LOW ENERGY VIDEO PROCESSING AND COMPRESSION HARDWARE DESIGNS

by

Ercan Kalalı

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Sabancı University

August 2018

LOW ENERGY VIDEO PROCESSING AND COMPRESSION HARDWARE DESIGNS

APPROVED BY:

Assoc. Prof. İlker Hamzaoğlu .....
(Thesis Supervisor)

Assoc. Prof. Müjdat Çetin .....

Assoc. Prof. Hüsnü Yenigün .....

Prof. Müştak Erhan Yalçın .....

Asst. Prof. Ramazan Yeniçeri .....

DATE OF APPROVAL: 30.07.2018

To my Mother and Father

To my beloved wife Ayşegül

ACKNOWLEDGEMENT

I would like to thank my supervisor, Dr. İlker Hamzaoğlu for all his guidance, support, and patience throughout my PhD study. I appreciate very much for his suggestions, detailed reviews, invaluable advices and life lessons. I particularly want to thank him for his confidence and belief in me during my study. It has been a great honor for me to work under his guidance.

I would like to thank to all members of System-on-Chip Design and Testing Lab; Hasan Azgın and Ahmet Can Mert for their great friendship and their collaboration during my studies.

Special thanks to my family and my love Ayşegül. This thesis is dedicated with love to them for their constant support and encouragement for going through my tough periods with me.

Finally, I would like to thank Sabancı University and Scientific and Technological Research Council of Turkey (TUBITAK) for supporting me throughout my graduate education.

ABSTRACT

LOW ENERGY VIDEO PROCESSING AND COMPRESSION HARDWARE DESIGNS

Ercan Kalalı
Electronics, PhD Dissertation, 2018

Thesis Supervisor: Assoc. Prof. İlker HAMZAOĞLU

Keywords: Median Filter, Gaussian Blur, Image Sharpening, HEVC, Intra Prediction, Fractional Interpolation, DCT, IDCT, Approximate Computing, Hardware Implementation, FPGA, Low Energy

Digital video processing and compression algorithms are used in many commercial products such as mobile devices, unmanned aerial vehicles, and autonomous cars. Increasing resolution of videos used in these commercial products increased computational complexities of digital video processing and compression algorithms. Therefore, it is necessary to reduce computational complexities of digital video processing and compression algorithms, and energy consumptions of digital video processing and compression hardware without reducing visual quality.

In this thesis, we propose a novel adaptive 2D digital image processing algorithm for 2D median filter, Gaussian blur and image sharpening. We designed low energy 2D median filter, Gaussian blur and image sharpening hardware using the proposed algorithm. We propose approximate HEVC intra prediction and HEVC fractional interpolation algorithms. We designed low energy approximate HEVC intra prediction and HEVC fractional interpolation hardware. We also propose several HEVC fractional interpolation hardware architectures. We propose novel computational complexity and energy reduction techniques for HEVC DCT and inverse DCT/DST. We designed high performance and low energy hardware for HEVC DCT and inverse DCT/DST including the proposed techniques.

We quantified computation reductions achieved and video quality loss caused by the proposed algorithms and techniques. We implemented the proposed hardware architectures in Verilog HDL. We mapped the Verilog RTL codes to Xilinx Virtex 6 and Xilinx ZYNQ FPGAs, and estimated their power consumptions using Xilinx XPower Analyzer tool. The proposed algorithms and techniques significantly reduced the power and energy consumptions of these FPGA implementations in some cases with no PSNR loss and in some cases with very small PSNR loss.

ÖZET

DÜŞÜK ENERJİLİ GÖRÜNTÜ İŞLEME VE SIKIŞTIRMA DONANIM TASARIMLARI

Ercan Kalalı
Elektronik Müh., Doktora Tezi, 2018

Tez Danışmanı: Doç. Dr. İlker HAMZAOĞLU

Anahtar Kelimeler: Orta Değer Filtresi, Gauss Bulanıklığı, Görüntü Keskinleştirme, HEVC, Çerçeve İçi Öngörü, Kesirli Aradeğerleme, Ayrık Kosinüs Dönüşümü, Ters Ayrık Kosinüs Dönüşümü, Yaklaşık Hesaplama, Donanım Gerçekleme, FPGA, Düşük Enerji

Sayısal video işleme ve sıkıştırma algoritmaları mobil cihazlar, insansız hava araçları ve otonom araçlar gibi birçok ticari üründe kullanılmaktadır. Bu ticari ürünlerde kullanılan video çözünürlüklerinin artması sayısal video işleme ve sıkıştırma algoritmalarının hesaplama karmaşıklığını arttırmaktadır. Bu yüzden, sayısal video işleme ve sıkıştırma algoritmalarının hesaplama karmaşıklığını ve sayısal video işleme ve sıkıştırma donanımlarının enerji tüketimlerini görsel kaliteyi düşürmeden azaltmak gerekmektedir.

Bu tezde, 2B orta değer filtresi, Gauss bulanıklığı ve görüntü keskinleştirme algoritmaları için yeniden uyarlanabilir 2B sayısal görüntü işleme algoritması önerilmektedir. Önerilen algoritmayı kullanarak düşük enerjili 2B orta değer filtresi, Gauss bulanıklığı ve görüntü keskinleştirme donanımları tasarlanmıştır. Yaklaşık HEVC çerçeve içi öngörü ve yaklaşık HEVC kesirli aradeğerleme algoritmaları önerilmektedir. Düşük enerjili yaklaşık HEVC çerçeve içi öngörü ve yaklaşık HEVC kesirli aradeğerleme donanımları tasarlanmıştır. Ayrıca, HEVC kesirli aradeğerleme algoritması için farklı donanım mimarileri önerilmektedir. HEVC DCT ve ters

DCT/DST için birkaç farklı hesaplama karmaşıklığı ve enerji azaltma teknikleri önerilmektedir. Önerilen teknikleri kullanarak, yüksek performanslı ve düşük enerjili HEVC DCT ve ters DCT/DST donanımları tasarlanmıştır.

Önerilen algoritma ve tekniklerin neden olduğu hesaplama azaltmaları ve video kalitesi kayıpları ölçüldü. Önerilen donanım mimarileri Verilog donanım tasarlama dili ile gerçeklendi. Verilog RTL kodları Xilinx Virtex 6 ve Xilinx ZYNQ FPGA'lerine sentezlendi ve bunların güç tüketimleri Xilinx XPower Analyzer aracı ile tahmin edildi. Önerilen algoritmalar ve teknikler, bu FPGA gerçeklemelerinin güç ve enerji tüketimlerini, bazı durumlarda PSNR kaybı olmaksızın, bazı durumlarda ise çok küçük PSNR kaybı ile önemli ölçüde azalttı.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AXI	Advanced eXtensible Interface
BRAM	Block RAM
CABAC	Context Adaptive Binary Arithmetic Coding
CU	Coding Unit
DBF	Deblocking Filter
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
FHD	Full High Definition
FPGA	Field Programmable Gate Array
HD	High Definition
HDMI	High Definition Multimedia Interface
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
IDCT	Inverse Discrete Cosine Transform
PSNR	Peak Signal to Noise Ratio
PU	Prediction Unit
QFHD	Quad Full High Definition
QP	Quantization Parameter
SAO	Sample Adaptive Offset
TU	Transform Unit
VCD	Value Change Dump

CHAPTER I

INTRODUCTION

Digital video processing and compression algorithms and hardware are used in many commercial products such as mobile devices, unmanned aerial vehicles, and autonomous cars [1]-[4]. To improve visual quality and compression efficiency, video sizes and computational complexities of digital video processing and compression algorithms are increased. For example, Quad Full HD (4K) and Ultra HD (8K) video resolutions started to be used instead of Full HD (2K) video resolution. This increases the energy consumptions of hardware implementations of these algorithms. This trend is expected to continue in the future as well. According to Cisco Visual Networking Index internet video traffic will be 82% of all consumer internet traffic by 2021 [5]. Also, 63% of video IP traffic will be consumed by mobile devices by 2021 [5]. Because of these developments, video coding algorithms with high coding efficiency should be designed. Therefore, Joint Collaborative Team on Video Coding (JCT-VC) recently developed a new video compression standard called High Efficiency Video Coding (HEVC) [6]-[8]. HEVC provides 50% better coding efficiency than H.264 video compression standard. HEVC uses larger block sizes, more prediction modes and more transform types than H.264 to obtain better coding efficiency. Therefore, HEVC has higher computational complexity than H.264.

1.1 HEVC Video Compression Standard

The video compression efficiency achieved by HEVC standard is result of a combination of several encoding and decoding tools such as intra prediction, motion estimation, deblocking filter, sample adaptive offset (SAO) and entropy coder. The top-level block diagrams of an HEVC encoder and decoder are shown in Figure 1.1 and Figure 1.2.

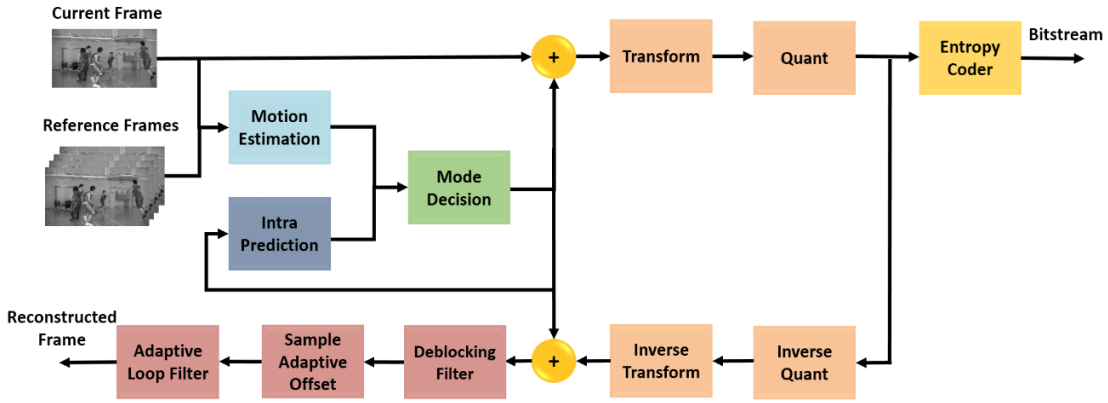

Figure 1.1 HEVC Encoder Block Diagram

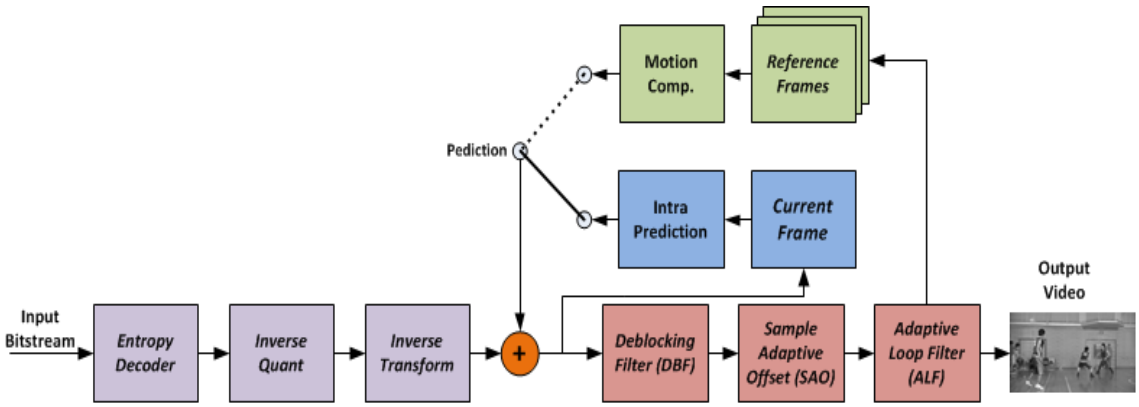

Figure 1.2 HEVC Decoder Block Diagram

As shown in Figure 1.1, an HEVC encoder has a forward (coding) path and a reconstruction (decoding) path. The forward path is used to encode a video frame by using spatial (intra) and temporal (inter) prediction modes. Then, residual data are coded after the transform and quantization processes, and bitstream is created. Since HEVC decoder does not have access to original frames, reconstruction path in the encoder is used to prevent a mismatch between encoder and decoder. In this way, both encoder and decoder use identical reference frames for intra and inter prediction.

