A 2014 MBIN/S DEEPLY PIPELINED CABAC DECODER FOR HEVC

Yu-Hsin Chen, Vivienne Sze

Massachusetts Institute of Technology, Cambridge, MA 02139

ABSTRACT

High Efficiency Video Coding (HEVC) is the latest video standard that specifies video resolutions up to 8K Ultra-HD (UHD) at 120 fps to support the next decade of video applications. This results in high throughput requirements for the Context Adaptive Binary Arithmetic Coding (CABAC) entropy decoder, which was already a well-known bottleneck in H.264/AVC. Several modifications were made to the HEVC CABAC to address the throughput challenges. This work leverages these improvements in the design of a high throughput HEVC CABAC decoder. The proposed design uses a deeply pipelined architecture to achieve a high clock rate. Additional techniques such as state prefetch logic, latched-based context memory, and separate finite state machines are applied to minimize stall cycles, while multi-bypass bins decoding is used to further increase the throughput. The design is synthesized in a IBM 45nm SOI process, and achieves throughputs up to 2014 and 2748 Mbin/s under common and worst-case test conditions, respectively, at 1.9 GHz operating frequency. The results show that the design is sufficient to decode video bitstreams in real-time at Level 6.2, or at Level 6.0 for applications requiring sub-frame latency.

Index Terms— CABAC, High Efficiency Video Coding (HEVC), H.265, Video Compression

1. INTRODUCTION

High Efficiency Video Coding (HEVC), developed by the Joint Collaborative Team on Video Coding (JCT-VC) as the latest video compression standard, was approved as an ITU-T/ISO standard in early 2013 [1]. HEVC achieves 2× higher coding efficiency than it's predecessor H.264/AVC, and supports resolutions up to 4320p, or 8K Ultra-HD (UHD) [2]. HEVC uses Context Adaptive Binary Arithmetic Coding (CABAC), a form of entropy coding, to achieve high coding efficiency [3]. However, CABAC is a well-known throughput bottleneck in H.264/AVC codecs, particularly in the decoder due to the highly serial dependencies caused by several feedback loops within the decoding flow. Efforts have been made to revise the CABAC in HEVC with many throughput-aware improvements, such as reduced memory requirements, reduced context-coded bins and grouping of bypass bins [4]. This work will describe an architecture that fully leverages these features to achieve a high-throughput CABAC decoder.

Previous works on the CABAC decoder, mostly for the H.264/AVC standard, attempt to expose the parallelism within the fixed algorithms. On one hand, pipelining is an effective way of extending parallelism at the temporal domain. However, tight feedback loops at the bin level make the pipelined architecture suffer from an excessive number of stalls [5, 6]. On the other hand, multi-bin per cycle decoding explores the parallelism by adding additional decoding logic. Many designs decode up to two bins per cycle [7, 8, 9, 10, 11], but this comes at the cost of decreased clock rate. Prediction-based decoding can be used in the multi-bin case to save the hardware overhead of prefetching decoding information for the extra bins [11, 12], though it also lowers the throughput due to prediction miss penalty and extra critical path delay.

This work proposes an architecture for the CABAC decoder in HEVC with the goal of achieving the highest possible throughput in terms of bins per second. Section 2 introduces the techniques used to exploit the parallelism for a high-throughput decoder. Specifically, it will describe a deeply pipelined architecture employing a multi-bypass bins decoding scheme that incorporates features such as the state prefetch logic, latch-based context memory and separate finite state machines to minimize stalls. Section 3 presents the experimental and analytical results for the average and worst-case conditions, respectively. The synthesized throughput, area and power will also be reported.

2. PROPOSED CABAC DECODER ARCHITECTURE

In this section, we discuss several architectural techniques that are motivated by the high-throughput features of CABAC in HEVC. These techniques seek to increase two key factors, namely clock rate (cycles per second) and average number of decoded bins per clock cycle, to realize a high-throughput CABAC decoder measured in bins per second.

