Fast H.264 intra prediction for network video processing

Christophe Van Praet[1], Guy Torfs[1], Arno Vyncke[1], Elena Matei[1], Paul Cautereels[2] and Johan Bauwelinck[1]
[1] *INTEC/IMEC, Ghent University*
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
[2] *Alcatel Lucent-Bell*
Copernicuslaan 50, 2000 Antwerp, Belgium
christophe.van.praet@intec.ugent.be

Abstract: This letter proposes a fast parallel and deeply pipelined architecture for realtime H.264 intra 4x4 prediction capable of handling up to 32 High Definition video streams (1920×1080 @ 30fps) simultaneously, while offering high flexibility and consuming only a fraction of resources available on modern FPGA's. The design has been validated on target using a state of the art Altera Stratix IV FPGA.
Keywords: H.264/AVC, intra prediction, parallel processing
Classification: Electron devices, circuits, and systems

References

[1] Iain E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia* , Wiley, ISBN 0470848375, 2003.
[2] ITU-T Recommendation, H.264 *Advanced video coding for generic audiovisual services*, March, 2010
[3] T.-C. Chen, C.J. Lian, and L.-G. Chen, "Hardware Architecture Design of an H.264/AVC Video Codec," Proc. 11th Asia and South Pacific Design Automation Conf., Yokohama, Japan, pp. 750-757, January, 2006.
[4] J. Li, and E. Abdel-Raheem, "Fast implementation of H.264 4x4 intra prediction," *IEICE Electron. Express* vol. 7, no. 5, pp. 332-338, March, 2010.
[5] G. Jin, J.-S. Jung, and H.-J. Lee, "An Efficient Pipelined Architecture for H.264/AVC Intra Frame Processing," Int. Sym. on Cir. and Sys., New Orleans, USA, pp. 1605-1608, May, 2007.
[6] J.W. Chen, C.-H. Chang, C.C. Lin, Y.-H. Ou Yang, J.-I. Guo, and J.-S. Wang, "A Condition-based Intra Prediction Algorithm for H.264/AVC," IEEE Int. Conf. on Mult. and Expo, Toronto, Canada, pp. 1077-1080, July, 2006
[7] Y.W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder," IEEE Trans. on Cir. and Sys. for Video Tech. vol. 15, no. 3, pp. 378-401, March, 2005
[8] H. Loukil, B. Kaanich, N. Masmoudi, and A. Ben Atitallah, "An Efficient Hardware Architecture Design for H.264/AVC INTRA 4×4 Algorithm," First Workshops on Image Processing Theory, Tools and Applications, Sousse, Tunisia, pp. 1-5, November, 2008.

1 Introduction

The H.264/AVC video compression standard achieves considerably higher bandwidth efficiencies than former standards by means of aggressive compression techniques, at the expense of dozens of compression modes and an increased number of computations [1][2]. While the complexity of the encoding process favors a software approach to achieve optimal compression ratios, the computational complexity demands for dedicated hardware for the most computation-intensive operations to allow real-time performance as is often required for modern-day applications [3].

Compression is achieved by means of inter or intra prediction, which respectively attempt to eliminate temporal redundancy in consecutive video frames or spatial redundancy within a single frame. For intra prediction, the encoder evaluates several prediction modes, and finally chooses the one believed to yield the smallest bitstream after encoding the resulting residuals.

In [4] an overview is presented of previously developed techniques to speed up the intra prediction process, by pipelining calculations, exploiting parallelism, changing the order in which blocks are processed [5] and even skipping prediction modes [5]. Furthermore, [4] presents a pipeline which supports the same modes as in this work, though no performance is mentioned. Other solutions reduce the required number of computations by using an algorithm to determine and skip less likely modes. One way is to analyze the mode selected for neighboring blocks [6], or by quickly analyzing subsampled blocks [7].

This letter proposes a highly parallel and deeply pipelined architecture which allows real-time 4×4 intra-encoding on 32 high definition frames simultaneously, while preserving all 9 luma modes and allowing high flexibility.

At present the design supports DC chroma mode, though the architecture can easily be extended to support all chroma modes. Furthermore the architecture supports the most-probable-mode feature [1].

