A Low-Complexity MPEG-2 to H.264/AVC Intra-Frame Transcoder Architecture for HD Video Sequences

Milica Orlandić, *Student Member, IEEE,* and Kjetil Svarstad, *Member, IEEE*

***Abstract*—The wide use of MPEG-2 in TV broadcasting and H.264/AVC in mobile multimedia application creates a need for supporting such standards. We present a novel high-throughput architecture of MPEG-2 to H.264/AVC intra transcoder. The computationally demanding yet regular units of the MPEG-2 decoder have been implemented with a high level of parallelism by processing 8 pixels in parallel, whereas H.264/AVC encoder engine utilizes a 4×4 block level pipeline. The synchronous communication between decoder and encoder and full pipeline of the system are achieved by an intermediate memory buffer mechanism. A wavefront macroblock level scanning order based on on-the-fly processing of a number of consecutive macroblocks and on-chip memory organization are proposed to increase efficiency of intra prediction algorithm. The proposed architecture is synthesized and implemented on Kintex 705-XC7K325T board. Achieved results represent a significant reduction of minimal required frequency compared to the state of the art for resolutions CIF, SD and HD1080p. Furthermore, the transcoder implementation supports QFHD resolution and requires 62 MHz to encode QFHD 2160p at 60 fps, whereas its encoder stage has maximal throughput of 1744 Mpixels/s that corresponds to processing of UHD 4320p resolution at 30 fps.**

Keywords - H.264/AVC encoder, MPEG-2 decoder, transcoder, intra prediction, FPGA, hardware implementation, reconstruction loop

I. Introduction

Data compression enables a communication system to transfer more information by removing redundancies present in the incoming data. It significantly reduces bitrates and thus the cost of communication. The variety of networks and user devices demand video streaming bitrate to adapt to limitations in communication networks such as channel bandwidth, and to capabilities of video devices such as screen size, storage or energy level. Numerous video compression standards have been proposed during the past decades, and the recent H.264/AVC standard outperforms prior video encoding standards due to the ability to achieve lower bitrates while retaining the quality. Coding efficiency plays a crucial role in current multimedia systems as the bandwidth requirement for video transmission in recent video applications constantly increases. Real-time streaming of video sequences is an additional requirement. As the number of networks, types of devices and content representation formats increase, interoperability between different systems and networks becomes more important. This creates a need for devices such as environment tags, multipoint control units, or servers that support the transcoding process between different standards and in particular video compression standards. The transcoding process converts a pre-coded bitstream into a bitstream that meets the bandwidth limitation conditioned by the network or by energy level of the end user. Special attention is given to transcoding between MPEG-2 and H.264/AVC. H.264/AVC is widely present in network and mobile multimedia applications, whereas MPEG-2's presence is significant in TV broadcasting. Though H.264/AVC is highly efficient compared to MPEG-2, the wide use of MPEG-2 creates a need for the coexistence of these technologies and transcoding mechanisms.

Low computational complexity is important for transcoder architectures since both video decoding and encoding are performed. Moreover, H.264/AVC and MPEG-2 compression algorithms differ in the transform coding, so the complexity reductions achieved in MPEG-2 to MPEG-4 transcoding are unsuitable. Published literature on MPEG-2 to H.264/AVC transcoding is sparse, particularly directed towards hardware implementation. The classification model SVM [1] is used to build a mode decision classifier for a low-complexity transcoding, whereas DCT coefficients are converted into H.264/AVC transform coefficients in [2] by use of additional matrix transformations. Hardware implementation of these hybrid transformations is computationally expensive and complex. Most common transcoding complexity reductions are achieved by mode decision techniques for intra prediction by the use of DCT coefficients [3], [4]. However, cascaded decoder-encoder methods result in best quality but their software implementations show high complexity, high processing time and consequently significant delays. High level of parallelism, a capability of a reconfigurable hardware medium, decreases substantially processing time and introduced delays, thus enables implementation of fast transcoding techniques.

The efficiency of the H.264/AVC standard compared to prior video standards is achieved at the expense of higher computational complexity. A number of advanced techniques are used in H.264/AVC such as improved directional spatial prediction for intra coding [5]. The standard offers higher flexibility and enables an accurate prediction based on the assumption of spatial similarity between blocks. The strong data dependency between consecutive 4×4 blocks and high reconstruction loop latency cause an increased number of cycles spent on processing one MB. Moreover, very intensive data transfers cause the memory access bandwidth to be a

M. Orlandić and Kjetil Svarstad are with the Department of Electronics and Telecommunication, Norwegian University of Science and Technology, Norway, e-mail: milica.orlandic@iet.ntnu.no.

Manuscript received Month ?,

severe performance bottleneck in current video processing systems.

Extensive research has been made on computational complexity reduction techniques of intra prediction algorithm such as an optimization of the 4×4 block processing order [6], early termination approach [7], edge classification [8], edge direction histogram [9] and three step intra prediction algorithm [10]. Furthermore, massive parallel architectures, such as Graphic Processor Unit (GPU) that accommodate a high number of parallel processors are used in a number of works [11]–[14] for data-parallel processing schemes for block-based intra prediction algorithms. Throughput is main target of these systems, so powerful platforms such as NVIDIA CUDA Platform [15] with 480 streaming processors are used in [11], [14]. However, synchronization between processors is costly due to increasing number of processors employed to achieve higher resolutions, whereas the sequential speed of these processors is low.

A number of efficient VLSI architectures for H.264/AVC intra-only encoder have been reported in [16]–[23]. Most of the implementations increase the encoder performance by dealing with the system bottleneck-intra 4×4 prediction. The first intra-frame H.264/AVC encoder [16] lowers bubble cycles by interlacing intra algorithms, but its throughput is limited to SD video format. Mode decision implementations in block-wise intra prediction algorithm are speeded up by modified 3-step technique in [17], by quality scalability algorithm [18] that adjusts complexity of intra 4×4 algorithm depending on video application or by TraDED technique in [22] based on transform characteristics. Several approaches for improving coding performance in [20] are focused on 4×4 block data-dependancy. A block-level and mode-level co-ordering strategy for intra 4×4 prediction is employed in [21]. High 64-pixel input parallelism is used in [23] to implement fast H.264/AVC High Profile Intra encoder. The high throughput that corresponds to ultra HD video is achieved by employing high input-parallelism and two prediction algorithms of sizes 8×8 and 16×16, thus by excluding intra 4×4 prediction.

FPGA boards are characterized by limited resources in respect to the memory, logic and IO, so the FPGA solutions are supposed to be highly area-efficient and to maximize the high throughput with the limited IO resources. Furthermore, architecture is characterized by low one-directional control flow between stages in order to support reconfigurability. Xilinx [24] proposes its own H.264/AVC encoder that supports all-intra capability. An FPGA video encoder [25] with intra 4×4 prediction can support 1080i video sequences, whereas implementation in [26] processes HD sequences of 1080p resolution at 60 fps. Intra 4×4 prediction algorithm computational complexity is lowered by exploiting the pixel equality and pixel similarity in [27]. However, the statistical analysis on image quality losses introduced by this technique is limited to CIF sequences. Our previous work [28] explores the idea of intra 4×4 prediction algorithm module reconfiguration in the case of variable throughput.

The goal of this paper is to implement a real-time MPEG-2 to H.264/AVC transcoding architecture that targets high definition sequences. If resources are limited and real-time transcoding of HD video sequences up to 2160p is a requirement, reduced complexity is necessary. We show that a highly pipelined structure supported by fast on-chip memory organization and by maximal parallel processing conditioned by technology limitations lowers the operating frequency, thus achieves the throughput that corresponds to ultra HD resolutions.

The paper is organized as follows. In Section II, the decoder and encoder structures are analyzed, in particular the intra prediction algorithm. A novel parallel hardware architecture of an MPEG-2 to H.264/AVC transcoder is proposed in Section III. The implementation results and performance comparison are presented in Section IV. Finally, conclusions are summarized in Section V.

II. Background

The objective of transcoding is to adapt the video stream to the end user capabilities. In the case of MPEG-2 to H.264/AVC transcoding, bitrate reduction is achieved while the high quality is maintained. A cascaded video transcoder is composed of a decoding stage followed by an encoding stage. Its structure is shown in Fig. 1. The MPEG-2 bitstream is decoded first to produce the reconstructed image, then the resulting image is used as source for the encoding stage. The generated H.264/AVC bitstream is the output of the complete transcoder.

