Design and Implementation of a High-Throughput CABAC Hardware Accelerator for the HEVC Decoder

Philipp Habermann
Embedded Systems Architecture
Technische Universität Berlin

p.habermann@tu-berlin.de

Abstract: HEVC is the new video coding standard of the Joint Collaborative Team on Video Coding. As in its predecessor H.264/AVC, Context-based Adaptive Binary Arithmetic Coding (CABAC) is a throughput bottleneck. This paper presents a hardware acceleration approach for transform coefficient decoding, the most time consuming part of CABAC in HEVC. In addition to a baseline design, a pipelined architecture and a parallel algorithm are implemented in an FPGA to evaluate the gain of these optimizations. The resulting baseline hardware design decodes 62 Mbins/s and achieves a 10× speed-up compared to an optimized software decoder for a typical workload at only a tenth of the processors clock frequency. The pipelined design gives an additional 13.5%, while the parallel design provides a 10% throughput improvement compared to the baseline. According to these results, HEVC CABAC decoding offers good hardware acceleration opportunities that should be further exploited in future work.

1 Introduction

High Efficiency Video Coding (HEVC [1]) is a new video coding standard that targets to halve the bitrate requirements while remaining comparable in perceptive image quality to its predecessor H.264/AVC. HEVC is also designed to better exploit the capabilities of todays multicore architectures. High-level coding tools for sub-picture parallelization were introduced, while low-level throughput limitations were removed to allow more efficient implementations.

A component of HEVC that can only hardly be parallelized is Context-based Adaptive Binary Arithmetic Coding (CABAC). The reason is that there are strong data dependencies which make CABAC a throughput bottleneck. This paper gives a brief overview of a master thesis that aims to evaluate the hardware acceleration opportunities for HEVC CABAC decoding. Therefore, a customized hardware decoder for transform coefficient coding [2] is built, as it is the most time-consuming part of CABAC in HEVC.

The paper is structured as follows. Section 2 presents related work while Section 3 explains the basic functionality of CABAC and describes the design of the hardware accelerator. Section 4 provides a performance evaluation of the resulting implementation. Finally, a conclusion is given in Section 5.

2 Related Work

Sze and Budagavi describe the general throughput improvements in HEVC CABAC compared to H.264/AVC in [3]. The improvements include a reduced total number of bins, a reduced amount of context-coded bins, grouped bins with the same context and grouped bypass-coded bins. Furthermore, there are less context selection dependencies and lower memory requirements.

A comparison of different H.264/AVC CABAC accelerator architectures is provided by Jan and Jozwiak in [4]. According to their analysis, a parallel pipeline approach is most promising and gives the highest throughput. Sze et al. present Parallel CABAC in [5]. This technique allows the decoding of mutiple bins in parallel by using n-ary instead of binary arithmetic coding. This increases the computational complexity and is only suitable for a low number of parallel bins.

The scope of this paper is to evaluate the hardware acceleration opportunities for HEVC CABAC decoding. Therefore, a hardware accelerator is built that covers the transform coefficient coding part while exploiting the throughput improvements introduced in HEVC. The above-mentioned optimizations are also implemented to evaluate the speed-up that can be achieved when using them in HEVC CABAC.

3 Design

Binary arithmetic coding is an efficient entropy coding method, where the encoded bitstream is represented by an offset inside an interval (range). To decode a binary symbol (bin), the range is divided into two subranges. The size of each subrange relative to the initial range is proportional to the probability with that each bin state can appear. The decoded bin state corresponds to the subrange where the offset is located in. This subrange becomes the new range for the decoding of the next bin. Context models are used to estimate the probabilities for bins that represent specific syntax elements. They are updated based on the results of previously decoded bins to provide a good estimation for different inputs. Beside these context-coded bins, there are also bypass-coded bins that have fixed equal probabilities for both bin states. They are coded in a less complex procedure that does not involve context models.

The CABAC decoding process is highly sequential due to strict bin-to-bin dependencies (see Figure 1). The decoding of a bin that is part of a syntax element starts with the contex model selection. The context model is used to perform the actual bin decoding. Depending on the result, the range and offset are updated, the involved context model is adapted and the next syntax element is chosen. As the context model selection for the next bin depends on the selection of the corresponding syntax element, it is not possible to overlap the decoding steps for consecutive bins without modifications.

Based on the analysis of the bin decoding process, three designs are implemented that differ in the amount of resources they use to increase the throughput. First, a baseline design implements the decoding process as seen in Figure 1. The second design implements a

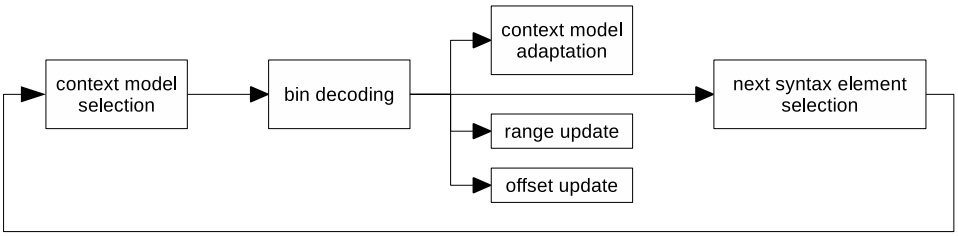

Figure 1: Steps in the decoding process of a context-coded bin. Context model selection and adaptation and the range update are not needed for bypass-coded bins.

