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

Abstract. The demands for high quality, real-time performance and multi-format video support in consumer multimedia products are ever increasing. In particular, the future multimedia systems require efficient video coding algorithms and corresponding adaptive high-performance computational platforms. The H.264/AVC video coding algorithms provide high enough compression efficiency to be utilized in these systems, and multimedia processors are able to provide the required adaptability, but the algorithms complexity demands for more efficient computing platforms. Heterogeneous (re-)configurable systems composed of multimedia processors and hardware accelerators constitute the main part of such platforms. In this paper, we survey the hardware accelerator architectures for Context-based Adaptive Binary Arithmetic Coding (CABAC) of Main and High profiles 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. The comparative analysis shows that the parallel pipeline accelerator architecture seems to be the most promising.

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, video telephony, camcoders, surveillance, medical imaging, and especially High Definition Television (HDTV) and new emerging Ultra HDTV (UHDTV) in video broadcasting domain, demand for highly efficient computational platforms. The problem is amplified by the quickly growing requirements of higher and higher quality, especially 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)$\sim$ 33Megapixel with a data rate of 24Gbps. Additionally, the latest standards video coding algorithms are much more complex due to the digital multimedia convergence and specifically access of multimedia through a variety of networks and different coding formats used by a single device, as well as, the slow vanishing of the old video coding standards (e.g. MPEG-2) and widespread adaptation of the new standards (e.g. H.264/AVC, VC1 etc). 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.

Despite of spectacular advances in microelectronic industry, the future multimedia systems cannot be realized using the conventional processor architectures or the existing multimedia processors. They require highly efficient specialized hardware architectures to satisfy the stringent functional and non-functional requirements, and be flexible enough to support multiple domains, standards and modes, and have to be implemented with SoC platforms involving embedded (re-)configurable hardware. In particular, (re-)configurable hardware accelerators are indispensable for the development of these specialized and demanding systems, as well as, new design and design automation methodologies to support development of such accelerators.

H.264/AVC [1] is the latest multi-domain video coding standard that provides the compression efficiency of almost 50% higher than former standards (e.g. MPEG-2) due to its advance coding tools. However, its computational complexity is about four times higher compared to its predecessors, and induces the necessity of the real-time video coding through a sophisticated dedicated hardware design.

H.264/AVC supports two entropy coding modes: Context Adaptive Variable Length Coding (CAVLC) and Context-based Adaptive Binary Arithmetic Coding (CABAC) [2]. CAVLC covers Baseline profile of H.264/AVC for low-end applications, like video telephony, while CABAC targets Main and High profiles for high-end applications, like HDTV. CABAC improves the compression efficiency 9%-14% as compared to CAVLC at the cost of an increase in complexity of 25-30% and 12% for encoding and decoding, respectively, in terms of access frequency [2][3][4]. Its purely software based implementation results in an unsatisfactory performance even for a low quality and resolution video (e.g. 30-40 cycles are required on average for a single bin decoding on DSP [3]). The situation is much worse for High Definition (HD) video as the maximum bin rate requirement of HD (level 3.1 to 4.2) in H.264/AVC, averaged across a coded picture, ranges from 121 Mbins/s to 1.12 Gbins/s [5]. This makes the software based implementation inadequate to achieve the real-time performance for HD video as a multi-giga hertz RISC processor would be required for HD encoding in real-time [6]. Moreover, the serial nature of CABAC paralyzes the other processes in video codec that could be performed in parallel, making CABAC a bottleneck in the overall codec performance. Consequently, to achieve the required performance, flexibility, low cost and low energy consumption, a sophisticated (re-)configurable 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. Numerous research groups from academia and industry all over the world have proposed different hardware architectures for CABAC using different hardware acceleration concepts and schemes. Our work reported in this paper is performed in the framework of a research project that aims to develop an adequate design methodology and propose supporting EDA tools for development of demanding (re-)configurable hardware accelerators.