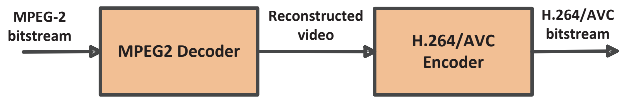

Fig. 1. Block diagram of Cascaded MPEG-2 to H.264/AVC Transcoder

A. Decoder

The computationally intensive units of an MPEG-2 decoder such as transform and quantization are relevant for hardware implementation due to their regular structure. The purpose of a linear transform is to decorrelate the original signal, resulting in redistributed energy concentrated in a number of coefficients. The 8×8 discrete cosine transform (DCT) has been used in many international video and image compression standards such as H.26x, JPEG, and the MPEG family. The compact matrix form of 2D IDCT transform is given by:

$$X_{idct} = \mathbf{C}^T\mathbf{X}\mathbf{C}, \tag{1}$$

where $\mathbf{X}$ is the inverse DCT transform matrix and $\mathbf{C}$ is a cosine matrix. It shows that two dimensional discrete cosine transform can be decomposed into two one dimensional transforms - along rows and columns. To achieve fast hardware implementation, floating point coefficients are scaled and approximated by integers. The inverse quantization consists of three steps: the inverse quantization itself, the saturation and mismatch control. The inverse quantization of AC coefficients is defined by:

$$X_{iq}(k_1,k_2) = \frac{(2\times X_{idct}(k_1,k_2)+k)\times QF(k_1,k_2)\times scaler}{32}, \quad (2)$$

where $k = 0$ when intra blocks are processed, and **QF** is quantization matrix. The factor *scaler* is derived from the data elements *quantizer_scale_type* and *quantizer_scale_code* according to ITU-T Recommendation H.262 [29]. Depending on the parameter *quantizer_scale_type*, *scaler* gets the values from one of the sets in the range $[1, 112]$. Furthermore, the saturation stage is performed in order to place the coefficients in the range $[-2048, 2047]$, whereas mismatch control reduces the drift between DCT and inverse DCT. This step attempts to eliminate bit patterns that have the highest contribution to mismatches by modifying the least significant bit of the AC coefficient with the highest frequency.

B. Encoder

The block diagram of H.264/AVC intra encoder is presented in Fig. 2. The H.264/AVC video standard partitions the frame into macroblocks (MB). An MB contains 16×16 pixels and is further partitioned into 16 sub-blocks of 4×4 pixels. There are a number of prediction modes with respect to prediction direction and block size. Three types of intra prediction are defined: Intra 4×4, 16×16 luminance and Intra 8×8 chrominance predictions in main, baseline and high profiles. High profiles can include additional Intra 8×8 luminance prediction. Luminance 16×16 and chrominance 8×8 predictions have four prediction modes, whereas luminance 4×4 and 8×8 have nine different prediction modes.

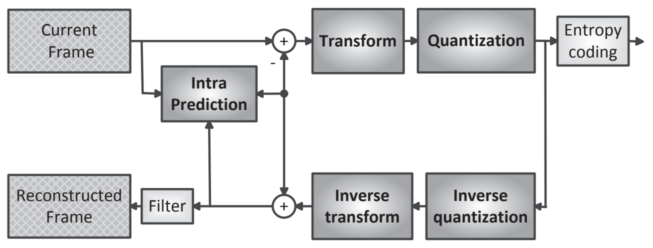

Fig. 2. Block diagram of H.264/AVC Intra Encoder

Nine intra 4×4 prediction modes (DC mode (mode 2) and eight directional modes) are illustrated in Fig. 3. In case of vertical and horizontal mode predictions, pixels are generated by copying the reconstructed pixels from upper or left blocks. DC mode is a mean value of the upper and/or left reference samples depending on block availability, whereas the diagonal group of modes are composed of N-tap filters given by:

$$Pred = (Eq + R) \gg S, \quad (3)$$

where Eq is the filter equation, R is rounding addition and S is shift operation. Most of the results of 2-tap equations are reusable for calculating the 3-tap filter equations. For example, prediction pixels b and c in VR mode are calculated by:

$$Pred_b = (Eq_b + R_b) >> 1, Pred_c = (Eq_c + R_c) >> 1, \quad (4)$$

where $Eq_b = a + b$, $Eq_c = b + c$ and R_b, $R_c = 1$. The intermediate values $Eq + R$ can be reused for calculation of 3-tap filters such as for calculation of prediction pixels a, h and g in prediction modes DDL, DDR, VR, respectively. In that case the 3-tap filter equation has a form $Pred_{a_{DDL}} = (Eq_b + R_b + Eq_c + R_c) >> 2$. After generation of prediction pixels, the prediction mode with the minimum value of the cost function Sum of Absolute Differences (SAD) is selected to be the best mode. The difference between the prediction and the actual pixel value, residual or prediction error, is output of the prediction module and is used in further processing.

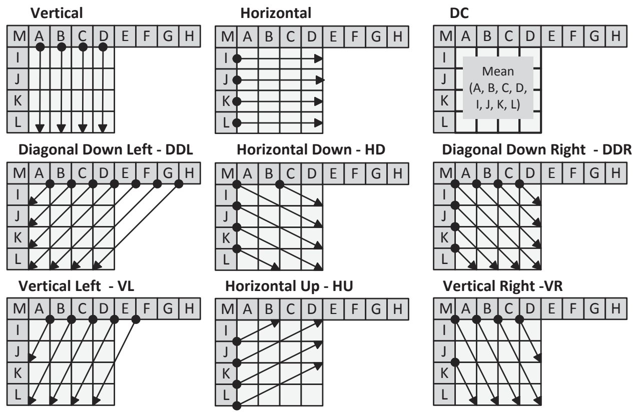

Fig. 3. Directional modes for Intra Prediction

The dependency chain is introduced due to the data dependency between blocks in the intra prediction process. Data dependency among 4×4 blocks is found to be the performance bottleneck of the whole system. Scanning order rearrangement of intra 4×4 blocks can improve the processing time to some extent. In [6] ten alternative scanning orders of sixteen 4×4 blocks within a macroblock are compared. The conventional and two alternative scanning block orders are depicted in Fig. 4. Block B (labeled 1) depends on Block A (labeled 0) since Block A contains pixels required to calculate predictions of Block B. Prediction of Block B starts after the completion of both intra prediction and reconstruction loop of Block A. Prediction of Block C (labeled 2) follows after the complete reconstruction process of Blocks A and Block B.

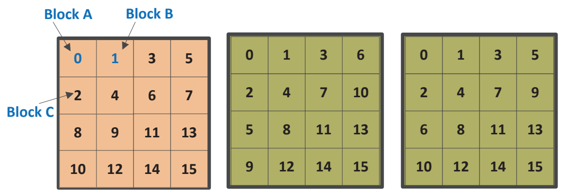

Fig. 4. Block ordering within one macroblock

C. Reconstruction loop

This stage consists of integer DCT, quantization and reconstruction path. The residual is transformed and quantized in order to decorrelate the pixels and map the results to a smaller set of values, whereas the reconstruction path generates reference samples needed for intra prediction. The H.264/AVC standard adopts integer DCT that is performed on 4×4 blocks and its implementation is simplified compared to the floating point 8×8 DCT used in MPEG-2 since it requires only addition and shift operations. The forward transform and quantization are defined as:

$$\mathbf{Y} = \mathbf{C_f X C_f^T}, \tag{5}$$

$$\mathbf{W_{i,j}} = \mathrm{sign}\left(\mathrm{abs}\left(\mathbf{Y_{i,j}}\right)\mathbf{M_f} + f\right) \gg qbits \tag{6}$$

where $\mathbf{X}$ is a 4×4 residual block, $\mathbf{C_f}$ is 4×4 integer DCT transform matrix, whereas parameter *qbits* is defined by:

$$qbits = 15 + floor(QP/6). \tag{7}$$

The inverse transform and quantization are presented by:

$$\mathbf{W'_{i,j}} = round\left(\mathbf{C_i}\left[\mathbf{W_{i,j}} \cdot \mathbf{V}\right]\mathbf{C_i^T}\right)\frac{1}{2^6}. \tag{8}$$

The scaling factors $\mathbf{M_f}$ and $\mathbf{V}$ in quantization and inverse quantization depend on quantization parameter QP and the position of the transform coefficient within 4×4 block [30]. The quantization parameter QP can take values in the range $[0, 51]$ and is an additional input argument of the transcoding system. The output of the system are quantized coefficients that are sent out to be encoded by entropy coder for further transmission. Quantized coefficients are also sent back to the reconstruction path for prediction purpose. Reconstructed residual and best mode prediction pixels are summed and used for the prediction of neighboring blocks. In the period between intra prediction processes of two consecutive blocks, in particular during the reconstruction phase, the intra prediction hardware remains idle. Hence a fast execution of the reconstruction path is an important requirement for the improved performance of the total system.

III. Hardware Architecture

The MPEG-2 to H.264/AVC video transcoder produces an output bitstream compliant to the H.264/AVC video bitstream syntax. The goal of the proposed design implemented on FPGA is a real-time transcoding targeting wide range of video resolutions that eases integration in current applications. The FPGA implementation of a low-complexity transcoder system implies an efficient low-area solution in order to place design within a single chip, to lower the power consumption and to provide basic design for complex reconfigurable system. Furthermore, the high throughput is an important requirement due to an increasing number of video sequences with high definition resolutions such as HD, QFHD and ultimate UHD. The top level hardware organization of the proposed transcoder design is shown in Fig. 5. It consists of inverse 8×8 DCT, inverse quantization and the memory buffer that rearranges the reconstructed pixels between decoder and encoder as parts of MPEG-2 decoder, the intra prediction unit and reconstruction loop engine from the H.264/AVC encoder.

