An H.264 Quad-FullHD Low-Latency Intra Video Encoder

Muhammad Usman Karim Khan, Jan Micha Borrmann, Lars Bauer, Muhammad Shafique and Jörg Henkel
Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{muhammad.khan, lars.bauer, muhammad.shafique, henkel}@kit.edu

***Abstract*—Video applications are moving from Full-HD capability (1920×1080) to even higher resolutions such as Quad-FullHD (3840×2160). The H.264 Intra-mode can be used by embedded devices to trade off the better encoding efficiency of H.264 temporal prediction (Inter-mode) against savings in area and power as well as saving the massive computational overhead of the sub-pixel motion estimation by using only spatial prediction (Intra-mode). Still, the H.264 Intra-mode requires a large computational effort and imposes severe challenges when targeting Quad-FullHD 25 fps real-time video encoding at moderate operating frequencies (we target 150 MHz) and limited area budget. Therefore, in this work we address the strong sequential data dependencies within H.264 Intra-mode that restrict the parallelism and inhibit high resolution encoding by a) decoupling of DC and AC transform paths, b) cycle-budget aware mode prediction scheduling while c) being area efficient. Using our proposed techniques, Quad-FullHD (3840×2160) 28 fps video encoding is achieved at 150 MHz, making our architecture applicable for high definition recording.**

I. Introduction and Challenges

H.264 (also called AVC, MPEG-4 part 10) is an industry standard for video coding and is jointly developed by the MPEG group of ISO/IEC MPEG and the VCEG group of ITU-T [1]. H.264 is a hybrid video coding standard that incorporates temporal (Inter) as well as spatial (Intra) predictions [2]. H.264 has become popular because of its high coding efficiency and is an integral part of video conferencing, HDTV, blu-ray disks etc. H.264 outperforms its predecessors by offering up to 60-times compression for common video content [3]. Therefore, it is the key encoding standard for high definition video.

Encoding efficiency comes at the cost of high computational complexity (especially temporal prediction modes which now can be done at quarter-subpixel resolution [2]) and by including a big set of compressing tools in the encoding loop [4], requiring already ~ 10 billion RISC instructions per second for encoding even SDTV (720×480) resolution at 30 fps [5]. H.264 is a block based standard, dividing luma frames into blocks of 16×16 pixels and chroma blocks into blocks of 8×8 (termed as Macroblocks, MB). For real-time encoding resolutions beyond Full-HD (1920×1080), like *Quad Full-HD* (QFHD, 3840×2160), 32,400 MBs of luma (plus 64,800 chroma MBs) need to be processed at 25 fps resulting in a data processing rate of 2.32 Gbps.

To cope with this tremendous amount of data processing, often sub-optimal encoding methods are used, leading to a degradation in encoding efficiency. Motion-compensated temporal prediction (Inter-mode) provides the highest compression efficiency, but it also comes at the highest computational cost. Therefore, for some applications requiring low-latency and high quality (like pocket cameras, digital video capturing, automotive, camcorders, movie backups, or high-end cinema), only in-frame spatial prediction is used due to its smaller computational workload and a smaller power footprint. Still, H.264 Intra-mode remains competitive with standards like Motion JPEG2000 [6].

978-3-9815370-0-0/DATE13/©2013 EDAA

A. Challenges of H.264 Intra-Encoding for HD

A high-level architecture of an H.264 Intra-encoder with its main functional blocks is shown in Fig. 1. MBs of an input frame are processed individually. They are scanned in raster scan order and each MB passes through the encoding loop (marked as dashed red arrow). The content of each MB is *predicted* based upon the pixel values of the left, upper, and left-upper MB (*Intra Prediction Generator* in Fig. 1). The residue $X^{(r)}$ (i.e. the pixel-wise difference of the predicted MB X' and the actual MB X) is sent to the *Transform* block. There, the residue is processed by the *Discrete Cosine Transformation* (DCT), *Hadamard Transformation* (HT), and *Quantization* (Q)) and sent to the *Entropy Coding*. Additionally, the data is locally decoded, i.e. processed by *Inverse Quantization* (IQ), *Inverse DCT* (IDCT), *Inverse HT* (IHT), and *Reconstruct*. The reconstructed MBs are required as a base for the next predictions. There are various data dependencies inside this loop and those with the highest significance for encoding performance are explained in the following.

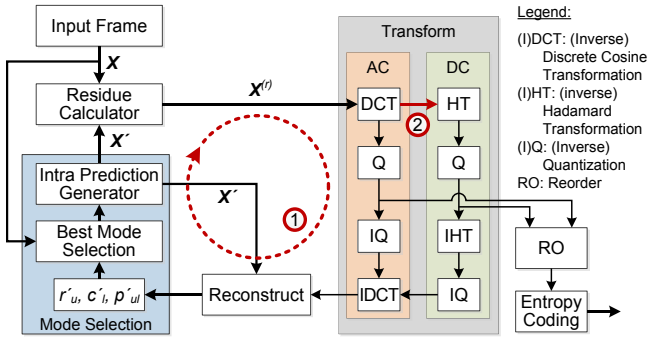

Fig. 1: H.264 Intra Processing Loop Architecture: the current input Macroblock is labeled X, the predicted and residue MBs are labeled X' and $X^{(r)}$; sequential data dependencies are shown by dashed arrows (① and ②)

Dependency (a): The main performance degrading dependency comes from the fact that before entering the encoding loop, the current MB has to wait for the previous MBs in the loop to be fully encoded (i.e. generation of a residue block which is then processed by DCT, HT, and Q) and then locally decoded (IQ, IDCT, IHT, and Reconstruct). This is depicted by the dashed arrow labeled ① in Fig. 1.