2.1. Architecture Overview

Figure 1 illustrates the block diagram of the proposed CABAC decoder architecture. The bitstream parser (BP) buffers the incoming bitstream and feeds the arithmetic decoder (AD) according to the AD decoding mode. There are four decoding modes, which invoke four different decoding processes: context-coded bin decoding (CTX), bypass bin decoding (BPS), multi-bypass bins decoding (mBPS) and terminate bin decoding (TRM). With the request of a decoding process,

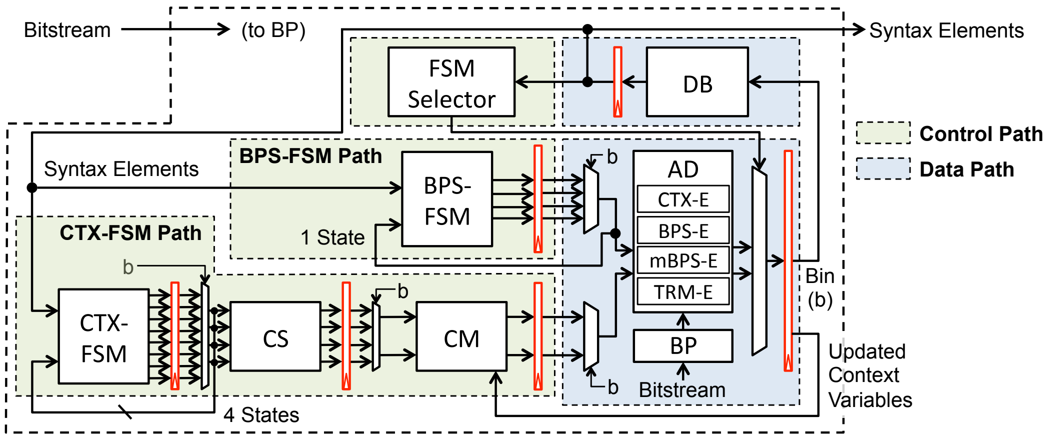


Fig. 1: The block diagram of the CABAC decoder for HEVC. Red blocks represent the stage registers used for deep pipelining.

the corresponding decoding engine (CTX-E, BPS-E, mBPS-E or TRM-E) would be activated to perform the operation. The decoded bins are then reassembled at the de-binarizer (DB) into syntax elements. The rest of the decoder is responsible for gathering decoding information for AD. First, the decoding mode at each cycle is determined by two finite state machines, BPS-FSM and CTX-FSM, according to the HEVC-compliant decoding syntax. Only one out of the two FSMs controls AD within a single cycle, which is decided by the FSM Selector based on previously decoded syntax elements. In addition, if the decoding mode invokes the CTX process, the estimation of bin probability as modeled by the context variables (CVs) is also required. CVs are initialized and stored in the context memory (CM), and the required one for decoding is retrieved by the context selector (CS). CS is only controlled by CTX-FSM. After the CTX process, the updated CV is written back to CM for future access.

Among the decoding engines in AD, CTX-E dominates the decoding complexity and contains the critical path of AD. Therefore, we optimize CTX-E using the techniques introduced in [13] for a 22% reduction in delay.

2.2. Deep Pipelining with State Prefetch Logic

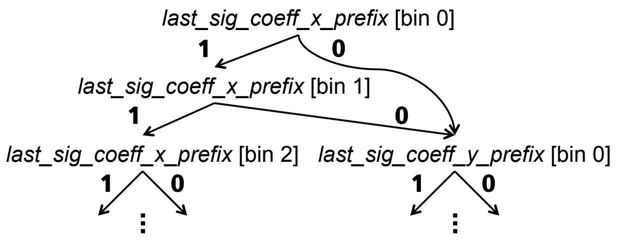
The architecture discussed in Section 2.1 is pipelined for high-throughput decoding. The pipeline stages are shown in Figure 1, in which the red blocks represent the stage registers. The function blocks are divided into two groups, the data path and the control path. The data path consists of two pipeline stages: AD and DB. The control path is further divided into two sub-paths based on the two FSMs. The BPS-FSM path only has one stage that controls the data path directly, while the CTX-FSM has a deeper three-stage pipeline, including CTX-FSM, CS and CM. The overall design is a deeply pipelined five-stage structure. If the next decoder state depends on the decoded syntax element of the current state, this architecture could impose up to four stall cycles.

State prefetch logic is introduced to eliminate the majority of stalls imposed by the tight dependencies described above. Figure 2a shows an example of how the prefetch logic works. Based on the binary value of the decoded bin at the current decoder state, there are two choices for the next states. As in the case of the syntax element *last_sig_coeff_x_prefix*, if its first bin is 1, the decoding will continue to its second bin; otherwise, the decoding will jump to the first bin of *last_sig_coeff_y_prefix*. The next state logic of the FSM will prefetch both of the possible states, and the decision is delayed until the bin being decoded. Following this manner, the FSM logic becomes a binary decision tree (BDT). The construction of the BDT is a trade-off between number of states and number of stalls. If the BDT is fully expanded, the number of states would grow exponentially that its logic delay becomes part of the critical path and affects throughput. To balance between the two aspects, the BDT is optimized to eliminate most of the throughput-critical stalls while keeping the number of states to a minimum. Figure 2b shows another case where the stalls are kept. The FSM stalls until the position of the last significant coefficient being decoded, so it can select the CVs for syntax elements *sig_coeff_flag* without fully expanding the BDT, which saves dozens of states.

