A macroblock-level analysis on the dynamic behaviour of an H.264 decoder

Florian H. Seitner, Ralf M. Schreier, *Member*, IEEE
Michael Bleyer, Margrit Gelautz, *Member*, IEEE

Abstract — *This work targets the optimization of multi-processor H.264 decoder implementations. We have extended the simulator of a multi-core VLIW media processor to enable cycle-accurate function profiling on a sub-macroblock level, which allows measuring the effects of coding modes on the computational complexity with very fine granularity. This knowledge helps the system designer to optimize the system performance and memory sizes to reduce system costs*[1].

Index Terms - H.264 decoder, dynamic behaviour, multi-processor implementation.

I. INTRODUCTION

Among current video coding algorithms, H.264/AVC has the highest computational complexity. Due to its use in international broadcasting standards such as DVB-H and DMB, it is also becoming relevant for mobile devices such as PDAs or mobile phones, which typically provide minimum system resources. If the computational requirements cannot be met with a single processing unit, multi-core systems often provide an elegant and power-efficient solution.

For an H.264/AVC decoder [1], the main decoding tasks such as syntax parsing and entropy decoding can be realized with dedicated physical hardware units. In software based multi-processor implementations, however, it is favourable to execute multiple basic tasks on a single processor. In this work, we focus on one possible method to accomplish dual-core splitting for the H.264 decoder that is shown in Fig. 1. Here, the stream parsing and entropy decoding processes are implemented on the first processor, while the IDCT and pixel-based operations are executed on the second processor. This approach reduces data dependency issues in the main decoder loop. However, the method raises the problem of finding a good trade-off between even work load balancing on the one hand and buffer minimization on the other hand.

The key challenge in designing such a system lies in the dynamic nature of video decoding. There are strong indications that the assumption of constant work load over time is not valid in a system with real-time constraints. It is a well-known fact that H.264 offers a large variety of different coding modes on the macroblock level and partitioning modes on the sub-macroblock level. These different modes occur with variable probabilities depending on the video content [2]. On the one hand, intra-coded macroblocks show significantly higher average data rates than predicted or skipped macroblocks, which results in a time-varying load of the entropy decoding unit on frame and macroblock levels. On the other hand, the complexity of the prediction and reconstruction processes highly depends on the macroblock partitioning mode and sub-pixel motion vector position.

In practical implementations, multiple processing units are usually interfaced with FIFO buffers as shown in Fig. 1. It is also possible to introduce additional frame buffers located between reconstructor and the display unit in order to compensate for long-term load variations of the whole system. Larger buffers generally cause higher latencies and consequently result in a more efficient usage of the computational system resources. In cost-sensitive markets, however, it is necessary to minimize these on-chip and off-chip memories to reduce silicon costs. For an efficient multi-processor portioning of the H.264 decoder, it is therefore essential to understand the dynamics of run-time execution.

In this work, the simulator of a multi-core VLIW media processor was extended to enable cycle-accurate function profiling on a sub-macroblock level which allows measuring the effects of coding modes on the computational complexity with very fine granularity. Several standard image sequences are used to extract profiling information. We then combine the profiling information with the stream information in order to identify correlations between execution time and macroblock characteristics.

Our study is related to several papers that profile the computational performance of an H.264 decoder [2-4]. These benchmark approaches decode whole video sequences and then report the execution time spent in each functional block of the decoder. While this information might be sufficient to roughly identify the bottleneck in the decoder implementation, these profiling results lack information about the varying computational demands of individual macroblocks. In a spirit similar to our work, a step towards macroblock-based profiling of an H.264 decoder has recently been taken in [5].

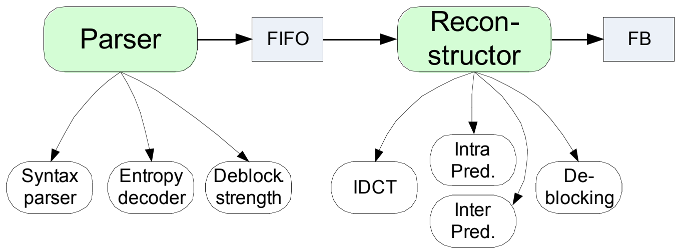


Fig. 1. Decoder functionality is divided into a parsing and a reconstruction part and distributed onto two processing units. The units are connected via a FIFO buffer.