This paper surveys several most interesting recently proposed hardware accelerator architectures for CABAC. Its main purpose is to deliver a critical insight in the proposed hardware accelerator solutions, and this way facilitate our own and other researchers further work on (re-)configurable accelerator architectures for future complex multimedia applications, 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 in the hardware accelerator domain, like throughput, frequency, resource utilization and power consumption. Based on the critical architecture comparisons 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 and classification. Using them, Section 4 presents a critical review of hardware accelerator architectures for CABAC, comparison of various architectures and discusses some promising design directions. Section 5 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 reduction of the SE alphabet size to binary in binarization not only minimizes the complexity of arithmetic coder, but also enables the subsequent context modeling stage to more efficiently model the statistical behavior of the syntax elements (SEs). Four basic binarization schemes are used in CABAC [2].

The ***context modeling*** process determines the probabilities of the bins using pre-defined context (probability) models, before they are encoded arithmetically. The context models are selected taking into account the neighboring information of the bins/SEs referred to as context. CABAC defines 460 unique context models, each of which correspond to a certain bin or several bins of a SE, and are

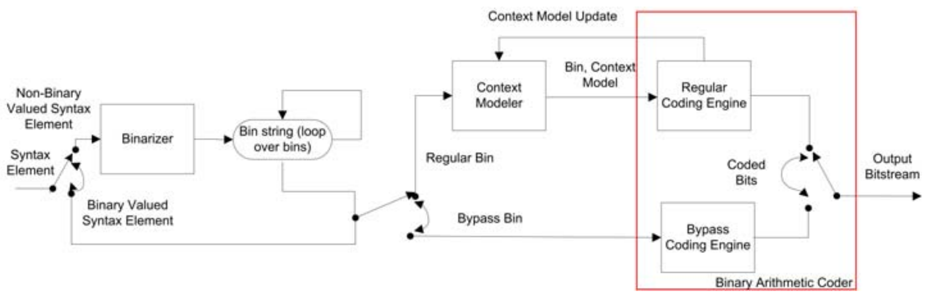

Fig. 1. Block Diagram of CABAC Encoder

updated after bin encoding in order to adopt the models to the varying statistics of the video data. Each context model comprises of the 6-bit probability state index (pStateIdx) and the most probable symbol (MPS) value of the bin [2].

CABAC utilizes the table-based binary arithmetic coder [7] to avoid the costly multiplication process in probability calculation. The ***binary arithmetic coding*** engine consists of two sub-engines: regular and bypass, as shown in Figure 1. The regular coding engine utilizes adaptive probability models, but the bypass coding engine assumes a uniform probability model to speed up the encoding process. To encode a bin, the regular coding engine requires the probability model (pStateIdx, MPS) and the corresponding interval range (width) R and base (lower bound) L of the current code interval. The interval is then divided into two subintervals according to the probability estimate (ρ_{LPS}) of the least probable symbol (LPS). Then, one of the subintervals is chosen as the new interval based on whether the bin is equal to MPS or LPS, as given in the following equations [2].

$$R_{new} = R - R_{LPS} \quad \text{and} \quad L_{new} = L \quad \text{if} \quad bin = MPS \tag{1}$$

$$R_{new} = R_{LPS} \quad \text{and} \quad L_{new} = L + R - R_{LPS} \quad \text{if} \quad bin = LPS \tag{2}$$

where $R_{LPS} = R \,.\, \rho_{LPS}$ represents the size of the subinterval associated with the LPS. The probability 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. In bypass encoding the probability estimation and update processes are bypassed, because uniform probability is assumed for a bypass bin.