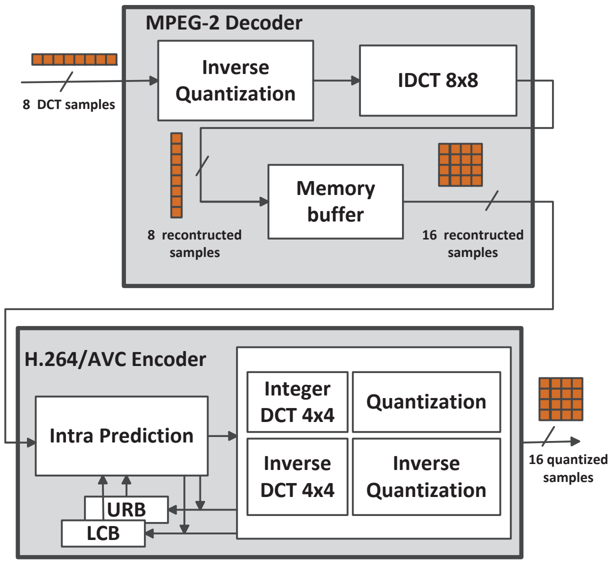

Fig. 5. Architecture of the proposed transcoder

A. MPEG-2 Decoder Implementation

The MPEG-2 intra decoder is an open-loop system where the bitstream is variable-length decoded (VLD) to extract the quantized DCT coefficients. The MPEG-2 quantized DCT coefficients are input to the proposed transcoder design while the variable-length coding is performed in the software due to its irregular structure and intense data-mapping. The quantized coefficients are sent to an inverse quantifier to compute the original DCT coefficient values. The proposed architecture for inverse quantization is shown in Fig. 6. The complete operation for one row of 8×8 is performed in 2 clock cycles. Firstly, both quantized DCT coefficients and parameters *quant_scale_type* and *quant_scale_code* are read. Eight quantized DCT coefficients, simultaneously sent as input, correspond to a row of the 8×8 block. In the first clock cycle, the values of row $\mathbf{QF}(i)$ based on the position within a block are fetched from ROM memory. The parameter $scaler$ that depends on the input parametes $quantizer\_scale\_code$ and $quantizer\_scale\_type$ is also fetched from the memory. The product P of two parameters $\mathbf{QF}(i)$ and $scaler$ is then calculated. The multiplication of the quantized DCT values and product P is performed in the second clock cycle followed by shifting the result by 5. The resulting coefficients are then saturated to lie in the range $[-2048, 2047]$. The mismatch control operation is performed in the ninth clock cycle when quantization of the entire 8×8 block is finished.

The 2D inverse DCT transform can be implemented by two 1D DCT computations performed on each row of the original block and then on each column of the intermediate result. Proposed 2D IDCT design performs Chen's algorithm [31] by processing 8 samples in parallel. The input data corresponds to a row of 8×8 de-quantized block. The input vector $(x_{00}, .., x_{07})$ is multiplied by 8 columns of the multiplication matrix $\mathbf{C}$ resulting in the output vector $(y_{00}, ..., y_{07})$ as shown

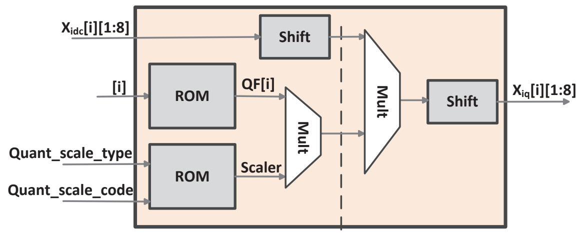

Fig. 6. Inverse Quantization architecture

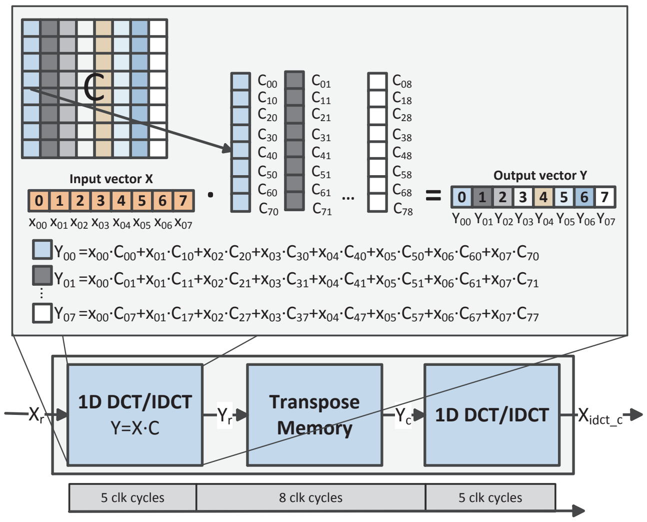
C
Input vector X
Output vector Y
Y00 =x00·C00+x01·C10+x02·C20+x03·C30+x04·C40+x05·C50+x06·C60+x07·C70
Y01 =x00·C01+x01·C11+x02·C21+x03·C31+x04·C41+x05·C51+x06·C61+x07·C71
Y07 =x00·C07+x01·C17+x02·C27+x03·C37+x04·C47+x05·C57+x06·C67+x07·C77
1D DCT/IDCT
Y=X·C
Transpose Memory
1D DCT/IDCT
5 clk cycles
8 clk cycles
5 clk cycles

Fig. 7. Block diagram of Inverse DCT in MPEG-2 Decoder

in Fig. 7. Computing the multiplication operation of an input vector and the multiplication matrix takes 5 clock cycles. Remaining seven rows of 8×8 block are processed in the same way and the results of the 1D IDCT are stored in a RAM memory with a delay of one clock cycle per row. The second 1D IDCT is performed on the samples stored in the RAM by performing vector multiplication of the intermediate matrix $\mathbf{Y}$ and the transposed multiplication matrix.

The design has two RAM blocks and each block can store one 8×8 block of data. When writing operation of resulting vectors into one RAM is completed, data that corresponds to block columns is read from the same RAM block. The calculation of the 1D IDCT for the another 8×8 block continues concurrently with reading data from the first RAM. Intermediate results of the 1D DCT are then stored in the second RAM. These two memories alternate write/read operations in time: while one is being read, the writing process takes place in the other. Writing one 8×8 block into RAM memory requires 8 clock, whereas 2D IDCT processing of one macroblock is completed in 49 clock cycles.

B. H.264/AVC Encoder

There are two stages, prediction and reconstruction, in the H.264/AVC intra encoder. Unlike conventional video encoding where the two engines are clearly separated, in H.264/AVC standard computation of prediction samples depends on the results of the reconstruction engine. High speed implementation of the encoder is a challenge due to the high data dependency. FPGA implementation is supposed to follow additional requirements such IO limitation, full usage of available logic and power consumption, so low complexity is of high importance due to heavy computational load of the encoding algorithm. The most suitable degree of parallelism required for intra prediction to meet different video specification is obtained as a tradeoff between IO/area constraint and high throughput requirement. Moreover, higher parallelism requires less number of processing cycles for the reconstruction engine, indirectly providing higher hardware utilization since the prediction engine decreases idle time for the reconstruction results.

In the conventional scanning, neighboring macroblocks are processed one by one due to their data dependency. The block-wise mode decision mechanism decides which among two intra prediction algorithm results are further processed. The prediction process of the right neighbor macroblock cannot start until the reconstruction of the current macroblock has been computed. This highly influences the pipeline efficiency due to the idle state of one of the units (intra prediction unit itself or reconstruction loop). With constant increase of the resolution this problem becomes critical when real-time processing is requested. The higher throughput is required due to the limited processing time when videos with resolutions such as QFHD and UHD are encoded. It can be achieved to some extent by lowering the complexity of the system by a number of techniques reported in the state-of-the-art. The proposed design relies on the assumption that the throughput is significantly increased when the macroblock dependency is removed. The fully pipelined transcoding system with low latency can be achieved by employing only 4×4 intra prediction. By removal of intra 16×16 prediction algorithm, there is no need for block-wise mode decision process, large number of buffers for storing intermediate macroblock prediction data for both algorithms and control logic.

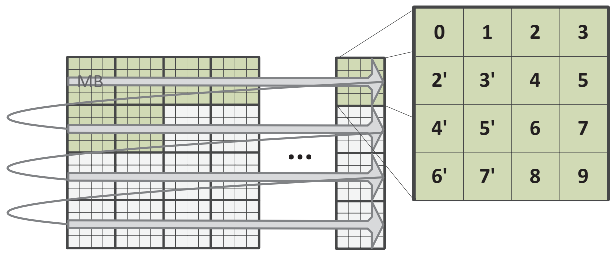

Fig. 8. Proposed macroblock and 4×4 block ordering

When the mode decision operation is discarded due to the use of only intra 4×4 prediction, there is no need for conventional scanning order. Furthermore, when the reconstructed neighboring samples required for prediction of 4×4 blocks are available, several blocks within few macroblocks can be processed simultaneously. The wavefront scanning block reordering proposed in our design is presented in the Fig. 8.