[1]This work was supported by the Austrian Federal Ministry for Transport, Innovation and Technology under the FIT-IT project VENDOR (Nr. 812429).

All authors are with the Institute for Software Technology and Interactive Systems, Vienna University of Technology, Austria (e-mail: [seitner, schreier, bleyer, gelautz]@ims.tuwien.ac.at).

The authors build a categorisation of macroblocks. For each macroblock type, they determine the frequency of its occurrence in the profiled video stream. Run-time measurements are, however, only provided for the decoding of whole video frames and not for the decoding of individual macroblocks which is the target of our study.

II. Profiling Environment

A. Hardware environment

A cycle-accurate simulator for a VLIW multimedia processor provides the profiling results used in this work. The RISC processor core can process four instructions in parallel which can be any combination of 32-bit arithmetic instructions and load-store operations. For parallel pixel operations, 16-bit SIMD instructions are provided. Each processor has a dedicated 64 KB local memory for fast data access and a 64 KB instruction cache. Data is transferred via DMA or direct access between the processor's local memory and 64 kB shared on-chip SRAM as well as up to 64 MB external memory. The processor can be programmed in C/C++ with intrinsic SIMD functions as well as in low-level assembly language for time-critical routines.

B. Decoder and test sequences

The simulations are based on a commercial H.264/AVC main profile decoder for embedded architectures that was optimized in terms of memory usage and support of DMA transfers. Additionally, the regular pixel-based processing functions of the decoder (e.g. interpolation, prediction, IDCT) were assembly optimized to make use of the SIMD processor commands.

The run-time behavior of the decoder was profiled for the five standard image sequences listed in Table I. The test sequences were encoded in H.264 baseline profile with a GOP size of 14 frames using the JM12.0 [6] encoder (CIF, deblocking active, all prediction modes allowed, SR +/-16 pixels, 3 reference frames). In order to obtain enough samples of intra-coded blocks, all sequences were additionally coded without temporal prediction.

To avoid impacts of the encoder rate control algorithm as well as coding quality effects on the decoder run-time, all sequences are coded with constant quantization parameters (QP) at an average SNR of 35 dB in all sequences. Initially, the quantization parameter QP_I for each intra-coded sequence is adjusted until reaching an average SNR of 35 dB for the complete test sequence. The IPP-coded sequence uses the same quantization value QP_I for intra frames and a matched quantization value QP_P for normalizing the inter-coded frames to ~35 dB. In Table I, the test sequences and the applied quantization parameters and resulting bitrates are summarized.

TABLE I
TEST SEQUENCES WITH INTRA / INTER QP SETTINGS

Sequence	QP_I	QP_P	Bitrate@25Hz
Foreman, 70 frames, CIF	32	32	242.3 kb
Paris, 70 frames, CIF	31	29	613.1 kb
Flowergarden, 70 frames, CIF	31	29	1.4 Mb
Mobile, 70 frames, CIF	29	28	1.9 Mb
Barcelona, 70 frames, CIF	29	27	2.3 Mb

C. Complexity mapping

In order to obtain detailed profiling results for the analysis, the cycle-accurate simulator of the VLIW media processor and its data memory system (DMS) was extended to enable function profiling on a sub-macroblock level. Hence, the simulator provides very detailed records of program execution from which the call graph as well as individual execution times for each function call can be extracted. This allows measuring the effects of coding modes on the computational complexity with very fine granularity. The simulator profiling information is subsequently combined with the bitstream information of the decoded video sequence to identify characteristics and correlations between execution time and macroblock characteristics. The functions called during the decoding process of a macroblock are mapped to the functional blocks parser and reconstructor of the decoder. The exact execution time for each macroblock is measured by accumulating the function cycle-counts of all sub-functions for each functional block.