two-stage pipeline. Therefore, the selection of the next syntax element is moved to the beginning of the decoding for the next bin. The first pipeline stage performs the syntax element and context model selection. The data path is duplicated to be able to speculate on both possible results of the previously decoded bin. This bin is then forwarded to select the correct context model. The second pipeline stage consists of the bin decoding, the range and offset update and the context model adaptation. Finally, the third design uses quaternary arithmetic coding to decode two bypass-coded bins in parallel.

4 Evaluation

All designs are implemented in the programmable logic of the Zynq-7020 System-on-Chip. It contains an ARM Cortex-A9 processor and a Xilinx Artix-7 FPGA, thereby allowing efficient hardware-software co-design. Three different videos are used to represent a wide spectrum of inputs: a small video (low bitrate), an average video (medium bitrate) and a big video (high bitrate). An optimized software decoder from the Embedded Systems Architecture Group at TU Berlin is used as a reference for a throughput comparison (see Figure 2). The speed-up of the baseline design compared to the software decoder is $8\times$ for the big video, $10\times$ for the average one and $14\times$ for the small video. The baseline hardware accelerator operates at a clock frequency of 66 MHz, which is only a tenth of the processors clock frequency. Due to a higher maximum clock frequency of 75 MHz caused by the pipelined architecture, the pipelined design can process 13.5 % more bins per second than the baseline design. The parallel design achieves a speed-up of 8 to 10 % over the baseline due to the simultaneous decoding of two bypass-coded bins.

The resulting hardware accelerator has also been integrated into the software decoder. A low-throughput software register based interface has been used with the main purpose of verifying the functionality of the hardware decoder. Due to the high data transfer overhead, the overall speed-up for this hybrid decoder is only 1.5 %.

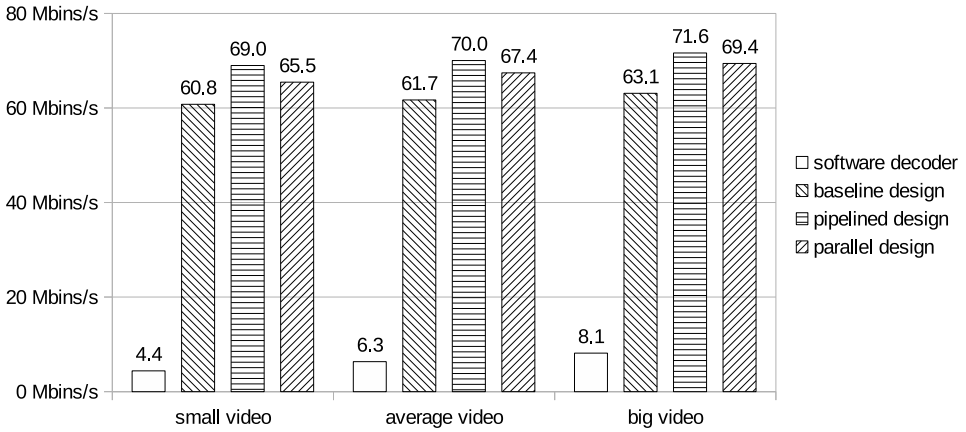

Figure 2: Throughput comparison of the hardware accelerator designs for different input videos.

5 Conclusions

Three different implementations of a hardware accelerator for HEVC transform coefficient decoding were presented. For an average video, a speed-up of 10× over software decoding is achieved at a tenth of the clock frequency, which is a promising result for a highly sequential process like CABAC. When the hardware accelerator is integrated into a software decoder, the theoretical speed-up is reduced due to a very high data transfer overhead caused by the fine-grained hardware acceleration. Currently, a complete CABAC hardware decoder is developed that significantly reduces the data transfer overhead.

References

[1] G. J. Sullivan, J. Ohm, Woo-Jin Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, issue 12, pp. 1649 - 1668, December 2012

[2] J. Sole, R. Joshi, Nguyen Nguyen, Tianying Ji, M. Karczewicz, G. Clare, F. Henry, A. Duenas, "Transform Coefficient Coding in HEVC", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, issue 12, pp. 1765 - 1777, December 2012

[3] V. Sze, M. Budagavi, "High Throughput CABAC Entropy Coding in HEVC", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, issue 12, pp. 1778 - 1791, December 2012

[4] Y. Jan, L. Jozwiak, "CABAC Accelerator Architectures for Video Compression in Future Multimedia: A Survey", Proceedings of 9th International Workshop on Embedded Computer Systems: Architectures, Modeling and Simulation, pp. 24 - 35, July 2009

[5] V. Sze, A. P. Chandrakasan, M. Budagavi, Minhua Zhou, "Parallel CABAC for low power Video Coding", Proceedings of 15th IEEE International Conference on Image Processing, pp. 2096 - 2099, October 2008