Real Time All Intra HEVC HD Encoder on FPGA

Sachille Atapattu, Namitha Liyanage, Nisal Menuka, Ishantha Perera and Ajith Pasqual
Dept. of Electronic and Telecommunication Engineering, University of Moratuwa, Sri Lanka
e-mails {sachille, namitha, ishantha}@paraqum.com, {nisal, pasqual}@ent.mrt.ac.lk

***Abstract*—Higher compression efficiency in HEVC encoders comes with increased computational complexity, making real time encoding of high resolution videos a challenging task. This challenge can be addressed by software, yet hardware solutions are more appealing due to their superior performance and low power consumption. This paper presents an FPGA based hardware implementation of an all intra HEVC encoder, which can encode 8 bits per sample, 1920x1080 resolution, 30 frames per second raw video, that is viable in real time even at low operating frequencies. A major obstacle to real time encoding in available architectures is the dependency created by reference generation. Moreover, each coding unit (CU) has to be processed in multiple configurations to determine the most efficient split and prediction mode representation, based on the bit stream generated. We propose a new three stage architecture to reduce these dependencies and increase parallelism. Feedback needed for CU split and prediction direction decision from binarization is avoided by a Hadamard based early decision method. Feedback constrained coefficient and reconstruction derivation module exploits several optimization techniques. All modules can operate at 200 MHz and the encoder can achieve real time encoding with a minimum operating frequency of 140 MHz. The design consumes 83K LUTs, 28K registers, and 34 DSPs when implemented on Xilinx Zynq ZC706.**

***Index Terms*—high efficiency video coding, video coding on fpga, intra coding, early CU partitioning, early mode decision, low latency encoding**

I. Introduction

Developed by Joint Collaborative Team on Video Coding (JCT-VC), H.265/High Efficiency Video Coding (HEVC) is the successor to the highly popular video encoding standard H.264/Advanced Video Coding (AVC). Formally ratified in early 2013, HEVC is designed to achieve over 50% bit rate reduction over H.264 and is expected to be the video codec of choice for the future. This improvement of coding efficiency comes with increased computational complexity, especially at the encoder end, making real-time encoding a major challenge.

Even though this can be addressed by software solutions, a huge demand for hardware HEVC encoders exists, especially in broadcasting and mobile applications, due to their superior performance, high reliability, and low power consumption. Hardware encoders can be implemented by either application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs). While ASIC solutions have higher performance, it is only economical for very large volume production as they require custom made silicon chip. Whereas FPGA based encoders, which are implemented using already existing configurable hardware platforms, are more cost effective for low volume production making it suitable for encoding solutions during adoption period. Moreover, it entails less risk as it could be easily configured to match an evolving standard.

HEVC introduces an enhanced prediction model over H.264. 16x16 macroblocks found in H.264 are replaced by 64x64 coding tree units (CTUs), where CTUs can be split recursively into coding units (CUs). Each CU can then be split recursively into transform units (TUs) and prediction units (PUs). HEVC further introduces 35 modes for intra prediction in contrast to 9 modes in H.264. The best compression efficiency can be achieved by using rate distortion optimization (RDO) where optimal mode (prediction direction and splitting structure) is searched in a computationally expensive brute force manner. HM reference encoder version 16.7 uses the sum of absolute transformed differences as the cost function to preselect few modes instead of performing full RDO for all modes [1]. However, considerable amount of modes still undergo full RD cost optimization resulting in a significant time to encode raw videos.

The number of complete hardware implementations for HEVC encoding still remains low. Although there are a few commercial hardware encoders, their architecture and algorithms remain proprietary. Very few works related to entire encoder implementations exist in literature, while many have focused on implementations of different parts of the encoder. Work by Miyazawa et al. [2] presents an implementation of an HEVC encoder for real time encoding of 1080p, 10 bits per sample, 60 frames/s video without specifying the operating frequency or the resource usage. HEVC encoder proposed by Zhu et al. [3] estimates the rate distortion (RD) cost from source image textures and preselects two modes for RDO processing. It can support real time encoding at 1080p, 44 frames/s video. However, the ASIC implementation has a high area cost with 2269k gates and runs at a 357 MHz operating frequency. Pastuszak et al. [4] presents a computationally scalable algorithm and a hardware architecture able to support intra encoding up to 2160p, 30 frames/s resolution. The encoder selects a set of candidate modes by processing 8x8 predictions computed from original samples and also takes advantage of simplified bin count based on RDO. The FPGA implementation of this work can encode 1080p video up to 35 frames/s. This implementation uses much higher numbers of lookup tables (LUTs) and digital signal processor (DSP) blocks than our proposed design.