III. Modeling

The modeling approach taken in this project is based on the fact that the decoding process is completely defined by the input data stream because the stream syntax elements control the program execution of the decoder precisely. Hence, it would be possible to model the decoder run-time behavior by analyzing all functions and their execution time with all possible input parameters. This approach would require a very detailed and time-consuming analysis of the source code.

To avoid this tedious analysis, a high-level approach is taken in which the most significant macroblock characteristics are identified. Investigating the correlations between single characteristics of a macroblock (e.g. coding modes, MV resolution) and its run-time complexity, strong linear dependencies can be observed. For describing this linear behavior we therefore use linear regression [7] to estimate the complexity of a macro-block.

Since parser and reconstructor as well as intra and inter macroblocks depend on different input characteristics, four separate linear models are applied to address each case (Intra-parser, Intra-recon, Inter-parser, Inter-recon). On one hand, this minimizes the input information required for the complexity estimate of each functional block. On the other hand, this addresses the differences in the run-time complexity and the dynamic behavior of the different macroblock types.

Due to the low number of input characteristics and the detailed knowledge about the entire decoding process, a manual modeling approach based on statistical investigations was preferred to an automatic machine learning approach. For building up a complexity model, a linear function depending on the highest correlating characteristic and a constant offset was used as a first estimate. The model parameters were determined with a multiple linear regression algorithm which minimizes the square error of the prediction. We then systematically added features in the equation until no further increase in the estimation precision could be achieved. One of the five sequences (Mobile) was used for calculating the model parameters, which were then used for run-time estimation of the other four sequences.