HEVC uses quad-tree block structure as shown in Figure 1.3. Therefore, each frame is divided into coding units (CU) in the forward path. These CUs can be 8x8, 16x16, 32x32 or 64x64 pixel blocks. CUs in I frames are encoded with only intra prediction modes. CUs in P and B frames are encoded with intra or inter mode depending on the mode decision. Intra and inter prediction modes use the prediction unit (PU) partitioning structure inside the CUs. Each PU size can be equal to or less than CU size. PU sizes can be 4x4, 8x8, 16x16 and 32x32 for intra prediction modes. However, inter prediction has 24 different PU sizes (4x8, 8x4, 8x8 etc.). After the prediction, mode decision determines whether the PU will be coded with intra or inter prediction based on PSNR and bit-rate. Then, prediction is subtracted from original video data and residual data is generated. Then, transformation and quantization are performed on the residual data. Transform units (TU) are used in the integer discrete cosine transform (DCT), and TU sizes can be from 4x4 up to 32x32. 4x4 TU size is only used for discrete sine transform (DST). Finally, entropy coder (context adaptive binary arithmetic coding) generates the encoded bitstream.

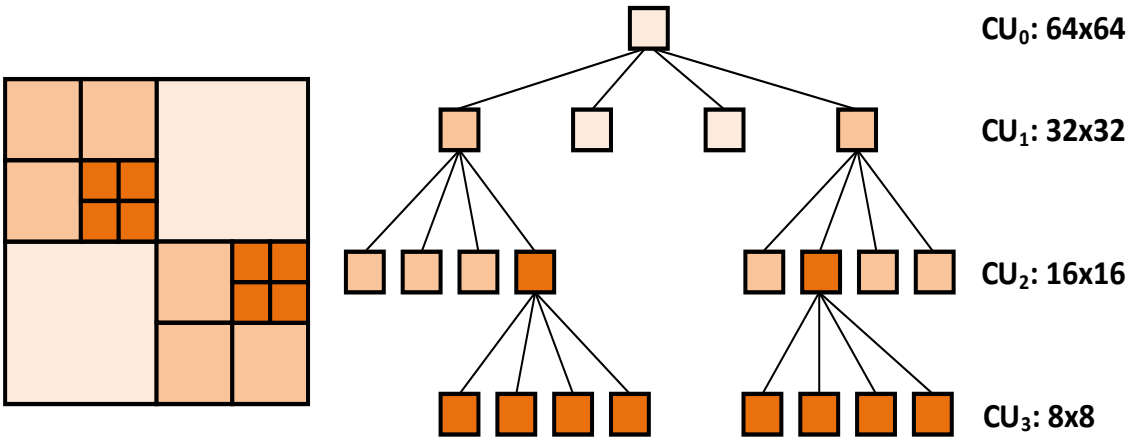

Figure 1.3 HEVC Quadtree Block Structure

Reconstruction path begins with inverse quantization and inverse transform. The quantized transform coefficients are inverse quantized and inverse transformed to generate the reconstructed residual data. Since quantization is a lossy process, inverse quantized and inverse transformed coefficients are not identical to the original residual data. The reconstructed residual data are added to the predicted pixels to create the reconstructed frame. DBF is, then, applied to reduce the effects of blocking artifacts in the reconstructed frame.

Intra prediction algorithm in HEVC predicts the pixels of a block from the pixels of its already coded and reconstructed neighboring blocks. In H.264, there are 9 intra prediction modes for 4x4 luminance blocks, and 4 intra prediction modes for 16x16 luminance blocks [9]. In HEVC, for the luminance component of a frame, intra prediction unit (PU) sizes can be from 4x4 up to 32x32 and number of intra prediction modes for a PU can be up to 35 [6, 7]. 33 of these 35 prediction modes are intra angular prediction modes, and the predicted pixels are generated by weighted average of two neighboring pixels. In addition to angular prediction modes, there are DC and planar prediction modes in the HEVC intra prediction algorithm.

Inter prediction algorithm in HEVC, first, performs integer pixel motion estimation. There are 24 different PU sizes and 593 different best motion vector candidates in the integer motion estimation of each 64x64 CU. There are different motion vector search algorithms for integer pixel motion estimation in the literature [7]. Integer motion vector search algorithm is not specified in the HEVC standard. However, full search, diamond search and TZ search algorithms are often used in the implementations. After the integer pixel motion estimation, fractional pixel (half and quarter) accurate variable block size motion estimation is performed in HEVC to increase the performance of integer pixel motion estimation. In H.264, 6-tap FIR filter is used for the interpolation of half pixels, and bilinear interpolation filter is used for the interpolation of quarter pixels [9]. In HEVC, one 8-tap FIR filter and two 7-tap FIR filters are used for the interpolation of half and quarter pixels [6, 7].

Integer discrete cosine transform (DCT) is used in HEVC similar to H.264. In H.264, transformation block sizes can be 4x4 or 8x8. In HEVC, TU sizes can be from 4x4 up to 32x32. In addition to DCT, HEVC uses discrete sine transform (DST) for the 4x4 intra prediction [6, 7]. HEVC performs 2D transform operation by applying 1D transforms in vertical and horizontal directions. The coefficients in HEVC 1D transform matrices are derived from DCT-II and DST-VII basis functions. However, integer coefficients are used for simplicity.

After the transform of residual data, transform coefficients are divided by a quantization step size, and the results are rounded. However, in the inverse quantization, only multiplication by the quantization step size is performed. Quantization step size is determined using the quantization parameter similar to H.264.

Entropy coder uses context adaptive binary arithmetic coding (CABAC) similar to H.264 with several improvements [10]. Entropy coder exploits statistical

redundancies to perform lossless compression. Binarization, context modeling and binary arithmetic coding are the three main parts of CABAC algorithm.

Deblocking filter algorithm reduces blocking artifacts on the edges of the prediction units. Decision making and filtering processes in deblocking filter are simplified in HEVC compared to H.264. Sample adaptive offset (SAO) is added to deblocking filter process in HEVC which is not used in the previous video compression standards [6, 7]. After the deblocking filter, SAO is used to reduce the ringing artifacts.

1.2 Thesis Contributions

As the complexity of video processing and compression algorithms are increasing, the energy consumptions of their hardware implementations are also increasing [11]. Therefore, in this thesis, we propose computation and energy reduction techniques for video processing and compression algorithms. Then, we designed and implemented low energy video processing and compression hardware.

We propose 2D adaptive median filter algorithm [12]. The proposed algorithm detects noiseless pixels, and it eliminates the sorting operation in the median filter. The proposed adaptive median filter algorithm does not perform any sort in the best case, and it sorts 15 pixels instead of 25 pixels in the worst case for a 5x5 window. Then, we generalize this novel low complexity algorithm for 2D adaptive digital image processing (DIP) [13]. We show that the proposed algorithm also reduces computational complexities of 2D gaussian blur and 2D image sharpening without reducing quality of output image.

We also designed and implemented 2D median filter, Gaussian blur and image sharpening hardware including the proposed 2D adaptive DIP algorithm using Verilog HDL. We quantified the impact of the proposed algorithm on the power consumptions of these hardware on a Xilinx Virtex6 FPGA using Xilinx XPower. The proposed algorithm reduced energy consumption of the median filter, Gaussian blur and image sharpening hardware up to 80%, 22% and 31%, respectively.

We propose an approximate HEVC intra angular prediction technique. The proposed technique uses closer neighboring pixels instead of distant neighboring pixels in an intra angular prediction equation if the distance between the neighboring pixels used in this intra angular prediction equation is larger than 2. The proposed approximate HEVC intra angular prediction technique causes negligible PSNR loss and bit rate

increase. Then, we designed and implemented approximate HEVC intra angular prediction hardware using Verilog HDL. The proposed hardware, in the worst case, can process 24 Quad Full HD fps. The proposed hardware is the smallest HEVC intra prediction hardware in the literature.

We propose two pixel correlation based computation and energy reduction techniques for HEVC fractional interpolation [14]. The proposed techniques compare pixels at the inputs of HEVC fractional interpolation operation. If these pixels are equal or similar, interpolation operation is skipped and one of the input pixels is selected as output. The proposed techniques significantly reduce the computational complexity of HEVC fractional interpolation with a negligible PSNR loss and bit rate increase. Also, we designed and implemented two HEVC fractional interpolation hardware including the proposed techniques using Verilog HDL. The proposed hardware, in the worst case, can process 30 Quad Full HD fps. They consume up to 39.7% and 46.9% less energy than original HEVC fractional interpolation hardware.

We propose low energy HEVC fractional interpolation hardware using Hcub MCM [15]. The proposed hardware calculates common sub-expressions in different FIR filter equations in HEVC fractional interpolation algorithm once, and the result is used in all the equations. We designed and implemented the proposed hardware using Verilog HDL. The proposed hardware, in the worst case, can process 30 Quad Full HD fps. It consumes up to 48% less energy than original HEVC fractional interpolation hardware.

We propose two approximate HEVC fractional interpolation filters [16]. Both of these approximate filters use one 4-tap and two different 3-tap FIR filters instead of using one 8-tap and two different 7-tap FIR filters. The proposed interpolation filters significantly reduce the computational complexity of HEVC fractional interpolation with a negligible PSNR loss and bit rate increase. Then, two approximate HEVC fractional interpolation hardware for all PU sizes are designed and implemented using Verilog HDL for each proposed approximate fractional interpolation filter. The proposed hardware, in the worst case, can process 45 Quad Full HD fps. They consume up to 67.1% less energy than original HEVC fractional interpolation hardware.

