Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2013, Article ID 834793, 9 pages
http://dx.doi.org/10.1155/2013/834793

Research Article

Asynchronous Realization of Algebraic Integer-Based 2D DCT Using Achronix Speedster SPD60 FPGA

Nilanka Rajapaksha,[1] Amila Edirisuriya,[1] Arjuna Madanayake,[1] Renato J. Cintra,[2] Dennis Onen,[3] Ihab Amer,[4] and Vassil S. Dimitrov[3]

[1] *Electrical and Computer Engineering, Auburn Science and Engineering Center (ASEC) 265, The University of Akron, Akron, OH 44325-3904, USA*
[2] *Signal Processing Group, Department of Statistics, Federal University of Pernambuco, 50740-540 Recife, PE, Brazil*
[3] *Department of Electrical and Computer Engineering, ICT 402, Schulich School of Engineering, University of Calgary, 2500 University Drive NW Calgary, Alberta, Calgary, AB, Canada T2N 1N4*
[4] *Advanced Micro Devices, 1 Commerce Valley Drive East, Markham, ON, Canada L3T 7X6*

Correspondence should be addressed to Arjuna Madanayake; arjuna@uakron.edu

Received 28 November 2012; Revised 23 February 2013; Accepted 25 February 2013

Academic Editor: Antonio G. M. Strollo

Transformation and quantization play a critical role in video codecs. Recently proposed algebraic-integer-(AI-) based discrete cosine transform (DCT) algorithms are analyzed in the presence of quantization, using the High Efficiency Video Coding (HEVC) standard. AI DCT is implemented and tested on asynchronous quasi delay-insensitive logic, using Achronix SPD60 field programmable gate array (FPGA), which leads to lower complexity, higher speed of operation, and insensitivity to process-voltage-temperature variations. Performance of AI DCT with HEVC is measured in terms of the accuracy of the transform coefficients and the overall rate-distortion (R-D) characteristics, using HM 7.1 reference software. Results indicate a 31% improvement over the integer DCT in the number of transform coefficients having error within 1%. The performance of the 65 nm asynchronous hardware in terms of speed of operation is investigated and compared with the 65 nm synchronous Xilinx FPGA. Considering word lengths of 5 and 6 bits, a speed increase of 230% and 199% is observed, respectively. These results indicate that AI DCT can be potentially utilized in HEVC for applications demanding high accuracy as well as high throughput. However, novel quantization schemes are required to allow the accuracy improvements obtained.

1. Introduction

High dynamic range (HDR) video and image transmission over digital communication channels is undergoing exponential growth [1]. With the increasing demand for high-definition programming, there exists a strong need for efficient digital video coding (DVC) that provides high data compression ratios which in turn leads to better utilization of network resources [2]. The H.264/AVC standard [3] does not provide the required compression ratios for emerging capture and display technologies such as ultra high definition (UHD) [4], multiview [5], and autostereoscopy [6]. To address such emerging needs, the Joint Collaborative Team on Video Coding (JCT-VC) has developed the successor for H.264/AVC, called High Efficiency Video Coding (HEVC) [4]. The HEVC standard aims at achieving a 50% reduction in data rate compared with its predecessors while maintaining low complexity computation. Video compression systems operating at high frequencies and resolutions require hardware capable of significant throughput with tolerable area and power requirements. Real-time video compression circuits having high numerical accuracy are needed for next-generation video [1], coding systems [2, 3, 7], and retina displays [8].

The two-dimensional (2D) 8×8 discrete cosine transform (DCT) is a fundamental operation in real-time video systems, which is adopted in compression standards, such as JPEG, MPEG-1, MPEG-2, H.261, H.263, H.264, and most recently

H.265/HEVC [3, 4, 9]. The DCT is the de facto standard in transform coding, with extensive applications in modern DVC standards, due to its superior energy compaction on par with the optimum Karhunen–Loève transforms, achieved at reasonably low computational complexity [9, 10]. The circuit realization of the 2D 8 × 8 DCT affects noise, distortion, circuit area, and power consumption of such compression systems. The 2D DCT implementation is essentially dependent on the one-dimensional (1D) DCT. The 8-point 1D DCT requires multiplications by numbers in the form $c[n] = \cos(n\pi/16)$, $n = 0, 1, \ldots, 7$. These constants impose implementation difficulties in terms of their machine representation, because they are irrational values. Fixed-point arithmetic DCT implementations usually employ rounding off to approximate such quantities, which introduces errors. Besides the numerical representation issues, error propagation, noise injection, noise coupling, and noise amplification are significant when considering fixed-point realizations.

Algebraic integer (AI) encoding [11] has been proposed for addressing this problem. Algebraic integers are defined as the roots of monic polynomials having integer coefficients. AI encoding maps irrational numbers to array of integers, which can be exactly manipulated in integer arithmetic hardware. Depending on the numbers to be encoded, AI mapping can be exact, which is the case for 8-point DCT multipliers [12]. It was previously shown in [11, 13] that AI number theoretic bases can lead to the exact computation of the 2D DCT. The resulting AI-based computation is required to be mapped back into usual fixed-point representation at a given precision using the final reconstruction step (FRS) [11], where an error is unavoidably introduced. However, this introduced error is controlled and not propagable. The AI-based DCT algorithms utilize a number of theoretical techniques, such as multiencoding [14], Booth encoding [15], and expansion factor technique [16]. AI-based architectures for the accurate computation of DCT coefficients lead to low chip area, low critical path delay, and low dynamic power consumption.