Dependency (b): HT cannot start execution until the whole MB is processed by DCT, but the Q and IQ blocks in the AC path can start processing earlier than the HT block. The transform block itself consists of two paths, one processing the AC part of the spatial frequencies, the other processing the DC part. The outputs from the DCT (first output is the DC output and the other are called AC outputs) are fed both to HT in the DC path and Q in the AC path as inputs. This dependency is shown as arrow labeled ② in Fig. 1. Additionally, to compute the IDCT, data from the DC path is required. Therefore, IDCT has to stall and wait for data from the DC stage.

Dependency (c): Moreover, the entropy-coding module also processes the DC output coefficients before the AC coefficients, but the DC coefficients are generated later. This adds to the latency of the output generation.

In this work, we present a new approach for a high throughput and area efficient Intra encoding loop, capable of handling QFHD sequences at 28 fps operating at 150 MHz (see system setup in Section IV). Instead of focusing on a single module of the encoding loop, we propose a complete H.264 Intra-encoding system, which offers low-latency and high-throughput.

Our novel contributions are:

1. decoupling of DC and AC spatial frequency path in the H.264 transform stage by eliminating their mutual dependency, thus decreasing the latency of the transform stage,
2. an area-economical, simple, and efficient edge feature extractor for helping the Intra-mode Decision mechanism,
3. advanced mechanisms for scheduling the calculation of different prediction modes and the Sum of Absolute Differences to process dimensions even larger than QFHD, and
4. implementation and functional verification of our contributions on a cost-optimized mid-range Altera Arria II GX FPGA.

Because of Dependency (a), the encoding loop (reconstructed MB going back to prediction) cannot be pipelined and this limits the throughput of the whole encoder. For example, in a QFHD (3840×2160) sequence there are 32,400 16×16 MBs per frame and at 25 fps there are 810,000 MB/s that need to be encoded. At a target frequency of 150 MHz we have a cycle budget of ⌊150,000,000 cycles/s ⁄ 810,000 MB/s⌋ = 185 cycles/MB in the loop. This cycle-budge also suggests that it is impossible to meet this constraint using a SW-only solution.

Therefore, our novel contribution targets low latency hardware modules in each stage of the encoding loop.

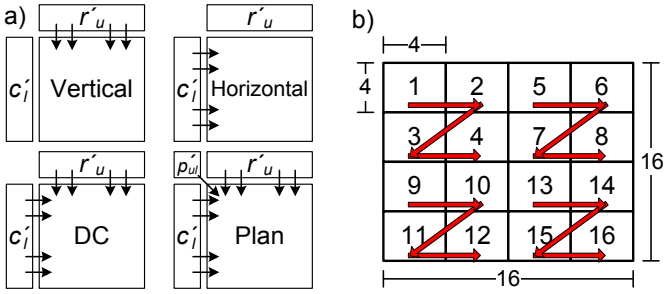

Fig. 2: (a) I16MB modes, (b) 4×4 reordering (inside an MB) required by the entropy coding

The encoding loop has two main computational hotspots, 1) transform and 2) mode selection. The transform block consists of 2D (I)DCT, 2D (I)HT, and (I)Q as shown in Fig. 1. Note that the building blocks of transform work on 4×4 image data. The mode selection block determines the best prediction mode for aiding compression by computing the *Sum of Absolute Difference* (SAD), or the *Sum of Absolute Transformed Differences* (SATD) of the input MB X and the predicted MB X'. Intra prediction of luma data can be done for one full 16×16 MB (I16MB mode) or for 16 individual 4×4 subblocks of one luma MB (I4MB mode) or for 4 individual 8×8 sublocks of the luma MB (I8MB mode). Chroma is predicted at 8×8 MB size. The prediction X' is generated from the reconstructed MBs' data. For an I16MB processing scheme, X and X' are both 16×16 and the values of X' can be generated using either *Vertical* (V), *Horizontal* (H), *DC*, or *Planar* (P) mode [2], and their dependencies with upper reconstructed MB's last row r'_u, left reconstructed MB's last column c'_l and upper left reconstructed pixel p'_{ul} are marked in Fig. 2a. The prediction X' (for the current MB X) that gives the least SAD or SATD is selected as the best prediction and is passed on to the residue generator stage to produce the residue $X^{(r)}$ as shown in Eq. 1.

$$X^{(r)}_{(i,j)} = X_{(i,j)} - X'_{(i,j)},\ \forall i,j = 1,...,16 \qquad (1)$$

Additionally, because of Dependency (b), the HT and IDCT computations have to stall until the coefficients from the AC path and the DC path are available, respectively.

Therefore, our novel contribution proposes an AC/DC-path decoupling scheme that overcomes this dependency while still being correct.