We propose a computation and energy reduction technique for HEVC DCT operation [17]. The proposed technique is a kind of adaptive zero prediction technique. Since most of the forward transformed and quantized high frequency coefficients in a TU become zero, the proposed computation reduction technique only calculates several

pre-determined low frequency coefficients of transform units (TUs), and it assumes that the remaining coefficients are zero. The proposed technique reduces the computational complexity of HEVC DCT significantly at the expense of slight decrease in PSNR and slight increase in bit rate.

We also designed and implemented two (lower utilization and higher utilization) low energy hardware for HEVC DCT including the proposed computation and energy reduction technique using Verilog HDL. In addition to proposed computation and energy reduction technique, Hcub MCM is used in the transform datapath, and an efficient transpose memory architecture is implemented. The proposed lower utilization hardware and higher utilization hardware can process 48 Quad Full HD and 53 Ultra HD video frames per second, respectively. The proposed technique reduced the energy consumption of the lower utilization hardware and the higher utilization hardware up to 17.9 and 18.9, respectively.

We propose a computation and energy reduction technique for HEVC IDCT/IDST [18]. The proposed technique calculates IDCT and IDST only for DC coefficient if the values of several predetermined forward transformed low frequency coefficients in a TU are smaller than a threshold. Otherwise, it calculates IDCT and IDST for all coefficients in the TU. The proposed technique significantly reduces computational complexity of HEVC inverse transform with a negligible PSNR loss and bit rate increase. Performing IDCT only for DC coefficient in a TU, on the average, achieves 98.87% reduction in addition and 98.70% reduction in shift operations.

We also designed and implemented a low energy HEVC 2D inverse transform (IDCT and IDST) hardware for all TU sizes including the proposed computation and energy reduction technique using Verilog HDL. Clock gating technique is used to reduce the energy consumption of the proposed hardware. Hcub MCM is also used in the transform datapath, and an efficient transpose memory architecture is implemented. The proposed hardware, in the worst case, can process 48 Quad Full HD fps. The proposed technique reduced the energy consumption of this hardware up to 32%.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

Chapter II presents the proposed 2D adaptive digital image processing algorithm. It describes the proposed low energy median filter, Gaussian blur and image sharpening

hardware including the proposed 2D adaptive DIP algorithm and presents their implementation results.

Chapter III, first, explains HEVC intra angular prediction algorithm. Then, it describes the proposed approximate intra angular prediction technique and the proposed approximate HEVC intra angular prediction hardware. It also presents the implementation results.

Chapter IV, first, explains the HEVC fractional interpolation algorithm. Then, it presents the proposed pixel correlation based computation and energy reduction techniques for the HEVC fractional interpolation, and their hardware implementations. After that, the proposed HEVC fractional interpolation hardware using multiplierless constant multiplication is explained. Also, the proposed approximate HEVC fractional interpolation filters and their hardware implementations are explained in Chapter IV. Finally, hardware comparison with the literature is presented.

The proposed computation and energy reduction technique for HEVC DCT algorithm is described in Chapter V. Then, the proposed lower utilization and higher utilization hardware implementations of HEVC DCT including the proposed computation and energy reduction technique are explained. After that, implementation results are presented.

Chapter VI explains the proposed computation and energy reduction technique for HEVC IDCT/IDST algorithm. Then, the proposed low energy hardware implementation of HEVC IDCT/IDST including the proposed computation and energy reduction technique is presented.

Chapter VII presents conclusions and future works.

CHAPTER II

LOW COMPLEXITY 2D ADAPTIVE IMAGE PROCESSING ALGORITHM AND ITS HARDWARE IMPLEMENTATION

Digital images are affected by the noise resulting from image sensors or transmission of images. Image denoising is performed to remove the noise from images. Several linear and non-linear filters are proposed for image denoising [19]. Although non-linear filters are more complex than linear filters, they are more commonly used for image denoising because they reduce smoothing and preserve image edges. 2D spatial median filter is the most commonly used non-linear filter for image denoising. It is a non-linear sorting-based filter. It sorts pixels in a given window, determines the median value, and replaces the pixel in center of the given window with this median value.

Since 2D median filter has high computational complexity, in this thesis, we propose a novel low complexity 2D adaptive median filter algorithm [12]. The proposed algorithm reduces the computational complexity of 2D median filter and produces higher quality filtered images than 2D median filter by exploiting pixel correlations in input image. We also designed a low energy 2D adaptive median filter hardware implementing the proposed 2D adaptive median filter algorithm for 5x5 window size. The proposed hardware is implemented using Verilog HDL. It is verified to work correctly on an FPGA board. It can work at 263 MHz, and it can process 105 full HD (1920x1080) images per second in the worst case on a Xilinx Virtex 6 FPGA. It has more than 80% less energy consumption than original 2D median filter hardware on the same FPGA.

Then, in this thesis, we generalize this novel low complexity adaptive algorithm for 2D digital image processing. We show that the proposed algorithm also reduces computational complexities of 2D Gaussian blur and 2D image sharpening without reducing quality of output image. These DIP algorithms also have high computational complexity. 2D Gaussian blur is commonly used for image smoothing and denoising. In this thesis, 2D Gaussian kernel shown in equation (1.1) is used. Output image is generated by convolving input image with this kernel. 2D image sharpening is used to sharpen images and enhance edges. In this thesis, 2D image sharpening kernel shown in equation (1.2) is used. Output image is generated by convolving input image with this kernel.

$$G = \begin{bmatrix} 3 & 4 & 5 & 4 & 3 \\ 4 & 6 & 7 & 6 & 4 \\ 5 & 7 & 8 & 7 & 5 \\ 4 & 6 & 7 & 6 & 4 \\ 3 & 4 & 5 & 4 & 3 \end{bmatrix} \gg 7 \tag{1.1}$$

$$S = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 2 & 2 & 2 & -1 \\ -1 & 2 & 8 & 2 & -1 \\ -1 & 2 & 2 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \gg 3 \tag{1.2}$$

We also designed a low energy 2D adaptive gaussian blur hardware and a low energy 2D adaptive image sharpening hardware implementing the proposed 2D adaptive gaussian blur and 2D adaptive image sharpening algorithms, respectively, for 5x5 window size. The proposed hardware are implemented using Verilog HDL. The proposed 2D adaptive gaussian blur hardware can work at 152 MHz, and it can process 74 full HD (1920x1080) images per second in the worst case on a Xilinx Virtex 6 FPGA. It has more than 22% less energy consumption than original 2D gaussian blur hardware on the same FPGA. The proposed 2D adaptive image sharpening hardware can work at 185 MHz, and it can process 105 full HD (1920x1080) images per second in the worst case on a Xilinx Virtex 6 FPGA. It has more than 31% less energy consumption than original 2D image sharpening hardware on the same FPGA.

Several median filter algorithms are proposed in the literature [20]-[23]. These algorithms can be classified into two groups. Median filter algorithms proposed in [20],

[21] optimize sorting process to reduce computational complexity of median filter algorithm without reducing quality of filtered images. Median filter algorithms proposed in [22], [23] increase quality of filtered images without increasing computational complexity of median filter algorithm. These algorithms try to detect noisy pixels and adaptively filter only these noisy pixels. However, the 2D adaptive DIP algorithm proposed in this thesis both reduces computational complexity of median filter algorithm and increases quality of filtered images by exploiting pixel correlations in input image.

Several median filter hardware are proposed in the literature [24]-[28]. In [24], an adaptive median filter hardware that detects noisy pixels in several iterations and filters only these noisy pixels is proposed. The proposed median filter hardware uses different sorting algorithms like bitonic and odd-even merge sort. In [25], sorting process of median filter algorithm is optimized. The proposed median filter hardware only finds correct positions of input pixels in the sliding window instead of sorting all pixels in the window. In [26], a histogram based median filter algorithm is proposed. It only performs well for large window sizes. In [27], low complexity bit-pipeline algorithm is proposed to decrease hardware area and increase performance. In [28], an energy efficient median filter hardware is proposed by optimizing memory read/write scheduling of median filter algorithm. However, performance and area of this hardware are not reported. The 2D adaptive median filter hardware proposed in this thesis is compared with these median filter hardware in Section 2.2.

Several Gaussian blur algorithms are proposed in the literature [29], [30]. These algorithms increase quality of output image by increasing computational complexity of Gaussian blur algorithm. However, the 2D adaptive DIP algorithm proposed in this thesis reduces computational complexity of Gaussian blur algorithm without reducing quality of output image by exploiting pixel correlations in input image.

Several Gaussian blur hardware are proposed in the literature [31]-[34]. In [31], a Gaussian blur hardware is proposed for real time stereo vision application for 5x5 window. In [32], nearest pixel approximation is used for Gaussian blur hardware implementation. This reduces hardware area. But, it also reduces quality of output image. In [33], a Gaussian blur hardware is proposed for feature extraction application. This hardware performs two 1D convolution operations instead of performing direct 2D convolution to decrease hardware area. In [34], modified Gaussian blur hardware is proposed to decrease rounding error in kernel coefficients. The 2D adaptive Gaussian

blur hardware proposed in this thesis is compared with these Gaussian blur hardware in Section 2.2.

Several image sharpening hardware are proposed in the literature [35], [36]. However, they are implemented as part of image up-scaling hardware. Their area and performance are not separately reported.

2.1 Proposed 2D Adaptive Digital Image Processing Algorithm

The proposed 2D adaptive DIP algorithm consists of two steps as shown in Figure 2.1. Pseudo code of the proposed 2D adaptive DIP algorithm for 5x5 window is given in Figure 2.2. The proposed algorithm, in the best case, does not perform any sorting or convolution operation. It, in the worst case, sorts or convolves 15 pixels instead of 25 pixels for 5x5 window.

In the first step, the proposed algorithm compares pixels in each row and column of the given window separately. If pixels in a row are similar, row comparison signal for that row is set to 1. Similarly, if pixels in a column are similar, column comparison signal for that column is set to 1. Then, if pixels in all rows are similar, PS_R signal is set to 1. Similarly, if pixels in all columns are similar, PS_C signal is set to 1. The proposed algorithm decides that pixels in a row or column are similar if their 4 most significant bits are the same.

In the second step, output value is determined. If there is full similarity (both PS_R and PS_C are 1), the pixel in center of the window is determined as output value of the window. If there is partial similarity (only PS_R or PS_C is 1), diagonal pixels in the window are sorted or convolved with 1D_1 kernel, and output of this operation is determined as output value of the window. If there is no similarity (neither PS_R nor PS_C is 1), diagonal, horizontal and vertical pixels are sorted or convolved with 1D_1, 1D_2 and 1D_3 kernels, respectively, and their output values (O1, O2, O3) are determined separately. Then, O1, O2, O3 are sorted or convolved with 1D_4 kernel, and output of this operation is determined as output value of the window. Finally, the pixel in center of the given window is replaced with the output value.