In Section 2, intra prediction is introduced, in Section 3 the novel architecture is proposed, Section 4 presents the results and in Section 5 conclusions are drawn.

2 H.264 Intra prediction

For luma pixels, the standard describes 4 prediction modes on 16×16 blocks and 9 on 4×4 blocks. For chroma, one of 4 modes has to be chosen per 8×8 block, though the predictions are calculated on 4×4 basis and are a subset of the luma modes. The decision-algorithm, responsible for selecting that prediction mode which is believed to result in the smallest bitstream, is not specified by the standard [2]. Though more complex algorithms have been developed in the past [5][7], for real-time implementation on dedicated hardware the brute force approach followed by summing of the residuals (Sum of Absolute Differences, SAD) to make the decision is popular [4].

The major bottleneck is the data dependency between consecutive blocks of an image. A block has to wait for processing until its left and upper neighbors have been predicted, encoded (lossy) and decoded again. Current

implementations try to reduce the bottleneck by changing the order in which the blocks are processed [5], reducing the number of supported modes [5], preselecting modes based on prior decisions [6] or analysis of subsampled blocks [7]. However, disabling, favoring or preselecting modes may –depending on the algorithm and the actual data– come at the expense of a lower compression ratio or decreased picture quality, usually expressed in terms of SNR.

Due to dependencies, simply changing the processing order, doesn't allow to fill a pipeline completely without skipping modes, hence the hardware is used inefficiently [4]. In [4], a solution is presented which prevents bubbles in the pipeline by clever disassembly, regrouping and scheduling of the required calculations. In this work we propose to keep the pipeline filled by processing other streams while waiting for our data to be processed and thus, dependencies to be resolved. In contrast with previous implementations, this allows the pipeline to have an arbitrary number of stages, enabling to balance the maximum operation frequency and the number of simultaneously processed streams.

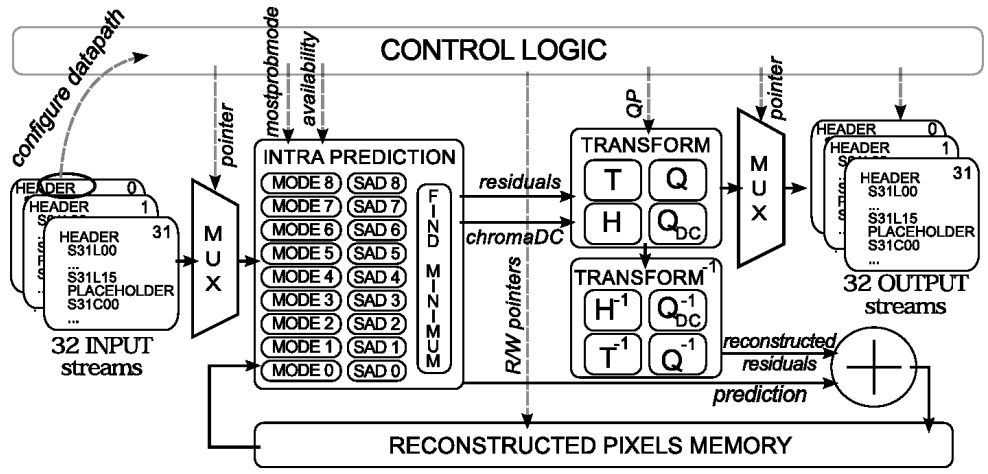
3 Proposed architecture

Fig. **1**.a represents the proposed architecture. In order to handle 32 HD video streams simultaneously and real-time, a high throughput –up to 3×10^9 pixels per second– is required. We opted for a fast datapath solution, with a control block to configure the datapath and exploited parallelism as much as possible.

