jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2010, VOL. 56, NO. 3, PP. 291-300
Manuscript received July 10, 2010: revised September, 2010. DOI: 10.2478/v10177-010-0039-7

Architecture Design of The Hardware H.264/AVC Video Decoder

Mikołaj Roszkowski, Andrzej Abramowski, Michał Wieczorek, and Grzegorz Pastuszak

***Abstract*—The need for real-time video compression systems requires a particular design methodology to achieve high troughput devices. The paper describes the architecture of the H.264/AVC decoder able to support SDTV and HDTV resolutions. The design applies many optimization techniques to reduce the resource consumption and maximize the throughput. The archietcture is verified with the software reference model JM16 and synhesized for FPGA technology. The maximal working frequency is 100 MHz for Stratix II devices.**

***Keywords*—H.264/AVC, video compression, FPGA, architecture design.**

I. Introduction

THE most modern, and the best in terms of offered capabilities of all video compression standards currently in use is undoubtedly H.264/AVC ([10], [6]). The standard itself evolved in time. The main purpose of the first version, published in 2003, was to provide efficient compression methods of standard and high definition video signals that would meet the requirements of an average home-user. This part of the standard is called Main Profile. However, this profile can not be employed for more advanced applications that require increased sample bit-depth or chroma sub-samplings other than 4:2:0. These features required the development of standard extensions (commonly known as FRExt extensions) that were gathered in a group of profiles known as High Profiles and became part of the standard in 2005.

FRExt extensions, except for the introduction of many new features, further improve compression efficiency. Tests (see [7]) prove that the use of H.264/AVC Main Profile allows the reduction of the bit-rate twice compared to MPEG-2 systems, while preserving the same perceived quality. Other tests (see [9]) proved that the use of High Profile provides further 10% decrease in bit-rate compared to Main Profile.

Compared to MPEG-2, the most important advances include Context Adaptive Variable Length Coding (CAVLC), Context Adaptive Binary Arithmetic Coding (CABAC), INTRA prediction, and variable block size INTER prediction. All these changes allow a tremendous performance improvement, but bring the increase in complexity. As a consequence, decoding and encoding of H.264/AVC sequences typically requires the use of signal processors, graphical processing units (GPU), or specialised integrated circuits. This means that the development of efficient hardware implementations of a H.264 decoder modules is a task of the fundamental importance to increase the adoption of the standard.

This paper presents the architecture of a H.264/AVC decoder designed to support a real time decoding of H.264/AVC High Profile (except for MBAFF processing mode) sequences with video resolutions up to 720x576 in both progressive and interlace mode in FPGA devices. If B frames are not used, also HDTV resolutions can be supported. The architecture was designed to be very flexible, and after some modifications to the INTER prediction module it should be capable of decoding 1080p sequences in real time for B frames. The reason these modifications were not introduced, is the fact that they require a significant increase of the resources used, far beyond the ones that were available to us. Nevertheless, the remaining modules are prepared to decode 1080p@fps sequences.

II. H.264/AVC Decoder Overview

A. H.264/AVC Stream Organisation

In order to fully understand the way the H.264/AVC decoder works it is necessary to become familiar with the H.264/AVC data-flow. Data in H.264/AVC compliant stream is organised in a hierarchical way. A sequence comprises of multiple frames, and each frame is divided into slices. Decoding of each slice is completely independent on the decoding of the other slices, which prevents transmission error propagation or even losing data from the whole frame. Slices are made up of macroblocks, which are the basic coding unit on which H.264/AVC decoder operates. Each macroblock contains 16x16 luma sample array with two accompanying chroma components, which are sent separately within a macroblock. Depending on the chroma sub-sampling, the size of the chrominance components vary. Particularly, For 4:2:0, 4:2:2, and 4:4:4 sub-samplings, chroma block size is equal to 8x8, 8x16, and 16x16, respectively. The macroblock layer is the only one containing actual sample data. The structures higher in the H.264/AVC stream hierarchy contain only control data specifying the exact way the macroblocks should be decoded.

B. Timing Constraints

The design of the system working in real time require a detailed analysis and specification of the timing constraints. In the case of the H.264/AVC decoder, the timing constraints the results from the video throughput, and they are specified by the number of clock cycles that can be allocated to one

The work is a part of the project "Integrated mobile system for counterterrorism and rescue operations", co-financed by the European Regional Development Fund within the framework of the 1. priority axis of the Innovative Economy Operational Programme, 2007-2013, submeasure 1.1.2 "Strategic R&D Research". Contract no. POIG.01.02.01-00-014/08.

M. Roszkowski, A. Abramowski, M. Wieczorek, and G. Pastuszak are with Institute of Radioelectronics, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland (e-mail: {M.Roszkowski, A.Abramowski, M.Wieczorek, G.Pastuszak}@ire.pw.edu.pl).

TABLE I
THROUGHPUT OF 4:2:0 SUB-SAMPLED SEQUENCES

Resolution	frames/s	throughput	
		bits/s	MB/s
1920x1080 (1080p)	25	622.080.000	204.000
	30	764.496.000	244.800
1280x720 (720p)	25	276.480.000	90.000
	30	331.776.000	108.000
720x576 (576p)	25	124.416.000	40.500
	30	149.299.200	48.600

TABLE II
MAXIMAL NUMBER OF CLOCK CYCLES AVAILABLE FOR ONE MACROBLOCK DECODING AT DIFFERENT OPERATING FREQUENCIES AND FOR DIFFERENT VIDEO RESOLUTIONS

Resolution	frames/s	frequency		
		80 MHz	100 MHz	120 MHz
1920x1080 (1080p)	25	392	490	588
	30	326	408	490
1280x720 (720p)	25	888	1111	1333
	30	740	925	1111
720x576 (576p)	25	1975	2461	2962
	30	1646	2057	2469

macroblock decoding. The latter measure is particularly useful as it is independent of the chroma sub-sampling. The throughput of the various sequences is presented in Table I, and the maximal amount of clock cycles available is summarised in Table II.