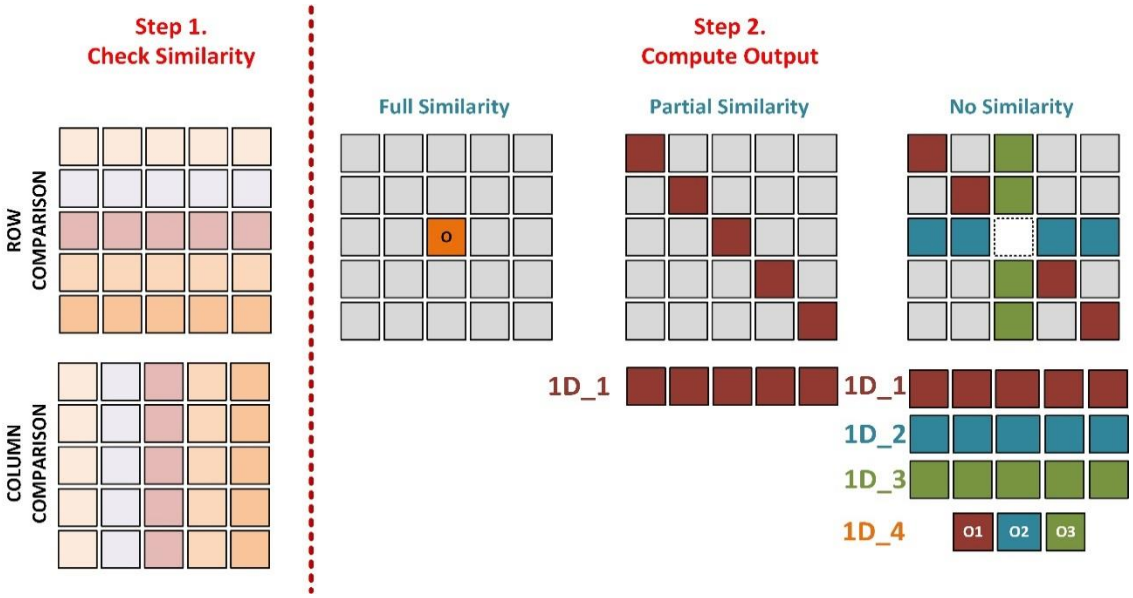

Figure 2.1 Proposed 2D Adaptive Digital Image Processing Algorithm

```
2D_Adaptive_DIP_Algorithm (Window) {
  RC = compare(MSB 4 bits of pixels in each row)
  CC = compare(MSB 4 bits of pixels in each column)
  PS_R = (RC[0] & RC[1] & RC[2] & RC[3] & RC[4])
  PS_C = (CC[0] & CC[1] & CC[2] & CC[3] & CC[4])
  if (PS_R is 1 and PS_C is 1)
     Output = Window(2, 2)
  else if (PS_R is 1 or PS_C is 1)
     Output = 1D_Operation (Diagonal Pixels) // 1D_1
  else {
     O1 = 1D_Operation (Diagonal Pixels)   // 1D_1
     O2 = 1D_Operation (Horizontal Pixels) // 1D_2
     O3 = 1D_Operation (Vertical Pixels)   // 1D_3
     Output = 1D_Operation (O1, O2, O3)    // 1D_4
  }
  Window(2, 2) = Output
}
```

Figure 2.2 Pseudo Code of Proposed 2D Adaptive Digital Image Processing Algorithm

1D kernels shown in equations (1.3), (1.4) and (1.5) are used in the proposed 2D adaptive gaussian blur algorithm.

$$1D\_1 = [3 \quad 6 \quad 8 \quad 6 \quad 3] / 26 \tag{1.3}$$

$$1D\_2 = 1D\_3 = [5 \quad 7 \quad 8 \quad 7 \quad 5] \gg 5 \tag{1.4}$$

$$1D\_4 = [1 \quad 2 \quad 1] \gg 2 \tag{1.5}$$

1D kernels shown in equations (1.6) and (1.7) are used in the proposed 2D adaptive image sharpening algorithm.

$$1D\_1 = 1D\_2 = 1D\_3 = \begin{bmatrix} -1 & 1 & 2 & 1 & -1 \end{bmatrix} \gg 1 \quad (1.6)$$

$$1D\_4 = \begin{bmatrix} -1 & 3 & -1 \end{bmatrix} \quad (1.7)$$

Number of windows with similar pixels in an image varies from image to image. We used HEVC video compression standard test videos [37] and commonly used image processing benchmark images [38] to determine percentage of similarities for different window sizes. Simulation results for 5x5 and 7x7 window sizes for one image from Traffic (2560x1600), People on Street (2560x1600), Basketball Drive (1920x1080), Tennis (1920x1080), Kimono (1920x1080), Park Scene (1920x1080), Vidyo1 (1280x720), Vidyo4 (1280x720), Kristen and Sara (1280x720), Four People (1280x720) videos [37], and Baboon (512x512), Barbara (512x512), Goldhill (512x512), Lena (512x512), Peppers (512x512) images [38] are shown in Table 2.1 and Table 2.2.

Table 2.1 Similarity Percentages (%) for 5x5 and 7x7 Windows (HEVC Images)

		Traffic	**People on Street**	**Basket**	**Tennis**	**Kimono**	**Park Scene**	**Vidyo 1**	**Vidyo 4**	**Kristen and Sara**	**Four People**
5x5	**F. S.**	13.32	13.30	18.29	25.39	20.23	14.64	19.16	22.16	21.06	20.17
	P. S.	2.34	1.68	4.22	4.25	3.67	3.90	4.27	3.71	2.01	4.66
	N. S.	84.54	85.02	77.49	70.36	76.10	81.46	76.57	74.13	76.94	75.17
7x7	**F. S.**	4.44	4.41	4.78	9.86	6.01	3.31	5.09	6.82	8.32	7.79
	P. S.	3.24	1.11	1.54	2.75	1.11	2.15	3.33	2.37	2.26	2.39
	N. S.	92.32	94.48	93.68	87.39	92.88	94.55	91.59	90.81	89.42	89.82

Table 2.2 Similarity Percentages (%) for 5x5 and 7x7 Windows (Benchmark Images)

		Baboon	Barbara	Goldhill	Lena	Peppers
5x5	F. S.	2.21	8.13	7.51	10.31	11.63
	P. S.	1.00	2.44	2.56	2.46	3.20
	N. S.	96.79	89.42	89.92	87.23	85.17
7x7	F. S.	2.47	3.39	3.45	3.23	3.77
	P. S.	2.04	2.10	2.07	2.06	2.04
	N. S.	95.48	94.51	95.48	94.71	94.19

We also quantified impact of the proposed 2D adaptive DIP algorithm on PSNR performance for 5x5 and 7x7 window sizes. For 2D median filter, salt & pepper noise is added to input images. Then, these images are filtered with original 2D median filter algorithm, and with the proposed 2D adaptive median filter algorithm. For 2D Gaussian blur, input images are convolved with the kernel shown in equation (1.1), and with the proposed 2D adaptive Gaussian blur algorithm. For 2D image sharpening, input images are convolved with the kernel shown in equation (1.2), and with the proposed 2D adaptive image sharpening algorithm. PSNR and visual quality results for Basketball Drive image are shown in Figure 2.3. PSNR values between output and input images are computed and shown in Table 2.3 and Table 2.4. These results show that the proposed 2D adaptive DIP algorithm produces higher PSNR values than original 2D DIP algorithms. This is because, if pixels in the window are similar, the proposed 2D adaptive DIP algorithm does not replace the pixel in center of the given window, and therefore preserves the input image.

Figure 2.3 Example Image for 2D Median Filter

Table 2.3 PSNR Values (dB) for HEVC Test Images

Image	W. Size	2D Median Filter				2D Gaussian Blur			2D Image Sharpening		
		S & P Noise	Orig.	Prop.	ΔPSNR (dB)	Orig.	Prop.	ΔPSNR (dB)	Orig.	Prop.	ΔPSNR (dB)
Traffic	5x5	18.189	32.515	34.582	**2.067**	30.132	33.170	**3.039**	27.400	30.160	**2.760**
	7x7		29.345	32.864	**3.519**	29.097	31.260	**2.163**	32.070	32.225	**0.155**
People on Street	5x5	18.156	29.157	33.334	**4.177**	28.295	31.216	**2.920**	26.555	29.214	**2.659**
	7x7		32.371	34.947	**2.576**	27.550	29.676	**2.126**	30.445	30.626	**0.177**
Basket	5x5	18.713	31.291	32.054	**0.763**	29.309	32.265	**2.956**	28.723	31.100	**2.371**
	7x7		30.046	31.191	**1.145**	28.332	29.915	**1.583**	29.903	30.863	**0.961**
Tennis	5x5	17.699	38.145	39.007	**0.862**	33.424	36.180	**2.756**	32.146	34.370	**2.224**
	7x7		35.149	37.729	**2.580**	32.792	34.535	**1.743**	34.501	35.113	**0.612**
Kimono	5x5	17.929	43.436	45.418	**1.982**	35.662	38.853	**3.191**	35.542	37.391	**1.849**
	7x7		39.796	43.904	**4.108**	33.050	33.749	**0.699**	33.585	33.912	**0.327**
Park Scene	5x5	18.077	31.648	34.125	**2.477**	30.510	33.108	**2.599**	28.569	31.862	**3.293**
	7x7		29.574	32.829	**3.255**	29.786	31.860	**2.074**	32.419	33.740	**1.321**
Vidyo1	5x5	18.211	35.080	36.812	**1.732**	30.914	34.850	**3.936**	29.857	32.913	**3.056**
	7x7		32.528	35.356	**2.828**	28.780	30.169	**1.389**	30.336	30.723	**0.387**
Vidyo4	5x5	18.215	35.200	36.383	**1.183**	28.971	31.062	**2.091**	28.465	29.671	**1.206**
	7x7		32.885	35.517	**2.632**	27.412	28.318	**0.906**	28.528	28.528	**0.000**
Kristen and Sara	5x5	17.977	31.316	32.677	**1.361**	28.613	31.840	**3.227**	28.533	30.924	**2.391**
	7x7		28.457	30.794	**2.337**	27.213	29.010	**1.797**	29.490	30.178	**0.688**
Four People	5x5	18.154	30.728	32.265	**1.537**	28.676	32.087	**3.411**	27.039	29.685	**2.645**
	7x7		28.601	31.287	**2.686**	27.353	29.294	**1.941**	29.844	30.124	**0.280**

Table 2.4 PSNR Values (dB) for Benchmark Images

Image	W. Size	2D Median Filter				2D Gaussian Blur			2D Image Sharpening		
		S & P Noise	Orig.	Prop.	ΔPSNR (dB)	Orig.	Prop.	ΔPSNR (dB)	Orig.	Prop.	ΔPSNR (dB)
Boat	5x5	18.526	27.044	28.880	**1.836**	24.682	26.854	**2.171**	23.715	26.103	**2.388**
	7x7		20.563	23.305	**2.742**	23.199	24.519	**1.320**	24.714	25.599	**0.885**
Barbara	5x5	18.461	23.142	24.923	**1.781**	22.933	25.156	**2.223**	24.050	25.825	**1.775**
	7x7		23.546	25.115	**1.569**	22.496	23.715	**1.219**	23.336	27.009	**3.672**
Goldhill	5x5	18.348	28.717	30.701	**1.984**	26.709	28.968	**2.259**	25.544	28.203	**2.659**
	7x7		27.226	30.239	**3.013**	23.919	24.821	**0.902**	24.821	25.574	**0.753**
Lena	5x5	18.459	30.971	32.927	**1.956**	26.313	27.952	**1.639**	25.603	27.284	**1.681**
	7x7		28.894	32.144	**3.250**	24.873	25.745	**0.872**	26.003	26.390	**0.387**
Peppers	5x5	18.100	31.801	33.865	**2.064**	26.434	28.041	**1.607**	25.823	27.490	**1.667**
	7x7		29.991	33.072	**3.081**	24.819	25.641	**0.822**	25.687	26.218	**0.531**

We also quantified impact of the proposed 2D adaptive DIP algorithm on visual quality using structural similarity (SSIM) metric. SSIM values between output images produced by original 2D DIP algorithms and output images produced by the proposed 2D adaptive DIP algorithm are computed and shown in Table 2.5 and Table 2.6. These

results show that the proposed algorithm reduces computational complexities of 2D DIP algorithms without reducing quality of output image.