3 Main Concepts of Hardware Acceleration

Hardware accelerator is an application-specific hardware sub-system that can implement a given function more effectively and efficiently than in software running on a conventional processor. A good example is the graphic accelerator. 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 data parallelism the multiple data instance of the same type are processed in parallel, provided the application allows and the resources are available. The functional parallelism simultaneously performs different operations on (possibly) different data instances. Also, the speculative execution can be used to enable for more parallelism. To design a high quality hardware accelerator, it is necessary to perform a thorough analysis of the application algorithms and exploit specific computational characteristics inherent to these algorithms. Dependent on the different characteristics discovered and accounted for result in different approaches to the design of hardware accelerators, and therefore, in the past a number of different basic architecture types were proposed:

- Straightforward datapath/controller hardware architecture
- Parallel hardware architecture
- Pipeline hardware architecture
- Parallel pipeline hardware architecture
- General purpose processor (GPP) augmented by loosely coupled hardware accelerator resulting from the HW/SW co-design approach
- 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.

4 Overview of Hardware Accelerators for CABAC

The accelerator architectures are analyzed here in a systematic conceptual way, when studying the computational characteristics of CABAC, and thus differently than in the sporadic fragmentary comparisons that can be found in the literature. Moreover, we focus on the main problems and solutions that drastically effect the achieved results. We will not actually consider the mixed HW/SW solution for CABAC, i.e. accelerated GPP augmented by loosely coupled hardware accelerator, because this option is not promising regarding the real-time requirements satisfaction due to the strong dependencies in the computations and the resultant high communication overhead. The performance of different approaches is analyzed and the results are compared, when focusing on the throughput, maximum frequency and area. In almost all of the reviewed papers no systematic analysis is provided or methods proposed on how to integrate the CABAC accelerator in a complete H.264/AVC en-/decoder.

Before considering the accelerator architectural approaches, 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 hampers 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 and a variable number of cycles, the next bin processing cannot be initiated, because the probability estimation of the next bin depends on the current interval range. These strong dependencies are some of the main challenges in the accelerator design, and a number of solutions are proposed to tackle these problems.

4.1 Straightforward Datapath/Controller Accelerators

The straightforward datapath/controller approach relies on the data flows in the algorithm of the software based implementation. This accelerates the computations to some degree, but does not exploit the true (parallel) nature of the application algorithm and improvement achievable using the hardware acceleration approach. This approach is followed in some CABAC accelerators, in the sense that processing is performed sequentially on a per bin basis, and possibilities are not explored for a multi-bin parallel processing. This always limits the performance to maximally 1 bin/cycle, as the simple serial hardware implementation without any optimizations takes as many as 14 cycles to encode a single bin [8]. Some optimization technique like pre-fetching and simple parallelism [8] etc. were proposed that enables to process one bin in 5 cycles. Chen *et al.* [9] proposed an FSM and a memory scheme for neighboring SEs, which results in the decoding throughput of 0.33∼0.50 bin/cycle. However, it decodes only CIF video at 30fps.

4.2 Parallel Hardware Accelerators

The inefficiency of the straightforward acceleration approaches to en-/decode in real-time HD video motivated the research community to exploit some alternative approaches to the design of CABAC accelerators. The most promising approach to achieve real-time performance for high resolution video is to process more than one bin/cycle, i.e. to utilize a parallel approach. However, in the en-/decoding of even a single bin complex interdependencies have to be resolved as discussed before, and consequently, the algorithm cannot be parallelized in its true basic nature. Utilizing the static and dynamic characteristics of the SEs that can be discovered through an analysis of CABAC algorithm for real video sequences, the parallelism can be achieved up to some level for some specific SEs, what can result in processing of more than one bin/cycle. However, in parallel en-/decoding of two or more regular bins the context models have to supplied to the coding engines. Due to the blocking dependencies, this cannot be performed in parallel. Also the context model fetching takes a substantial time. The details of these characteristics of SEs and the corresponding parallel schemes are discussed below.