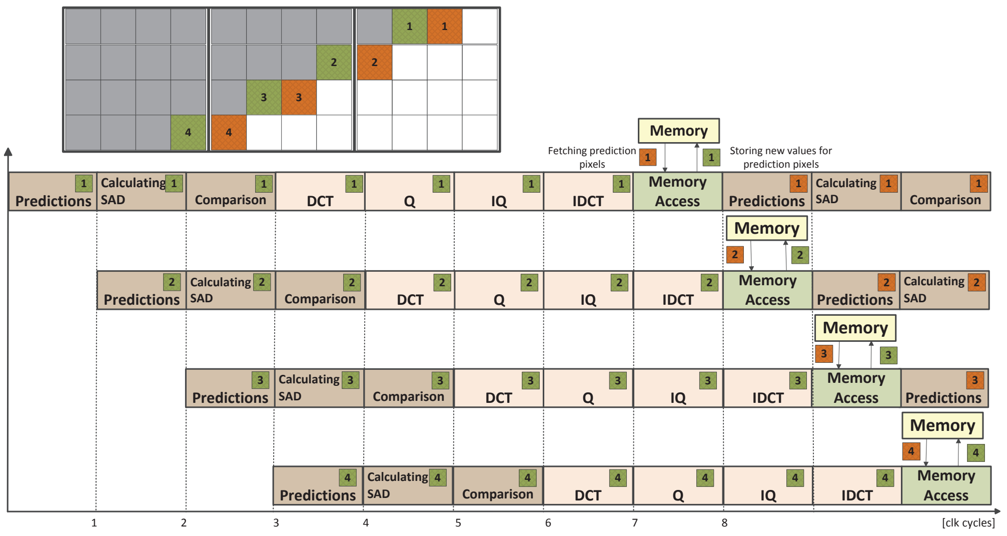

Fig. 10. Dataflow of on-the-fly macroblock processing within a 4×4 block processing cycle

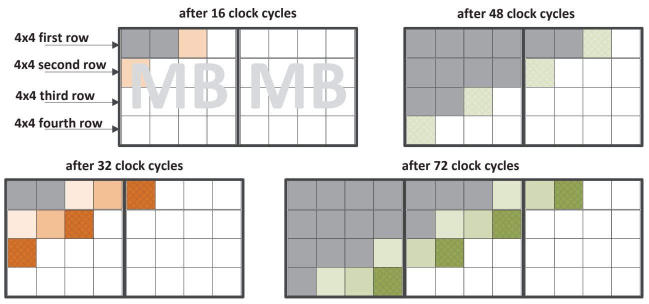

Fig. 9. Block scanning order within number of macroblocks

For the blocks with the equal block numbers the prediction pixels are available at the same time point and prediction can be performed independently in respect to the blocks that wait on the prediction.

1) Intra Prediction Implementation: In order to decrease latency of both stages within the encoder, a 16-pixel input parallelism is proposed. The intermediate memory buffer between MPEG-2 decoder and H.264/AVC encoder produces output that corresponds to a 4×4 block. Processing order of 4×4 blocks over a set of macroblocks is described in Fig. 9. The intra prediction cycle for one 4×4 block, that includes both intra prediction itself and reconstruction loop stage, takes 8 clock cycles. The period of 8 clock cycles required for intra prediction of one 4×4 block is defined as a 4×4 block processing cycle. Due to data dependency between neighboring blocks, sub-blocks (4×4 blocks) from the same row are processed every 4×4 block processing cycle. The wavefront technique on the 4×4-block level has been employed. Initially, only the first row of 4×4 blocks is processed. After two processing cycles, two blocks from the first row are predicted and their reconstructed prediction samples are available for the prediction of the neighboring 4×4 block below. Then, blocks from both rows can be processed in pipeline on the same hardware module with a delay of one clock cycle. After 48 initial clock cycles, each 4×4 row within a number of macroblocks has two predicted 4×4 blocks more than the row below. The requirement to have the upper and left reconstructed neighboring pixels is fulfilled and maximal pipeline for one macroblock line is employed. This implementation is limited to four 4×4 blocks being processed in pipeline due to the 8-pixel input parallelism of the complete transcoder. Maximal pipeline structure for given parallelism level is thus achieved after latency of 48 clock cycles. However, the proposed H.264/AVC encoder architecture has higher input parallelism capability and this can give doubled throughput at the expense of additional decoder block.

The dataflow through intra prediction and reconstruction engine phases is illustrated in Fig. 10. As described, four 4×4 blocks within a macroblock can be processed in a pipelined manner. The prediction module calculates 9 prediction modes and absolute differences of the original pixels and generated prediction pixels in the first clock cycle. The cost functions for 9 modes are compared and the best prediction is determined by the minimal SAD in the third clock cycle. The prediction error with the minimum SAD is sent to the reconstruction engine. Reconstruction process (forward transform, quantization, inverse quantization and inverse transform) takes 4 clock cycles. Reconstructed prediction samples are then computed by summing the reconstructed prediction error and best intra prediction samples. The samples of interest are then sent to the local memory, whereas the data required for the prediction process of new block are fetched from the memory. The processing cycles for blocks from lower rows are delayed one clock cycle in respect to upper-row neighbor blocks.

The boundary cases are also considered. Signals *first_row* and *first_column* that are set if 4×4 blocks from the first row or first column are processed, control the choice of prediction modes. Upper neighboring pixels do not exist for 4×4 blocks from the first row, so the signal *first_row* enables prediction modes that do not require upper reconstruction pixels such as horizontal (H), horizontal up (HU) and DC mode as presented in Table I. DC prediction mode is simplified in the case of first row or column and mean value calculation is performed based on four available pixels.

TABLE I
PREDICTION MODES IN CORNER CASES

First row blocks *first_row*	**First column blocks** *first_column*
DC(I, J, K, L)	DC(A, B, C, D)
H	V
HU	DDL
HU	VL

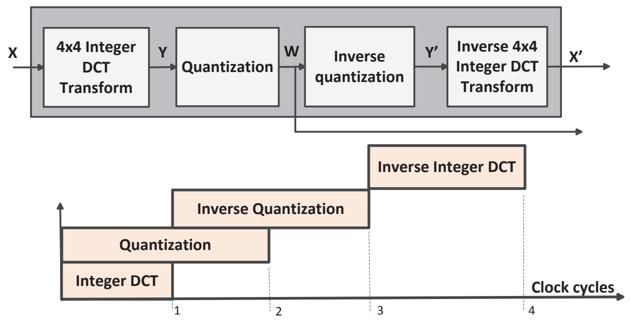

Fig. 11. Block diagram of reconstruction loop

2) Recontruction Engine Implementation: Reconstructed blocks required for prediction of the currently processed block are the result from the reconstruction loop engine. Low latency is the main requirement since a reconstruction loop is on the critical path of the encoding process. The 16 data items (4×4 block) in parallel are input to the transform unit. Computation of direct and inverse transforms takes one clock cycle, whereas quantization and inverse quantization take 2 clock cycles as presented in Fig. 11. Due to pipelined implementation structure entire reconstruction path takes 4 clock cycles.

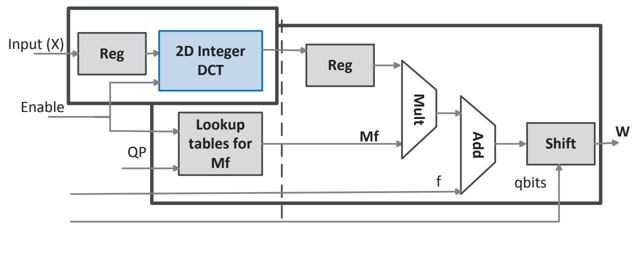

Fig. 12. Pipelined forward DCT and quantization architecture

The functional block diagrams for proposed forward integer DCT and quantization modules are depicted in Fig. 12. The parameters for quantization matrix, such as $\mathbf{Mf}$, are fetched from the local memory depending on the input QP in parallel with transform computations performed in the transform module. During the second clock cycle transformed residual coefficients are then quantized. The 2-dimensional transform is computed in one clock cycle by splitting into two 1D transforms as depicted in Fig. 13. Due to high input data-parallelism the second 1D transform is performed on the fly and the transposed buffer is not required for storing intermediate results.

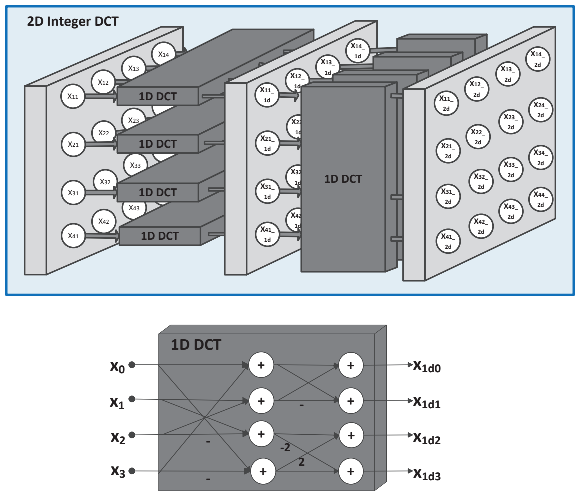

Fig. 13. Two-dimensional integer DCT transform

C. *Memory organization in MPEG-2 to H.264/AVC Transcoder*

Operations that include communication with memory consume a considerable amount of power and time, particularly when off-chip memory is used. To reduce memory power consumption, for purpose of MPEG-2 to H.264/AVC transcoding on-chip memory resources are proposed. A number of buffers and registers are used for storing intermediate data in both decoder and encoder parts of the transcoder.