Table 2.5 Structural Similarity (SSIM) Values for HEVC Test Images

Image	W. Size	2D Median Filter	2D Gaussian Blur	2D Image Sharpening
Traffic	5x5	0.974	0.987	0.968
	7x7	0.951	0.984	0.982
People on Street	5x5	0.976	0.987	0.977
	7x7	0.957	0.985	0.985
Basket	5x5	0.984	0.985	0.970
	7x7	0.981	0.984	0.967
Tennis	5x5	0.984	0.988	0.980
	7x7	0.978	0.989	0.979
Kimono	5x5	0.991	0.994	0.989
	7x7	0.985	0.996	0.990
Park Scene	5x5	0.967	0.981	0.976
	7x7	0.950	0.980	0.968
Vidyo1	5x5	0.985	0.988	0.983
	7x7	0.979	0.988	0.985
Vidyo4	5x5	0.987	0.990	0.976
	7x7	0.980	0.989	0.982
Kristen and Sara	5x5	0.984	0.987	0.987
	7x7	0.973	0.987	0.984
Four People	5x5	0.975	0.982	0.977
	7x7	0.959	0.980	0.978

Table 2.6 Structural Similarity (SSIM) Values for Benchmark Images

Image	W. Size	2D Median Filter	2D Gaussian Blur	2D Image Sharpening
Boat	5x5	0.946	0.969	0.968
	7x7	0.914	0.967	0.937
Barbara	5x5	0.884	0.931	0.955
	7x7	0.891	0.953	0.840
Goldhill	5x5	0.946	0.973	0.965
	7x7	0.921	0.971	0.932
Lena	5x5	0.970	0.982	0.980
	7x7	0.951	0.982	0.962
Peppers	5x5	0.973	0.983	0.972
	7x7	0.961	0.984	0.951

2.2 Proposed 2D Adaptive Digital Image Processing Hardware

The proposed 2D adaptive DIP hardware architecture is shown in Figure 2.4. An input pixels buffer is used to store pixels in a 5x5 window. This on-chip buffer reduces the required off-chip memory bandwidth. After the pixels are loaded into this buffer, 40x4 bit comparators in the comparison unit compare the pixels in each row and column. Based on the comparison results, similarity control signals PS_R and PS_C shown in Figure 2.2 are generated.

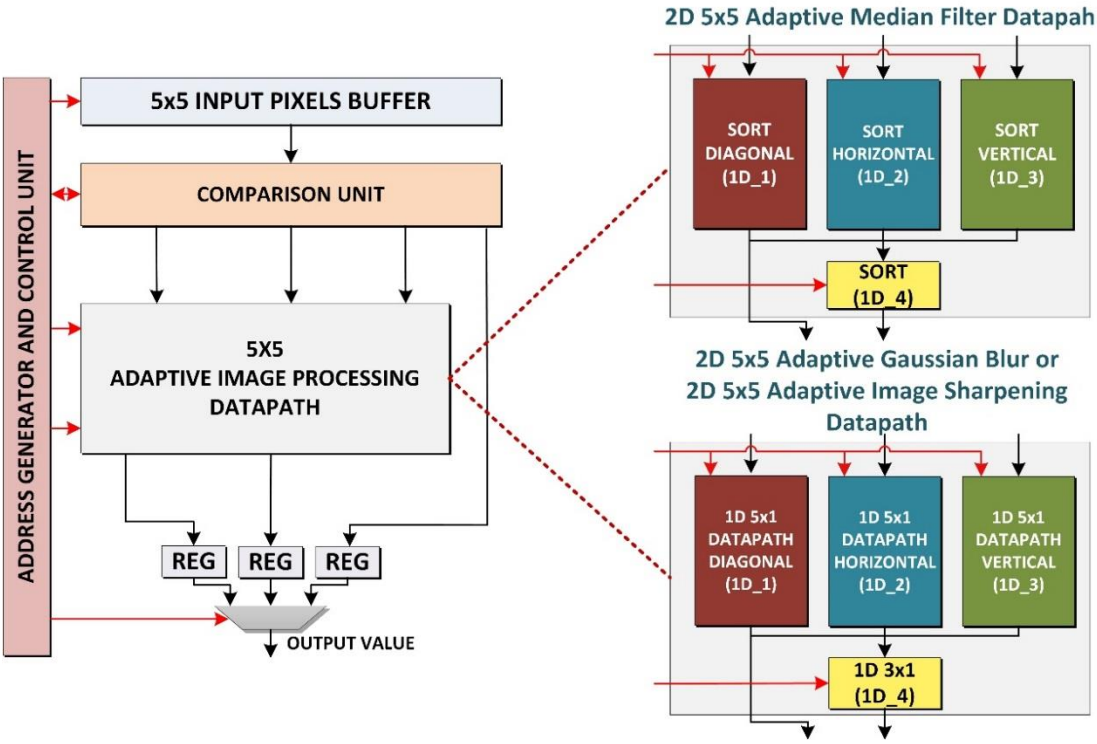

Figure 2.4 Proposed 2D Adaptive Digital Image Processing Hardware

The proposed hardware, in the best case, does not perform any sorting or convolution operation. It, in the worst case, sorts or convolves 15 pixels instead of 25 pixels for 5x5 window. These 15 pixels are sorted or convolved in 3 parallel datapaths. Each datapath has 4 pipeline stages to increase throughput. The proposed hardware produces 1 output per clock cycle.

If there is full similarity, the pixel in center of the window is selected in output multiplexer as the output value. If there is partial similarity, only diagonal 1D datapath (1D_1) is enabled, and the other datapaths are disabled to reduce power consumption. If there is no similarity, all datapaths are enabled, and the output of 1D 3x1 datapath (1D_4) is selected in output multiplexer as the output value.

In the proposed 2D adaptive median filter hardware, 1D 5x1 datapaths (1D_1, 1D_2, 1D_3) sort the given 5 pixels, and determine median value. 1D 3x1 datapath (1D_4) sorts the outputs of 1D_1, 1D_2, 1D_3 datapaths, and determines median value. In the proposed 2D adaptive Gaussian blur hardware and image sharpening hardware, 1D 5x1 datapaths (1D_1, 1D_2, 1D_3) convolve the given 5 pixels with corresponding 1D kernels. 1D 3x1 datapath (1D_4) convolves the outputs of 1D_1, 1D_2, 1D_3 datapaths with corresponding 1D kernel.

The proposed 2D adaptive DIP hardware and original 2D DIP hardware are implemented using Verilog HDL. The Verilog RTL codes are verified with RTL simulations. The RTL simulation results matched the results of software implementations of 2D DIP algorithms. The Verilog RTL codes are synthesized and mapped to a Xilinx Virtex 6 FPGA. The FPGA implementations are verified with post place and route simulations. The post place and route simulation results matched the results of software implementations of 2D DIP algorithms.

FPGA implementation of the proposed 2D adaptive median filter hardware uses 136 slices, 327 LUTs, 150 DFFs, and it can work at 263 MHz. FPGA implementation of the original 2D median filter hardware uses 208 slices, 634 LUTs, 226 DFFs, and it can work at 250 MHz.

FPGA implementation of the proposed 2D adaptive Gaussian blur hardware uses 144 slices, 291 LUTs, 160 DFFs, and it can work at 152 MHz. FPGA implementation of the original 2D Gaussian blur hardware uses 152 slices, 367 LUTs, 301 DFFs, and it can work at 152 MHz.

FPGA implementation of the proposed 2D adaptive image sharpening hardware uses 88 slices, 172 LUTs, 160 DFFs, and it can work at 185 MHz. FPGA implementation of the original 2D image sharpening hardware uses 100 slices, 178 LUTs, 259 DFFs, and it can work at 143 MHz.

The proposed 2D adaptive median filter hardware is verified to work correctly on an Xilinx Zynq ZC7200 FPGA board as shown in Figure 2.5. The FPGA board includes an FPGA, a dual core ARM microprocessor, a high speed AXI bus, 128 MB DDR3 memory, 16 MB quad flash memory, HDMI and Ethernet interfaces. The camera captures 60 fps full HD (1920x1080) images. The proposed hardware filters these images. The filtered images are displayed on HDMI monitor and sent to computer using Ethernet.

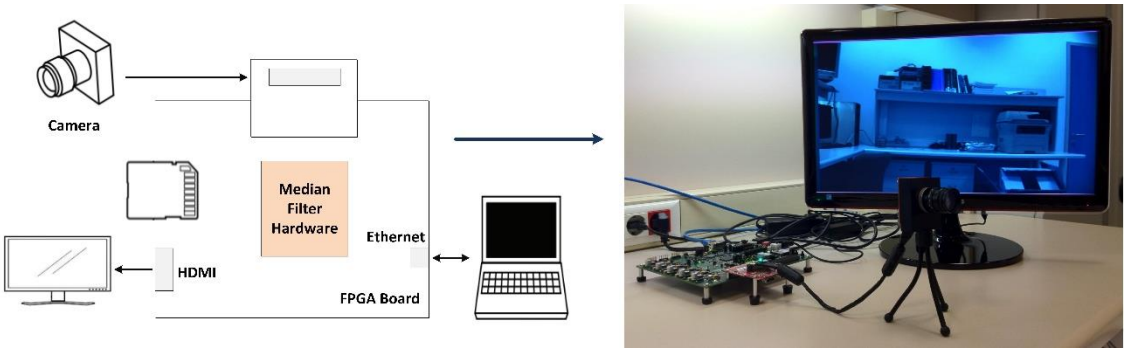

Figure 2.5 Proposed 2D Adaptive Median Filter Hardware Implementation on an FPGA Board

We estimated power consumptions of all FPGA implementations using Xilinx XPower Analyzer for one image from Tennis (1920x1080), Kimono (1920x1080), Park Scene (1920x1080) and Basketball Drive (1920x1080) videos [37]. In order to estimate power consumption of an FPGA implementation, post place and route timing simulation is performed, and signal activities are stored in a VCD file. This VCD file is used for estimating power consumption of the FPGA implementation. For all FPGA implementations, only internal power consumption is considered. Input and output power consumptions are ignored.