III. PROPOSED DECODER

A. Design Goals

The primary goal was to create an almost fully H.264/AVC High Profile compliant decoder supporting 576p sequences and capable of operating at the frequency of 100 MHz in Arria 2 devices. The only not-supported feature of the H.264/AVC High Profile is the macroblock adaptive frame field coding (MBAFF), temporal direct prediction modes, and implicit weighted prediction. All of these significantly increase resource usage and macroblock processing time. Moreover, the MBAFF frame coding mode do not seem to be widely used.

The second goal was to make the design that could be easily adapted to support higher video resolutions including 1920x1080. As it is shown in Table II, there are only 408 clock cycles available for the decoding of a macroblock in 1080p@30fps sequence. This implies that a decoding of one sample can take maximally one clock cycle for 4:2:0 sub-sampling. Therefore, if the decoder was to be easily upgradable to support higher video resolutions, most modules should have throughput close to one sample per clock cycle.

The final goal was to keep the size of the decoder relatively small. It was decided that the whole decoder should not use more than about 30K ALUTs of an Arria 2 device. For that reason, whenever it appeared that the support of 1080p sequences in the module would entail excessive resource usage, only 576p support was implemented.

B. Modules Overview

The basic unit the decoder operates on is a macroblock. Although some of the modules need to operate on blocks of sizes smaller than the macroblock size, the inter-module synchronisation is done on a macroblock level. The macroblock processing in the proposed decoder is divided into four main pipeline stages:

- Decoding of elementary stream data
- Residua and INTER prediction calculation
- Original sample values reconstruction.
- Deblocking (loop) filter process

The block diagram showing all modules of the designed H.264/AVC decoder is presented in Fig. 1. The first stage is entropy decoding of the input stream. In the case of the H.264/AVC High-Profile either CAVLC or CABAC entropy coding can be used, so the entropy decoder module is comprised of two sub-modules: CAVLC decoder and CABAC decoder. The parsing result of the elementary stream is stored in the embedded dual-port RAM module and registers.

The second stage includes the data preparation, dequantisation, inverse transform, and INTER prediction modules. The data preparation unit is vital to the correct functioning of the whole decoder. It is responsible for the calculation of all prediction parameters like INTRA prediction mode, motion vectors, reference frames etc. It also reorders the data for a use by the next modules by performing operations like the zig-zag scan or the Hadamard transform on DC samples.

The dequantisation module calculates transform coefficients using the data decoded by the entropy decoder in the previous stage, and sends them to the inverse transform module. The inverse transform module computes sample residua — values equal to the difference between original sample and its prediction (either INTRA or INTER). The residua are stored in the RAM for further use in reconstruction process. In parallel to the residua calculation process, INTER prediction module determines INTER prediction for the current macroblock, if necessary. This involves the restoration of correct motion vectors and reference frames from data preparation unit, fetching reference block data from the external memory, and performing actual prediction. Afterwards, predicted data are buffered for the reconstruction.

The third stage is a reconstruction process, combined with INTRA prediction in the case of INTRA coded macroblocks. During the process the original sample values are restored as a sum of residua calculated by inverse transform and appropriate prediction values. The reason why INTRA prediction is done together with reconstruction, unlike INTER prediction, is its strong dependence on just-reconstructed data. The INTRA prediction computes prediction values for the block using samples neighbouring with the block. Since in most cases INTRA prediction module operates on blocks of sizes much smaller than the macroblock, block neighbouring samples are usually those just reconstructed ones. In consequence, to achieve desired throughput a tight coupling of the reconstruction and INTRA prediction modules is necessary.

The last decoder stage is the deblocking (loop) filtering process. It smooths the final image, reducing the visibility of distortions introduced in the encoding process, The filtering takes place on borders of the blocks of sizes corresponding to the ones used by the transform. It operates on samples reconstructed in the previous stage, and filtered samples are