The different stages of the transcoder require different degree of parallelism. The maximum parallelism of each stage is dependent of stage data dependency and physical IO constraints. The decoder supports 8 data-item input/output parallelism each clock cycle. The output of the decoder is a column of 8×8 decompressed pixel block. The encoder, on the other side, processes 16 pixels in parallel that correspond to a 4×4 block. The 4×4 blocks from the same row are sent with periodicity of a 8 clock cycles. In order to lower delays introduced by change of parallelism between stages, an intermediate buffer is introduced. The data is not sent from the buffer in the straightforward manner and a control unit within a buffer is required. The input columns of a 8×8 block are processed in the order presented in Fig. 14 by first transmitting data from two upper 8×8 blocks followed by data from two lower blocks.

The transcoding buffer has a structure that consists of 4 RAM (RAM1, RAM2, RAM3, RAM4) blocks and each

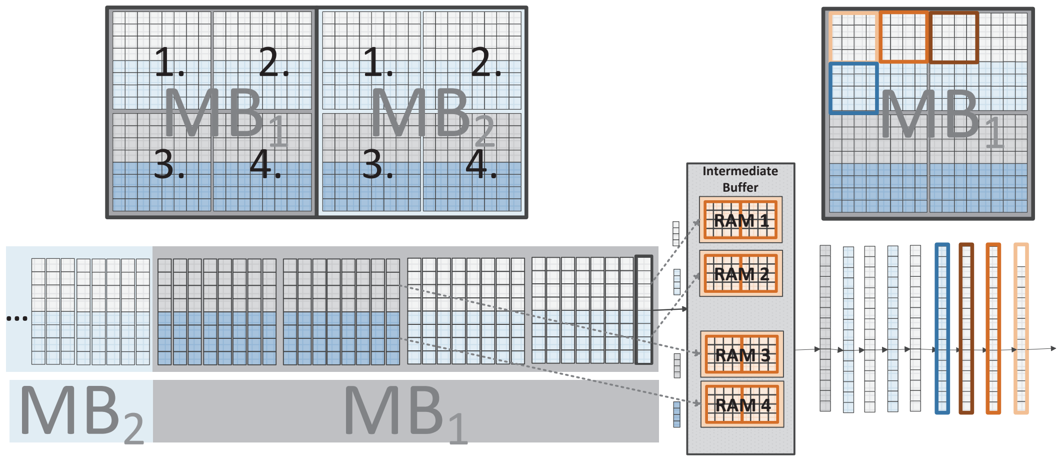

Fig. 14. Dataflow through memory buffer between decoder and encoder

memory block is reserved for storing data from one 4×4 block row. Each of the RAMs can store eight 4×4 data blocks. The buffer control splits an 8-data input into two arrays of four data items: upper data array is stored in the RAM reserved for the first row, lower array is stored in the the second RAM. After upper two 8×8 blocks have been input into the buffer in 16 clock cycles, writing into two RAM memories stops. Other two RAMs store the data from the lower 8×8 blocks and are active for next 16 clock cycles. The incoming lower two 8×8 data blocks (third and forth) are stored in the same manner as the upper blocks and the storing process for one macroblock data into the intermediate buffer takes 32 clock cycles. The complete process is depicted in Fig. 14.

The reading operation from the buffer is less regular than the writing operation. The buffer must produce the output data in the order required by the scanning operation in Fig. 8. The reading operation can start after enough data is written to the buffer - at least four columns with 8 pixel input data. However, writing and reading operations are required to be synchronized in order to omit additional delays introduced by resolving the issues of overwriting, empty reads and infinite buffer size. Fig. 15 shows the dataflow that ensures safe communication between decoder output and encoder input. It also shows that minimal data storage requirement needed for intermediate results is eight 4×4 data blocks per RAM unit. The transfer of the first row data blocks is critical for achieving the synchronization. The latency of 8 clock cycles between writing and reading operation that corresponds to time required for writing one 8×8 block into the buffer provides the synchronization. In Fig. 15, the first 4×4 blocks (5^1, 9^1, 13^1 blocks) from three different macroblocks have constant time delay of 8 clock cycles between writing and reading operation. When the condition that the upper neighbors are reconstructed for all four rows, the buffer sends out data from each row in consecutive clock cycles. The four output control signals of the intermediate buffer inform the intra prediction module about order number of the row which contains sent data.

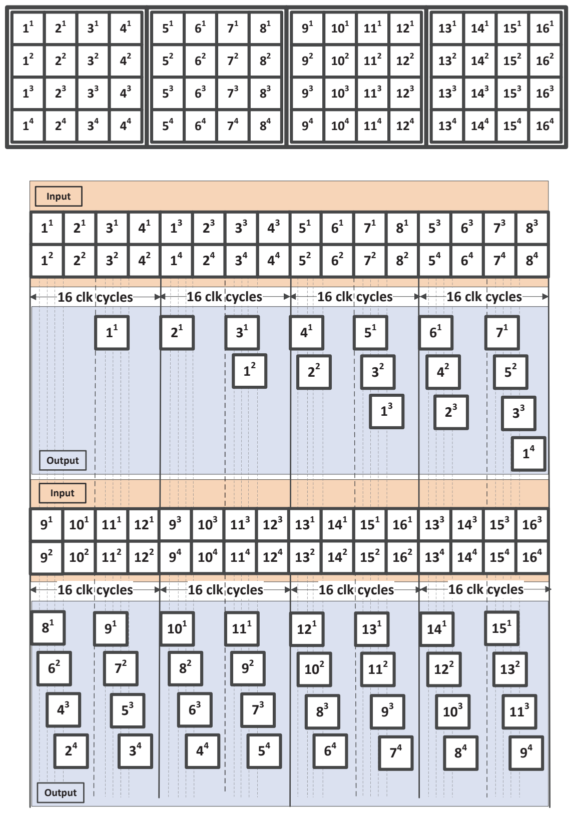

Fig. 15. Time diagram of intermediate memory buffer dataflow

The intra prediction engine requires a substantial amount of memory operations. External memory access is avoided by storing all reconstructed pixels on chip in two local buffers: upper-row buffer (URB) and left-column buffer (LCB). In Fig. 16 the proposed memory allocation process for reference pixels in URB buffer is presented. The upper-row buffer can store entire frame line. A sliding window approach is proposed

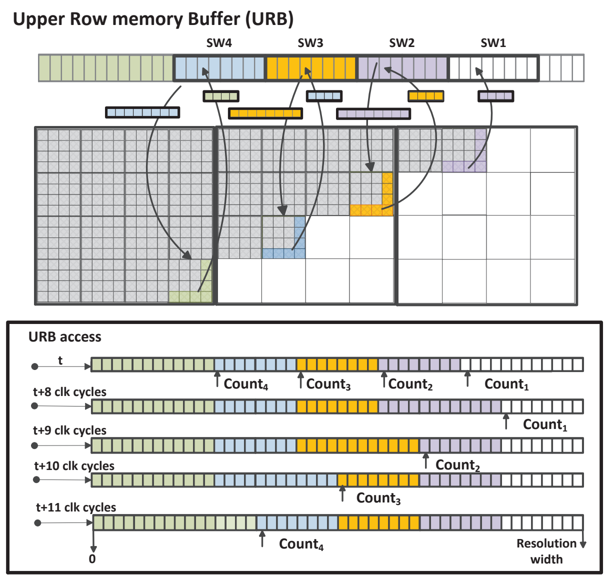

Fig. 16. Block diagram of URB memory access process

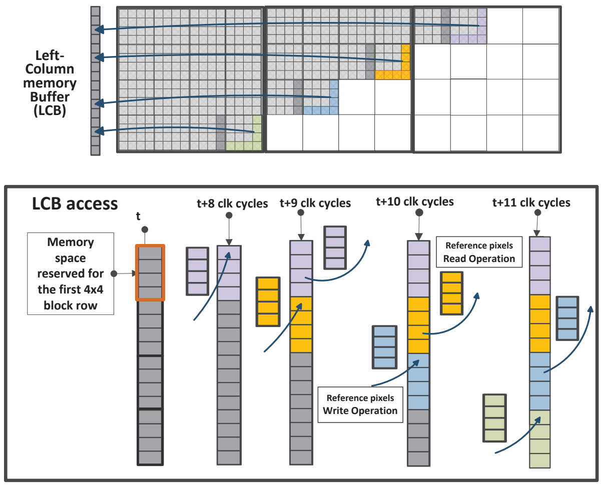
Left-Column memory Buffer (LCB)
LCB access
t
t+8 clk cycles
t+9 clk cycles
t+10 clk cycles
t+11 clk cycles
Memory space reserved for the first 4x4 block row
Reference pixels Read Operation
Reference pixels Write Operation

Fig. 17. Block diagram of LCB memory access process

to fetch the reference pixels from the URB. Each row of 4×4 blocks has its own sliding window (SW1, SW2, SW3, SW4) and a counter ($Count_1$, $Count_2$, $Count_3$, $Count_4$) that counts the current position of the sliding window within the frame width. The sliding window fetches eight reference pixels from URB that correspond to A-H for the current 4×4 block. These pixels are then stored in the local registers for use in the prediction process. The value of the reference pixel D is kept in a separate register as it represents M prediction pixel in the prediction process for the next neighbor block. After the intra prediction is executed, four reconstructed pixel values from the bottom pixel line of the reconstructed block are stored in the same position where data used for prediction is fetched from. For example, eight reconstructed pixel values for the second row 4×4 block are fetched from the address that $Count_2$ points to, and after the prediction and reconstruction processes four reconstructed pixels are stored in the first four prediction pixel places at the same address in memory buffer URB. The counter is then incremented by memory space that corresponds to 4 reconstructed samples, position of sliding window changes and the communication process between URB and intra module repeats.