Yu *et al.* [3] proposed the first parallel architecture for CABAC decoding. Unlike the conventional approaches [8][9] that take a number of cycles to decode a single bin, this architecture decodes 1$\sim$3 bin/cycle. The parallelism in this architecture 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. To reduce the context memory accesses, relevant context models of a SE or group of SEs are accessed in blocks and are stored in a high speed register bank. However, it results in an extra cost of the register bank. The architectures [10][11][12][13][14] are based on the same concept, but after some specific extensions are capable to en-/decode HD video. In [15] sixteen cascaded regular decoding units are used for more speed up for frequent SEs. However, due to dependencies the throughput remains less than 1 bin/cycle, and it causes an increase in the critical path latency and circuit area. In [16] five different architectures for CABAC encoder are designed and analyzed for area/performance tradeoff. Results show that two regular with bypass bins architectures perform better for the high quality video than the others.

A predictive approach is employed by Kim *et al.* [17]. Unlike the architectures [3][10][11][12][14][15], in which there is a latency due to the cascaded arithmetic coding engine, this architecture initiates decoding of two bins simultaneously by prediction. However, due to mis-prediction only 0.41 bin/cycle is achieved, although with a high frequency of 303MHz.

Algorithmic optimizations can expose far more parallelism than available in the original application algorithm. A novel algorithm is proposed by Sze *et al.* [5] which is fundamentally parallel in nature and deterministically en-/decode several (N) bins with different context at the same time. The context models for different bins are determined simultaneously using conditional probabilities, what is different than in the predictive strategy [17] and the cascaded approaches [3][10][11][12][14][15]. The two possible context models that could be used for the second bin are determined by taking into account the two possible values of the first bin (0 or 1). Its software implementation (N=2) enables 2 bins/cycle at a cost of 0.76% increase in bit rate compared to the original CABAC algorithm. However, such optimizations require 3 to 4 multiplications for two bins en-/decoding as well comparators, which could make its hardware implementation costly in resources.

4.3 Pipeline Hardware Accelerators

Although, the parallelism in the form of multi-bin processing in CABAC outperforms the conventional approach, it increases the complexity of the architecture, specially in the renormalization and context management. The cascaded multiple processing engines also increase the critical path delay. Moreover, the hardware resources are much increased with not much gain from acceleration [4]. In addition, the multi-bin processing only accelerates the decoding of certain frequent SEs, is not equally well effective for all SEs, and the number of cycles per bin processing varies. Therefore, the pipeline concept of hardware acceleration is also utilized in CABAC, with the prime goal of achieving the real-time performance for HD video. A number of pipeline schemes are proposed to effectively overcome the problems and complexities of other schemes discussed earlier. 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, which is a read after write (RAW) data hazard. 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 of the purely pipelined architecture from the maximum of 1 bin/cycle to a lower value. Below the details of the pipeline schemes, solutions for pipeline hazards and performance of proposed pipeline accelerators are discussed.

Zheng *et al.* [18] proposed a two stage pipeline decoding architecture for residual SEs 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. The proposed architecture supports HD1080i video. This SLA approach is also used in pipeline architectures [19][20][21]. Yi *et al.* [22] proposed a two stage pipeline decoding architecture, instead of 4 usual stages, to reduce the pipeline latency and to increase the throughput. The data hazard are removed using the forwarding approach, and the structural hazards by using a context model reservoir (CMR) with context memory. However, the SE switching causes stalls due to CMR update, and this limits the throughput to an average of 0.25 bin/cycle. This problem is solved in [23] by using a SE predictor that increases the throughput to 0.82 bin/cycle.