In this paper, we present an FPGA based hardware implementation of an all intra HEVC encoder which can encode HD quality, 8 bits per sample, 30 frames/s raw video in real time. The architecture is optimized to achieve real time

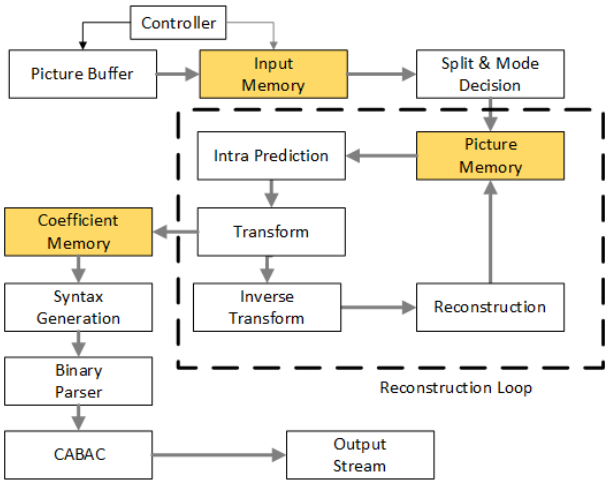

Fig. 1. Overall architecture of encoder

encoding with an operating frequency of 140 MHz while using minimum hardware resources. The design was emulated in a Xilinx Zynq ZC706 platform. Our design introduces following novelties in the architecture to the best of our knowledge.

- Reconstruction loop architecture avoiding multiple feedback and shared between luma and chroma
- Early split structure and mode decision entirely avoiding RD optimization process
- Pipelined TU coding architecture of 4x4 block per cycle
- Intra prediction architecture optimized for 4x4 blocks to predict 19 modes in parallel

The rest of the paper is organized as follows. Section II describes the overall architecture of the encoder. Key modules are described in detail in Section III. Implementation results are presented in Section IV.

II. Overall Architecture

The system level architecture of our encoder design is shown in Fig. 1. The encoder consists of a three stage, highly pipelined architecture, devoid of any feedback between stages, to support real time encoding. To enable separate development and verification, each stage is made independent of others. Memories with subspaces are placed between stages to establish a coherent method of passing data to the next stage. The architecture also focuses on increasing parallelism in modular level and reducing resource consumption.

First stage of the architecture consists of a controller module for input picture buffer and an internal memory along with split and early mode decision module. Split and early mode decision module selects the suitable quad-tree structure and prediction direction for each CU in the quad-tree. These data along with the raw pixels are passed to the picture memory.

The second stage includes the reconstruction loop, where dependencies are created by reference pixels required by intra prediction. Therefore, this stage is optimized to reduce latency of the loop. To achieve the desired latency and to minimize resource usage, our design proposes novel architectures for intra prediction and a combined architecture for forward and inverse transform. The modules are also shared between luma and chroma color planes to further reduce the resource usage. The final stage consists of syntax generation unit, binary parser and entropy coder which operate sequentially to achieve the required output bit rate for HEVC encoding of 30 frames/s video.

TU and PU partitions are kept as same as CU partition except for 8x8 CUs to reduce the amount of comparisons needed to be performed. Since the maximum TU block size is 32x32, this implementation provides CU blocks only up to 32x32.

III. Key Modules

A. Early Mode Decision

Proposed methods of deducing the optimal split structure determined from computationally expensive full RDO process fall into two broad approaches. One attempts to speed up decision by early terminating the RDO process, and the other assists mode decision by using low complexity RD cost estimators [5]. Even though some such implementations reduce feedback requirement for certain stages [5], feedback from reconstruction loop is essential for full RD cost comparison of quad tree structure in both these approaches. This feedback restricts full parallel implementation of encoding algorithms. Early split and mode decision devoid of any feedback information from reconstruction loop is one of the key novelties in the architecture we propose.

While different RD cost estimators can be employed, a Hadamard transform based RD cost estimation is selected to evaluate early decision approach in this implementation. Original pixels are used in place of the reconstructed pixels for reference generation in split and mode decision. This enables the subsequent encoding to be carried out to a predetermined split and mode structure.