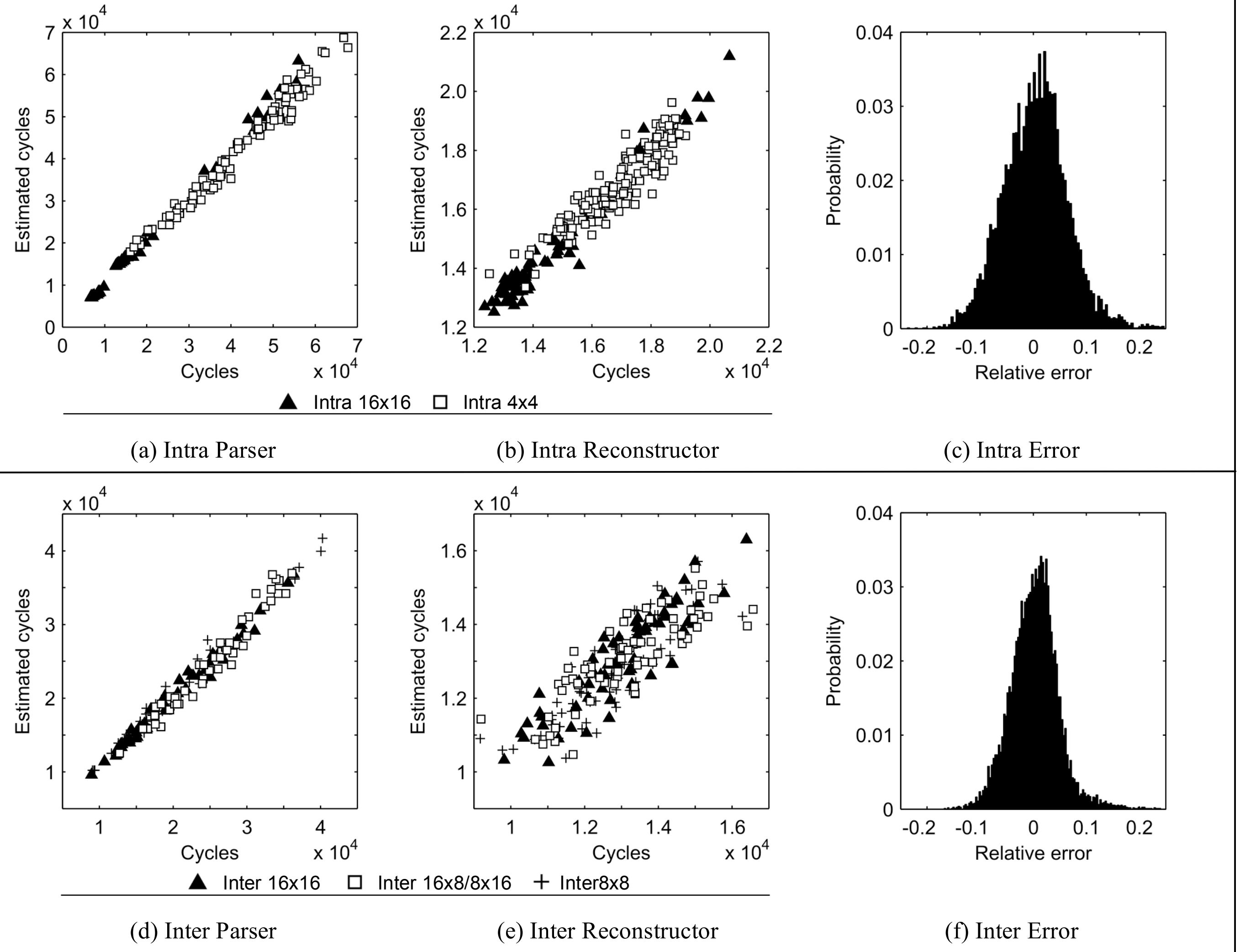


Fig. 2. Correlation between estimated and measured macroblock run-time complexity: (a) intra-coded in parser, (b) intra-coded in reconstructor, (d) inter-coded in parser, (e) inter-coded in reconstructor . Error distributions for intra (c) and inter (f) macroblock estimation.

Parsing time for sequence, picture and slice headers as well as corresponding setup functions of the reconstructor are not considered.

IV. RESULTS

In the case of the stream parsing unit, we assumed that the execution time mostly depends on the number of syntax elements which need to be identified, decoded and removed from the input stream buffer. For medium and high quality bitstreams, the majority of a coded macro-block's syntax elements and bits are consumed by luma and chroma DCT coefficients. For low-bitrate data streams, this situation is different. Here, the significant amount of syntax elements is consumed by macroblock mode selection, CBP, quantizer changes and prediction data (i.e. intra prediction mode signaling, inter prediction MVs).

For intra-coded macroblocks a highly linear relationship between a macroblock's number of syntax elements and the macroblock's parser run-time complexity can be observed (correlation coefficient $\rho_{X,Y}$ =0.986). When using the number of syntax elements as the only model parameter, run-time estimation with an average relative error of 4.9% can already be achieved. Including the coded macroblock size (in bits) further reduces the average error to 3.9% (=1537 cycles absolute). Results obtained using this two-input model are shown in Fig. 2a. Here, the estimated parser cycle count is plotted against the measured values for the intra-coded macroblocks of the Mobile sequence. It can be seen that the run-time complexities for both, Intra16x16 and Intra4x4 macroblocks are well described by the same model.

The computational complexity of Intra16x16 blocks is usually lower than that of Intra4x4, since they are typically used for coding homogeneous regions with small and only few residuals. Additionally, Intra16x16 blocks require less syntax elements for the prediction than Intra4x4 blocks.

For inter-coded macroblocks, the parsing stage behaves similar to the one for intra macroblocks (Fig. 2d). The number of a macroblock's syntax elements is strongly correlated with the computational complexity in the parsing module ($\rho_{X,Y}$ =0.981). Based on the number of syntax elements a complexity estimation with an average relative error of 5.0% is achieved. This estimation can further be improved by adding the encoded macroblock size in bits (4.6%, 1024 cycles).

For the reconstruction of an intra-coded macroblock, we assumed that the mode prediction and the deblocking steps cause the major part of the run-time complexity. Using the luma and chroma prediction modes as well as the number of deblocked edges as model input, a relative error of 4.3% was derived. The input information for edge deblocking consists of three values: number of non-filtered edges, number of filtered edges with strength 1-3 and number of strongly filtered edges with strength 4. Adding the number of active bits in the coded block pattern (CPB) to the previous two inputs improved the estimation and yields a relative error of 2.7% (453 cycles) on average. The CBP bits describe for which 4x4 sub-macroblock the IDCT has to be computed.