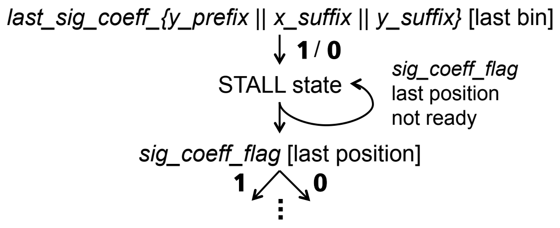
The number of possible next states at each pipeline stage grows exponentially with the depth of the pipeline. To resolve one state at the AD stage for decoding requires CTX-FSM to compute next eight possible states at every cycle since it occurs three stages before AD. BPS-FSM, though only has the control path depth of one, still needs to compute next four possible states due to the multi-bypass bins decoding scheme, which will be discussed in Section 2.5. At each pipeline stage, the bins decoded by AD at previous cycle are used to select the correct inputs states from all input states, as shown by the mux at the beginning of stage CS, CM and AD in Figure 1.

2.3. Latch-Based Context Memory

The state prefetch logic addresses the stall issue for the deeply pipelined architecture; nevertheless, it also brings higher


last_sig_coeff_x_prefix [bin 0]
1
0
last_sig_coeff_x_prefix [bin 1]
1
0
last_sig_coeff_x_prefix [bin 2]
last_sig_coeff_y_prefix [bin 0]
1
0
1
0

(a)


last_sig_coeff_{y_prefix || x_suffix || y_suffix} [last bin]
1 / 0
STALL state
sig_coeff_flag last position not ready
sig_coeff_flag [last position]
1
0

(b)

Fig. 2: Parts of the binary decision tree (BDT) for the FSM prefetch logic. (a) A fully expanded BDT avoids the need for stall cycles. (b) Stalls are kept to save dozens of extra states.

computational requirements to the hardware. This concern is most significant for CM, where two context variables need to be prefetched as CM occurs in the stage before AD. Conventional architectures use SRAM for low area and low power memory access. Some works [5, 8] implement extra caches to achieve multiple reads and writes within the same cycle. But these designs cannot support truly random high-bandwidth memory access as required by the state prefetch logic. HEVC has 3× fewer contexts than H.264/AVC. Thus, the required CM space reduces to 1 kbit, which makes the all-cache CM implementation a more practical option. An all-cache implementation is also beneficial for Wavefront Parallel Processing (WPP), a high level parallelism tool introduced in HEVC. WPP requires the CABAC decoder to replicate its context states for parallel processing, and an all-cache CM can shorten the latency of this process.

Latch-based cache requires smaller area and consumes less power when compared to register-based cache. To ensure glitch-free latch access, Figure 3 demonstrates the design of the latch enable signal. The signal coming into the enable (EN) port of the latch is constrained to settle within the first half of the clock; thus only half of the clock cycle is available for latch updates. This is not an issue for the deeply pipelined architecture, as both the write data (wData) and write address (wAddr) signals come from the pipeline stage registers, and the timing requirement can be easily met.

Table 1 lists the comparison of area and power between all three possible memory implementations. The latch-based design takes only 15% more area than SRAM, but reduces power consumption by 1.7×.

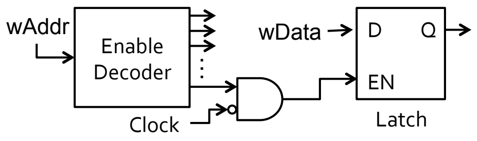


Fig. 3: Glitch-free latch enable design.

Memory Design	Area (eq. gate count)	Power (mW)
SRAM	11.7K	8.10
Register-based	20.2K	13.10
Latch-based	13.4K	4.68

Table 1: Comparison between different types of 1kbit memory implementation for CM. The power is measured under a 2GHz clock and 100% memory access pattern.