The left register can store one macroblock column and it consists of 16 memory addresses. Four sample addresses belong to each 4×4 block line in the macroblock. Every four reference pixels within the LCB correspond to the right-most column of the previously decoded 4×4 block. After prediction process of the current block last column of 4×4 block is stored in the memory space reserved for the currently processed block line. Complete process is presented in Fig. 17.

TABLE II
SIMULATION ENVIRONMENT

	Simulation Environment	
Software Codec	H264/AVC Encoder JM18.4 [32]	
Sequences	Foreman	352x288
	Bus	352x288
	City	352x288
	Harbour	352x288
	Park run	1280x720
	Shield	1280x720
	Stockholm	1280x720
	Mobilcal	1280x720
	Blue Sky	1920x1080
	Pedestrian area	1920x1080
	Riverbed	1920x1080
	Rush Hour	1920x1080
Sequence Format [CIF]	4:2:0, 30 Hz	
Sequence Format [HD720]	4:2:0, 50 Hz	
Sequence Format [HD1080]	4:2:0, 25 Hz	
GOP	All-intra	
QP	2,10, 15, 20, 25, 28, 32	

TABLE III
STATISTICS IN VIDEO SEQUENCES

QP	Average PSNR level [dB]			
	1280x720		1920x1080	
Scenario	1.	2.	1.	2.
10	51.93	51.93	51.78	51.76
15	47.31	47.31	47.63	47.64
20	42.32	42.30	43.88	43.86
25	38.17	38.15	41.68	41.60

1) Intra Prediction Analysis: There is a wide variation between practical implementations of H.264/AVC codec despite the fact that H.264/AVC is an industry standard. The number of coding options introduces a tradeoff between compression and complexity. Unlike the traditional approaches, the performance requirements for hardware implementations are not just quality and bitrate, but also complexity of a module implementation. The proposed design relies on the fast and efficient highly-pipelined intra 4×4 prediction implementation, whereas the other block-wise algorithms for luminance

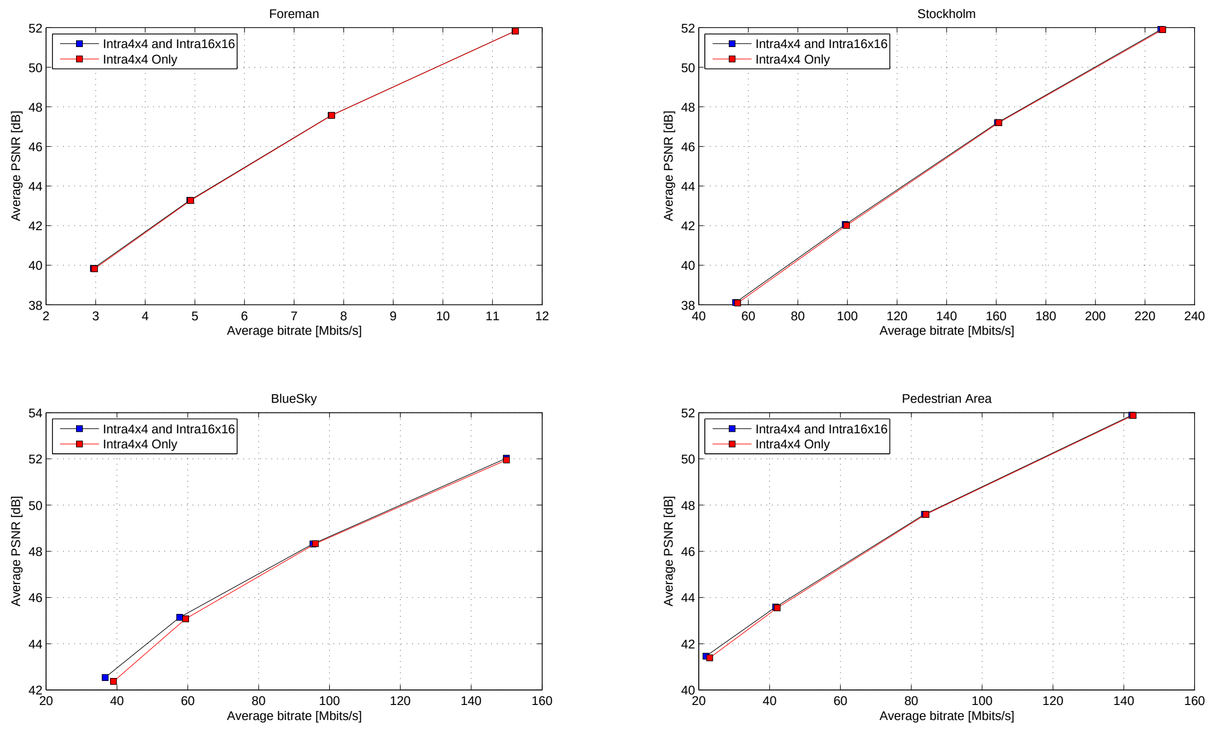

Fig. 18. The average RD curves comparison for various sequences

TABLE IV
PSNR AND BITRATE CHANGES

Video Sequences	Δ PSNR(dB)	ΔBitrate(%)
Foreman	-0.057	1.02
Bus	-0.042	0.33
City	-0.009	0.12
Harbour	-0.003	0.01
Park run	-0.007	0.12
Shield	-0.037	0.43
Stockholm	-0.090	1.32
Mobilcal	-0.004	0.35
Blue Sky	-0.493	7.40
Pedestrian area	-0.176	5.94
Riverbed	-0.011	0.26
Rush Hour	-0.264	17.63

frame component such as intra 16×16 are excluded. The implementation that lowers the complexity increases parallelism possibilities and lowers the delays but it also introduces losses in both quality and bitrate. The experiment is performed for two scenarios: the first that includes H.264/AVC compression standard with both intra 16×16 and 4×4 prediction algorithms and the second scenario that includes only intra 4×4 prediction algorithm. The experiment is performed into the JM18.4 [32] reference software to evaluate the compression performance with QP values (10, 15, 20, 25) on twelve test video sequence of various resolutions (CIF, HD720p and HD1080p) and frame rates (25 fps, 30 fps, 50 fps). The simulation environment is described in Table II. The rate distortion optimization (RDO) mode decision is set off. The statistical analysis on the video sequence content is performed in the first case scenario. The intra 16×16 prediction mode selection rate increases as chosen quantization parameter value gets higher due to introduced degradation of the video quality such as blurred object edges and texture. As expected, the average percentage of intra 16×16 blocks increases with HD resolution. The exclusion of intra 16×16 introduces average PSNR losses: 0.028 dB for CIF, 0.043 dB for HD720p and 0.23 dB for HD1080p for chosen QP values. The PSNR minimum limit for good quality over 40 dB and average PSNR levels for HD resolutions are presented in Table III. To evaluate the losses in the case when intra 4×4 prediction is only used algorithm, the difference of PSNR and bitrate for quantization parameters values (10, 15, 20, 25) are calculated by using the RD curves fitting method [33]. RD curves for video sequences Foreman (CIF), Stockholm (720p), Pedestrian Area (1080p) and Blue Sky (1080p) are presented in Fig. 18. The PSNR and bitrate changes are given in Table IV. The experimental results have shown that encoder with both intra prediction and only intra 4×4 prediction modes have low quality losses for quantization parameter QP value up to 30. Performance improvement results in average bit-rate increase of compared to full algorithm in HD sequences. However, bitrate losses including Rush Hour at 17.63% are considered comparable with the results achieved in recent optimizing techniques for intra prediction [18], [22], whereas computational complexity is lowered drastically since both intra 16×16 algorithm and a very-time consuming mode decision process are excluded from the algorithm. Furthermore, performance improvement in terms of speed, throughput and area efficiency are significant. Therefore, proposed design can be used for moderate compression efficiency with significant performance improvement but negligible loss in image quality and bitrate.

TABLE V
SYNTHESIS RESULTS

Module	Virtex Kintex 705 - XC7K325T				
	FPGA Slices	FF Utilization	LUTS	RAM (kbits)	Multipliers
MPEG-2 Decoder					
IQ	517	448	1206	-	16
IDCT	2022	4421	6441	-	-
Intermediate Buffer	206	187	315	16.4	-
H.264/AVC Encoder					
DCT	200	145	720	-	-
Q	673	247	1576	-	16
IQ	604	176	1431	-	16
IDCT	389	485	1232	-	-
Intra Preduction	2015	2644	5686	19.5	-

TABLE VI
CHARACTERISTICS OF THE TRANSCODING SYSTEM

Module	Frequency [MHz]	Maximum Throughput [MB/s]
MPEG-2 Decoder	154	1232
Intra Prediction	118	944
Reconstruction Loop	109	872

IV. RESULTS

The complexity of the H.264/AVC standard is lowered by only employing intra 4×4 prediction algorithm. The level of noise introduced by the compression is an important requirement for for real-time video processing. Two experimental scenarios are performed to examine limitations introduced by low-complexity encoding algorithm. It is shown that the QP values up to 25 provide PSNR level of reconstructed image above 38 dB which is considered as good quality compression. The PSNR drop due to encoding with only intra 4×4 prediction is comparable with losses in other optimizing implementations such as [18], [22], whereas encoder complexity is significantly lowered due to the high-complexity mode decision algorithm and intra 16×16 prediction being removed from the architecture.