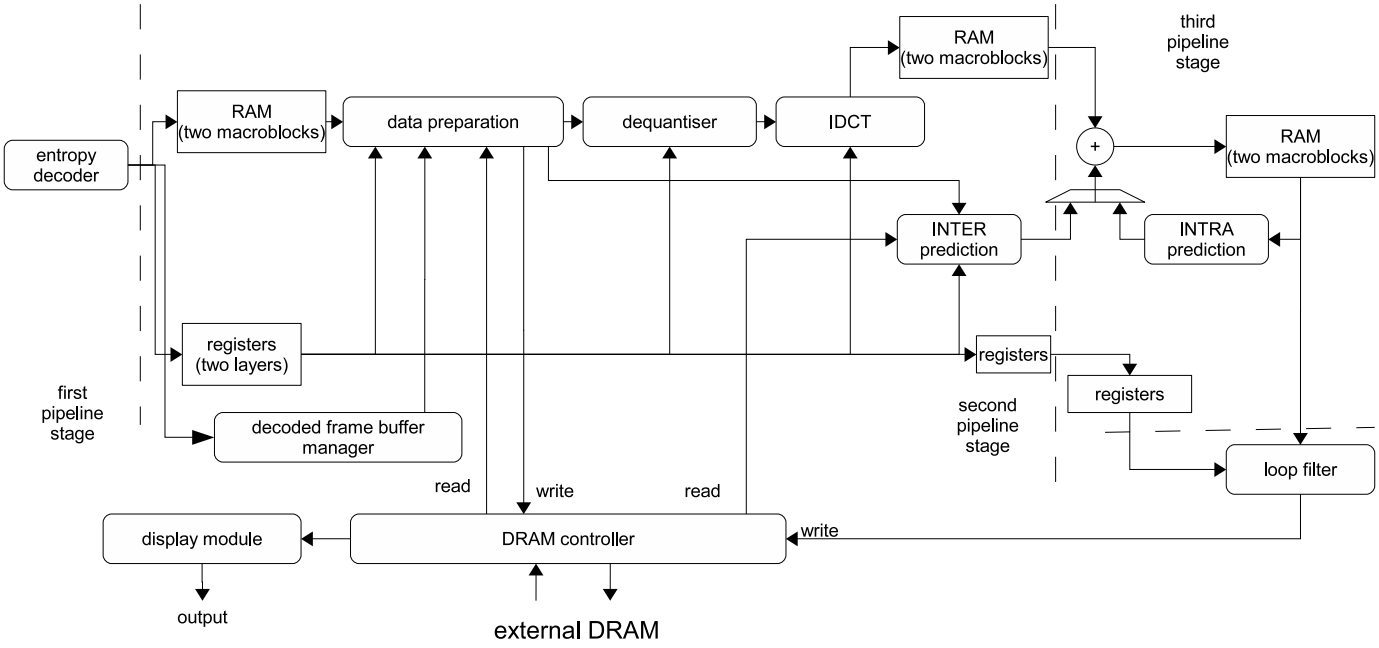


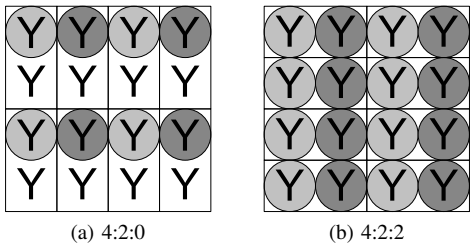
Fig. 1. Proposed H.264/AVC decoder block diagram.

stored into the external memory. These samples are displayed on the screen, and they can be later accessed for the use in the INTER prediction.

Except for the modules working in the four-stage pipeline, there are two additional modules that cannot be fitted into particular macroblock pipeline stage. These are: the external memory controller module, the decoded frame buffer manager, and the module responsible for sending decoded data to the screen. The external memory controller acts as an arbiter between different modules requiring external memory access like the INTER prediction module, the data preparation module, or the loop filter module. The decoder frame buffer manager is a module operating on slice and frame level rather than macroblock one. It assigns space in external memory for frame storage, maintains reference lists, and controls which frame is to be send to screen next.

C. Data Organisation

1) Embedded RAMs: One of the greatest challenges in the design of the H.264/AVC decoder is ensuring the best performance of the designed pipeline. For that reason, each pipeline stage is terminated with a RAM module and register set. Each RAM module can keep data from two macroblocks. One macroblock address space contains data that is a source for the next pipeline stage, whereas the second is for the data calculated by the previous stage. When both pipeline stages are ready, the macroblocks' data is swapped by simple change of the address spaces. This allows the smoothing out of the latencies of the different pipeline stages and makes the better use of available clock cycles. The RAM modules keep only actual sample data, and the accompanying control data will be stored in registers. The RAM located after the entropy decoder stage keeps one sample in a memory cell, while the remaining ones keep four samples. These four samples constitute one column of a 4x4 block. Such a format is selected since the H.264/AVC standard specifies the inverse transform returning column based data.



(a) 4:2:0 (b) 4:2:2

Fig. 2. Organisation of the color components in the external DRAM for a 4x4 block; Light grey — placement of a Cb sample, dark grey — placement of a Cr sample.

2) External DRAM: The organisation of the data in the external memory is very important for the design of INTER and data display modules. The samples' data in the external memory is grouped into frames. The decoder assigns to each frame a separate address space from a pool of free address spaces. The size of the pool is equal to 16 as this is the maximal number of reference frames that can be used.

Colour components of a frame are stored in an interleaved fashion, which varies depending on the chroma sub-sampling. In the case of the 4:2:0 sub-sampling, one memory cells keeps two luma samples from a column and one chroma sample (see Fig. 2 (a)). In the case of the 4:2:2 sub-sampling, one memory cell holds two luma samples from one row and two accompanying chroma components (see Fig. 2 (b)). When 4:4:4 chroma sub-sampling is used one memory cell keeps whole pixel consisting of one luma and two accompanying chroma samples.

Such a manner of storing samples has many advantages. First of all, for 4:2:0 and 4:4:4 sub-samplings, one memory cell

can hold samples with bit-depths higher than standard 8 bits, even when using simple 32-bit wide memory. This makes the High Profile sample's extended bit depth implementation possible with cheaper hardware. The second benefit is the simplified architecture of the display module. The data is most often sent to the screen using BT.656 standard, in which colour components are transferred also in an interleaved way. Having the data already organised in such a way reduces the number and size of necessary buffers — only one is needed for all data instead of one for each component. The last gain of such an approach is easy to achieve parallel processing of image components in the INTER prediction.

The main drawback of the interleaved approach are trough-put limits for real-time 1080p@30fps sequences processed by the INTER prediction module. To support this kind of sequences, the INTER prediction module should accept four samples per clock cycle from an input (see section IV-F), which is nearly impossible with such a way of storing samples. On the other hand, INTER module capable of taking four input samples in a clock cycle would surely be too big to meet the assumed maximal resource usage (see section III-A).