Because of Dependency (c), reordering of the 4×4 blocks as needed for Entropy Coding, shown in Fig. 2b, should be incorporated in the design without adding additional latency along the reconstruction feedback loop.

Thus, from the above discussion, it is clear that low-latency HW design of H.264 Intra-encoder requires optimizations along the encoding loop modules.

TABLE I COMPARISON OF POWER AND VISUAL QUALITY TRADE-OFFS FOR I16MB VS. I16MB WITH I4MB; POWER VALUES OBTAINED BY ALTERA QUARTUS POWER ANALYZER WITH A REPORTED POWER ESTIMATION CONFIDENCE BETWEEN 96% AND 98%

Test Sequence	Power (16×16) [mW]	Power (4×4) [mW]
Traffic (2560×1600)	178.78	504.74
People (2560×1600)	178.78	507.46
Riverbed (1920×1080)	181.39	513.77
Basketball (1920×1080)	181.39	511.71
Shields (1280×720)	183.69	510.88
Kimonol (1920×1080)	181.39	496,34
Tennis (1920×1080)	181.39	487.40
Average	**180.97**	**505.99**

B. *ANALYSIS OF I4MB AND I16MB*

As already described in Section I, Intra Prediction for Luma can be done at 4×4 block granularity (I4MB) or 16×16 MB granularity (I16MB). 4×4 blocks are predicted in the order presented in Fig. 2b. For both modes, the data dependencies from the reconstructed previous block determine the overall encoding performance. We have noticed that sequentially testing each of the four I16MB prediction modes fits the tight cycle budget for QFHD resolutions. For I4MB prediction, the aforementioned dependencies at 4×4 block level limit the cycle budget for the traversal of one sub-block to 11 cycles when a clock frequency of 150 MHz is used. This demands parallel mode computations and even more effort than I16MB.

We implemented both prediction modes, did a thorough analysis for high resolutions, and show power consumption results in Table I. The average power consumption of I16MB mode prediction is only 35.8% of the average power consumption of I4MB mode prediction (i.e. using I4MB requires 2.8× more power). Therefore, for intra-only encoders encoding high-resolution videos, it becomes attractive to implement I16MB only and we focus on I16MB in the following. However, our core techniques like the proposed AC/DC decoupling, the likely-mode prediction, and the prediction generator (as shown in Fig. 3 and explained in Section III) are scalable and applicable to I4MB prediction as well.

The rest of the paper is organized as follows. Section II discusses related work and Section III presents the details of our proposed H.264 Intra encoder architecture. An FPGA prototype of our architecture and experimental results are presented in Section IV. Section V concludes the paper.

II. Related Work and Our Novel Contributions

Encoding HD videos requires new methods and tools to administer the necessary data processing tasks and dependencies of H.264. For example, authors in [5] discuss an H.264 Intra encoder chip operating at 54 MHz. However, it is only capable of handling a 720×480 4:2:0 video at 30 fps. Additionally, the authors use a parallel structure for computing the modes that increases silicon area overhead. In [7], a fast mode selection preprocessor based on spatial domain filtering is discussed. If a likely mode based on a matching edge masks is found, other modes are skipped. If no dominant edge can be extracted, all prediction modes are computed. A four-stage pipeline for edge extraction increases the latency of their design and processing one MB requires 416 cycles at a maximum possible clock rate of 66 MHz. The design in [8] presents a 1080p at 25 fps Intra encoder operating at 100 MHz. It takes about 440 cycles to compute the Intra predictions and makes the 16×16 mode computation independent of 4×4 by changing the 4×4 processing order. In [9], a QFHD at 60 fps Intra prediction architecture is proposed that replicates hardware for high throughput and uses out-of-order execution with the need to reorder the data before entropy coding, which incurs large area overhead. That architecture needs to execute at least at 310 MHz to achieve QFHD while still using sub-optimal encoding methods. Ref. [10] presents a low-latency 1080p at 61 fps Intra encoder architecture operating at 150 MHz, but it computes the modes in parallel (scarifying area) and takes about 300 cycles to encode one MB. In [11], authors propose a fast method of selecting the best intra-prediction modes, based upon the texture flow. Ref. [12] presents an open loop (OL) method to determine the most likely mode based upon the original image data rather than the reconstructed data.

Moreover, the above-mentioned designs do not adapt the H.264 Intra computation schedule to different cycle budgets, given by different resolutions and frame rate requirements of the encoder and hence are not extensible to higher resolutions. In addition, either they have a large silicon footprint at high working frequencies, or they exercise non-optimal coding strategies to save area.

III. Low Latency H.264 Intra Encoding Loop

The proposed architecture of the H.264 Intra Encoding loop is depicted in Fig. 3. Compared to the conventional Intra encoding loop (see Fig. 1), we mark our differences with a star symbol and discuss the impact of each block separately. We only discuss the luma path of the loop, but the discussion also applies to the chroma path, as their differences are negligible. Our encoding pipeline is fed from DDR3 memory containing the actual video input and the stored reconstructed frame. Data is accessed in MB granularity where one line of a 16×16 MB is transferred per cycle over a 128-bit data bus.