Power and energy consumptions of the proposed 2D adaptive DIP hardware and the original 2D DIP hardware are shown in Figure 2.6. As shown in this figure, the proposed 2D adaptive median filter hardware has 42% and 85% less power and energy consumption than the original 2D median filter hardware. The proposed 2D adaptive Gaussian blur hardware has 22% less power and energy consumption than the original 2D Gaussian blur hardware. The proposed 2D adaptive image sharpening hardware has 31% less power and energy consumption than the original 2D image sharpening hardware.

Comparison of the proposed 2D adaptive median filter hardware with the median filter hardware proposed in the literature is shown in Table 2.7. 2D median filter hardware shown in this table process 5x5 pixel 2D windows whereas 1D median filter hardware shown in this table process 25 pixel 1D windows. Although the adaptive median filter hardware proposed in [24] increases quality of output image, this hardware has large area. Sorting process is optimized in [25] without reducing output image quality. But, its hardware area is 10 times larger than the proposed 2D adaptive median

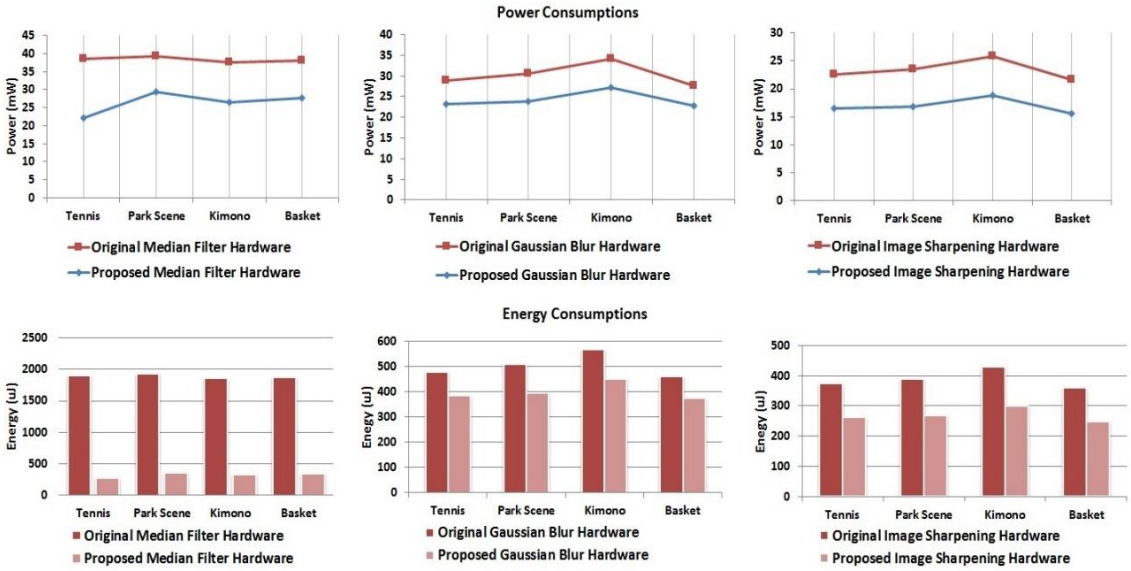

Figure 2.6 Power and Energy Consumptions of FPGA Implementations for Full HD (1920x1080) Images

Table 2.7 Median Filter Hardware Comparison for 5x5 Window

	FPGA	**# of Slices**	**Max. Speed (MHz)**	**Performance (fps)**
[24]	Xilinx Virtex II	1506	305	140 Full HD
[25]	Altera Cyclone II	1309	94	23 Full HD
[26]	Xilinx Virtex II	2300	333	35 Full HD
[27]	Xilinx Virtex II	660	318	Not Reported
Proposed	Xilinx Virtex II (Scaled)	366	140	56 Full HD
	Xilinx Virtex VI	136	263	105 Full HD

filter hardware. Histogram based median filter proposed in [26] gives better results for large window sizes, but it is very costly for small window sizes. Low complexity bit-pipeline algorithm proposed in [27] has smaller hardware area than the other median filter hardware in the literature. But, the proposed 2D adaptive median filter hardware has much smaller area than this hardware. In addition, the median filter hardware proposed in [27] does not increase quality of output image.

Optimized memory scheduling based median filter hardware proposed in [28] reduces energy consumption of median filter hardware up to 53%. However, the proposed 2D adaptive median filter hardware reduces energy consumption of median filter hardware more than 80%. In addition, performance and area of this hardware are not reported.

Comparison of the proposed 2D adaptive Gaussian blur hardware with the Gaussian blur hardware proposed in the literature is shown in Table 2.8. The hardware proposed in [31] has much larger area and lower performance. Although, the hardware proposed in [32] has lower area, it has 0.4 dB average quality loss. The hardware proposed in [33] has larger area, and its performance is not reported. The hardware proposed in [34] increases quality of output image. But, it has much larger area, and its performance is not reported.

Table 2.8 Gaussian Blur Hardware Comparison for 5x5 Window

	FPGA	**# of Slices**	**Max. Speed (MHz)**	**Performance (fps)**
[31]	Xilinx Virtex 5	3775	141	50 Full HD
[32]	Xilinx Virtex 6	52	159	Not Reported
[33]	Altera Cyclone III	545	Not Reported	Not Reported
[34]	Xilinx Spartan 3E	2637	Not Reported	Not Reported
Proposed	Xilinx Virtex 6	144	152	74 Full HD

CHAPTER III

AN APPROXIMATE HEVC INTRA PREDICTION HARDWARE

Intra prediction algorithm predicts the pixels of a block from the pixels of its already coded and reconstructed neighboring blocks. In H.264, there are 9 intra prediction modes for 4x4 luminance blocks, and 4 intra prediction modes for 16x16 luminance blocks. In HEVC, for the luminance component of a frame, intra prediction unit (PU) size can be from 4x4 up to 32x32 and number of intra prediction modes for a PU is 35.

In this thesis, an approximate HEVC intra angular prediction technique is proposed. The proposed technique uses closer neighboring pixels instead of distant neighboring pixels in an intra angular prediction equation if the distance between the neighboring pixels used in this intra angular prediction equation is larger than 2. The proposed approximate HEVC intra angular prediction technique causes negligible PSNR loss and bit rate increase.

In this thesis, an approximate HEVC intra angular prediction hardware is designed and implemented using Verilog HDL. The common-sub expressions in the constant multiplication operations used in HEVC intra angular prediction equations are calculated once and the results are used to generate different constant multiplications in the proposed hardware. Therefore, Hcub multiplierless constant multiplication algorithm is used [40]. The proposed hardware is the smallest HEVC intra prediction hardware in the literature [42]-[53].

3.1 HEVC Intra Prediction Algorithm

HEVC intra prediction algorithm predicts the pixels in prediction units (PU) of a coding unit (CU) using the pixels in the available neighboring PUs [6]. For the luminance component of a frame, 4x4, 8x8, 16x16 and 32x32 PU sizes are available. As shown in Figure 3.1, there are 33 angular prediction modes (Mode) corresponding to different prediction angles (Angle) for each PU size. In addition, there are DC and planar prediction modes for each PU size. An 8x8 PU, four 4x4 PUs in it, and their neighboring pixels are shown in Figure 3.2.

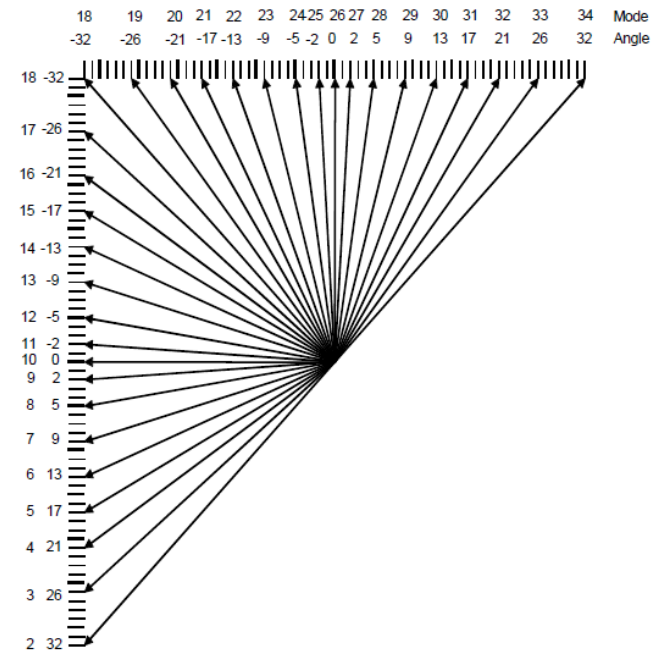

Figure 3.1 HEVC Intra Prediction Mode Directions

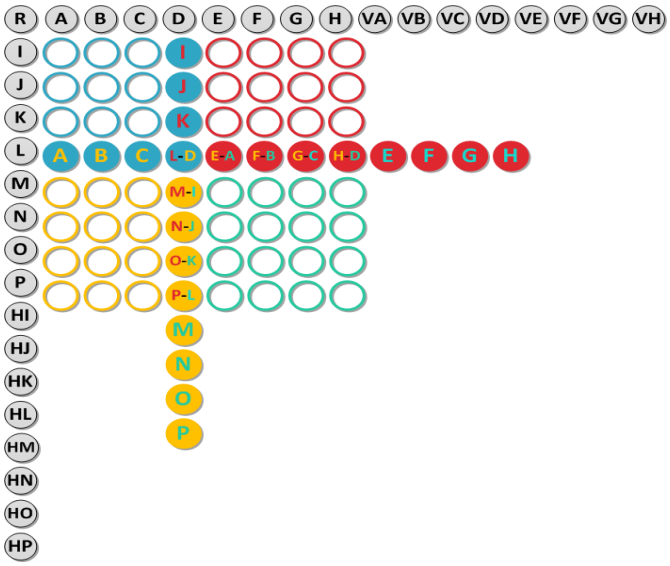

Figure 3.2 Neighboring Pixels of 4x4 and 8x8 PUs

In HEVC intra prediction algorithm, first, reference main array is determined. The pixels in the reference main array are used in the intra prediction equations. If the prediction mode is equal to or greater than 18, reference main array is selected from above neighboring pixels. However, first four pixels of this array are reserved to left neighboring pixels, and if prediction angle is less than zero, these pixels are assigned to the array. If the prediction mode is less than 18, reference main array is selected from left neighboring pixels. However, first four pixels of this array are reserved to above neighboring pixels, and if prediction angle is less than zero, these pixels are assigned to the array.