IV. Architecture

A. Entropy Decoder

Sequence/Picture control and slice data are embedded in separate NAL (Network Abstract Layer) units. This provides the simple internal synchronisation as each NAL unit starts with a unique byte sequence which must not appear within a NAL code-stream. After detecting a new NAL unit an appropriate decoding schema is started according to the order specified in the standard. The architecture consists of four major parts: the elementary stream buffer, the Finite State Machine (FSM) controller with headers parser, the Exponent-Golomb decoder, and the CAVLC decoder. The design provides the simple interface to the arithmetic decoder module. When syntax elements are coded using the CABAC mode, the FSM controller disables other processing units (e.g., CAVLC) and forward stream data to the CABAC path using the hand-shake.

1) Elementary Stream Buffer: The architecture of the elementary stream buffer is shown in Figure 3. It communicates with three alternative decoding paths and synchronises them with the external data source.

The buffer loads new data whenever they are available on the input and there are enough invalid bits in the buffer. In each clock cycle, the decoder needs a variable number of bits to parse appropriate parameter. When there is not enough valid data in elementary stream buffer the work of the module is halted.

2) Syntax Element Parser: The superior module of the proposed H.264/AVC binary decoder is the FSM controller, which controls the order of syntax elements. To synchronise the decoder with the input stream, the FSM looks for the NAL start sequence in the elementary stream buffer and detects the NAL type by reading the following byte. In dependence on the type, an appropriate decoding process is invoked. There are separate paths for each NAL type. The paths include states

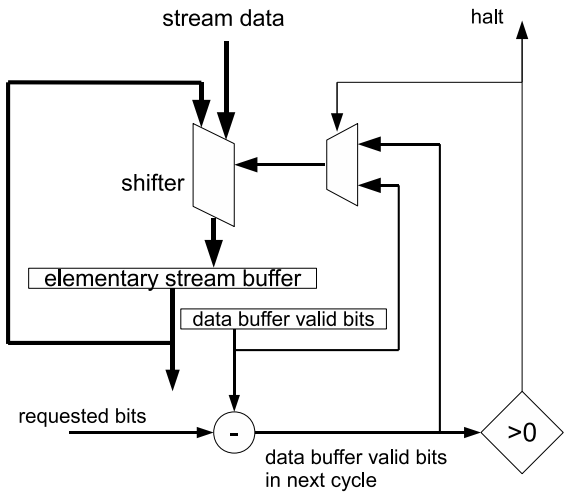


Fig. 3. Proposed architecture of elementary stream buffer.

associated with successive syntax elements embedded in the code stream. Transitions between states follows the order of syntax elements specified in the H.264/AVC standard.

3) Exponent-Golomb Decoder: The submodule refers to the stream buffer and detects the number of leading zeros. In fact, this detector supports the unary code used also for other syntax elements. The number of leading zeros is used to retrieve and normalise the suffix in the cascade of two multiplexers. Particularly, the multiplexers shift the input down and insert sequence of zeros on the MSB positions. Instead of determining the value of an appropriate power of two, the normalised string contains the concatenated one-value bit on the MSB position (e.g., taken from the input). To get the proper code number, the normalised string is decremented. Thank to the simplified structure (combined retrieving and normalisation and the decrementation), the circuit has the reduced complexity compared to the straightforward implementation

4) CAVLC Decoder: The sequential nature of the binary-decoding algorithm imposes the general structure of architecture, which is well-described in literature for the CAVLC decoder. Particularly, there are two optimisation methods for the basic CAVLC decoder - the multi-symbol decoding for high throughputs and the simplification of coef_token tables to save resources. CAVLC coefficient token is decoded using six tables (four for luma and two for chroma). An appropriate table is chosen by the number of non-zero coefficients in neighbouring blocks. Only one of those tables can be replaced with simple arithmetic operations (for $nC > 8$). Other luma tables are rather complex and require a large address space to map directly all binary representations. The detailed analysis of the coefficient-token codes allows us to notice that they can be represented as a number of leading zeros followed by a suffix composed of maximally three bits. Since the number of leading zeros cannot exceed 14, four bits are enough to represent the prefix length. As a consequence, only 7 address bits are needed to specify all coefficient token tables, which is much less than in the straight-forward representation.

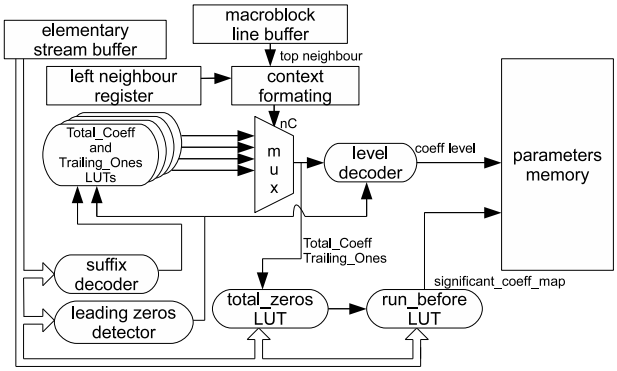


Fig. 4. Proposed architecture of CAVLC decoder.

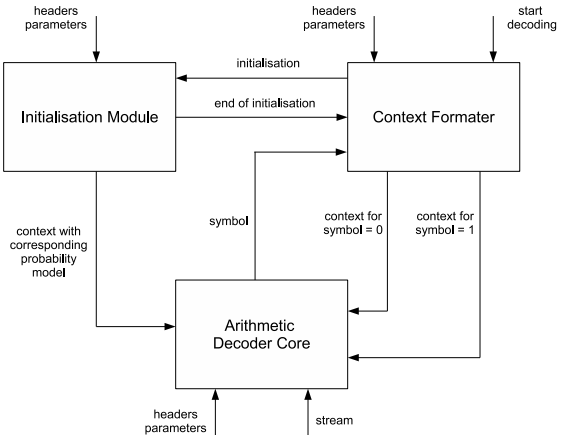


Fig. 5. Proposed CABAC decoder architecture.