This work is based on the 8 × 8 2D AI-based DCT digital hardware architecture in [16], which is an improvement of architectures proposed in [17, 18]. The architecture is based on the low-complexity Arai DCT algorithm [19], which formed the building block of each 1D DCT, using AI number representation. The 8-point Arai algorithm only needs 5 multiplications to generate the 8 output coefficients. The entire signal flow graph is free of quantization errors, unlike conventional fixed-point implementations. Error propagation is eliminated throughout intermediate computation, resulting in zero error correlation among the final DCT coefficients. Errors in the architecture are confined to the FRS, which as mentioned before maps the resulting doubly AI-encoded DCT coefficients into fixed-point representations [11]. This allows the selection of individual levels of precision, for each of the 64 DCT coefficients, which makes the architecture capable of arbitrarily high accuracy. The architecture is multiplier-free because it operates over the error-free AI-encoded number representation.

We study the quantization effects of AI-based DCT architecture [16] in the domain of DVC, using the HEVC standard. The HEVC employs a scaled DCT approximation proposed by Chen et al. [20], for real-time DCT implementation, in order to reduce complexity. By replacing Chen's fast DCT algorithm in the HEVC standard with the AI-based Arai DCT, the potential improvements to video image quality in terms of signal-to-noise ratio (SNR) are investigated. Simulations with HM 7.1 reference software [21] are used for this purpose. Unlike its predecessor H.264/AVC, which only supported block sizes of 4 × 4 and 8 × 8, HEVC supports DCT transforms of block sizes of 16 × 16, 32 × 32, and 64 × 64. The higher-order DCTs are utilized to improve energy compaction, and to reduce quantization energy when compressive images contain large homogeneous areas. This study is limited to block sizes 4 × 4 and 8 × 8. The 4 × 4 DCT was implemented using the 8 × 8 DCT, by zero padding the input. All statistical analysis was conducted using standard video sequences available in online databases [22].

In addition to the evaluation of the quantization noise performance, the applicability to high-throughput applications is investigated, by implementation on high-throughput asynchronous quasi delay-insensitive (QDI) logic [23]. The architecture is suitable for asynchronous QDI logic, because its feedforward structure can be pipelined, using the asynchronous QDI methods, such that the maximum throughput and speed of operation are achieved. In addition to higher throughput, asynchronous QDI logic has the advantages of (i) robustness to changes in circuit delays due to process-voltage-temperature (PVT) variations and (ii) simplified application-specific integrated circuit (ASIC) design effort in clock tree routing, due to absence of a global clock [23]. We use a 65 nm CMOS field programmable gate array (FPGA) device SPD60, from Achronix Semiconductor Corporation [24], for asynchronous QDI logic realization. Performance comparison in terms of speed of operation is conducted using implementations on conventional 65 nm synchronous Xilinx Virtex-5 FPGA devices.

This paper unfolds as follows. In the next section the existing transform as well as the novel AI-based DCT transform is reviewed. The methodology used for investigating the performance of the novel AI-based DCT in the HEVC standard and the results obtained are given in Section 3. The asynchronous QDI logic is introduced in Section 4, followed by the results obtained by hardware implementation in Section 5.

2. Review and Background

2.1. HEVC Integer Cosine Transform. The integer cosine transform (ICT) required in the HEVC specification is a scaled approximation of Chen's fast algorithm for computing the 1D DCT [20]. Since the computational architecture is based on a regular butterfly structure, the ICT has lower circuit complexity at the physical layer. However, the ICT algorithm is not optimal in terms of speed or accuracy [4, 20]. Despite such suboptimality, the ICT algorithm has the advantage of being extensible to larger transform sizes such as 32 × 32 and 64 × 64, leading to its adoption in HEVC. An overview of the Chen's fast DCT algorithm is given

Table 1: Multiplicands required in the 8-point Chen's fast DCT algorithm.

Exact value	$\cos(\pi/32)$	$\cos(\pi/8)$	$\cos(3\pi/16)$	$\cos(\pi/4)$	$\cos(5\pi/16)$	$\cos(3\pi/4)$	$\cos(3\pi/8)$
Approximation (scaled by $\sqrt{2}$)	89/64	83/64	75/64	64/64	50/64	36/64	18/64

in Section 2.2 to help the reader understand its difference compared with AI-based DCT architecture.