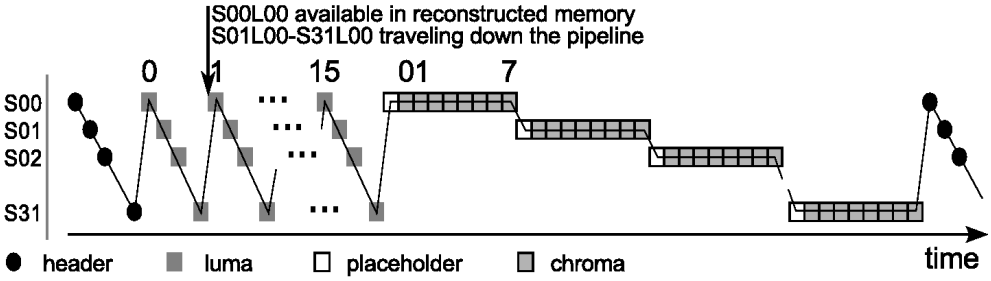
3.1 Parallellism and pipelining

The datapath bus transfers one 4×4-block per cycle. All operations are performed directly on this 4×4-block level. As shown in Fig. **1**.a, all 9 luma predictions are calculated in parallel. The control block disables irrelevant modes, depending on availability and type of block (luma/chroma). The residuals for every mode and their energy (SAD) are also evaluated in parallel. The minimizing algorithm picks the lowest of 2 SADs, favoring the mostprobmode signaled by the control block when two SADs would be equal. The first stage of the algorithm executes 4 comparisons in parallel, then 2, then 1 and finally 1 again, yielding the chosen mode, predicted pixels and residuals. Aiming for minimum hardware cost and 32 HD streams, in our implementation of the proposed architecture processing one 4×4-block takes 19 cycles whereafter the result is stored in memory for an additional 13 cycles, yielding 32 cycles in total (see Fig. **1**.c).

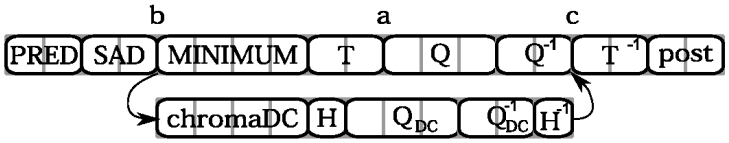
To allow for a very flexible system, e.g. where every stream can have its own frame size or quantization parameters (QP), every macroblock –consisting of 16 luma 4×4-blocks, 4 chroma U and 4 chroma V 4×4-blocks– is preceded by a header. Every header contains a QP for chroma and one for luma and flags to indicate the start of a new frame, the beginning of a new line and the end of the line. The header is decoded by the controller, a complex state machine, which uses the information in the headers to signal the intra prediction block about the availability of previous blocks (and hence about



(a) Overview of the presented architecture



(b) Order of intake of 4x4 subblocks



(c) Pipeline stages

Fig. 1. Presented architecture

valid prediction modes) for the particular stream to which the incoming block belongs. It also signals the quantization and dequantization block about the appropriate QP. After processing, the header contains the selected prediction modes for all the 4×4-blocks it affects, which signals the entropy encoder.

3.2 Description of the processing order

To efficiently fill the pipeline, the blocks must be processed in a specific order. This order is represented in Fig. **1**.b. Let SXX indicate Stream number XX (0..31), LYY Luma subblock YY (0..15) and CZZ Chroma subblock ZZ (0..7). During the first 32 cycles the system reads one header-block from each of the input streams and uses the information provided in those headers (SxxHEADER) to configure the datapath. Hereafter, S00L00 is fed to the pipeline, followed by the first block of the other streams (SxxL00). As the presented implementation has 19 pipeline stages (as illustrated in **1**.c), after 19 cycles the S00L00 has finished propagating down the pipeline. The transformed residuals are multiplexed to their respective output stream and the reconstructed pixels needed for future predictions are stored in the memory.

32 cycles after S00L00 was sent into the pipeline, its reconstructed pixels are retrieved from the internal memory and the system is ready to process S00L01. This process goes on until S31L15 has been processed.

Next a placeholder block is sent into the system to store the two 2×2 processed DC blocks originating from S00C00–S00C03 and S00C04–S00C07. Hereafter S00C00–S00C07 are processed, followed by a new placeholder and S01C00–S01C07. This goes on till S31C07. Finally new headers are expected at the input.