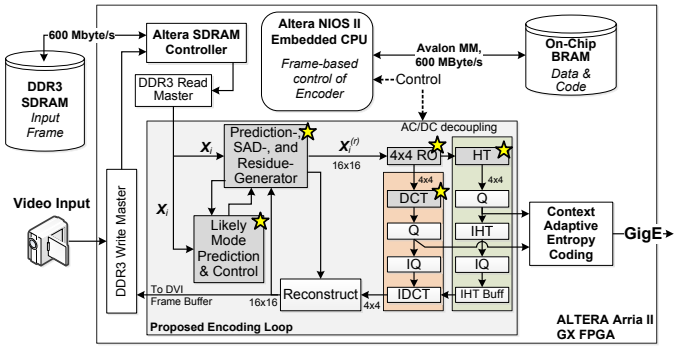

Fig. 3: Proposed encoding loop; HT is decoupled from DCT; a likely mode predictor works on the input data and presents its output to the predictor stage

A. 4×4 Reordering and HT Lookahead

To decrease the latency of the transform stage (Dependency (b) in Section I.A), we propose a method to decouple the AC path from the DC path in the transform block. As mentioned in Section I.A, the inner blocks of the transform unit require a 4×4 block as input. Thus, we must subdivide the residue 16×16 block into 16 4×4 blocks using the 4×4 reorder stage (RO). Our residue generator stage provides one line of $X^{(r)}$ (16 pixels in 1 cycle, i^{th} line of the MB given by $X_i^{(r)}$). Whenever four $X_i^{(r)}$ are accumulated in the input registers of the 4×4 RO stage, the RO generates four 4×4 blocks and pushes them to the input FIFO of the 2D-DCT stage. As mentioned in Section I.A, HT cannot commence until all 4×4 blocks are processed by the DCT. However, by simplifying the DCT formula (the DCT used by H.264 is not an actual discrete cosine transform but a derived one with similar characteristics [2]), we observe that the n^{th} output DC value (which is the n^{th} HT input value HT_{in}^n) obtained from the DCT by processing the n^{th} 4×4 residue block $X^{(r)n}$ can be calculated by Eq. 2.

$$HT_{in}^n = \sum_{i=1}^{4}\sum_{j=1}^{4} X_{i,j}^{(r)n},\ n = 1,\ldots,16 \qquad (2)$$

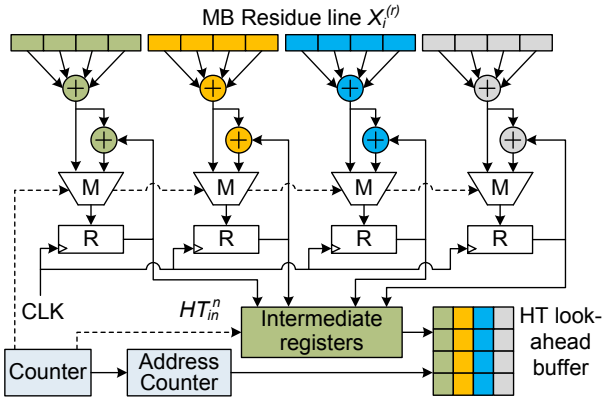

Fig. 4: HT Lookahead buffer filler responsible for HT precomputations

That shows that HT_{in}^n can be generated by adding all the entries in the 4×4 block. Thus, we generate the DC outputs at the RO stage instead of the DCT stage, effectively decoupling DCT from HT. This process is shown in Fig. 4. A residue line can add to the current accumulators (shown as registers *R*) of the look-ahead HT memory or it can trigger a new DC coefficient generation, controlled by the counter administering the MUXes *M*.

The entropy coding requires the arrival of 4×4 blocks in a Z-fashion, as shown in Fig. 2b. Our RO unit generates the 4×4 blocks for the 2D-DCT unit as required by the entropy coding on the fly, eliminating the need for an extra RO unit in front of entropy coding, further reducing the latency of the whole encoder (addressing Dependency (c) from Section I.A).

B. Mode Decision

The Mode Decision unit has to select one mode out of the four available modes (i.e. V, H, DC, and P). It can apply full search to select the best mode (i.e. the best X') using either SATD or just SAD between X and X'. The Best Mode Decision is rather slow if the generation of X' and the cost calculation for every mode is computed sequentially using one hardware unit. On the other extreme, the parallel implementation of all four modes would demand a significant amount of hardware. Therefore, we propose that only 16 adders/subtractors need to be used and thus, SAD of a

full luma MB can be generated in 16 cycles. The V mode can start executing immediately for the current MB because r'_u is always available (except the top row of the image that does not have a V mode) in the reconstruct buffers as the scanning order of the MBs is raster scan. This is also useful for DC and P mode where some computations depend upon r'_u. However, the H, DC, and P modes have to wait until all pixels in c'_l are available. A full-reconstructed 16×16 MB is available from the reconstruction block (generating one reconstructed MB line) after 20 cycles. Hence, the V mode of MB X_n can be computed in parallel to X_{n-1} in the encoding loop, because it does not depend upon c'_l. Therefore, implementing all SAD units in parallel does not result in proportional performance gain.