2.2. 8-Point Chen's Fast DCT Algorithm. The fast algorithm proposed by Chen et al. [20] for computing the 8-point DCT requires 26 additions and 12 multiplications. The seven *different multiplicands* employed in this algorithm are given in Table 1. The *exact* DCT coefficients can be obtained by scaling the output obtained from the butterfly structure given in [20] by 4. In the suggested reference C++ software implementation [20], Chen's algorithm was implemented in 16-bit arithmetic, using *short* variables to represent intermediate results. Suitable approximations for the irrational constants in the DCT are scaled by a factor of $\sqrt{2}$, leading to Table 1. In order to compute the 2D DCT, the 1D transform is applied twice: first in a row-wise manner, then column-wisely, with a transposition operation in between. Hence, the errors that are introduced in the rounding operation after the row and column transform calculations propagate through the 2D DCT computation and are present as additive noise at the final quantizer stage. The additive noise injected within the 2D transformation algorithm can effect a visible impact on picture fidelity especially for low levels of compression.

2.3. Review of 2D AI DCT

2.3.1. AI Encoding and Decoding. Encoding real numbers to AI representation involves a mapping function $f_{\text{enc}}(x;\mathbf{z}) = \mathbf{a}$, where x is a real number, $\mathbf{z}$ is a fixed array of algebraic integers, and $\mathbf{a}$ is an array of integers, which represents x in the AI basis. Any real number can be represented in the AI basis with arbitrary precision, and there are some real numbers that can be represented *without* error [25].

For the Arai DCT algorithm [19], the required multiplication constants are [17, 18] $c[4] = \cos(4\pi/16)$, $c[6] = \cos(6\pi/16)$, $c[2] - c[6] = \cos(2\pi/16) - \cos(6\pi/16)$, $c[2] + c[6] = \cos(2\pi/16) + \cos(6\pi/16)$. These values can be encoded according to the following 2D AI basis array [16]:

$$\mathbf{z} = \begin{bmatrix} 1 & z_1 \\ z_2 & z_1 z_2 \end{bmatrix}, \tag{1}$$

where $z_1 = \sqrt{2+\sqrt{2}} + \sqrt{2-\sqrt{2}}$ and $z_2 = \sqrt{2+\sqrt{2}} - \sqrt{2-\sqrt{2}}$. Encoding real numbers from the given AI basis gives a representation of the form [16]

$$f_{\text{enc}}(x;\mathbf{z}) = \begin{bmatrix} x^{(a)} & x^{(b)} \\ x^{(c)} & x^{(d)} \end{bmatrix}, \tag{2}$$

where $x^{(a)}$, $x^{(b)}$, $x^{(c)}$, and $x^{(d)}$ are the encoded integer coefficients. Error-free and sparse representation of $c[4]$, $c[6]$, $c[2]-c[6]$, and $c[2]+c[6]$ using the defined AI basis are given in Table 2 [16–18]. This representation employs small integers, which are suitable for fast arithmetic circuits.

Table 2: 2D AI encoding of Arai DCT constants.

$c[4]$	$c[6]$	$c[2]-c[6]$	$c[2]+c[6]$
$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix}$

In the proposed architecture, AI encoding requires no operation, since input data are assumed to be integer. In fact, an integer n satisfies the following rule $f_{\text{enc}}(n;\mathbf{z}) = \left[\begin{smallmatrix} n & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Decoding operation is done in the FRS block, which implements the following operation: $f_{\text{dec}}(\mathbf{a};\mathbf{z}) = x^{(a)} + x^{(b)} z_1 + x^{(c)} z_2 + x^{(d)} z_1 z_2$.

2.3.2. Arai AI DCT Architecture. The 2D DCT is expressed by $(\mathbf{C}\cdot(\mathbf{C}\cdot\mathbf{A})^T)^T$ for an 8×8 input $\mathbf{A}$ [26], where $\mathbf{C}$ is the usual DCT matrix [10]. This shows that the 2D DCT consists of (i) the column-wise application of the 1D DCT to the input $\mathbf{A}$, (ii) a transposition operation, and then (iii) the row-wise application of the 1D DCT.

Here the basic building block of the 2D AI DCT architecture is taken as the 1D AI DCT block [16, 18] with 8 inputs and 22 AI integer outputs, as shown in Figure 1. The block diagram of the architecture which consists of five subcircuits [16] is shown in Figure 2. The five subcircuits are reviewed as (i) an input decimator circuit, (ii) an 8-point AI-encoded 1D DCT block for column-wise computation, (iii) an AI-based transposition buffer, (iv) four parallel instantiations of the 8-point AI-encoded 1D DCT block for row-wise computation, and (v) the FRS circuit for decoding AI-encoded 2D DCT coefficients. The last transposition is obtained via wired cross-connections.

The implementation does not cover the input decimator circuit, which can be realized using the high-speed serializer/deserializer (SerDes) intellectual property cores. The reader is directed to [16], where the architecture is described in detail.