B. CABAC

H.264/AVC provides three different modes for the binary arithmetic coding: non-equiprobable, bypass and terminate. In the most important non-equiprobable mode, the appropriate context must be provided to decode an individual symbol. This context is determined with reference to the current syntax element, the symbol position, previous symbols, left and/or top neighbours and header parameters. Its value allows a selection of the correct probability model, consisting of the state index and the More Probable Symbol (MPS) value. In dependence on the current index and two MSB bits of the range register, the H.264/AVC standard specifies the division of the range into two subranges, corresponding to Less and More Probable Symbol values (LPS/MPS). In the decoder, the chosen probability estimate is subtracted from the range register and compared with the content of the offset register. The result of this comparison identifies the decoded symbol, determines the update of the selected probability model and the context for the next symbol. To keep the required precision, the range and offset registers have to be renormalized. To restore an original syntax element value, successive symbols are grouped and mapped to numbers.

The main difficulty in designing an efficient CABAC architecture is the inter-symbol dependency, which creates a tight feedback loop. Particularly, a proper model is necessary to decode the next symbol, and the decoded symbol is needed to prepare the context that selects following probability model. These constraints exclude employing pipelining to accelerate the design. The proposed solution is based on the fact, that an incoming symbol may take only two values: 0 or 1. As a result, it is possible to avoid waiting for this bit at the context generator by the calculation of two possible contexts in advance [3].

The design consists of three separate modules: the initialisation module, the arithmetic decoder core and the context generator, which is presented in Fig. 5. The first element restores probability models for all contexts at the beginning of each slice, using parameters m and n stored in the ROM memory. The model contains the MPS value and the index, identifying LPS probability estimate.

The primary task of the arithmetic decoder core is to decode each symbol in one clock cycle, regardless of selected routine. In order to save hardware resources all symbols are decoded using the same logic. For bypass mode it is realised by employing several multiplexers, whereas the termination mode exploits a suitable probability model for the context equal to 276. Decoded symbol selects the appropriate context and the bypass mode flag, which allows reading of the probability model for the following symbol from the dual-port RAM memory. Simultaneously an update of probability model is prepared and stored in the same memory.

The context generator is designed to prepare two alternative contexts and two bypass mode flags. In addition, this module controls the work of the arithmetic decoder and restores original syntax elements from the sequence of successive decoded symbols. Syntax elements are processed in the order defined in the H.264/AVC standard. Therefore, the CABAC architecture incorporates a Finite-State Machine (FSM) to identify the type of the syntax element and the order of processed data. Two alternative sets of information for each possible symbol value are prepared in each clock cycle. Each set contains the new FSM state, parameters describing the current macroblock and the syntax element value. The multiplexer, controlled by the currently decoded symbol, chooses one of those sets, which permits the correct data to be stored in registers and transferred outside. Based on these two sets and neighbour variables, the context generator prepares two corresponding context and bypass mode flags for next symbol decoding. A register contains variables decoded for the left macroblock, whereas parameters describing the top neighbour are read from the dual-port memory, keeping data for the whole line above the current macroblock.

C. Data Restoration Module

Data received from the entropy decoder must be adjusted to the troughput of further modules. The proposed solution consists of two independent units and an arbiter, responsible for management of access to the RAM memory with decoded syntax elements from current macroblock. First unit performs reverse coefficients' zyg-zag reorder, whereas the second ex-

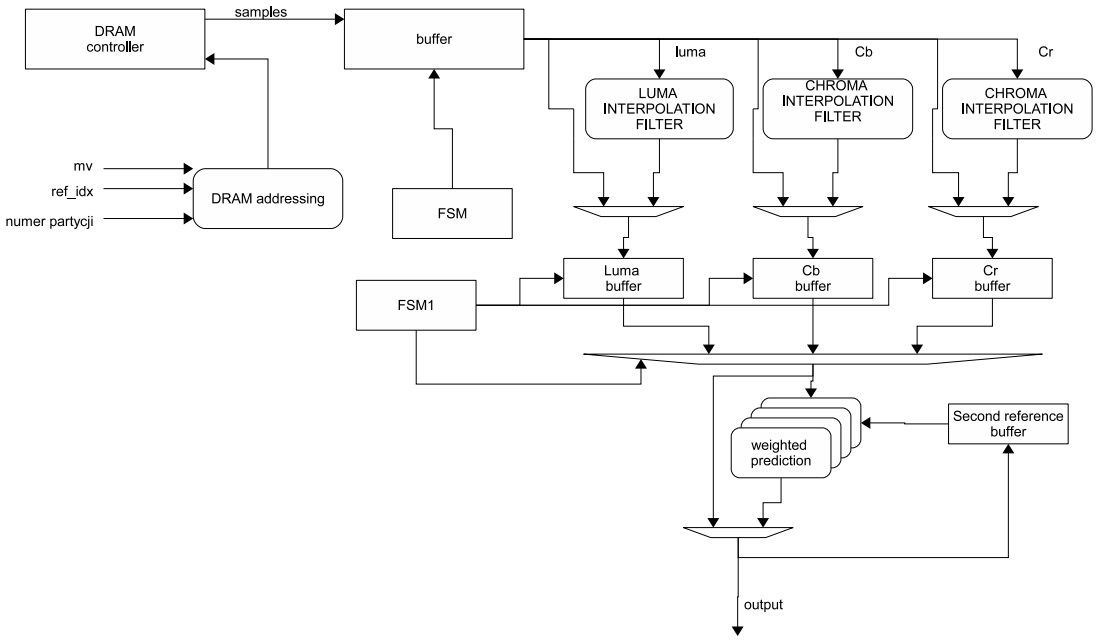


Fig. 6. Inter prediciton module architecture overview.

tracts original values of prediction parameters, such as INTRA prediction modes or motion vectors with frame indexes, and calculates strength for the loop filter for each border within the macroblock. All tasks of the second module requires access to the data from the previous macroblocks. This is solved by the storage of parameters describing left neighbour in a set of registers and employing a special memory to keep necessary information about the upper line of macroblocks. Each memory cell keeps data corresponding to a single partition from the lowest line of the macroblock.