Li *et al.* [4] proposed a three stage dynamic pipeline codec architecture. The pipeline is dynamic in the sense that the pipeline latency varies between one and two cycles depending on the bin type. No pipeline stalls occur for the BB and the RB of value MPS with the interval range (R) in its limit. For data hazards removal a pipeline bypass scheme is used and for structural hazards a dual-port SRAM. The bin processing rate of [18][23] is higher than of [4], because of the coarse pipeline stages with efficient context management. Tian *et al.* [24] proposed a three stage pipeline encoding architecture. Two pipeline buffers are introduced to resolve the pipeline hazards and the latency issue of [4], what results in the throughput of exactly 1 bin/cycle. Chang [25] proposed a three stage pipeline architecture that combines together the different speed up methods earlier proposed like: pipeline stalls reduction due to SEs switching, context model clustering for decreasing context memory access, and two-bin arithmetic decoding engine. The architecture achieves the average throughput of 0.63 bin/cycle at a comparatively high frequency, as shown in Table 1.

Table 1. Comparison of Different Hardware Accelerator Architectures

Design Approach	**Freq. MHz**	**Throughput Bin(s)/Cycle**	**VLSI Tech. TSMC(μm)**	**Circuit Area (gates)**	**Resolution Support**
Datapath/Control					
[8] Codec	30	0.2	Virtex-II	80,000(Inc.)*	SD480i@30fps
[9] Decoder	200	0.33∼0.5	0.13	138,226(Inc.)	CIF@30fps
Parallel					
[3] Decoder	149	1∼3	0.18	0.3mm^2+32x105reg	SD480@30fps
[14] Encoder	186	1.9∼2.3	0.35^{AMS}	19,426(Exc.)	CIF, HD
[15] Decoder	45	<1	0.18	42,000(Exc.)	HD1080i@30fps
[17] Decoder	303	0.41	0.18	-	SD480i@30fps
Pipeline					
[4] Codec	230	$0.60^{Enc}/0.50^{Dec}$	0.18	0.496mm^2(Inc.)	HD1080i@30fps
[18] Decoder	160	1	0.18	46,400(Inc.)	HD1080i@30fps
[22] Decoder	225	0.25/0.82**[23]**	0.18	81,162+12.18KB	HD1080p@25fps
[24] Encoder	186	1	0.35^{AMS}	19,100(Exc.)	-
[25] Decoder	250	0.63	0.18	35,615(Exc.)	HD1080p@30fps
Parallel pipeline					
[21] Decoder	200	1.27	0.18	28,956+10.81KB	HD1080i@30fps
ASIP/ISE					
[31] Decoder	120	0.021/0.028**	-	-	-

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

4.4 Parallel Pipeline Hardware Accelerators

The parallel pipeline schemes combine the acceleration features of both approaches, what often result 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 cannot fully utilize this approach, because it will make the accelerator architecture very complex or may even be impossible to design, its limited practical application is possible by utilizing the characteristics of SEs, like the processing of a single regular bin with one or more bypass bins in parallel pipeline fashion. This approach drastically improves the throughput which is the requirement of the future high quality and resolution systems.

Shi *et al.* [21] 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. Structural hazards are solved using two dual-port SRAMs and data hazards using forwarding technique and redundant circuitry. Two bypass bins are processed in parallel with no pipeline stalls due to switching from the regular to bypass mode and back to regular mode, what makes this architecture unique. Due to the processing of multiple bypass bins in pipeline average throughput of 1.27 bins/cycle is achieved.

4.5 ASIP/ISE Based CABAC Accelerators

The configurability and extensibility makes ASIP interesting option for the high-end adaptive applications. The extensibility in the form of ISE could be used to cope with evolving standards and results in an efficient real-time processing for high resolution video applications.

Flordal *et al.* [26] proposed a multi-standard (JPEG2000, H.264) CABAC encoder. A Multi-branch instruction is proposed here for the renormalization in CABAC. Unfortunately, this approach results in unsatisfactory performance for HD video. The work of Osorio *et al.* [27] is also in this direction, but it is based on an array of simple processors implementing the different tasks of various entropy en-/decoders (e.g for CABAC 5 processors array). Comparative study with Texas Instruments TMS320C6711 VLIW DSP shows only results for QCIF resolution video, however no real figures are given for HD video.