The proposed MPEG-2 to H.264/AVC transcoder architecture is described in VHDL. At first, a Matlab model was developed. A number of video sequences (Foreman, Bus) of a CIF resolution have been used to produce input vectors. The output results of the Matlab reference model was compared with the JM18.4 model. The test vectors for intra prediction, integer DCT, quantization, inverse quantization and inverse DCT modules were designed based on the Matlab specification model and the correctness of each hardware module was validated using Active HDL. The individual modules within the decoder and encoder have been synthesized and implemented on a Kintex 705 - XC7K325T development board to determine the hardware resource costs and to analyze performance. The resulting netlist of the complete architecture has been generated using Xilinx ISE 14.4. The post-place and route results are presented in Table V. The implementation takes less than 10% of the total LUTs of the Kintex FPGA.

The architecture includes computationally demanding modules within the MPEG-2 to H.264/AVC transcoder. Data dependencies have been widely examined and mutually independent blocks are processed in the pipelined manner. The implementation of the pipeline module provides high utilization of the hardware resources and thus high throughput. The dataflow diagram of the complete system state in the initial stage is presented in Fig. 19. Maximum transcoding pipeline is achieved after 76 clock cycles. Full pipeline for transcoder is highly dependent of input parallelism and outputs one macroblock of quantized and transformed residual data every 32 clock cycles as presented in Fig. 20. The performance characteristics are presented in Table VI. The intra prediction engine, i.e. encoder, can handle double throughput compared to complete transcoder due to 16-pixel/clock input. By reconfiguring the transcoding structure and doubling the input parallelism the throughput can be doubled. This is achieved by modifying the design through an additional decoder module as described in the paper.

The design contains an on-chip memory organization that supports fast video processing. The intermediate buffer reorganizes data from the decoder and ensures the synchronization between decoder and encoder, whereas the intra prediction module contains two memory buffers URB and LCB for storing reconstructed prediction values needed for prediction of the subsequent blocks.

The design supports 8-pixel input parallelism by processing 8 pixels every clock cycle. A maximum throughput of 871 MB/s is limited by the lowest maximum frequency of the reconstruction loop. The encoder processing cycle is set to 8 clock cycles to ease synchronization between coding stages and the proposed dataflow solution achieves full synchronization of the MPEG-2 decoder and the H.264/AVC stages. Table VII presents a comparison to related VLSI implementations [18]–[23] and present FPGA solutions [25], [26]. The proposed block scanning order takes significantly less clock cycles compared to previous work. Furthermore, it achieves a throughput that is significantly higher than the throughput in state of the art architectures, in particular FPGA implementations. Higher throughput is only achieved in [23], but the implementation is limited to high profile and is characterized by high input parallelism of 64 pixels. The existing FPGA boards are limited when regards to IO pins. Xilinx encoder specification [24] states a maximal throughput of 2.5 clocks/pixel with maximal frequency of 260 MHz, whereas the proposed encoder implementation within the transcoder provides 0.0625 clocks/pixel with maximal frequency of 109 MHz. The processing speed of the proposed implementation

TABLE VII
COMPARISON TO RECENT WORKS ON INTRA PREDICTION ARCHITECTURES

Design	[18]	[19]	[20]	[21]	[22]	[23]	[25]	[26]	Proposed Encoder	Proposed Transcoder
Technology	ASIC 90 nm	ASIC 180nm	ASIC 130nm	ASIC 130nm	ASIC 90nm	ASIC 65nm	FPGA Stratix III VLX110T	FPGA Virtex-5 EP3SL150	FPGA Kintex 705 XC7K325T	FPGA Kintex 705 XC7K325T
Max. Throughput (Mpixels/s)	-	128	62	337	151	1991	62	113	1744	872
Input Parallelism	4-pixel	4/16-pixel	32-pixel	16-pixel	16/8-pixel	64-pixel	16-pixel	16-pixel	16-pixel	8-pixel
Max. Supported Resolution	1080p 30 fps	1080p 60 fps	1080p 30 fps	2160p 30 fps	1080p+ 480p+CIF	4320p 60 fps	1080p 30 fps	1080p 60 fps	4320p 30 fps	2160p 60 fps
Cycles/Macrobl.	N/A	<300	464	163	<454	35	272	< 300	16	32
Frequency [MHz]	180	150	114	215	135	273	130	168	109	109
Min. Frequency 4320p/30 fps	-	-	-	-	-	136	-	-	62	-
Min. Frequency 2160p/60 fps	-	-	-	-	-	68	-	-	31	62
Min. Frequency 2160p/30 fps	-	-	-	158	-	34	-	-	16	31
Min. Frequency HD1080p/60 fps	-	146.8	-	80	-	16.6	-	146	8	16
Min. Frequency HD1080p/30 fps	152	73.4	114	40	110	8.3	66	73	3	8
Min. Frequency HD720p/30 fps	116	32.4	51	18	49	3.8	32.4	29	1.5	3
Min. Frequency SD/30 fps	43	12.1	19	5.9	16	1.3	10	10.9	0.5	1

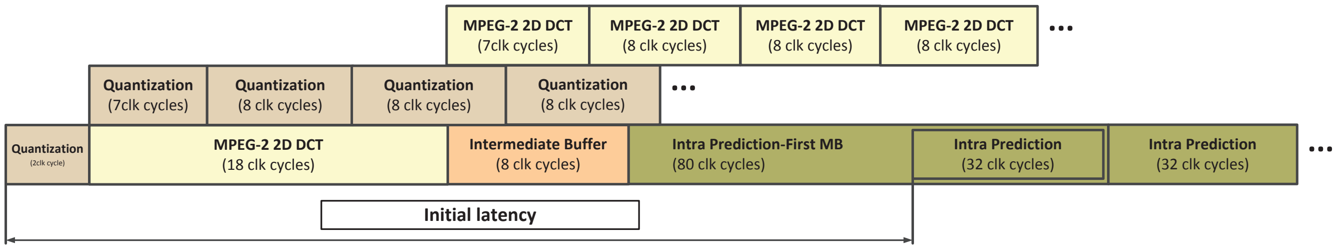
MPEG-2 2D DCT (7clk cycles)
MPEG-2 2D DCT (8 clk cycles)
MPEG-2 2D DCT (8 clk cycles)
MPEG-2 2D DCT (8 clk cycles)
...
Quantization (7clk cycles)
Quantization (8 clk cycles)
Quantization (8 clk cycles)
Quantization (8 clk cycles)
...
Quantization (2clk cycle)
MPEG-2 2D DCT (18 clk cycles)
Intermediate Buffer (8 clk cycles)
Intra Prediction-First MB (80 clk cycles)
Intra Prediction (32 clk cycles)
Intra Prediction (32 clk cycles)
...
Initial latency

Fig. 19. Dataflow diagram of the proposed MPEG-2 to H.264/AVC transcoder

is 17.2 times faster than the Xilinx solution.

The throughput for the transcoder implementation corresponds to real-time processing of QFHD video sequences with a frame rate of 60 fps. Generally, the architecture provides real-time transcoding of various resolutions such as CIF, SD and HD video sequences (2160p, 1080p and 720p) with various frame rates at relatively low operation frequencies ($\leq$ 62 MHz). Compared to state of the art for H.264/AVC encoding, the proposed encoder module executes in full pipeline by processing two macroblock lines in a wavefront manner and it can achieve maximum throughput of 1744 Mpixels/s which corresponds to real-time processing of an UHD video resolution 4320p at 30 fps.

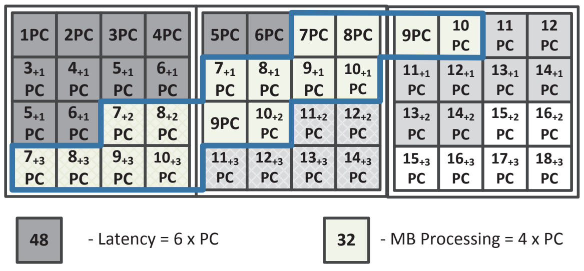
1PC 2PC 3PC 4PC
5PC 6PC 7PC 8PC
9PC 10 PC 11 PC 12 PC
3_{+1} PC 4_{+1} PC 5_{+1} PC 6_{+1} PC
7_{+1} PC 8_{+1} PC 9_{+1} PC 10_{+1} PC
11_{+1} PC 12_{+1} PC 13_{+1} PC 14_{+1} PC
5_{+1} PC 6_{+1} PC 7_{+2} PC 8_{+2} PC
9PC 10_{+2} PC 11_{+2} PC 12_{+2} PC
13_{+2} PC 14_{+2} PC 15_{+2} PC 16_{+2} PC
7_{+3} PC 8_{+3} PC 9_{+3} PC 10_{+3} PC
11_{+3} PC 12_{+3} PC 13_{+3} PC 14_{+3} PC
15_{+3} PC 16_{+3} PC 17_{+3} PC 18_{+3} PC
48 - Latency = 6 x PC
32 - MB Processing = 4 x PC

Fig. 20. Dataflow of the proposed intra prediction processing

V. CONCLUSION

An efficient low-complexity architecture of MPEG-2 to H.264/AVC intra frame transcoder is presented. The proposed architecture includes an implementation of a number of complex and data-intensive computational modules. A block reordering method provides parallel processing of 4×4 blocks in several macroblocks simultaneously. This decreases idle time of the module caused by intra 4×4 data dependencies. Additionally, the local neighboring pixel buffers are placed on chip in order to reduce the memory access delays enabling the processing of video sequences of different resolutions such as CIF, SD and HD resolutions. The different throughput requirements in decoder and encoder parts are successfully synchronized by use of an intermediate buffer. Compared to the state of the art, the proposed design uses a significantly lower number of clock cycles for real time processing of HD video sequences. Thus, proposed FPGA MPEG-2 to H264/AVC transcoder solution achieves high throughput that corresponds to real-time processing QFHD video resolution at 30 fps and 60 fps.