3. AI-Based Arai DCT in HEVC

Let $\mathbf{A}$ be a $4\cdot 2^p \times 4\cdot 2^q$ input data block, where $p, q \in \{0,1\}$. One may calculate the 2D DCT of $\mathbf{A}$ by zero padding it and submitting it to the 8×8 DCT algorithm. Zero padding $\mathbf{A}$ furnishes the 8×8 matrix $\mathbf{B}$ defined according to the following block diagonal matrix:

$$\mathbf{B} = \operatorname{diag}\left(\mathbf{A}, \mathbf{0}_{8-4\cdot 2^p, 8-4\cdot 2^q}\right), \tag{3}$$

where $\mathbf{0}_{m,n}$ is a zero matrix of size $m\times n$. Let us consider that an application of the 2D DCT on $\mathbf{A}$ and $\mathbf{B}$ results in $\mathbf{A}'$ and $\mathbf{B}'$, respectively. Moreover, let $a_{i,j}$ and $b_{i,j}$ be the

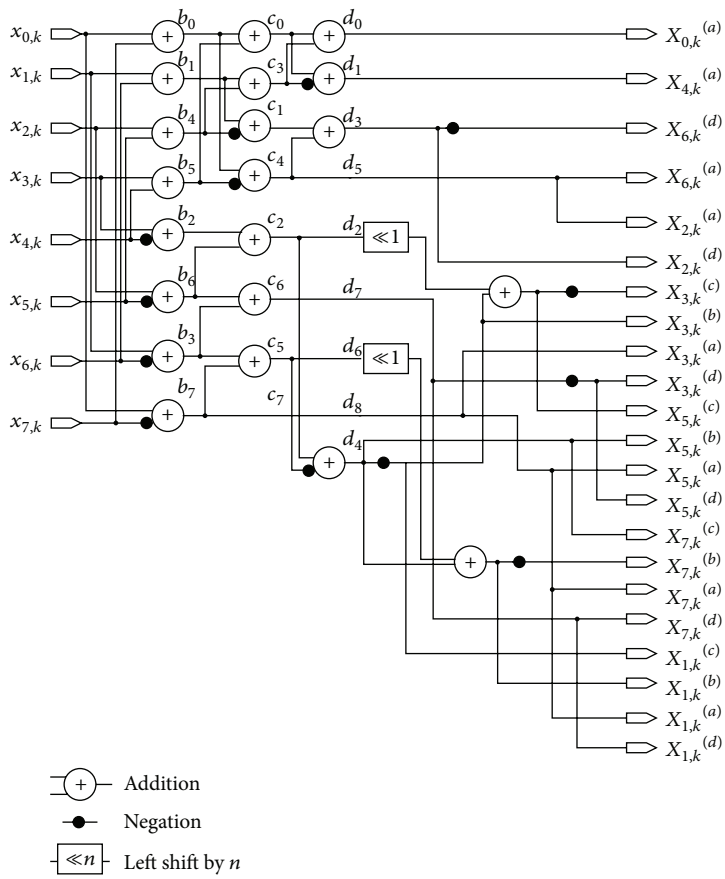

Figure 1: 1D AI Arai DCT block used in Figure 2 [16, 18].

(i, j)th element in matrices $\mathbf{A}'$ and $\mathbf{B}'$, respectively. Then the following relationship holds true:

$$a_{i,j} = K \cdot b_{2^{(2-p)} \cdot i, 2^{(2-q)} \cdot j}, \tag{4}$$

where $K = \sqrt{2^{(2-p)} \cdot 2^{(2-q)}}$. This expression can be directly obtained from the 2D DCT definition. Above relationship guarantees that the data blocks of size 4×4, 4×8, and 8×4 can be computed using the 8×8 DCT. For the 4×4 block size, we have $K = 2$; for the remaining cases $K = \sqrt{2}$.

3.1. Evaluation Setup. Using sample video sequences, the performance of the proposed AI DCT algorithm was compared with the ICT scheme [4]. Firstly, the accuracy of the existing transform unit was analyzed, followed by the effect of the new AI transform. Analysis was performed for the overall encoding process, by obtaining rate-distortion (R-D) curves associated with the standard test sequences. The configuration used for this evaluation, including the quadtree picture-partitioning settings, is given in Table 3. The configuration omits the use of larger-sized transforms and restricts the transform operation to 8×8 and 4×4 transforms. Nonsquare quadtree transform (NSQT) coding tool was also disabled to enable the aforementioned criterion [4, 27].

3.2. Accuracy of the Transform Block. The errors in the calculated transform coefficients, with respect to coefficients calculated using floating point MATLAB `dct2` function, were obtained both for the original implementation and the proposed implementation. Success rates were obtained as the percentage of coefficients having an error ratio less than a selected threshold value. The results are displayed in Figure 3. Notice that for errors within 1%, the proposed algorithm effects a 31% improvement when compared with the scheme described in [4, 20] .