An arbiter manages access to the RAM memory with syntax elements. Usually, the first unit has the priority, as a specific number of coefficients is required to fill up the inverse transform buffer. As many coefficients are equal to zero, there is no need to check data from the memory for them. This allows us to read prediction parameters during that time. The second unit gets the higher priority only if the inverse transform buffer is full.

D. Inverse Transform

The H.264/AVC High Profile allows the use of two transform sizes: 4x4 and 8x8. To simplify the dataflow, it seems reasonable to work only on the 8x8 block basis, which can be easily achieved by grouping four 4x4 blocks into one 8x8 block. Though the 4x4 and 8x8 transforms' definitions are not identical, such a grouping makes resource sharing between this two modes easier. As a consequence, the transforms takes columns of eight samples from the input and returns columns of eight samples as a result.

E. Dequantiser

The dequantiser module is responsible for the dequantisation of the coefficients fed by the entropy decoder module.

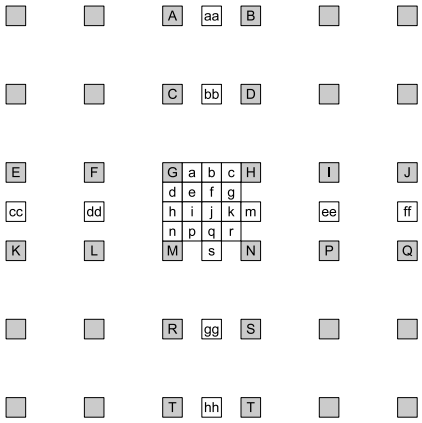


Fig. 7. Luma interpolated pixel positions. Grey samples — original samples; aa,bb,b,j,s,gg,hh,cc,dd,h,m,ee,ff — half-pixels; the rest — quarter-pixels.

One coefficient is dequnatised at a clock cycle. Afterwards, the dequantised coefficients are buffered to form columns of eight elements, which are sent to the inverse transform module.

F. Inter Prediction

The INTER prediction module is one of the most resource-consuming and computational-demanding operation in the whole H.264/AVC decoder. This comes from the fact that the standard allows motion vectors with the quarter-pel precision. Such a precisions requires the interpolation of half and quarter pixels from reference sample values. The locations of all possible interpolated pixels is presented in Fig. 7. The interpolation is a two stage process. First, half-pixels are interpolated from six original samples values arranged either in a column or a row. In the second stage, the quarter-pixels are computed as a mean of two selected half-pixels and original pixel values. In general, to obtain one interpolated quarter-pixel value an array

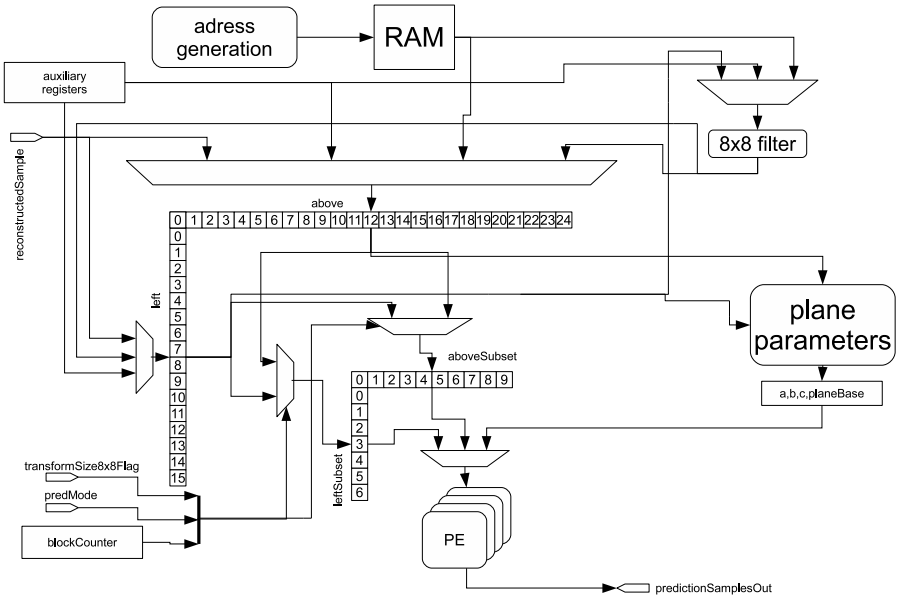


Fig. 8. Intra prediction module overview.

of 6x6 original pixels has to be used. The standard permits the use of prediction blocks of different sizes, varying from 4x4 to 16x16. The way the interpolation works means that for a 4x4 block there are 5 time more input samples than output values. For the larger blocks, this ratio decreases and for 8x8 blocks it is equal 2.6. The chroma interpolation process is much simpler, as it uses simple linear approximation from the four neighbouring samples.