In Fig. 2b, the cycles spent on intra macroblock reconstruction and our estimations are visualized. Compared to the parsing process less variation in the computational complexity of the reconstruction for intra-coded macroblocks occurs. However, more input parameters are necessary to estimate the complexity accurately.

The correlation between the reconstruction estimation and the measurements for inter-coded macroblocks is shown in Fig. 2e. A good estimation is achieved by using the motion vector (MV) resolution (Full-pel | Half-pel | Quarter-pel) and the edge deblocking information. Depending on the vertical and horizontal MV resolutions, interpolation up to quarter pixel accuracy has to be done in the decoder. Additionally, adding the number of active CBP bits results in an average error of 4.2% (538 cycles).

The run-time complexity of parsing and reconstruction shows significant differences in the run-time complexity. For intra macroblocks on average 71% (42568 cycles) are spent for the macroblock parsing and 29% (16701 cycles) on reconstruction. For inter macroblocks the parsing process is less demanding than the reconstruction and the complexity changes to 37% (13000 cycles) for the parser and 63% (22058 cycles) for the macroblock reconstruction. For both, intra and inter frames, the complexities of parser and reconstructor are unequally distributed. Additionally, the measured run-time for the macroblock parsing indicates strong variations in the computational complexity of macroblocks. For a multi-core system using such a functionally divided decoder an unequal core usage will lead to stall cycles in either parser or reconstructor and decrease the parallelization efficiency. In a real-time system, which typically operates on a macroblock level rather than on a frame level, this leads to strong variations in the core usages.

TABLE II
AVERAGE RELATIVE ERROR (IN %)

Sequence	Intra parser	Intra recon	Inter parser	Inter recon
Foreman, 70 frames, CIF	8.2	2.6	8.4	6.5
Paris, 70 frames, CIF	5.0	3.3	5.2	6.5
Flowergarden, 70 frames, CIF	4.5	2.6	5.2	5.1
Mobile, 70 frames, CIF	3.9	2.7	4.6	4.2
Barcelona, 70 frames, CIF	3.7	2.2	5.8	7.9

After calibrating the models with the Mobile sequence we applied them on the other four video sequences for verification. In Table II the average error (in percent) which was achieved by the models is shown. It can be seen that for all sequences the functional blocks are well estimated. However, for the parser estimation of intra- and inter-coded macroblocks the relative error increases for sequences with lower bitrates and few syntax elements / bits per macroblock (e.g. Foreman). This effect is caused by the linear regression and the calibration sequence used for setting up the model. The linear regression reduces the sum of square errors between measurements and estimations and, therefore, tries to find a solution which best fits the majority of the training samples. Therefore, best estimations will be obtained for macroblocks with similar characteristics to those macroblocks occurring most often in the calibration sequence. The Mobile sequence used for calibrating contains a high amount of texture and movements. Coding the sequence at an operating point of 35 dB on average requires a medium / high number of syntax elements per macroblock. Hence, more accurate parser estimations will be achieved for macroblocks with a similar high number of syntax elements.

A maximum relative error of 3.3% indicates an accurate complexity estimation for the intra reconstruction part. Compared to the parser the complexity estimation for the intra reconstruction also works well at varying bitrates.

The estimation for the inter reconstruction state in general achieves less accurate results than the estimation for intra-coded macroblocks. This could be caused by the larger number of input parameters of the inter reconstruction model which can be described less accuratly by the linear regression model.

V. CONCLUSION

Our results are intended to guide the process of designing a multi-processor solution for an H.264 decoder. The run-time complexity of an H.264 decoder can be well estimated at a high level of abstraction and by using only few syntax elements for the estimation. A profile from a single sequence and linear regression can be used to calibrate the model parameters for intra- and inter-coded macroblocks. This model can then be applied to new test sequences to estimate their run-time complexity. The estimation results for low complexity sequences imply the use of different parser models for low and high bitrates. Modeling the parts of the inter reconstruction functionality (e.g. Deblocking) would provide simpler models and could improve the estimation.