3.3. Performance of the Overall Encoding Process. Effect of the proposed transform on the overall performance of the encoding process was analyzed by obtaining R-D curves for standard video sequences. The curves were obtained by varying the quantization point (QP) from 0 to 8 and obtaining the peak signal-to-noise ratio (PSNR) of the reconstructed sequence, with reference to the original sequence, along with the bits/frame of the encoded video. Figures 5, 6, and 7 depict the obtained R-D curves for the `Basketballpass`

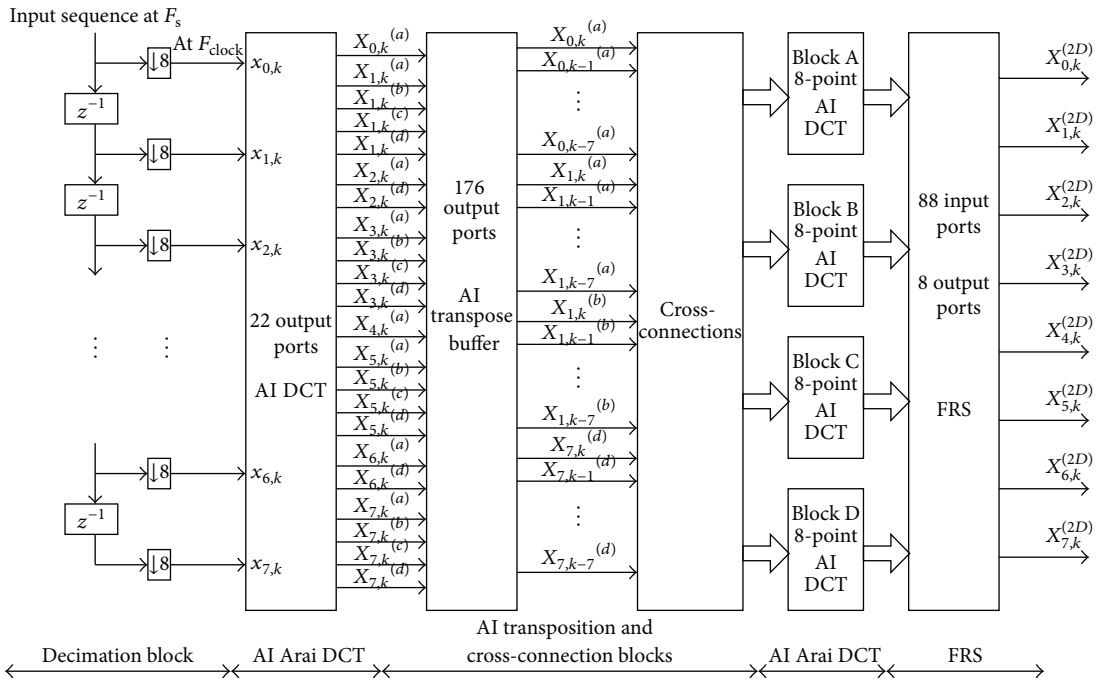

Figure 2: The 2D AI-DCT consists of an input section having a decimation structure, 1D 8-point AI-DCT block for column-wise DCTs, a real-time AI transpose buffer [16], four parallel 1D 8-point AI-DCT blocks for row-wise DCTs, and the FRS [16].

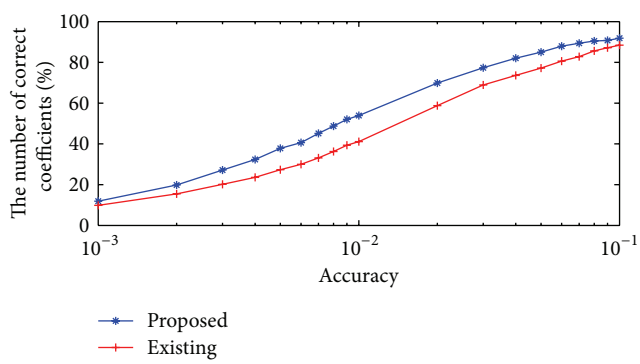

Figure 3: Accuracy comparison between the standard integer DCT and proposed AI-based Arai DCT.

(416 × 240), `BlowingBubbles` (416 × 240), and `BQMall` (832 × 480) using group of pictures (GOP) IPPP... sequence using intracoding. Figure 4 provides a comparison between the original video, encoded video using existing algorithm, and encoded video with proposed algorithm using a frame obtained using the test video sequence Basketballpass at QP = 32.

The R-D curves (Figures 5–7) indicate that the difference in the rate points of the current and proposed algorithm is almost negligible. This is due to the accuracy degradation in the quantization process. The quantization noise injected at the quantizer dominates the total error diminishing the accuracy gains obtained by the superior transform block. This clearly shows that in order for high-performance HEVC video systems to benefit from the improved accuracy of AI-based architectures, the quantizer block of the codec must be modified or improved from its default design.

4. Asynchronous QDI FPGA Implementation

Achronix Semiconductor FPGAs utilize asynchronous QDI logic [23] which enables rapid asynchronous implementation of designs without requiring ASICs. The clockless nature of the asynchronous implementation leads to reduced design complexity, lower energy consumption, and higher speed of operation [23]. It is also robust since the asynchronous operation is independent of changes in delay resulting from PVT variations [23]. A synchronous I/O frame surrounds the clockless logic fabric of the device. Inputs and outputs of the design are made synchronous to a user-defined clock that determines the speed of operation.

The speed of operation of Achronix devices is maximized by means of asynchronous fine-grained pipeline stages termed "picoPIPE," which are based on a 3-wire handshake scheme, in contrast to a single-wire connection with a global clock signal in synchronous FPGA devices. The correct operation is ensured by a local handshake protocol between adjacent pipeline stages. Adding a large number of pipeline stages to the datapath of the user design leads to high-throughput architectures which have speed of operation in the best case equal to the reciprocal of handshake time delay between two pipeline stages.