After the reference main array is determined, ildx which is used to determine positions of the pixels in this array that will be used in the intra prediction equations and iFact which is used to determine coefficients of these pixels are calculated as shown in (3.1a) and (3.1b), respectively. If iFact is equal to 0, neighboring pixels are copied directly to predicted pixels. Otherwise, predicted pixels are calculated as shown in (3.2).

$$iIdx = \big((y+1) * Angle\big) \gg 5 \tag{3.1a}$$

$$iFact = \big((y+1) * Angle\big) \,\&\, 31 \tag{3.1b}$$

$$\begin{gathered} pred[x,y] = ((32 - iFact) * \\ refMain[x + iIdx + 1] + iFact * \\ refMain[x + iIdx + 2] + 16) \gg 5 \\ x = 0\ to\ (PU_{size} - 1), y = 0\ to\ (PU_{size} - 1) \end{gathered} \tag{3.2}$$

All the intra prediction equations can be obtained from (3.2). As an example, reference main array and prediction equations for the 8x8 intra prediction mode 6 with prediction angle 13 are shown in (3.3a) and (3.3b), respectively. The neighboring pixels used in these equations can be seen in Fig. 2.

$$refMain = [0,0,0,0,0,0,0,0,R,A,B,C,D,E,F,G,H,VA,VB,VC,VD,VE,VF,VG,VH] \tag{3.3a}$$

$$\begin{aligned} &pred[0,0] = pred[1,0] = [19*A + 13*B + 16] >> 5 \\ &pred[2,0] = pred[3,0] = [19*B + 13*C + 16] >> 5 \\ &pred[4,0] = \\ &pred[5,0] = pred[6,0] = [19*C + 13*D + 16] >> 5 \\ &pred[7,0] = \quad [19*D + 13*E + 16] >> 5 \\ \\ &pred[0,1] = pred[1,1] = [6*B + 26*C + 16] >> 5 \\ &pred[2,1] = pred[3,1] = [6*C + 26*D + 16] >> 5 \\ &pred[4,1] = \\ &pred[5,1] = pred[6,1] = [6*D + 26*E + 16] >> 5 \\ &pred[7,1] = \quad [6*E + 26*F + 16] >> 5 \end{aligned} \tag{3.3b}$$

*pred[0,2] = pred[1,2] = [25*C + 7*D + 16] >> 5*
*pred[2,2] = pred[3,2] = [25*D + 7*E + 16] >> 5*
pred[4,2] =
*pred[5,2] = pred[6,2] = [25*E + 7*F + 16] >> 5*
*pred[7,2] = [25*F + 7*G + 16] >> 5*

*pred[0,3] = pred[1,3] = [12*D + 20*E + 16] >> 5*
*pred[2,3] = pred[3,3] = [12*E + 20*F + 16] >> 5*
pred[4,3] =
*pred[5,3] = pred[6,3] = [12*F + 20*G + 16] >> 5*
*pred[7,3] = [12*G + 20*H + 16] >> 5*

*pred[0,4] = pred[1,4] = [31*E + 1*F + 16] >> 5*
*pred[2,4] = pred[3,4] = [31*F + 1*G + 16] >> 5*
pred[4,4] =
*pred[5,4] = pred[6,4] = [31*G + 1*H + 16] >> 5*
*pred[7,4] = [31*H + 1*I + 16] >> 5*

*pred[0,5] = pred[1,5] = [18*F + 14*G + 16] >> 5*
*pred[2,5] = pred[3,5] = [18*G + 14*H + 16] >> 5*
pred[4,5] =
*pred[5,5] = pred[6,5] = [18*H + 14*VA + 16] >> 5*
*pred[7,5] = [18*VA+14*VB + 16] >> 5*

*pred[0,6] = pred[1,6] = [5*G + 27*H + 16] >> 5*
*pred[2,6] = pred[3,6] = [5*H + 27*VA + 16] >> 5*
pred[4,6] =
*pred[5,6] = pred[6,6] = [5*VA + 27*VB + 16] >> 5*
*pred[7,6] = [5*VB + 27*VC + 16] >> 5*

*pred[0,7] = pred[1,7] = [24*H + 8*VA + 16] >> 5*
*pred[2,7] = pred[3,7] = [24*VA + 8*VB + 16] >> 5*
pred[4,7] =
*pred[5,7] = pred[6,7] = [24*VB + 8*VC + 16] >> 5*
*pred[7,7] = [24*VC + 8*VD + 16] >> 5*

3.2 Proposed Approximate HEVC Intra Angular Prediction Technique

In this thesis, data reuse technique is first used for reducing amount of computations performed by HEVC intra prediction algorithm [40]. In HEVC, intra 4x4, 8x8, 16x16 and 32x32 luminance angular prediction modes have identical equations. There are identical equations between luminance angular prediction modes of different PU sizes as well. Data reuse technique calculates the common prediction equations for all 4x4, 8x8, 16x16 and 32x32 luminance angular prediction modes only once and uses the result for the corresponding prediction modes. There are 33792, 8448, 2112 and 528 prediction equations in 32x32, 16x16, 8x8 and 4x4 luminance angular prediction modes, respectively. As shown in Table 3.1, using data reuse technique, the numbers of prediction equations that should be calculated for 32x32, 16x16, 8x8 and 4x4 luminance angular prediction modes are reduced to 3735, 1507, 593 and 201, respectively.

A 32x32 CU includes one 32x32 PU, four 16x16 PUs, sixteen 8x8 PUs and sixty four 4x4 PUs. As shown in Figure 3.2, an 8x8 PU and some of the 4x4 PUs have common neighboring pixels. They also have common prediction equations. 4x4, 8x8, 16x16 and 32x32 PUs also have common neighboring pixels and common prediction equations. Therefore, data reuse technique is used for calculating predicted pixels of a 32x32 PU and predicted pixels of the corresponding four 16x16 PUs, sixteen 8x8 PUs and sixty four 4x4 PUs. In this way, the number of prediction equations that should be calculated for a 32x32 CU is reduced from 135168 to 14848.

Table 3.1 Prediction Equation Reductions by Data Reuse

	4x4 PU	**8x8 PU**	**16x16 PU**	**32x32 PU**	**32x32 CU**
# of Pred. Equations	528	2112	8448	33792	135168
# of Pred. Equations with Data Reuse	201	593	1507	3735	14848
Reduction (%)	61.93	71.92	82.16	88.94	89.02

Since we use data reuse technique, instead of calculating intra prediction equations of different prediction modes and PUs separately, we calculate all necessary intra prediction equations together and use the results for the corresponding prediction modes and PUs. As shown in Figure 3.3, there are much more intra prediction equations using closer neighboring pixels than intra prediction equations using distant neighboring pixels. Intra angular prediction equations using neighboring pixels that have larger than 2 distance between them are only 4% of intra angular prediction equations. Therefore, in this thesis, an approximate HEVC intra angular prediction technique is proposed. If distance between the neighboring pixels used in an intra angular prediction equation is larger than 2, the neighboring pixel that has 2 distance with the first neighboring pixel is used instead of second neighboring pixel. Otherwise, original neighboring pixels are used. For example, in Figure 3.3, neighboring pixel C is used instead of neighboring pixel D in the intra prediction equations using neighboring pixels A and D. Original neighboring pixels are used in the intra prediction equations using neighboring pixels A and C.

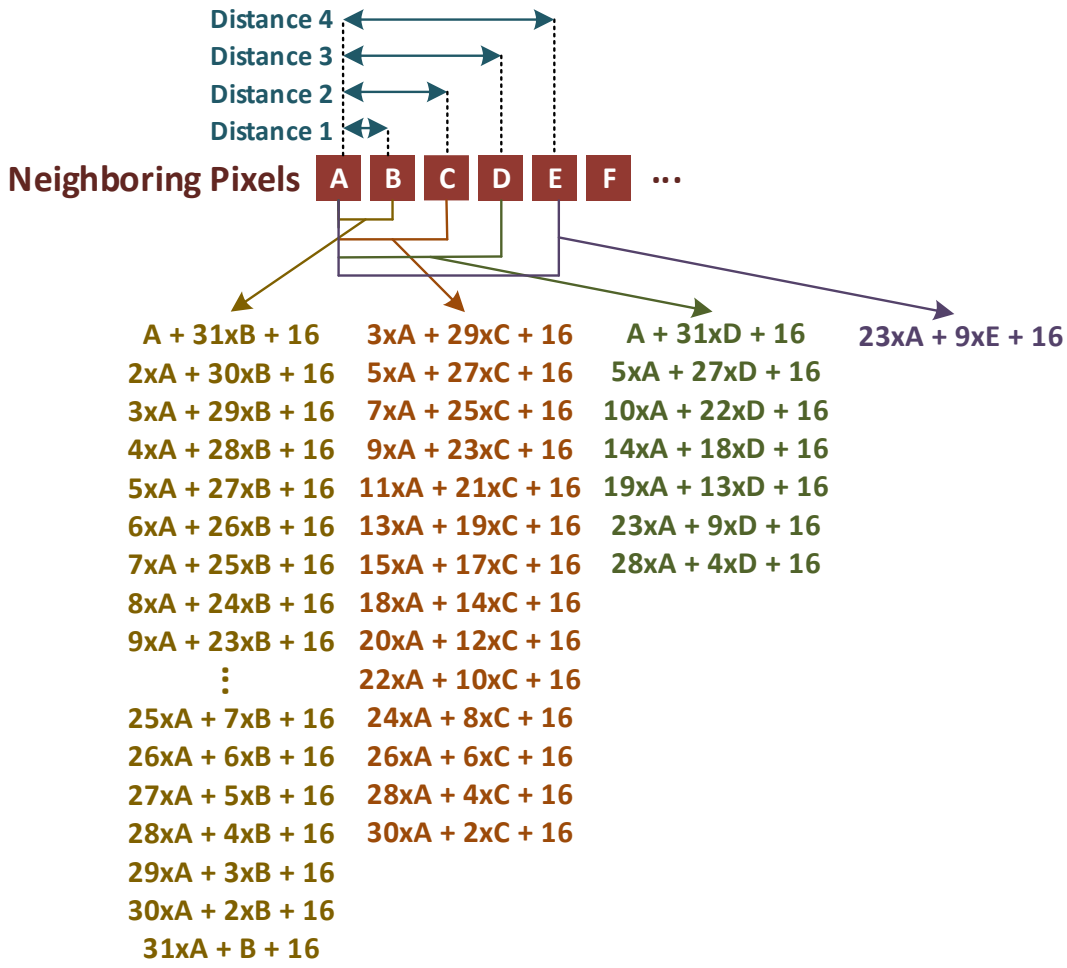

Figure 3.3 Example Intra Angular Prediction Equations for Different Distances

The proposed approximate HEVC intra angular prediction technique is integrated into intra angular prediction in HEVC HM software encoder 15.0 [39]. First ten frames of some of the HEVC test videos [37] are coded with all intra (AI) test configuration and four different quantization parameters (QP) using HEVC HM 15.0 with three different HEVC intra angular predictions; original, the proposed approximate HEVC intra angular prediction using neighboring pixels that have 1 distance between them (D1), and the proposed approximate HEVC intra angular prediction using neighboring pixels that have at most 2 distance between them (D2). The resulting rate-distortion performances are shown in Table 3.2. D2 causes negligible PSNR loss and bit rate increase because neighboring pixel intensities are similar as they are close to each other in the video frame. Since D2 has a negligible impact on PSNR and bit rate, it is implemented in the proposed approximate HEVC intra angular prediction hardware instead of D1.