In future work the macroblock-based profiling information can be exploited to determine an improved splitting of decoder functions across multiple processors. The granularity of the profiling will therefore be increased to provide flexibility in the modeling of functional blocks. The results can further aid to determine optimized approximations of buffer sizes.

It is a long-term perspective of this project to build up a high-level simulation for the major video codecs, which allows simulation of the dynamic behaviour in different multi-processor system configurations.

Acknowledgment

The authors would like to thank ON DEMAND Microelectronics (www.odm.at) for providing the processor simulator and infrastructure for this work and in-depth knowledge in the field of video coding.

References

[1] T. Wiegand, G.J. Sullivan, G. Bjntegaard, A. Luthra: "Overview of the H.264/AVC Video Coding Standard", *IEEE Trans. on Circuits and Systems for Video Technol.*, vol. 13, no. 7, pp. 560- 576, July 2003.

[2] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro: "H.264/AVC Baseline Profile Decoder Complexity Analysis", *IEEE Trans. on Circuits and Systems for Video Technol.*, vol. 13, no. 7, pp. 704-716, July 2003.

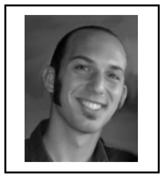
[3] V. Lappalainen, A. Hallapuro, T. D. Hämäläinen: "Complexity of Optimized H.26L Video Decoder Implementation", *IEEE Trans. on Circuits and Systems for Video Technol.*, vol. 13, no. 7, pp. 717-725, July 2003.

[4] T.-T. Shih, C.-L. Yang, Y.-S. Tung: "Workload Characterization of the H.264/AVC Decoder", *Proc. of the 5th IEEE Pacific-Rim Conference on Multimedia*, pp. 957-966, 2004.

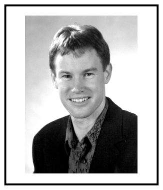
[5] H. Kalva, B. Furht: "Complexity Estimation of the H.264 Coded Video Bitstreams", *Computer Journal*, vol. 48, no. 5, pp. 504-513, 2005.

[6] Joint Model Version 12.2. http://iphome.hhi.de/suehring/tml/.

[7] J. Neter, M.H. Kutner, W. Wasserman, C. Nachtsheim: "Applied Linear Regression Models", 4th ed., Irwin Professional Pub, 2004.



Florian H. Seitner received his diploma degree (M.S.) in Computer Science from the Vienna University of Technology in 2006. Since September 2006 he is employed as a project assistant at the Vienna University of Technology where he works on his Ph.D. degree. He is currently doing research on modeling of video coding algorithms on embedded multi-core architectures.



Ralf M. Schreier was born in Friedrichhafen, Germany in 1974. He received the diploma degree (M.Eng.) in electrical engineering (systems engineering) from the University of Ulm in 2000 with a best thesis award donated by the VDE/VDI. From April 2000 to August 2006 he was a research fellow at the Microelectronics Department at the University of Ulm doing research in the field of low-delay video coding algorithms and real-time DSP implementations. Since September 2006 he is with the Institute of Software Technology and Interactive Systems of the TU Vienna. He is member of IEEE and the German TV and Cinema Technology Society (FKTG).

Michael Bleyer received his M.S. and Ph.D. degrees in Computer Science from the Vienna University of Technology in 2002 and 2006, respectively. He is currently employed as a post-doc researcher at the Vienna University of Technology. His research interests include stereo matching, optical flow computation, combinatorial optimization and video coding.

Margrit Gelautz received her PhD degree in Computer Science from Graz University of Technology, Austria, in 1997. She worked on optical and radar image processing for remote sensing applications during a postdoctoral stay at Stanford University. In her current position as an associate professor at Vienna University of Technology, her research focuses on image and video analysis for multimedia applications. She is Women in Engineering (WIE) officer of IEEE Austria Section.