However, the existence of unavoidable feedback loops and "reconvergent paths" [28] causes lower speed of operation. These two phenomena are the most frequently occurring

(a) (b) (c)

Figure 4: Frame from Basketballpass test video: (a) original video, (b) encoded video using existing HM encoder, and (c) encoded video using proposed DCT algorithm.

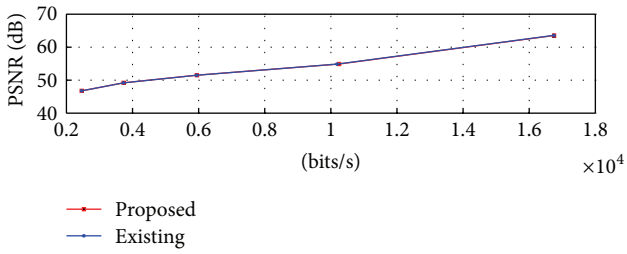

Figure 5: R-D curves for Basketballpass (416 × 240) with standard HEVC quantizer block.

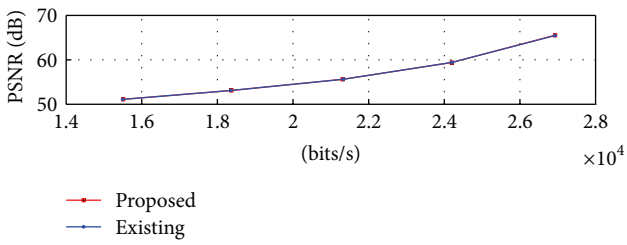

Figure 6: R-D curves for BlowingBubbles (416 × 240) with standard HEVC quantizer block.

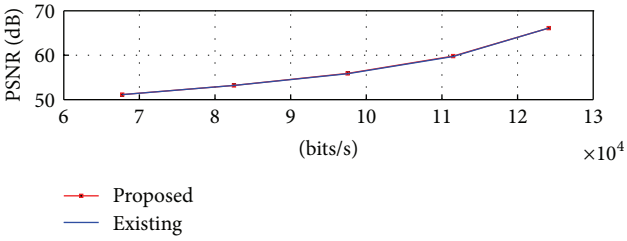

Figure 7: R-D curves for BQMall (832 × 480) with standard HEVC quantizer block.

types of critical paths in Achronix FPGAs, which are very different from causes of critical paths of synchronous FPGAs. The critical paths of synchronous FPGAs are determined by delayed paths in the circuit constrained by user-defined specifications such as clocks, input, and output delays. A netlist that contains "reconvergent paths"—nodes where a path with fewer pipeline stages (shorter path) reconverges with a path with more pipeline stages (longer path)—leads to slow speed of operation for asynchronous logic. In that case, data in the shorter path arrive earlier to the node and have to wait for data in the longer path to propagate forward. The performance limitation for a critical reconvergent path is the forward delay of data in the longer path and backward delay of the acknowledge signal in the shorter path [28]. Achronix tools documentation [28] details methods for eliminating reconvergent paths. In the next section, we describe and employ some of them. The AI DCT architecture is most suitable for implementation on this device because reconvergent paths form the only speed limitation. The Achronix SPD60 FPGA device based on asynchronous QDI logic [23] was employed for implementation and testing of the architecture on-chip.

Table 3: Configuration settings.

Parameter	Value
Software version	HM-7.1rc2
MaxCUWidth	64
MaxCUHeight	64
MaxPartitionDepth	4
QuadtreeTULog2MaxSize	3
QuadtreeTULog2MinSize	2
QuadtreeTUMaxDepthInter	2
QuadtreeTUMaxDepthIntra	2

5. FPGA Implementation and Test

The architecture described was implemented using MATLAB/Simulink and Xilinx System Generator (XSG) blocks. The FRS was designed using VHDL and imported to Simulink implementation through the XSG tools. The resulting design was tested using MATLAB/Simulink for correct operation and could be used for direct implementation in Xilinx devices.

We generated behavioral hardware description language (HDL) using the XSG tool. VHDL code was synthesized for asynchronous QDI logic by submitting it to the tool flow for implementation in Achronix SPD60 FPGAs. The design was downloaded to the SDP60 device after bit-file generation, and

Table 4: Resource utilization, speed of operations, and power consumption of the DCT design on synchronous FPGA and asynchronous FPGA implementations for various input fixed-point bus widths.