2.4. Separate FSM for Bypass Bins

HEVC has fewer context-coded bins than H.264/AVC, resulting in a larger proportion of bypass bins. In addition, most of the bypass bins are grouped together to reduce the amount of switching between bypass bins and context-coded bins. These observations lead to the design of separate finite state machines (FSMs) for context-coded bins (CTX-FSM) and grouped bypass bins (BPS-FSM). The FSM Selector is used to select the FSM that should be enabled for each cycle. CTX-FSM can handle all decoding modes except the mBPS mode, while BPS-FSM only operates for the grouped bypass bins with the BPS or mBPS mode. BPS-FSM does not manage all bypass bins, however, as that complicates the logic in the FSM Selector to switch between FSMs frequently, creating a longer critical path. As discussed in Section 2.2, the CTX-FSM path is divided into three stages to support CS and CM while maintaining a short critical path. As CS and CM are not needed for bypass bins, the BPS-FSM path only has one stage. This reduces the stalls for the bypass coded bins, particularly for the syntax element *coeff_abs_level_remaining* where the binarizaton of the next syntax element depends on the value of the current one.

2.5. Multi-Bypass Bins Decoding

Grouping of bypass bins also increases the benefit of decoding more than one bypass bin per cycle. The logic latency of BPS-E is around half of the optimized CTX-E. Therefore, without increasing the AD stage delay, the deeply pipelined architecture can decode up to two bypass bins per cycle with mBPS-E, which concatenates two BPS-Es. The corresponding decoding mode, mBPS, is used for the bins of *coeff_sign_flag* and *coeff_abs_level_remaining*. Since the number of bins combined from these two syntax elements account for at least 10% to 20% of the total number of bins in common video bitstreams, this scheme can improve the throughput significantly. To support mBPS, the BDT width of the BPS-FSM control path needs to be doubled, as the prefetch logic has to compute next four possible states instead of two.

Class	Sequence	QP	Coding Struct.	Bit Rate (Mbps)	Bins/Cyc.
A (WQXGA)	Traffic	22	AI	101.89	0.83
			LD	13.83	0.70
			RA	13.24	0.73
	People On Street	22	AI	104.72	0.85
			LD	37.56	0.78
			RA	32.83	0.78
	Nebuta	22	AI	403.02	1.06
			LD	239.32	1.02
			RA	216.38	1.01
	Steam Locomotive	22	AI	100.39	0.92
			LD	30.52	0.82
			RA	23.55	0.82
B (Full HD)	Kimono	22	AI	22.25	0.90
			LD	5.21	0.81
			RA	4.80	0.82
	Park Scene	22	AI	52.75	0.85
			LD	7.97	0.70
			RA	7.69	0.75
	Cactus	22	AI	105.39	0.83
			LD	20.05	0.74
			RA	18.44	0.75
	BQ Terrace	22	AI	180.18	0.90
			LD	52.82	0.77
			RA	39.64	0.78
	Basketball Drive	22	AI	71.14	0.81
			LD	19.86	0.76
			RA	17.40	0.77

Table 2: Simulated decoding performance of the proposed design for common test bitstreams [14]. Each sequence is coded with three configurations: All Intra (AI), Low Delay (LD) and Random Access (RA).

3. EXPERIMENTAL RESULTS

3.1. Experimental and Synthesis Results

Table 2 shows the simulated decoding performance of the proposed architecture for common test bitstreams [14]. In general, the bins per cycle of high bit-rate bitstreams, especially the all-intra (AI) coded ones, is higher than that of low bit-rate bitstreams. For example, *Nebuta*, with a bit-rate up to 400 Mbps, can be decoded at 1.06 bins/cycle.

The design is synthesized in a IBM 45nm SOI process, and achieves a maximum clock rate of 1.9 GHz. For the AI coded *Nebuta* bitstream, the throughput reaches 2014 Mbin/s. This throughput is already sufficient for the real-time decoding of Level 6.2 video bitstreams. The total gate count of the CABAC decoder is 85.3K (WPP not supported). The context memory and the line buffers take up around 30% of this area. The power consumption at synthesis is 26.0 mW. Table 3 summarizes the comparison with previous works.

3.2. Analytical Worst Case Performance

HEVC defines the maximum bin-rate limits at three granularities: within a coding-tree unit (CTU) (based on the constraint of maximum bits per CTU), within a frame (based on the constraint of maximum bins per slice-coded network abstraction layer unit), and across frames (based on the maximum bit-

	Lin [7]	Liao [8]	Choi [11]	**This Work**
Standard	AVC	AVC	HEVC	HEVC
Tech.	UMC 90nm	UMC 90nm	Samsung 28nm	IBM 45nm SOI
Gate Count	82.4K	51.3K	100.4K	85.3K
Max. Freq. (MHz)	222	264	333	1900
Bins/Cyc.	1.96	1.84 (130 Mbps)	1.30	1.06 (403 Mbps)
Throughput (Mbin/s)	435	486	433	2014

Table 3: Comparison on the results of different CABAC decoder implementations. WPP is not supported in this work.