To fully support real-time processing of 1080p@30fps sequences, INTER prediction module should take four samples per clock cycle from the input and incorporate four parallel luma interpolators. As this would use more resources than were available to us, it has been decided that only one luma interpolator will be used, which is enough to support 720x576 sequences. To further reduce the size of the buffer used by the interpolator unit, the maximal block size that can be processed is 8x8. The blocks of larger sizes are simply divided into 8x8 blocks, which are processed consecutively.

The overview of the module architecture is presented in Fig. 6. At the beginning, the external memory address is determined to fetch appropriate prediction samples. The downloaded data are buffered and sent to the interpolation modules, if necessary. Note that in the case of chroma 4:4:4 sub-sampling, the chroma component interpolation is done in the same way as luma interpolation, in contrast to the other sub-samplings. This means that to process all sub-samplings two sets of chroma interpolators are required. If the prediction does not uses the pixel interpolation, interpolation modules are bypassed. Following the interpolation, the sample storage order is changed from an interleaved to separate colour components, as the reconstruction and the transformation store data in this way. This is done by a simple buffering of samples of different components in three separate buffers. If the weighted prediction is required, the sample goes through weighting prediction module. Otherwise, they are sent directly to the output. In the case of the prediction obtained from two reference blocks, the interpolated samples from the first block are also buffered locally, waiting for weighting with the samples from the second reference block, as soon as they are ready.

G. Intra Prediction

The designed INTRA prediction module supports all INTRA prediction modes defined in the standard: 4x4,8x8, and 16x16. It is based on a widely used architecture consisting of four parallel processing elements (PE) [5], [12], [2], [11]. Each processing element is capable of calculating exactly one prediction per clock cycle. The overview of the module is presented int Fig. 8. As it can be seen in the Fig. 8, except for PEs there is also separate module for calculating INTRA 16x16 *plane* parameters. It is because for the calculation of this mode requires some entry parameters, which can not be easily and timely computed using solely PEs.

A parallel calculation of four samples is necessary to ensure that the average throughput of the module is at least one prediction per clock cycle. This is the consequence of the dataflow defined in H.264/AVC. The INTRA prediction usually operates on data blocks smaller than macroblock, and samples necessary for prediction are available only after the calculation of prediction and reconstruction of the previous data block. The referred samples are reconstructed samples neighbouring with the block from the left side and above. In the case of the 8x8 prediction modes, these samples undergo a simple low-pass filtering process.

The transfer of the neighbouring samples from the reconstruction module to the INTRA prediction module takes significant amount of time, which cannot be used for the prediction. The samples are transferred one at a clock cycle, to reduce the complexity of the 8x8 mode pre-filtering, as in such case it is needed to filter only one sample per clock cycle. Despite slow transfer rates, the calculation of four prediction

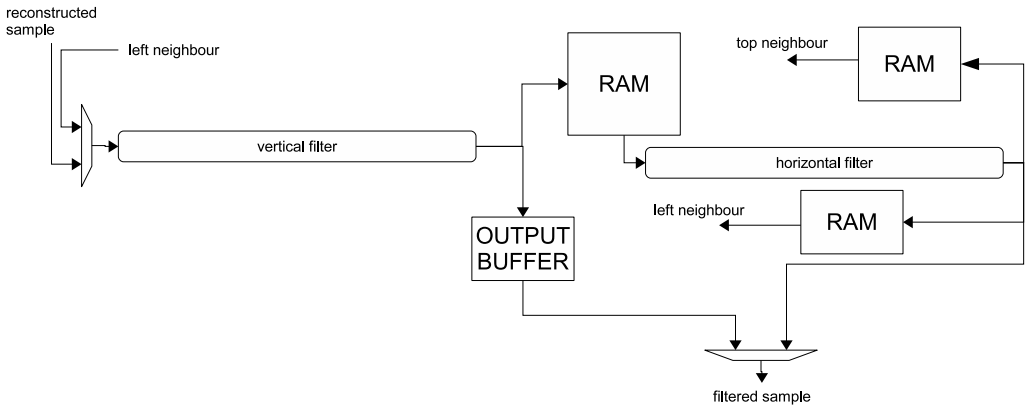


Fig. 9. Loop filter architecture overview.

values in parallel allows us to achieve a throughput of one sample per clock cycle.

In the case of the blocks lying at the macroblock's top border, the samples neighbouring above belong to the macroblock adjacent to the currently processed one. Hence, the whole previous image line has to be buffered to calculate the prediction. For the sake of the simplicity and the reduction of external memory access, this line is buffered in the embedded RAM module indicated in Fig. 8. The samples referred for the currently processed macroblock prediction are stored in registers. The first layer of registers keeps all samples that will be used for the prediction. The second layer (registers indicated as "Subset" in Fig. 8) conveys the samples for the prediction of a 4x4 block. Blocks of larger sizes are divided into 4x4 ones. The division into two register layers makes achieving higher operating frequencies much easier.

H. Loop Filter

The loop filter module reads exactly one sample from the input and produces exactly one sample per clock cycle. The filtering process is defined by the H.264/AVC standard to operate on successive macroblocks in a frame. For each macroblock, the filtering of vertical borders takes place first, followed by the filtering of horizontal borders. Which borders are filtered depend on the transform size used, i.e., either 4x4 or 8x8 block's borders. To reach one sample per clock cycle throughput, the architecture is decomposed into two identical and independent filters. The first is responsible for filtering vertical borders, whereas the second filters horizontal ones(see Fig. 9. The samples for the first filter are provided in the raster-scan order: from left to right in a consecutive lines that make up a macroblock. The filters are separated by the RAM buffer responsible for changing data order from the raster-scan to column oriented, which is required by the horizontal filter.