Fixed-point width	Xilinx Virtex-5 xc5vlx330 (synchronous)							Achronix SPD60 (asynchronous)				% Increase in speed
	Slices	Slice FFs	Slice LUTs	Freq. (MHz)	Quies. power (W)	Dyn. power (W)	Total power (W)	RLBs	SEQ-DFFs	LUT4s	Freq. (MHz)	
3	3262 (6%)	5310	9684	113.49	2.952	1.358	4.310	4531 (77.06%)	31352	36248	349.6	208.04
4	3684 (7%)	5811	10317	110.63	2.952	1.409	4.361	4852 (82.52%)	33767	38818	335.8	203.53
5	3786 (7%)	6082	10987	109.12	2.952	1.392	4.344	5174 (88%)	36193	41393	330.6	230.6
6	4032 (7%)	6516	11592	111.74	2.952	1.420	4.372	5496 (93.47%)	38609	43968	334.2	199.09
7	4080 (7%)	6925	12312	111.84	2.952	1.447	4.399	5772 (98.18%)	40974	46184	302.7	170.65
8	4341 (8%)	7407	12943	107.48	2.952	1.492	4.444	—	—	—	—	—

tested successfully on chip using the Achronix SPC60 plug-in card, using a serial connection established via an HDL wrapper.

Subsequently, the design was submitted to the FPGA tool flows of both Achronix and Xilinx to investigate their maximum speed of operation, power consumption, and FPGA resource utilization. Note that the HDL of the architecture without the wrapper for serial communication is used in the synthesis engine. Resource utilization, maximum speed of operation, and power consumption results of the compilations are given in Table 4 for different fixed-point input widths L in the hardware design. The architecture having $L = 8$ was the most desirable design with the acceptable levels of tolerances. However, architectures with smaller values of L were also implemented and tested on chip for completeness. For comparison with the asynchronous implementation in terms of speed of operation and resource utilization, the design was compiled to a larger Xilinx synchronous FPGA device—Virtex-5 xc5vlx330—with the same technology node as the Achronix device. Designs were not physically implemented and tested on the Xilinx device but only sent through the Xilinx tool flow in order to obtain the synthesis results for purposes of comparison. The speed of operation results were obtained from the static timing analysis of the compiled designs, considering the default options of the Xilinx tools and the same clock constraint of 125 MHz.

The critical paths for every Achronix compilation reported were also reconvergent paths. Such reconvergent paths can be eliminated in order to further increase the speed of operation as described in Achronix documentation [28]. To eliminate a reconvergent critical path, delay was adjusted by the introduction of extra delay elements to the path with lower number of pipeline stages. The Achronix ACE design tools facilitate these operations either by the addition of a constraint to the place and route tool or by changing HDL description by instantiating a macro "`ACX_SLACKMATCH`" at the shorter path. The place-and-route tool for Achronix does delay balancing automatically up to a certain extent. However, larger speeds require changing the input HDL and the constraints as described above and a recompilation. Notice that the speed of operation results given in Table 4 was obtained without optimization. Nevertheless, the asynchronous designs show faster (by more than 230%) operation speeds compared to the synchronous counterpart.

The resource utilization for various designs on Achronix FPGA is given in Table 4 for completeness, except for the 8-bit architecture, which could not be accommodated in the SPD60 device. The FPGA resource utilization for Achronix device is given in terms of reconfigurable logic blocks (RLBs), sequential delay flip-flops SEQ-DFFs, and 4-input look-up tables (LUT4s). Xilinx FPGA resource utilization is given in terms of slices, slice flip-flops (FFs), and slice look-up tables (LUTs). Although there is no direct comparison to asynchronous FPGA designs because of the vastly different logic fabrics, we provide corresponding estimates for 65 nm Achronix SDP60 realizations at the same range of L values. The configurable logic blocks (CLBs) in Xilinx Virtex-5 devices are equivalent to RLBs in Achronix FPGAs. Further, Xilinx LUTs can be substantially compared with Achronix 4-input LUTs. The increase in CLB, RLB, FF, and LUT consumptions, be it synchronous Xilinx or asynchronous Achronix FPGAs, both at 65 nm CMOS technology node, varied as shown in Table 4. The Achronix device was capable of operating significantly faster than the Xilinx FPGA device.

Both estimated dynamic and quiescent power consumptions for the 65 nm Xilinx Virtex-5 xc5vlx330 device are provided in Table 4. However, the power consumption estimation of the Achronix device was not possible with the current version of the Achronix design automation tools.

6. Conclusion

The performance of a recently proposed AI-based DCT architecture in the presence of quantization for HEVC was investigated along with its applicability to high-throughput applications using asynchronous QDI implementation.

Software implementation of the proposed algorithm was integrated into HM 7.1 and its performance in terms of accuracy was compared with the currently employed 8×8 DCT transform based on Chen's fast DCT algorithm. Results indicate significant improvement in the accuracy of transform coefficients, specifically a 31% improvement is

observed for an error percentage within 1%. R-D curves obtained for quantization values of $0 \leq \text{QP} \leq 32$ indicate that the overall improvement in terms of R-D performance is negligible. These results suggest the potential usage of AI-based DCT algorithms in applications that require high accuracy typically in high resolution front ends. Moreover, they also indicate the necessity of advanced transform techniques that are capable of utilizing the gains obtained through improved transform techniques.

Achronix FPGA devices are used to implement the hardware architecture on asynchronous QDI logic without using the time-consuming and expensive flow of ASIC implementation. Comparison results in terms of speed of operation through static timing analysis between the synchronous and asynchronous FPGA show that the asynchronous FPGA could outperform the Xilinx FPGA device by showing a maximum of 230% increase in operation speed.