Fig. 5 shows our Mode Decision block. $FIFO_{Xi}$ contains X and this data is then written to a shared on-chip memory that stores line-by-line. The pipelined Edge Detector (discussed in Section III.C) unit predicts the order of modes to test by finding the most-dominant edge for X. Using this order, predictions X' are generated by the prediction generator (utilizing r'_u, c'_l, and p'_{ul}) and the residual data $X^{(r)}$ is evaluated by calculating the SAD and stored in the same on-chip buffer as X at distinct address space. The residue calculator generates $X_i^{(r)}$ and passes it to both the SAD unit and the residue memory block. After all residues are stored into this memory, the residue resulting in the lowest SAD value is written to the 4×4 RO stage. Note that if the amount of cycles needed for encoding does not comply to the resolution and framerate requirements, our SAD unit can be configured to use the down-sampled version of the current MB for residual calculation, where the down-sampling factor denoted by *ds* means that every *ds* line of X is used for the SAD computation. When $ds > 1$, then the number of cycles for one SAD computation decreases. For example, for one luma MB with $ds = 2$, it takes 8 cycles for the SAD computation rather than 16 with $ds = 1$. However, if *ds* is larger than 1, then $X^{(r)}$ is also down-sampled and thus it must be updated with the full residue after final mode selection.

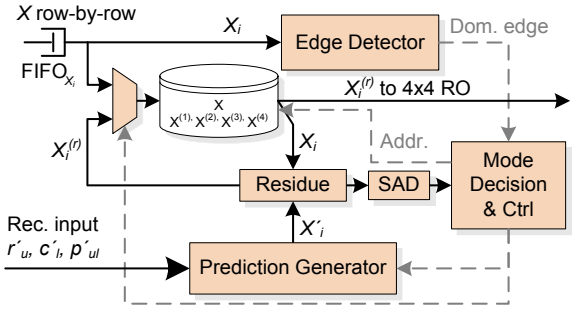

Fig. 5: Mode Decision unit for Intra 16×16

C. Edge Based Mode Prediction

If the encoding loop cycle latency does not comply with the allotted budget, it becomes necessary to sacrifice some encoding efficiency to meet the performance goals. As the transform loop is essential to the encoding process, usually less cycles of the overall budget are available to the Mode Decision block. Unlikely predictions are not selected as candidates for residue generation by using a preprocessing stage. This procedure is not required for parallel SAD implementations but it is useful for a sequential mode decider. Various algorithms are proposed in literature for mode prediction and elimination of unlikely modes [7, 11] where texture-based edge extraction information is used to determine the probable modes. Once the reconstructed data is available, the most suitable modes are applied first and the other modes are either delayed or even skipped. However, an edge extraction procedures for a 16×16 block will require 256 iterations (requiring a tan^{-1} function and a divider) plus dominant mode search cycles. Therefore, it cannot be embedded as a stage in the encoding loop or parallel to the encoding loop for QFHD sequences (encoding loop must finish within 182 cycles for 150 MHz at 25 fps). It can, however, be implemented as a separate pipeline stage outside the encoding loop. This introduces latency and limits the throughput of the encoding process. In addition, the area overhead might become too large if parallel edge extractors are employed. Moreover, the edge-threshold is decided at design time but it is not applicable to every video scene. In contrast to this approach, we propose a lightweight and efficient mode prediction process that uses a modified version of edge extraction procedures, but can still extract the dominant edge information from the input MB and does not require an edge-threshold, which would have to be adapted to the scene conditions.

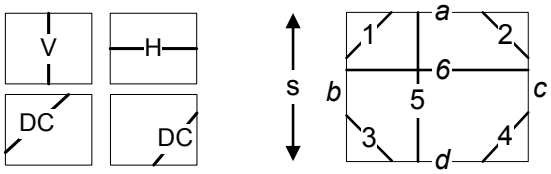

Fig. 6: Our strongest edge detection approach

For the current MB X_n, the proposed method can run in parallel with the residue calculator and transform stage and can generate the likely modes before the reconstructed previous MB X_{n-1} is available for the prediction generation stage. The crux of the method involves the estimation of edge pixels at the borders of the MBs by computing a sequential *running-difference rd*. This means that we use only one subtractor and one ABS (absolute) unit for both the top and bottom borders, and one subtractor and one ABS unit each for the left and right borders to detect the dominant edge direction. For example, at the vertical border *b* or *c* of the MB in Fig. 6, we compute rd_i as shown in Eq. 3.

$$rd_i = \left| X_{i,j} - X_{i+1,j} \right|, \ \forall i = 1,...,15, \ j \in \{1, \ 16\} \qquad (3)$$