Furthermore, to reduce resource usage, each CU sized original pixel array and reference pixel array are subsampled to a 4x4 block for prediction and Hadamard cost calculation. Top level architecture of early mode decision module is shown in Fig. 2. Intra prediction on all 35 prediction modes are done for each 4x4 block generated by sub-sampling process. The residual generated from the prediction undergoes a Hadamard transformation process. The selection unit picks the best mode for each block based on distortion results and makes a split decision if this best mode cost surpasses the cumulative cost from the optimal smaller block structure.

The intra prediction module applied in early mode decision is different from the intra prediction module implemented within the reconstruction loop. As intra prediction within early mode decision module is no longer bounded by latency, more focus is given on maximizing throughput and parallel implementation of submodules. The highly parallel architecture consists of 17 angular prediction modules, one planar prediction module and one DC prediction module. Each module is designed to output 4 pixels (1 row in a 4x4 block) per cycle.

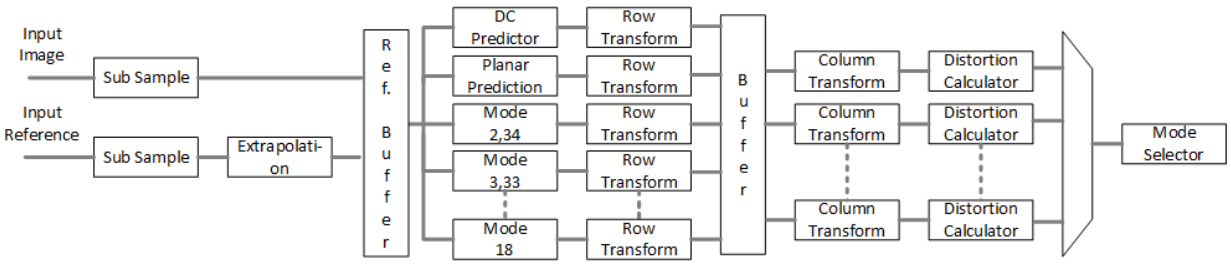

Fig. 2. Early split architecture combining intra prediction and Hadamard transformation

Intra prediction operation for any row for modes 2-17 can be obtained by shifting the horizontal reference samples used to predict the row above in the same block. This reduces the need to implement separate multiplications for all 16 pixel locations. Prediction values for modes 18-34 are obtained by interleaving horizontal and vertical reference buffers and transposition of original block, as these modes are diagonal mirror images of modes 2-16, further reducing resource usage. As each intra prediction module is dedicated to one single mode, the two variable multiplication in intra prediction equation [6] can be implemented as a constant factor multiplication for each pixel requiring merely shifters and adders.

Hadamard transformation is implemented in two stages. Each incoming row undergoes a row transformation. The output of row transformation is stored in an intermediate buffer until all 4 row transformations are completed. These data then undergo a column transformation one column per cycle. Using the given architecture, Hadamard distortion parameters for all 35 modes for a certain block can be calculated in 19 clock cycles with a throughput of 8 cycles per block.

B. Intra Prediction

Several architectures have been proposed to increase the throughput of intra prediction. However, as intra prediction is traditionally implemented in mode decision, most algorithms including [7], [8], and [9] focus on optimizing prediction of same block using different prediction directions. In this work, more focus is given on reducing the latency of prediction since mode decision is taken earlier.

The picture memory consists of two buffers named vertical and horizontal buffers which are equal to picture width (1920 pixels) and height (1080 pixels) alongside a 32x32 raw pixel block. Intra prediction unit can access 4 reference samples from each buffer and a 4x4 block of raw pixels per cycle. These reference values are filled to two internal reference buffers. Extrapolation and pre-filtering of reference values are performed at 8 elements per cycle. Same internal buffers are updated to reduce the resource usage.

The multiplication operations in planar prediction mode is replaced by a series of shifters and adders to reduce the need of 32 multipliers per 4x4 block. For DC mode, an adder tree is implemented to calculate the DC value. The output blocks from the prediction unit undergoes post filtering for specific prediction directions at a maximum of 4 pixels per cycle.