As a future work, the effect of novel quantization techniques such as frequency-dependent quantization [29] on the performance should be investigated. Also AI-based algorithms for large block size transforms and their inverse transformations should be studied. Work should also be done on integrating the Arai scaling factors into the quantizer.

Acknowledgments

This work was supported by The University of Akron, OH, USA; Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and FACEPE, Brazil; and NSERC, Canada.

References

[1] B. R. Lim, R. H. Park, and S. Kim, "High dynamic range for contrast enhancement," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1454–1462, 2006.

[2] D. R. Bull, E. J. Delp, S. Takamura, T. Wiegand, and F. Wu, "Introduction to the issue on emerging technologies for video compression," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, pp. 1277–1281, 2011.

[3] H. Kalva, "The H.264 video coding standard," *IEEE Multimedia*, vol. 13, no. 4, pp. 86–90, 2006.

[4] M. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: the new gold standard for video compression: how does HEVC compare with H.264/AVC?" *IEEE Consumer Electronics Magazine*, vol. 1, pp. 36–46, 2012.

[5] A. Vetro, S. Yea, M. Zwicker, W. Matusik, and H. Pfister, "Overview of multiview video coding and anti-aliasing for 3D displays," in *Proceedings of the 14th IEEE International Conference on Image Processing (ICIP '07)*, vol. 1, pp. 17–20, September 2007.

[6] A. M. Tekalp, A. Smolic, A. Vetro, and L. Onural, "Special issue on 3-D media and displays," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 536–539, 2011.

[7] B. Bennett, C. Dee, M. H. Nguyen, and B. A. Hamilton, "Operational concepts of MPEG-4 H.264 for tactical DOD applications," in *Proceedings of the IEEE Military Communications Conference (MILCOM '05)*, vol. 1, pp. 155–161, October 2005.

[8] B. Jones, "Apple retina display," 2013, http://prometheus.med.utah.edu/~bwjones/2010/06/apple-retina-display/.

[9] S. A. Khayam, *The Discrete Cosine Transform (DCT): Theory and Application*, Michigan State University, 2003.

[10] V. Britanak, P. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms*, Academic Press, New York, NY, USA, 2007.

[11] V. S. Dimitrov, G. A. Jullien, and W. C. Miller, "New DCT algorithm based on encoding algebraic integers," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '98)*, vol. 3, pp. 1377–1380, May 1998.

[12] R. Baghaie and V. Dimitrov, "Systolic implementation of real-valued discrete transforms via algebraic integer quantization," *Computers and Mathematics with Applications*, vol. 41, no. 10-11, pp. 1403–1416, 2001.

[13] I. Amer, W. Badawy, V. Dimitrov, and G. Jullien, "On the refinement of the DCT/IDCT scaling factor sensitivity," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '08)*, pp. 337–340, June 2008.

[14] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, London, UK, 4th edition, 1975.

[15] K. Y. Khoo, Z. Yu, and A. N. Willson, "Improved-booth encoding for low-power multipliers," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 1, p. 62, June 1999.

[16] A. Madanayake, R. J. Cintra, D. Onen et al., "A row-parallel 8×8 2-D DCT architecture using algebraic integer based exact computation," *IEEE Transactions on Circuits and Systems For Video Technology*, vol. 22, pp. 915–929, 2012.

[17] V. Dimitrov and K. Wahid, "On the error-free computation of fast cosine transform," *International Journal: Information Theories and Applications*, vol. 12, no. 4, pp. 321–327, 2005.

[18] V. Dimitrov, K. Wahid, and G. Jullien, "Multiplication-free 8×8 2D DCT architecture using algebraic integer encoding," *Electronics Letters*, vol. 40, no. 20, pp. 1310–1311, 2004.

[19] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *IEICE Transactions*, vol. E71-E, pp. 1095–1097, 1988.

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

[21] "SVN repository of the hm 7.1 reference software," 2013, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-7.2rc1/.

[22] "Standard videosequences for video compression testing," 2013, ftp://hvc:US88Hula@ftp.tnt.uni-hannover.de/testsequences.

[23] J. Teifel and R. Manohar, "An asynchronous dataflow FPGA architecture," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1376–1392, 2004.

[24] "Achronix semiconductor corporation," 2013, http://www.achronix.com/.

[25] J. H. Cozzens and L. A. Finkelstein, "Range and error analysis for a fast Fourier transform computed over $Z[\omega]$," *IEEE Transactions on Information Theory*, vol. 33, pp. 582–590, 1987.

[26] T. Suzuki and M. Ikehara, "Integer DCT based on direct-lifting of DCT-IDCT for lossless-to-lossy image coding," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2958–2965, 2010.

[27] "HEVC reference software manual," http://phenix.int-evry.fr/jct/doc_end_user/documents/5_Geneva/wg11/JCTVC-E447-v1.zip.

[28] "Achronix CAD Environment User Guide," v2.3.0, October 2009.

[29] I. Richardson, *The H. 264 Advanced Video Compression Standard*, John Wiley & Sons, New York, NY, USA, 2011.