The pixel location *i* where rd_i is maximum is declared as the point where the edge passed. Let $E_b=i$ be the location of the edge passing through *b* at *i*, generating the maximum rd_i given by rd_b. Similarly, we find edge *E* and running-difference *rd* at each border and declare the edge passing the two borders with maximum *rd*. We come up with six distinct possibilities, as shown in Fig. 6 where only one edge out of the six can occur. Using this edge detection approach, we computed the probabilities of modes for every line for various HD sequences and we devise Algorithm 1 for selecting the precedence order *m* of prediction generation. When the difference between all rd_i are less than some threshold (we keep it at constant 5 as we could not observe dependencies on the input data), we conclude that there was no edge and hence the algorithm detects $Line_0$, or no line. Note that the hardware works on a single line of input, i.e. X_i. In addition, in our Algorithm 1, the P mode is never selected first and intermediate values for P can be generated in parallel to the other modes preceding it, therefore, P mode prediction/residue computations take the same amount of cycles as the other modes. We define Depth *D* of the prediction as the number of allowable SAD computations. The value of *D* can be altered to compute SADs for the most likely modes, in order to meet the cycle budget of the encoding loop. With $D = 1$, there is no need of SAD computations and the most probable mode is used to gener-

TABLE II AVERAGE PSNR AND MBPS CHANGES IN PERCENTAGE FOR VARIOUS CONFIGURATIONS OF THE PROPOSED ENCODER VS. OPEN LOOP (OL) [12]; EACH VALUE REPRESENTS THE AVERAGE RESULTS FOR QP SWEEPS FROM 18 TO 32 (STEP SIZE = 2)

	Average PSNR [%] (QP 18...32 average, step size = 2)					Average Mbps [%] (QP 18…32 average, step size = 2)				
Sequence	***ds* = 2**	***D* = 3**	***D* = 2**	***D* = 1**	***OL* = 1**	***ds* = 2**	***D* = 3**	***D* = 2**	***D* = 1**	***OL* = 1**
Traffic (2560×1600)	-0.002	-0.005	-0.002	0.06	-0.96	0.15	0.18	0.82	2.2	12.74
People (2560×1600)	0.0002	-0.002	0.01	0.12	0.005	0.1	0.05	0.65	0.73	4.6
Riverbed (1920×1080)	-0.002	0.002	0.004	-0.016	-0.39	0.07	0.1	1.01	3.82	8.29
Basketball (1920×1080)	0.002	0.002	-1.01	-1.02	-0.19	0.36	0.37	1.48	4.2	15.76
Shields (1280×720)	-0.0012	0.0009	0.0041	0.012	-0.134	0.204	0.026	0.25	2.31	7.58
Kimonol (1920×1080)	-0.004	-0.004	-0.009	-0.003	-0.57	0.11	0.11	1.1	3.4	14.13
Tennis (1920×1080)	-0.004	0.0003	-0.028	-0.075	-0.503	0.303	0.92	3.84	9.45	20.22

input: Current MB X, size s
output: Dominant modes m

$d_w, E_w \leftarrow \text{EdgeFeature}(X), \forall w = a, b, c, d$
$Line_{out} \leftarrow \text{DominantLine}(d_w, E_w)$
switch $Line_{out}$ **do** // Note: there is an implicit 'break' after each case
  case $Line_0$
    $m \leftarrow$ (DC, P, H, V) // No edges found
  case $Line_1$
    if $E_a - E_b > s/2$ **then** $m \leftarrow$ (DC, H, P, V)
    elseif $E_a > E_b$ **then** $m \leftarrow$ (DC, P, H, V)
    elseif $E_b - E_a < s/2$ **then** $m \leftarrow$ (DC, P, V, H)
    else $m \leftarrow$ (DC, V, P, H)
  case $Line_2$
    if $E_a - E_c > s/2$ **then** $m \leftarrow$ (H, DC, P, V)
    elseif $E_a > E_c$ **then** $m \leftarrow$ (DC, H, P, V)
    elseif $E_c - E_a < s/2$ **then** $m \leftarrow$ (DC, P, H, V)
    else $m \leftarrow$ (DC, P, V, H)
  case $Line_3$
    if $E_b - E_d > s/2$ **then** $m \leftarrow$ (V, DC, P, H)
    elseif $E_b > E_d$ **then** $m \leftarrow$ (DC, V, P, H)
    elseif $E_b - E_d < s/4$ **then** $m \leftarrow$ (DC, P, V, H)
    else $m \leftarrow$ (DC, P, H, V)
  case $Line_4$
    if $E_c - E_d > 3s/4$ **then** $m \leftarrow$ (DC, V, P, H)
    elseif $E_c > E_d$ **then** $m \leftarrow$ (DC, P, V, H)
    elseif $E_d - E_c < 3s/4$ **then** $m \leftarrow$ (DC, P, V, H)
    else $m \leftarrow$ (DC, H, P, V)
  case $Line_5$
    $m \leftarrow$ (V, DC, P, H)
  case $Line_6$
    $m \leftarrow$ (H, DC, P, V)

Algorithm 1: Edge based likely mode selection

ate prediction and its residue is forwarded directly to the transform stage. For $1 < D \leq 4$, we start with the most probable mode and compute SADs.

IV. PROTOTYPE AND EXPERIMENTAL RESULTS

Our Intra encoder was developed using a Matlab/C and Modelsim co-simulation framework and it was prototyped on an Altera DK-DEV-2AGX260N FPGA Development Kit containing a cost-effective Arria II GX EP2AGX260 FPGA. For functional verification, we feed video data from an SD camera to our HD pipeline and the output bit-stream (generated by CAVLC [1]) is passed to the output node as AVB packets [13] via Gigabit Ethernet. The camera is connected to the FPGA prototyping board as shown in Fig. 7. The whole encoder uses a single clock domain of 150 MHz, the DDR3 SDRAM is clocked at 300 MHz. All hardware blocks are written in VHDL. An Altera NIOS-II embedded CPU is used for frame-by-frame control signaling and for future extensions of the prototype.