Level	*4.0*	*4.1*	*5.2*	*6.0*	*6.1*	*6.2*
Per CTU	1550	3100	24800	24800	49600	99200
Per Frame	292	585	2540	2510	5020	12900
Multi-Frame	40	67	320	320	640	1070

Table 4: The worst-case bin-rate (Mbin/s) limits.

rate). These granularities correspond to decoding latencies of a CTU, a frame and multiple frames. Reducing latency is important for applications such as video conferencing where sub-frame latency is required. The worst-case limits are listed in Table 4. The limits tend to be lower when larger latency is tolerated since workload can be averaged across CTUs and frames. The decoder throughput needs to be higher than these limits to guarantee real-time low latency decoding.

The worst case scenario is computed for a CTU size of 16×16 luma samples and the maximum total bins with the maximum number of context-coded bins. The maximum amount of context-coded bins per 16×16 CTU is 884 assuming 4×4 prediction units and 4×4 transform units, and the bit limit per 16x16 CTU with bit-depth of 8 is 5120. Considering the maximum bin-to-bit compression ratio for the context-coded bins at 0.0273, the maximum amount of bypass bins is 5096. Taking the number of bypass bins decoded with the mBPS mode and also the stalls into account, the design in this work decodes at 1.44 bins/cycle under the theoretical worst case. The corresponding throughput is 2748.3 Mbin/s. In Table 4, we shade in grey all bin-rate limits that can be achieved by this design in real-time for each granularity.

4. CONCLUSION

In this paper, we propose the hardware architecture of a CABAC decoder for HEVC. The design features a deeply pipelined structure and reduces stalls using techniques such as the state prefetch logic, latch-based context memory and separate FSMs. It can also decode up to two bypass bins per cycle. The decoder achieves up to 1.06 bins/cycle for high bit-rate common test bitstreams, and 1.44 bins/cycle under the worst-case scenario. With the synthesized clock rate at 1.9 GHz, the throughput reaches above 2000 Mbin/s, which is sufficient to real-time decode video bitstreams at Level 6.2 (8K UHD at 120 fps), or at Level 6.0 (8K UHD at 30 fps) for applications requiring sub-frame latency.

5. REFERENCES

[1] *High efficiency video coding*, ITU-T Recommendation H.265 and ISO/IEC 230082, April 2013.

[2] G. J. Sullivan, J. Ohm, T. K. Tan, and T. Wiegand, "Overview of the high efciency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 22, no. 12, pp. 1649–1668, December 2012.

[3] 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 (TCSVT)*, vol. 13, no. 7, pp. 620–636, July 2003.

[4] V. Sze and M. Budagavi, "High throughput CABAC entropy coding in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 22, no. 12, pp. 1778–1791, December 2012.

[5] Y. Yi and I.-C. Park, "High-speed H.264/AVC CABAC decoding," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 17, no. 4, pp. 490–494, April 2007.

[6] Y.-T. Chang, "A novel pipeline architecture for H.264/AVC CABAC decoder," in *Asia Pacific Conference on Circuits and Systems*. IEEE, November 2008, pp. 308–311.

[7] P.-C. Lin, T.-D. Chuang, and L.-G. Chen, "A branch selection multi-symbol high throughput CABAC decoder architecture for H.264/AVC," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2009, pp. 365–368.

[8] Y.-H. Liao, G.-L. Li, and T.-S. Chang, "A highly efficient VLSI architecture for H.264/AVC level 5.1 CABAC decoder," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 22, no. 2, pp. 272–281, February 2012.

[9] K. Watanabe, G. Fujita, T. Homemoto, and R. Hashimoto, "A high-speed H.264/AVC CABAC decoder for 4K video utilizing residual data accelerator," in *Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, March 2012, pp. 6–10.

[10] J.-W. Chen and Y.-L. Lin, "A high-performance hardwired CABAC decoder for ultra-high resolution video," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1614–1622, August 2009.

[11] Y. Choi and J. Choi, "High-throughput CABAC codec architecture for HEVC," *Electronics Letters*, vol. 49, no. 18, pp. 1145–1147, August 2013.

[12] M.-Y. Kuo, Y. Li, and C.-Y. Lee, "An area-efficient high-accuracy prediction-based CABAC decoder architecture for H.264/AVC," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2011, pp. 15–18.

[13] V. Sze and A. P. Chandrakasan, "A highly parallel and scalable CABAC decoder for next generation video coding," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 47, no. 1, pp. 8–22, January 2012.

[14] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards–including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 22, no. 12, pp. 1669–1684, December 2012.