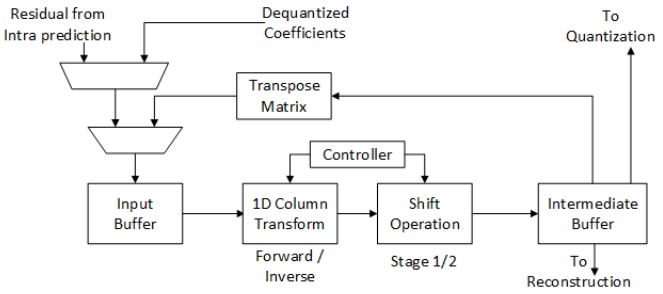

Fig. 3. Architecture for combined forward and inverse transform

At the final stage, predicted values are substituted from the original values and the difference is passed to the transform module. The predicted pixel values are passed alongside to the reconstruction module. The intra prediction operation occurs at one 4x4 block per cycle with a latency of 6 cycles and can run on a frequency up to 200 MHz.

As our architecture uses the same luma prediction direction for chroma mode, chroma prediction begins soon after the luma prediction finishes. This reduces the resource usage and utilizes idle time due to data dependencies in the luma reconstruction loop.

C. Forward and Inverse Transform

Transform modules contain 2D matrix multiplications which have a very high area cost when implemented in hardware directly. [10] and [11] propose multiplierless, butterfly structure hardware architecture and [12] proposes a unified forward and inverse transform architecture to reduce the area cost for the transforms. In this architecture, discrete sine transform (DST) is implemented separately from discrete cosine transform (DCT). The forward and inverse transforms for DST are implemented separately supporting 4x4 blocks. A combined transform architecture is used for DCT which can support 4x4, 8x8, 16x16, and 32x32 block sizes. The input matrix first undergoes a column transformation followed by stage 1 shift operation. The row transform is achieved by having the column transformation applied to the transpose of the resultant matrix. Fig. 3 shows high level architecture of this module.

The implemented DCT can perform one stage of transform for all 4 columns of a 4x4 block, single column of a 8x8 block, 1/2 of a single column of a 16x16 block, and 1/4 of a single column of a 32x32 block in one pass. In addition

to transformation, this module performs quantization, sign bit hiding, and dequantization. The highly pipelined design enables processing of both luma and chroma blocks in the same module to reduce the resource usage. However, this architecture is highly optimized for HD quality, 30 frames/s video and may not be sufficient for higher resolution videos.

D. Syntax Generation

The syntax generation unit converts PU and TU data to HEVC syntax. This consists of a block selection unit, coefficient syntax generation unit, prediction syntax generation unit, and a syntax ordering unit. To prevent any latency built up in syntax generation from affecting the time critical reconstruction loop, coefficient memory is used to separate the two sections. Coefficient memory unit consists of 2 subspaces, one undergoing reconstruction loop and the other subspace undergoing coefficient coding, to improve parallelism.

Block ordering unit picks 4x4 coefficient blocks in coding order, depending on the prediction direction and transform block size and pushes them to the coefficient syntax generation unit. The coefficient syntax generation unit is a 7 stage pipeline design which can extract syntax elements and other parameters needed for binary conversion and context assignment. The unit has a throughput of a 4x4 block per cycle and a latency of 7 clock cycles and can run on a frequency up to 250 MHz on a Zynq ZC706 FPGA. The throughput of a 4x4 block per cycle is useful when improving the performance of binary parsing and entropy coding unit to accommodate higher resolutions and frame rates, as it prevents pipelines in those modules from slowing down due to data dependencies. The syntax ordering unit picks the TU and PU level data and orders them according to HEVC syntax and passes them to binary parser, where each syntax element is passed to the entropy coding module with a context table and a context increment for each bin position.

E. Entropy coding

Common bottleneck in coefficient coding is the lengthy feedback loop in the entropy coder. Several approaches have been proposed to increase the throughput of entropy coding. [13] is one of the early works to surpass 1 bin per cycle in HEVC encoding on FPGA. [14] proposes an architecture which could encode 4 entropy coding bins per cycle targeting 90 nm. However, as work mentioned in this paper is targeting commercial FPGA designs the throughput is kept at 1 bin per cycle for entropy coding. [13] has shown this is sufficient to encode 60 frames/s video beyond 2560x1600 resolution.

The proposed entropy coding unit has 5 stages as shown in Fig. 4. At stage 1, variables valMPS and PStateIdx are pre-calculated and updated for each bin to reduce the critical path delay. At stage 2 and 3, values of ivlCurrRange and ivlLow at the end of renormalization operation are also pre-calculated to remove the feedback required from renormalization module. ivlLow and ivlCurrRange update of bypass coding is also implemented in parallel to the 3 stages. Stage 4 is the renormalization pipeline which generates the final bit stream in multiple steps with a throughput of 1 renormalize operation per cycle. The latency of this module is no longer critical as the final encoder states are not dependent on the result of the renormalize operation. The final stage of the entropy coder is the emulation preventer, which also adds HEVC headers to the bit stream and outputs the final bit stream in 8 bit units.