TABLE III SYNTHESIS RESULTS FOR OUR I16MB ENCODING LOOP; NOTE THAT THE TOTAL AREA (LAST ROW) EXCEED THE SUM OF THE COMPONENTS, BECAUSE THERE IS FURTHER GLUE LOGIC BETWEEN THE COMPONENTS (E.G. FIFOS)

Module	MHz	ALUT	Regs	Memory Size [Kbit]
4 × 4 RO+HT	321.34	438	1,604	0
Transform	167.87	7,901	3,958	0
Mode Pred.	171.14	1,426	747	256
Edge Detector	385.8	283	525	0
Reconstruct	475.74	460	969	0
Total	N/A	10,583	8,088	562

TABLE IV COMPARING THE PROPOSED ENCODER WITH STATE OF THE ART

Encoder	MHz	Resolution	FPS	Modes
[5]	54	720×480	31	Parallel
[7]	62.21	1920×1080	30	Parallel
[14]	140	1920×1080	30	Sequential
[8]	100	1920×1080	25	Parallel
[10]	150	1920×1080	61	Parallel
Our proposed architecture	150	3840×2160	28	Sequential

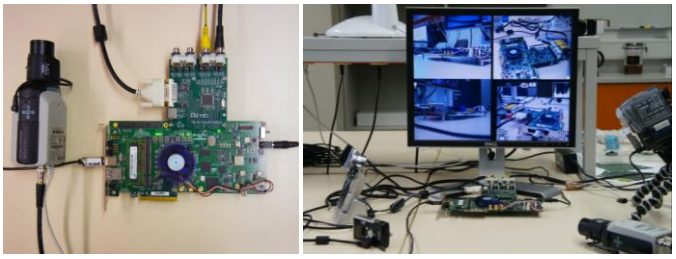

Fig. 7: Prototype implementation on an Altera Arria II GX FPGA

The total time taken by the transform loop is 56 cycles. Each X' generation requires 17 cycles, including the P prediction mode. The area usage and maximum frequency of the modules of our encoding loop implementation are given in Table III. The M9K embedded SRAM block memories are used as FIFOs to connect the encoding stages and the total amount of area of the encoding loop also includes these FIFOs.

For evaluating the encoding quality, we present a plot of Peak Signal to Noise Ratio (PSNR) in decibels (dB) against the Bitrate at depth D = 1 for various sequences shown in Fig. 8. These curves also suggest that the likely mode predictions PSNR curve matches the full search Intra mode (also called Closed Loop (CL)) selection process closely. As a comparison to the likely mode selection procedure presented in this paper, the Open Loop (OL) algorithm [12] PSNR curve for D = 1 is also plotted and is outperformed by our proposed scheme. The percentage change in PSNR and bitrate with different encoder configurations is given in Table II. Comparison of the proposed scheme with state-of-the-art Intra encoders is given in Table IV.

TABLE V DIMENSIONS SUPPORTED BY THE PROPOSED ENCODER FOR 25 FPS OPERATING AT 150 MHZ

	$D = 1$	$D = 2$	$D = 3$	$D = 4$
$ds = 1$	4894×2752	4564×2568	4294×2416	4068×2288
$ds = 2$	4894×2752	4564×2568	4416×2484	4280×2408

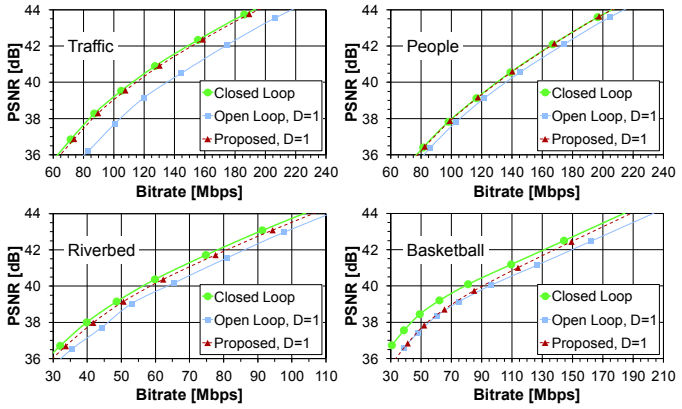

Fig. 8: PSNR vs. Bitrate plot for proposed and open loop $D=1$; each value represents the average results for QP sweeps from 18 to 32 (step size = 2)

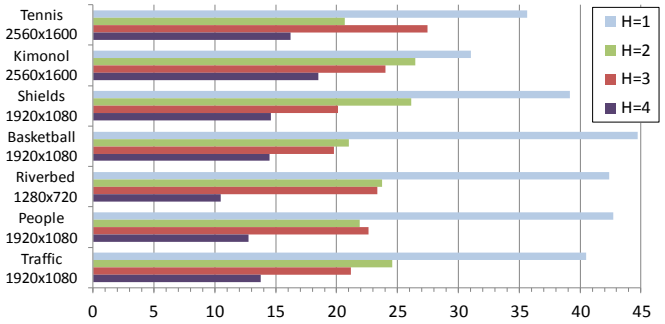

Fig. 9: Hit rates in percent (avg. over QPs 18...32, step size=2) for various sequences; H: priority of full search within mode schedule