Unfortunately, the dataflow defined by the H.264/AVC standard requires some already horizontally filtered data to be vertically filtered again at the next macroblock border. As a result another RAM buffer is incorporated for such samples. Moreover, access to four lower lines of the macroblock above is necessary to properly filter top macroblock border. Hence, another RAM buffer capable of keeping four image lines is added. The use of an embedded RAM is preferred over the usage of an external memory as typical FPGA has enough memory blocks available, and such a memory is far simpler to use. It also lowers the requirements for the data transfer rates to the external memory.

TABLE III
SYNTHESIS RESULTS OF THE PROPOSED H.264/AVC DECODER FOR THE ARRIA 2 DEVICES

	Without CABAC	With CABAC
Parameter	**Value**	**Value**
ALUTs	26486	31155
Memory ALUTs	183	183
Registers	12953	13758
DSP blocks	11	12
memory used[bits]	583352	704824

V. IMPLEMENTATION RESULTS

The proposed decoder is verified to work using reference data generated by the JM16 H.264/AVC reference software.

The decoder has been synthesised for the relatively-low-cost Altera Arria 2 FPGA devices in two versions: with and without CABAC entropy decoder. The synthesis is performed using Quartus II Version 9.1 and the device used as a synthesis target is EP2AGX125EF29I5 Arria 2. The results are gathered in Table III, and the operating frequency of 100 MHz has been achieved for the version without CABAC. The version with the CABAC module can operate only at about 92 MHz. However, much higher frequencies can be achieved for Stratix 3/4 devices allowing HDTV with B frames. The whole design occupies around 30000 ALUTs in both cases. This resource consumption should leave enough space in an average-size Aria 2 FPGA for some accompanying units like audio decoder and transport stream decoders. This would allow the construction of the whole TV decoding system on a single chip.

Most decoders described in the literature target ASIC chips [8], [1], [4]. Moreover, some of them supports only Baseline Profile [1], and none of them supports High Profile, although all of them claim to support real time decoding of the

1080p@30fps sequences. This differences make the comparison with the proposed decoder, targeted mostly for medium-sized, difficult. Furthermore, the decoders presented in the literature were not compiled for FPGA, so any comparisons of the resource usage is virtually impossible.

VI. Conclusion

The H.264/AVC standard is nowadays the most advanced commonly used video encoding standard. However, its efficiency comes at the price of the increased complexity. As a result, only hardware or hardware supported implementations can provide optimal performance. This paper presents the proposal of an H.264/AVC architecture optimised for FPGA structures, capable of processing in real-time SDTV and HDTV sequences encoded in High Profile. The design is flexible enough to easily upgrade the decoder to support higher resolutions, by modifying an implementation of the INTER prediction unit, but at the expense of significantly higher resource usage.

References

[1] T.-W. Chen, Y.-W. Huang, T.-C. Chen, Y.-H. Chen, C.-Y. Tsai, and L.-G. Chen, "Architecture design of h.264/avc decoder with hybrid task pipelining for high definition videos," in *IEEE International Symposium on Circuits and Systems 2005*, 23-26 2005, pp. 2931 – 2934 Vol. 3.

[2] T.-C. Chen, H.-C. Fang, C.-J. Lian, C.-H. Tsai, Y.-W. Huang, T.-W. Chen, C.-Y. Chen, Y.-H. Chen, C.-Y. Tsai, and L.-G. Chen, "Algorithm analysis and architecture design for hdtv applications - a look at the h.264/avc video compressor system," *IEEE Circuits and Devices Magazine*, vol. 22, pp. 22–31, May–June 2006.

[3] H. Eeckhaut, M. Christiaens, D. Stroobandt, and V. Nollet, "Optimizing the critical loop in the h.264/avc cabac decoder," in *IEEE International Conference on Field Programmable Technology, 2006.*, dec. 2006, pp. 113 –118.

[4] Y. Hu, A. Simpson, K. McAdoo, and J. Cush, "A high definition h.264/avc hardware video decoder core for multimedia soc's," in *Consumer Electronics, 2004 IEEE International Symposium on*, sept. 2004, pp. 385 – 389.

[5] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Analysis, Fast Algorithm, and VLSI Architecture Design for h.264/avc Intra Frame Coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, pp. 378–401, May 2005.

[6] *Recommendation ITU-T H.264(2007) — Corrigendum 1*, Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, January 2009.

[7] *Report of The Formal Verification Tests on AVC (ISO/IEC 14496-10 — ITU-T Rec. H.264)*, JVT, Test and Video Group, December 2003,Waikoloa.

[8] C.-C. Lin, J.-W. Chen, H.-C. Chang, Y.-C. Yang, Y.-H. O. Yang, M.-C. Tsai, J.-I. Guo, and J.-S. Wang, "A 160k gates/4.5 kb sram h.264 video decoder for hdtv applications," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 170 –182, jan. 2007.

[9] D. Marpe, T. Wiegand, and S. Gordon, "H.264/MPEG4-AVC Fidelity Range Extensions:Tools, Profiles, Performance, and Application Areas," in *IEEE International Conference on Image Processing 2005.*, vol. 1, September 2005, pp. 593–596.

[10] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression*. John Wiley & Sons, 2003.

[11] W. T. Staehler, E. A. Berriel, A. A. Susin, and S. Bampi, "Architecture of an hdtv Intraframe Predictor for a h.264 Decoder," in *IFIP International Conference on Very Large Scale Integration*, October 2006, pp. 228–233.

[12] C.-H. Tsai, Y.-W. Huang, and L.-G. Chen, "Algorithm and architecture optimization for full-mode encoding of h.264/avc intra prediction," in *48th Midwest Symposium on Circuits and Systems, 2005.*, vol. 1, August 2005, pp. 47–50.