Nunez *et al.* [28] extended the ISA of SPARC-compatible Leon CPU with 7 instructions just to integrate the CABAC in H.264/AVC encoder. The CABAC algorithm is implemented without context management and binarization as a single hardware accelerator unit in a pipeline style. Similarly, in [29] two new instructions are proposed for Trimedia TM3270 media-processor that accelerate only the arithmetic coding part of the CABAC, which supports only D1 resolution video. Tensilica 388VDO and Silicon HiveFlex VSP2500 video processors also utilizes specific instruction set extensions for CABAC implementation that support D1 and HD resolution videos, respectively. Since Multiprocessor SoC (MPSoC) are becoming more and more popular in accelerating the back-end of H.264/AVC. Osorio *et al.* [30] proposed a novel microprogrammed CABAC decoder for MPSoC based H.264/AVC Codec. Rouvinen *et al.* [31] utilize the Transport Triggered Architecture (TTA) for implementation of CABAC. Easy customization of TTA resources and programmable visible interconnect structure give many possibilities for the designer to get an optimized ASIP solution. Nine transport buses with other special functional units (SFU) are proposed for according to the CABAC requirement. However, 36 and 48 cycles are consumed in the MPS and LPS bins decoding, respectively. This solution is thus not suitable for HD applications but its flexibility favors multi-standard codec design. Most of the ASIP/ISE approaches discussed so far do not support HD video. We conclude that for HD video any ASIP/ISE other than implementing the whole coarse-grain CABAC accelerator as a single instruction seems to be less effective due to the introduction of extra clock cycles for each instruction fetch.

4.6 Comparison

Most of the results are compared in the previous sections along with the architecture discussion. However, the overall view is also important. The straightforward approach 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 fluctuates between a certain maximum and minimum, as it depends on the type of SE. It may result in 2, 3 or even 4 bins/cycle if supported by the architecture as in [12][14]. In the purely pipeline approach the throughput never goes above more than 1 bin/cycle, but independent of SEs it remains at 1 or close to 1 bin/cycle. However, in the parallel pipeline approach some extra performance is obtained from the characteristics of the SEs that enable to process some bins in parallel, like in reference [21] for the bypass bins. This result in average throughput of more than one bin/cycle for HD video. The decoding rate of 254Mbins/s at operating frequency of 200MHZ of this approach is much higher than ∼45Mbins/s required for HD1080i 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 to maintain the throughput consistently, simple control and minimal area. Also, the parallel approach seems to be more effective in case of CABAC encoding due to the availability of next bin(s) of a SE. However, in CABAC decoding the next bin information is available only after the processing of the current bin, so pipelined architectures perform better. These kind micro-architectural decisions could be employed in high-level synthesis (HLS) tools to automate the design of such complex accelerators.

5 Conclusion

In this paper, we reviewed numerous approaches to the hardware accelerator architectures for CABAC from the viewpoint of the hardware acceleration concepts and performances. The features and issues involved in each architectural approach are discussed with focus on the real-time, high resolution and high quality video processing capabilities. From the analysis and comparisons it follows that the parallel pipeline accelerator approach seems to be the most promising, because of high and steady throughput, simple control and hardware efficiency as compared to other architectures. However, the computational requirements of the current and future multimedia systems are ever increasing and require further research on accelerator architecture concepts, as well as adequate design methodologies and EDA tools for the development of 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. 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)
5. Sze, V., et al.: Parallel cabac for low power video coding. In: 15th IEEE International Conference on ICIP 2008, October 2008, pp. 2096–2099 (2008)
6. Shojania, et al.: A high performance cabac encoder. In: NEWCAS, pp. 315–318 (2005)
7. Marpe, D., et al.: A highly efficient multiplication-free binary arithmetic coder and its application in video coding. In: ICIP 2003, September 2003, pp. 263–266 (2003)
8. Ha, V., et al.: Real-time mpeg-4 avc/h.264 cabac entropy coder. In: 2005 Digest of Technical Papers. In: International Conference on ICCE, January 2005, pp. 255–256 (2005)