In contrast to the others approaches, our proposed encoder can be adjusted to balance the workload and encoding methods. Moreover, hardware is saved by reusing the same structure for SAD computations of all the four modes and by realizing only one DCT folded butterfly [15] instead of 16 parallel DCTs. As compared to [9], our approach only uses one prediction unit (instead of two parallel units) at 150 MHz instead of 310 MHz. However, our approach can also be extended in a similar fashion and is more flexible because the parameters *d*s and *D* are controllable. In order to determine the total number of MBs mb_{tot} that can be processed by the encoder, given the total cycles allocated per MB c_{mb} in the loop clocked at f_s MHz with a target FPS of f_{ps}, we calculate mb_{tot} as shown in Eq. 4. Using mb_{tot}, we can calculate the maximum image dimensions for given dimension ratios. In Table V we show the maximum sustainable picture dimensions at 16:9 for different encoder configurations. We use f_s of 150 MHz at 25fps. As seen, the maximum dimensions supporting all 16×16 modes at 25 fps with no row down-sampling is 4068×2288. In Fig. 9, we report the average hit-rate of the proposed likely mode selection algorithm per frame for various sequences. H is defined as the mode priority of the full search mode in the ordered likely candidates m generated by our Algorithm 1 and $H = 4$ is the worst misprediction.

$$mb_{tot} = \frac{f_s}{f_{ps} \times c_{mb}} \times 10^6 \qquad (4)$$

V. CONCLUSION

A novel area-efficient design of likely Intra mode prediction based on the edge image information of input MB is presented. The proposed design is capable of encoding QFHD at 28 fps sequences at 150 MHz. By addressing the H.264 Intra-prediction standard-inherent data dependencies, we present an I16MB encoding loop for encoding high definition videos, while still being area efficient. We propose a Mode Decision algorithm, which yields comparable results to existing work at much smaller area footprint. By decoupling the DC path from the AC path, we reduce the latency of the transform stage and present a static mode-decision schedule together with a very simple edge detector to allow even higher resolutions by mode skipping. Only one DCT/IDCT block is used. The SAD computation unit uses only one 16-adder/subtractor unit and the mode selection procedure is sequential, thus saving area.

ACKNOWLEDGMENT

This work was partly funded by BMWI as a ZIM project. We want to thank Marcus Eggenberger for his help in implementation.

REFERENCES

[1] ITU-T Rec. H.264 and ISO/IEC 14496-10:2005 (E) (MPEG-4 AVC), "Advanced video coding for generic audiovisual services", Tech. Rep., 2005.

[2] T. Wiegand *et al.*, "Overview of the H.264/AVC video coding standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[3] J. Golston, "Comparing media codecs for video content", *Embedded Systems Conference, Texas Instruments*, 2006.

[4] J. Ostermann *et al.*, "Video coding with H.264/AVC: tools, performance, and complexity", *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, 2004.

[5] Y.-W. Huang *et al.*, "Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder", *IEEE Trans. on Circuits and Systems for Video Techn.*, vol. 15, no. 3, pp. 378–401, 2005.

[6] D. Marpe *et al.*, "Performance evaluation of Motion-JPEG2000 in comparison with H.264/AVC operated in pure intra coding mode", in *Wavelet Applications in Industrial Processing*, vol. 5266, 2004, pp. 129–137.

[7] J.-C. Wang *et al.*, "A fast mode decision algorithm and its VLSI design for H.264/AVC intra-prediction", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 10, pp. 1414–1422, 2007.

[8] M. Roszkowski and G. Pastuszak, "Intra prediction hardware module for high-profile H.264/AVC encoder", in *Signal Processing Algor., Architectures, Arrangements, and Appl. Conf.*, 2010, pp. 62–67.

[9] G. He *et al.*, "Intra prediction architecture for H.264/AVC QFHD encoder", in *Picture Coding Symposium*, 2010, pp. 450–453.

[10] C. Diniz *et al.*, "A high throughput H.264/AVC intra-frame encoding loop architecture for HD1080p", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011, pp. 579–582.

[11] F. Pan *et al.*, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 7, pp. 813—822, 2005.

[12] T. A. da Fonseca, Y. Liu, and R. L. de Queiroz, "Open-loop prediction in H.264/AVC for high definition sequences", in *Simpósio Brasileiro de Telecomunicações*, 2007.

[13] A. K. Bartky, "Draft AVBTP over IEEE 802.3AVB stream data format", http://grouper.ieee.org/, 2007, version 0.02.

[14] Y.-k. Lin *et al.*, "A 140-MHz 94 K gates HD1080p 30-frames/s Intra-only profile H.264 encoder", *IEEE Transactions on Circuits and Systems*, vol. 19, no. 3, pp. 432–436, 2009.

[15] H. S. Malvar *et al.*, "Low-complexity transform and quantization in H.264/AVC", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598–603, 2003.