An area-efficient implementation gives possibility to place additional coding techniques that improve performance on a FPGA chip. Future work includes high-throughput inter prediction and exploration of the run-time reconfiguration of the transcoder implementation by changing the level of complexity depending on the real time variations in requirements (resolution, bitrate, quality).

References

[1] Z.-Y. Lu, K.-B. Jia, and W.-C. Siu, "Low-complexity intra prediction algorithm for video down-sizing transcoder," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*. IEEE, 2011, pp. 1–4.

[2] Y. Su, J. Xin, A. Vetro, and H. Sun, "Efficient mpeg-2 to h. 264/avc intra transcoding in transform-domain," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 1234–1237.

[3] H. Kalva and B. Petljanski, "Exploiting the directional features in mpeg-2 for h. 264 intra transcoding," *Consumer Electronics, IEEE Transactions on*, vol. 52, no. 2, pp. 706–711, 2006.

[4] L.-L. Wang and W.-C. Siu, "H.264 fast intra mode selection algorithm based on direction difference measure in the pixel domain," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 1037–1040.

[5] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

[6] S. Smaoui, H. Loukil, A. Ben Atitallah, and N. Masmoudi, "An efficient pipeline execution of h. 264/avc intra 4× 4 frame design," in *Systems Signals and Devices (SSD), 2010 7th International Multi-Conference on*. IEEE, 2010, pp. 1–5.

[7] B. Meng, O. C. Au, C. W. Wong, and H. K. Lam, "Efficient intra-prediction algorithm in h. 264," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3. IEEE, 2003, pp. III–837.

[8] Z. Wei, H. Li, and K. N. Ngan, "An efficient intra mode selection algorithm for h. 264 based on fast edge classification," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, 2007, pp. 3630–3633.

[9] F. Pan, L. S. Rahardja, K. P. Lim, L. D. Wu, W. S. Wu, C. Zhu, W. Ye, and Z. Liang, "Fast intra mode decision algorithm for h. 264-avc video coding," in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, vol. 2. IEEE, 2004, pp. 781–784.

[10] C. C. Cheng and T. S. Chang, "Fast three step intra prediction algorithm for 4× 4 blocks in h. 264," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 1509–1512.

[11] B. Pieters, C.-F. Hollemeersch, J. De Cock, P. Lambert, and R. Van de Walle, "Data-parallel intra decoding for block-based image and video coding on massively parallel architectures," *Signal Processing: Image Communication*, vol. 27, no. 3, pp. 220–237, 2012.

[12] M. C. Kung, O. Au, P. Wong, and C.-H. Liu, "Intra frame encoding using programmable graphics hardware," in *Advances in Multimedia Information Processing–PCM 2007*. Springer, 2007, pp. 609–618.

[13] S. Sun, D. Wang, and S. Chen, "A highly efficient parallel algorithm for H.264 encoder based on macro-block region partition," in *High Performance Computing and Communications*. Springer, 2007, pp. 577–585.

[14] H. Su, N. Wu, C. Zhang, M. Wen, and J. Ren, "A multilevel parallel intra coding for h. 264/avc based on cuda," in *Image and Graphics (ICIG), 2011 Sixth International Conference on*. IEEE, 2011, pp. 76–81.

[15] NVIDIA Corporation. (2010) NVIDIA CUDA compute unified device architecture: Programming guide version 3.2.

[16] 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," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 3, pp. 378–401, 2005.

[17] Y.-K. Lin, C.-W. Ku, D.-W. Li, and T.-S. Chang, "A 140-mhz 94 k gates hd1080p 30-frames/s intra-only profile h. 264 encoder," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 3, pp. 432–436, 2009.

[18] J.-W. Chen, H.-C. Chang, J.-S. Wang, and J.-I. Guo, "A dynamic quality-adjustable H.264 intra coder," *Consumer Electronics, IEEE Transactions on*, vol. 57, no. 3, pp. 1203–1211, August 2011.

[19] C. Diniz, B. Zatt, C. Thiele, A. Susin, S. Bampi, F. Sampaio, D. Palomino, and L. Agostini, "A high throughput h. 264/avc intra-frame encoding loop architecture for hd1080p," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 579–582.

[20] H.-C. Kuo, L.-C. Wu, H.-T. Huang, S.-T. Hsu, and Y.-L. Lin, "A low-power high-performance H.264/AVC intra-frame encoder for 1080pHD video," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 6, pp. 925–938, June 2011.

[21] H. Ren, Y. Fan, X. Chen, and X. Zeng, "A 16-pixel parallel architecture with block-level/mode-level co-reordering approach for intra prediction in 4k× 2k h. 264/avc video encoder," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. IEEE, 2012, pp. 801–806.

[22] G.-L. Li, T.-Y. Chen, M.-W. Shen, M.-H. Wen, and T.-S. Chang, "135-mhz 258-k gates vlsi design for all-intra h. 264/avc scalable video encoder," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 636–647, 2013.

[23] G. He, D. Zhou, W. Fei, Z. Chen, J. Zhou, and S. Goto, "High-performance H.264/AVC intra-prediction architecture for ultra high definition video applications," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 1, pp. 76–89, Jan 2014.

[24] Xilinx Corporation. (2012) H.264 high profile encoder. [Online]. Available: www.xilinx.com

[25] A. B. Atitallah, H. Loukil, and N. Masmoudi, "Fpga design for h. 264/avc encoder," *International Journal of Computer Science, Engineering and Applications*, vol. 1, no. 5, 2011.

[26] C. M. Diniz, A. A. Susin, and S. Bampi, "Fpga design of h. 264/avc intra-frame prediction architecture for high resolution video encoding," in *Programmable Logic (SPL), 2012 VIII Southern Conference on*. IEEE, 2012, pp. 1–6.

[27] Y. Adibelli, M. Parlak, and I. Hamzaoglu, "Computation and power reduction techniques for h. 264 intra prediction," *Microprocessors and Microsystems*, vol. 36, no. 3, pp. 205–214, 2012.

[28] M. Orlandić and K. Svarstad, "A low complexity h.264/avc 4x4 intra prediction architecture with macroblock/block reordering," in *ReConFigurable Computing and FPGAs (ReConFig), 2013 International Conference on*. IEEE, 2013.

[29] ITU-T and I. J. 1, "Generic coding of moving pictures and associated audio information - part 2: Video," *ITU-T Recommendation H.262 and ISO/IEC 13 818-2 (MPEG-2)*, 1994.

[30] I. Richardson, *The H.264 Advanced Video Compression Standard*. Wiley, 2010. [Online]. Available: http://books.google.no/books?id=LJoDiPnBzQ8C

[31] W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *Communications, IEEE Transactions on*, vol. 25, no. 9, pp. 1004–1009, 1977.

[32] (2012) Reference software for h.264/avc codec jm18. [Online]. Available: http://www.iphome.hhi.de/suehring/uml/index.htm

[33] G. Bjontegard, "Calculation of average psnr differences between rd-curves," *ITU-T VCEG-M33*, 2001.

Milica Orlandić received the B.S. and M.S. degrees in electrical engineering from University of Montenegro in 2007 and 2009 respectively. She is currently holding a position as Ph.D. candidate at the Norwegian University of Science and Technology, Trondheim, Norway.

Her research interests include video and image processing, digital hardware design and reconfigurable systems on FPGA.

Kjetil Svarstad received the M.Sc. and Ph.D. degrees from NTH (now NTNU, Norwegian University of Science and Technology) in 1984 and 1989 in VLSI Design and architectures. From 1990 to 2005 he worked at SINTEF and participated in pilot FPGA and ASIC development projects in satellite communication and digital TV broadcasting, amongst others. He has been a board member for a technology startup company and an electronic arts center, and for several years leader of the national VHDL users' group in Norway. From 2006 he has been a professor at the Department of Electronics and Telecommunication at NTNU. His teaching and research are in areas such as system-on-chip design, system level languages and descriptions, run-time reconfigurable systems on FPGA, formal and assertion based verification, and probabilistic analysis of state systems.