9. Chen, J.W., et al.: A hardware accelerator for context-based adaptive binary arithmetic decoding in h.264/avc. In: ISCAS 2005, May 2005, pp. 4525–4528 (2005)
10. Mei-hua, et al.: Optimizing design and fpga implementation for cabac decoder. In: International Symposium on HDP 2007, June 2007, pp. 1–5 (2007)
11. Bingbo, L., et al.: A high-performance vlsi architecture for cabac decoding in h.264/avc. In: 7th International Conference on ASICON, October 2007, pp. 790–793 (2007)
12. Deprá, D.A., et al.: A novel hardware architecture design for binary arithmetic decoder engines based on bitstream flow analysis. In: SBCCI, pp. 239–244 (2008)
13. Jian, et al.: A high-performance hardwired cabac decoder. In: ICASSP 2007, pp. 37–40 (2007)
14. Osorio, R.R., et al.: High-throughput architecture for h.264/avc cabac compression system. IEEE Transactions on CSVT, 1376–1384 (November 2006)
15. Zhang, P., et al.: High-performance cabac engine for h.264/avc high definition real-time decoding. In: International Conference on ICCE 2007, January 2007, pp. 1–2 (2007)
16. Pastuszak, G.: A high-performance architecture of the double-mode binary coder for h.264.avc. IEEE Transactions on CSVT, 949–960 (July 2008)
17. 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)
18. Zheng, J., et al.: 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)
19. Eeckhaut, H., et al.: Optimizing the critical loop in the h.264/avc cabac decoder. In: IEEE International Conference on FPT 2006, December 2006, pp. 113–118 (2006)
20. Yang, Y.C., et al.: A high throughput vlsi architecture design for h.264 cabac decoding with look ahead parsing. In: Multimedia and Expo., pp. 357–360 (2006)
21. Shi, B., et al.: Pipelined architecture design of h.264/avc cabac real-time decoding. In: 4th IEEE International Conference on ICCSC 2008, May 2008, pp. 492–496 (2008)
22. Yi, Y., et al.: High-speed h.264/avc cabac decoding. IEEE CSVT, 490–494 (2007)
23. Son, W., et al.: Prediction-based real-time cabac decoder for high definition h.264/avc. In: IEEE International Symposium on ISCAS 2008, May 2008, pp. 33–36 (2008)
24. Tian, X.a.: Implementation strategies for statistical codec designs in h.264/avc standard. In: 19th IEEE International Symposium on RSP, June 2008, pp. 151–157 (2008)
25. Chang, Y.T.: A novel pipeline architecture for h.264/avc cabac decoder. In: IEEE Asia Pacific Conference on APCCAS 2008, December 2008, pp. 308–311 (2008)
26. Flordal, O., et al.: Accelerating cabac encoding for multi-standard media with configurability. In: 20th International IPDPS 2006, April 2006, p. 8 (2006)
27. Osorio, R.R., et al.: Entropy coding on a programmable processor array for multimedia soc. In: International Conference on ASAP 2007, July 2007, pp. 222–227 (2007)
28. Nunez, et al.: Design and implementation of a high-performance and silicon efficient arithmetic coding accelerator for the h.264 video codec. In: ASAP 2005, pp. 411–416 (2005)
29. van de Waerdt, J.W., et al.: The tm3270 media-processor. In: 38th IEEE/ACM International Symposium on Microarchitecture 2005, pp. 331–342 (2005)
30. Osorio, R.R., et al.: An fpga architecture for cabac decoding in manycore systems. In: International Conference on ASAP 2008, July 2008, pp. 293–298 (2008)
31. Rouvinen, J., et al.: Context adaptive binary arithmetic decoding on transport triggered architectures. In: SPIE Conference Series (March 2008)