The reason chroma blocks are processed in a different order is that the DC components of each four chroma-blocks in a macroblock undergo an additional transformation (H) to reduce their correlation and, hence, improve compression. It is possible to process them consecutively as the prediction modes for chroma subblocks never rely on pixels to the right of the actual subblock, contrary to luma subblocks [2].

Fig. **1**.c illustrates how these 2×2 DC blocks are processed. Normally the DC components only become available after transformation to the spatial frequency domain (point "a" in Fig. **1**.c). In the proposed architecture, however, they are calculated along with the SAD 6 cycles earlier in the pipeline ("b"). Once calculated, the 4 DC components of 4 chroma subblocks need to be collected (indicated by "chromaDC" in Fig. **1**.c), prior to applying the Hadamard transformation H and dedicated quantization Q_{DC}. They are immediately reconstructed to be available right after regular dequantization Q^{-1} ("c" in Fig. **1**.c) to allow reconstruction according to the standard [2] minimizing the required number of pipeline stages. The transformed chroma DC values are stored on the placeholder to be readily available to the succeeding encoding process, which also handles the chroma DC separately [2], to reduce latency.

4 Results

The design has been verified to run at more than 220 MHz on the DK-DEV-4SGX230N Altera Stratix IV development kit. As the architecture requires 26 cycles (header, placeholder, 16 luma, 8 chroma) per macroblock, this allows 32 High Definition streams (1920×1080) to be processed at 30 fps simultaneously and in real-time. The design uses approximately 16k Altera logic cell combinationals, 16k Altera logic cell registers, 72 DSP 18-bit elements and 36 DSP 18×18. All DSP blocks are used for multiplications in the transform block. For more details on resource utilization, see table **I**.

Table **II** compares the proposed architecture with other state of the art implementations in terms of logic cost and processing speed. Despite the strong parallelism and the deep pipelining in the proposed architecture, the hardware cost per stream remains relatively low. This shows that the hardware is used very efficiently indeed. Furthermore, the throughput for similar operation frequencies proves to be very high. The memory usage is higher because every 4×4-block is kept in memory for 13 cycles.

There is still margin to improve performance on slower devices or to sup-

Table I. Resource utilization summary

Block name	LC Comb.	LC Reg.	Block Mem bits	DSP 18-bit	DSP 18×18
Control	3621	4862	499	0	0
Reconstruction Memory	114	14	435200	0	0
Intra prediction	6911	7913	1470	0	0
Transforms	4807	3294	60	72	36
Multiplexers	240	0	0	0	0
Top	15693	16083	437229	72	36

Table II. Comparison

	This work	[3]	[4]	[8]
clockfrequency (MHz)	200	120	–	160
kbit memory	437	5	–	7.5
kbit memory/stream	13.6	5	–	7.5
macroblocks/1000 cycles	36	–	4.9	2.3
gates (estimated)	1984k	121k	–	220k
gates/stream (estimated)	62k	121k	–	220k
target device	Stratix IV	0.18um CMOS	Virtex 4	Stratix II

port even higher resolutions or frame rates. In the presented implementation the pipeline has 19 stages and the predicted pixels reside in memory for another 13 cycles. These memory-cycles could be traded for extra pipeline stages where bottlenecks occur, allowing even higher clock-rates.

5 Conclusion

The new design achieves a very high throughput while offering high flexibility by means of flags (new frame/slice, new line, and endline) signalling the position of each 4×4 subblock along with quantization parameters for chroma and luma. By cleverly exploiting the need of encoding several streams, it is possible to almost entirely avoid so-called bubbles in the presented pipeline.

As described earlier, the architecture can be extended to process an additional 13 HD streams. As the architecture can operate on any frame size, smaller frames could be processed at a higher frame-rate or lower resolution streams could even share a single queue given additional multiplexers and buffers to interleave and de-interleave streams on a frame-basis. Furthermore, its small size (21% of Altera EP4SGX230KF40C2) allows for many instantiations on a single FPGA, drastically increasing the throughput even further.

Acknowledgments

The authors would like to thank the Special Research Fund of Ghent University, the Agency for Innovation by Science and Technology in Flanders (IWT) for funding the Vampire Project and Alcatel Lucent-Bell (ALU) Antwerpen for their collaboration and financial support.