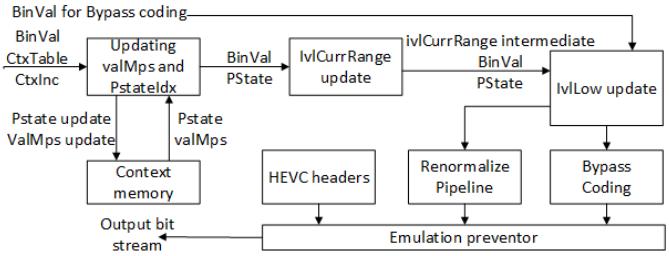

Fig. 4. Top level architecture of entropy coding unit

TABLE I
RESOURCE UTILIZATION OF KEY MODULES

Module	LUTs	Registers	DSP
Intra prediction in reconstruction loop	6116	3682	33
Intra prediction in early mode decision	6853	3789	
Hadamard transformation	2825	1540	
Picture memory handling	1343	985	
Forward and inverse transformation	54308	12789	
Prediction data coding	1233	547	
Residual coding pipeline	5233	2447	
Binary parser	796	790	
Entropy coder	3177	969	1

IV. RESULTS

The HEVC encoder proposed by the authors are specified in Verilog and is targeting Zynq ZC706 FPGA platform. The overall encoder is synthesized at 140 MHz, while individual modules are synthesized at 200 MHz during standalone implementations. The design has been verified at 30 frames/s and is capable of operating at higher frequencies. Table I shows the resource utilization of key modules. The total resource utilization is 83548 LUTs, 28303 registers, and 34 DSPs.

It can be seen that transformation modules use most of the resources (65%). In earlier implementations, we used separate reconstruction loops for luma and chroma, which required 114K LUTs with transformation requiring 72% of it.

When comparing with existing literature, most of the work are implemented in ASIC targeting TSMC libraries. Therefore, it is not possible to achieve a proper performance comparison. However, Miyazawa et al. [2] and Pastuzak et al. [4] have given some results for FPGA implementations of their architectures. Along with them, this work is compared with the work of Zhu et al. [3] in Table II. It can be seen that the proposed architecture requires the least amount of resources (LUT count and DSP count) when implemented in FPGA. However, the lack of specific results for configurations HD, 8 bit, and 30 frames/s prevents us from making a more meaningful comparison.

TABLE II
COMPARISON WITH EXISTING LITERATURE

Design	[2]	[3]	[4]	This work
Technology	FPGA	TSMC 90nm	FPGA[1]	FPGA
Frequency	-	357 MHz	100 MHz[2]	140 MHz
Resolution	1080p	1080p	1080p	1080p
Frames/s	60	44	60	30
LUT count	-	2269k gates	93184	83548
DSP count	-	-	481	34

[1]Synthesized on Arria II GX, which uses 8-input LUTs.
[2]Some modules run at 200 MHz.

TABLE III
CODING OVERHEAD DUE TO EARLY SPLIT STRUCTURE

Video	BD rate	Video	BD rate
Kimono	10.51	Old town cross	8.67
Park Scene	8.60	Station 2	11.40
Cactus	25.56	Sunflower	19.22
In to tree	8.50	Crowd run	8.76
Average			12.66

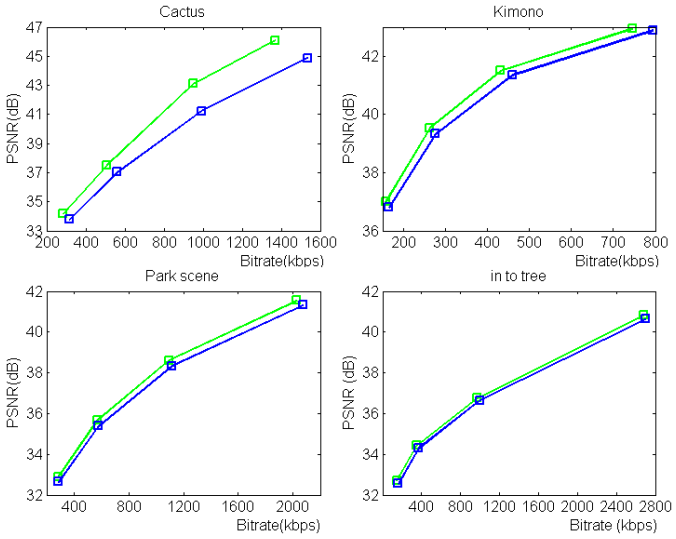

Fig. 5. PSNR vs bit rate plots of encoding. Green line indicates results of HM v16.7 and blue line indicates results of early split decision tested in HM

Coding performance of our architecture depends on the success of the early split and mode decision. Therefore, the early split structure is compared with HM reference encoder version 16.7 [1] for evaluation of several HD test video sequences at quantization parameter (QP) = 22,27,32,37. The metric used for the test is Bjontegaard Delta Rate (BD- Rate). As shown in Table III, the proposed early split decision technique in our architecture results in an average bit rate increment of 13% for real time encoding. Videos mentioned in Table III, Kimono, Park Scene, and Cactus are class B videos in JCT-VC common test conditions. The rest are from the test media collection at [15]. The results due to early split is further elaborated in the plots shown in Fig. 5.

V. CONCLUSION

The architecture suggested in this paper uses a novel design of isolating split decision and mode decision from the feedback loop. Due to eliminating RD optimization for multiple modes, the proposed encoder is able to reduce the encoding time and resource utilization significantly. However, a coding overhead of around 13% is added due to the current early split algorithm. This trade-off is justifiable when the resource reduction and real time 30 frames/s encoding is considered. Moreover, this coding overhead could be further reduced by improving the early split algorithm. Therefore, the authors consider this as a viable path for real time HEVC encoders on FPGA.

REFERENCES

[1] "HM software repository," accessed 2016. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

[2] K. Miyazawa, H. Sakate, S. i. Sekiguchi, N. Motoyama, Y. Sugito, K. Iguchi, A. Ichigaya, and S. i. Sakaida, "Real-time hardware implementation of hevc video encoder for 1080p hd video," in *Picture Coding Symposium (PCS), 2013*, Dec 2013, pp. 225–228.

[3] J. Zhu, Z. Liu, D. Wang, Q. Han, and Y. Song, "Hdtv1080p hevc intra encoder with source texture based cu/pu mode pre-decision," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, Jan 2014, pp. 367–372.

[4] G. Pastuszak and A. Abramowski, "Algorithm and architecture design of the h.265/hevc intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 210–222, Jan 2016.

[5] J. Zhu, Z. Liu, D. Wang, Q. Han, and Y. Song, "Fast prediction mode decision with hadamard transform based rate-distortion cost estimation for hevc intra coding," in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 1977–1981.

[6] ITU-T, "High efficiency video coding - Recommendation ITU-T H.265," *ITU-T H-Series recommendations audiovisual and multimedia systems*, 2014.

[7] A. Abramowski and G. Pastuszak, "A novel intra prediction architecture for the hardware hevc encoder," in *Digital System Design (DSD), 2013 Euromicro Conference on*, Sept 2013, pp. 429–436.

[8] Y. Jiang, D. Llamocca, M. Pattichis, and G. Esakki, "A unified and pipelined hardware architecture for implementing intra prediction in hevc," in *Image Analysis and Interpretation (SSIAI), 2014 IEEE Southwest Symposium on*, April 2014, pp. 29–32.

[9] E. Kalali, Y. Adibelli, and I. Hamzaoglu, "A high performance and low energy intra prediction hardware for high efficiency video coding," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2012, pp. 719–722.

[10] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer dct architectures for hevc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 168–178, Jan 2014.

[11] S.-M. H. J.-S. Park, W.-J. Nam and S. Lee, "2-d large inverse transform (16x16, 32x32) for hevc (high efficiency video coding)," *Journal of Semiconductor Technology and Science*, vol. 12, no. 2, p. 203211, Jun 2012.

[12] M. Budagavi and V. Sze, "Unified forward+inverse transform architecture for hevc," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, Sept 2012, pp. 209–212.

[13] B. Peng, D. Ding, X. Zhu, and L. Yu, "A hardware cabac encoder for hevc," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, May 2013, pp. 1372–1375.

[14] D. Zhou, J. Zhou, W. Fei, and S. Goto, "Ultra-high-throughput vlsi architecture of h.265/hevc cabac encoder for uhdtv applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 497–507, March 2015.

[15] "Xiph.org video test media," accessed 2016. [Online]. Available: https://media.xiph.org/video/derf/