Table 3.2 BD-Rate(%) and BD-PSNR(dB)

	D1		D2	
Video Sequence	**BD-Rate (%)**	**BD-PSNR (dB)**	**BD-Rate (%)**	**BD-PSNR (dB)**
People on Street	0.3057	-0.0174	0.0238	-0.0014
Traffic	0.0867	-0.0047	-0.0154	0.0008
Tennis	0.2515	-0.0076	0.0196	-0.0005
Kimono	0.1204	-0.0040	0.0348	-0.0009
Basketball Drive	0.4870	-0.0114	0.0657	-0.0013
Park Scene	0.1032	-0.0045	0.0165	-0.0008
Vidyo1	0.8689	-0.0422	0.0962	-0.0044
Vidyo4	0.5559	-0.0248	0.0488	-0.0023
Kristen And Sara	0.8100	-0.0413	0.1525	-0.0072
Four People	0.6710	-0.0390	0.2079	-0.0120
Keiba	0.1294	-0.0071	-0.0110	0.0000
Party Scene	0.3019	-0.0239	0.0308	-0.0029
Race Horses	0.3769	-0.0242	0.0137	-0.0008
Basketball Drill	1.4598	-0.0687	0.1130	-0.0060
Average	**0.4663**	**-0.0229**	**0.0569**	**-0.0028**

3.3 Proposed Approximate HEVC Intra Prediction Hardware

The proposed approximate HEVC intra prediction hardware implementing angular prediction modes for all PU sizes (4x4, 8x8, 16x16 and 32x32) including data reuse and the proposed approximate technique is shown in Figure 3.4.

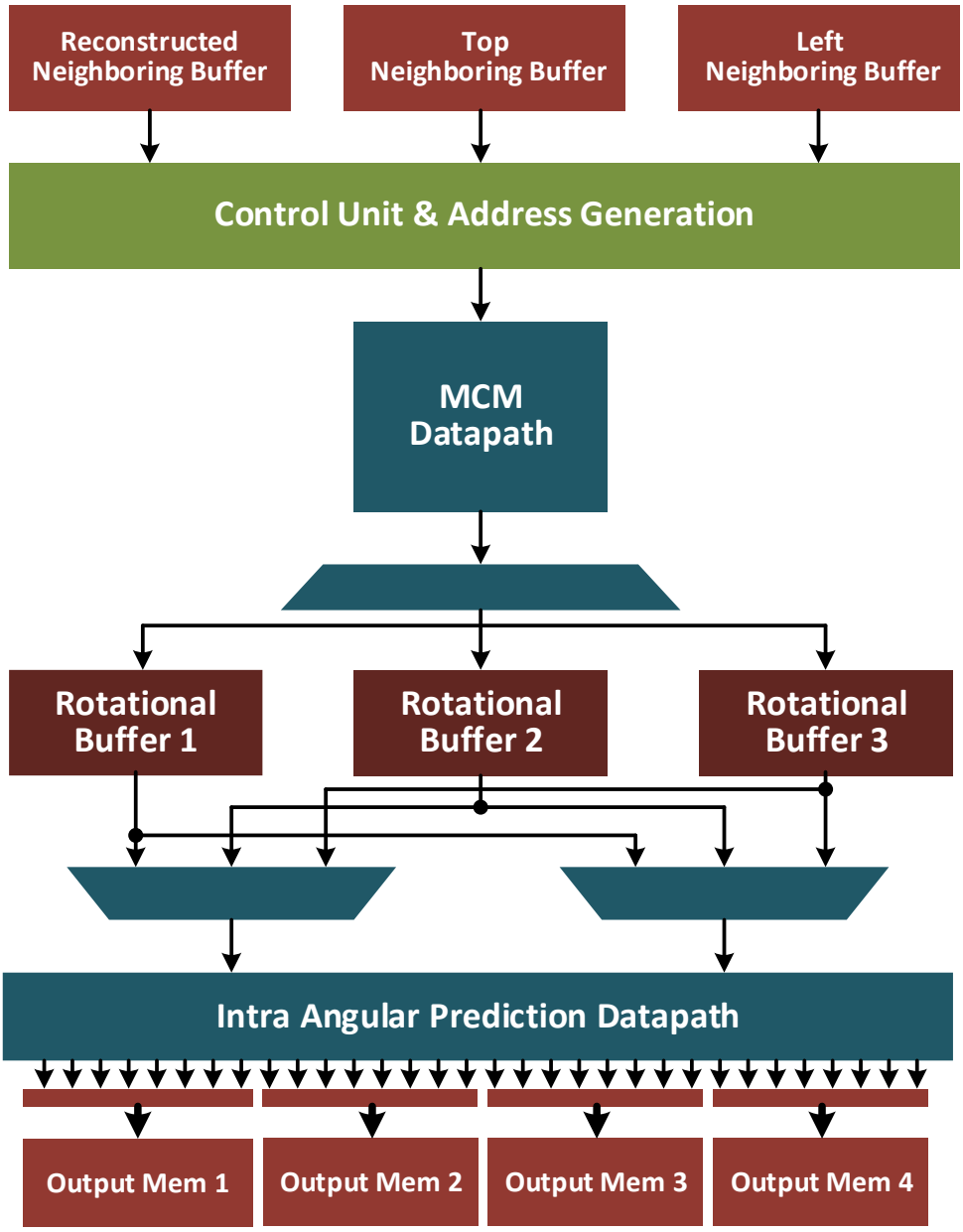

Figure 3.4 Proposed Approximate HEVC Intra Prediction Hardware

Three local neighboring buffers are used to store neighboring pixels in the previously coded and reconstructed neighboring PUs. After a PU in the current CU is coded and reconstructed, the neighboring pixels in this PU are stored in the corresponding buffers. These on chip neighboring buffers reduce the required off-chip memory bandwidth. More on-chip memory accesses are required when the intra angular prediction equations use distant neighboring pixels. Since the proposed approximate intra angular prediction technique uses closer neighboring pixels, it reduces number of on-chip memory accesses.

As shown in Figure 3.3, one neighboring pixel is multiplied with different constants in different prediction equations. Therefore, in the proposed hardware, multiple constant multiplication (MCM) hardware is used to efficiently implement constant multiplications using add and shift operations. The proposed MCM hardware multiplies an input pixel with constants 1, 2, 3, …, 31 by calculating common parts in these constant multiplications once and using them to perform all constant multiplications.

The proposed MCM datapath is shown in Figure 3.5. In the proposed MCM hardware, Hcub MCM algorithm is used to reduce number and size of adders, and adder tree depth [40]. The proposed MCM datapath takes only one neighboring pixel in every two cycles and performs multiplications with constants 1, 3, 5, 7, 9, 11, 13, 15. Multiplications with constants 2, 4, 6, 8, 10, 12, 14, 16 are performed by using these multiplication results and shift operations. Multiplications with constants 17, 18, 19, …, 31 are performed by adding 16 to these multiplication results.

As shown in Figure 3.3, since the number of HEVC intra angular prediction equations using distant neighboring pixels is small and MCM hardware multiplies an input pixel with constant 1, 2, 3, …, 31, MCM hardware will perform many unnecessary constant multiplications for distant neighboring pixels. Since the number of HEVC intra angular prediction equations using closer neighboring pixels is large and the proposed approximate intra angular prediction technique uses closer neighboring pixels, it performs few unnecessary computations.

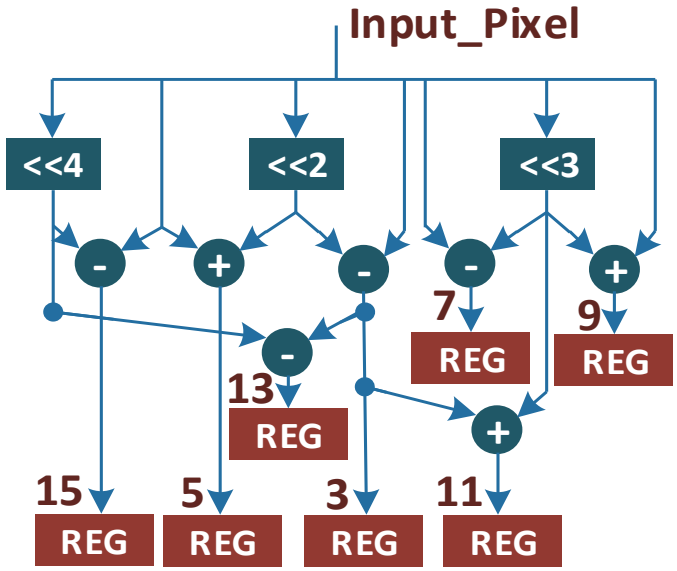

Figure 3.5 Proposed MCM Datapath

As shown in Figure 3.4, three rotational buffers are used in the proposed hardware. As shown in Figure 3.6, first, constant multiplication results of neighboring pixels A and B are stored to rotational buffers 1 and 2, respectively. While the intra prediction equations using both neighboring pixels A and B are calculated, constant multiplication results of neighboring pixel C are stored to rotational buffer 3. After the intra prediction equations using neighboring pixel A are calculated, there is no need to store the constant multiplication results of neighboring pixel A in rotational buffer 1. Therefore, while the intra prediction equations using both neighboring pixels B and C are calculated, constant multiplication results of neighboring pixel D are stored to rotational buffer 1. After the intra prediction equations using neighboring pixel B are calculated, there is no need to store the constant multiplication results of neighboring pixel B in rotational buffer 2. Therefore, while the intra prediction equations using both neighboring pixels C and D are calculated, constant multiplication results of neighboring pixel E are stored to rotational buffer 2. This process repeats rotationally. Therefore, constant multiplication results of a neighboring pixel should be stored 6 cycles in a rotational buffer.

Since the proposed approximate intra angular prediction technique uses closer neighboring pixels instead of distant neighboring pixels, it reduces the number of necessary rotational buffers. If original intra angular prediction equations using distant neighboring pixels are calculated, more rotational buffers will be used to store constant multiplication results of more neighboring pixels.

Since the proposed approximate intra angular prediction technique uses closer neighboring pixels instead of distant neighboring pixels, it also reduces the number of necessary clock cycles. If original intra angular prediction equations using distant neighboring pixels are calculated, additional clock cycles will be used to calculate the intra prediction equations using distant neighboring pixels. For example, in Figure 3.6, additional clock cycles will be used to calculate the intra prediction equations using both neighboring pixels A and D.

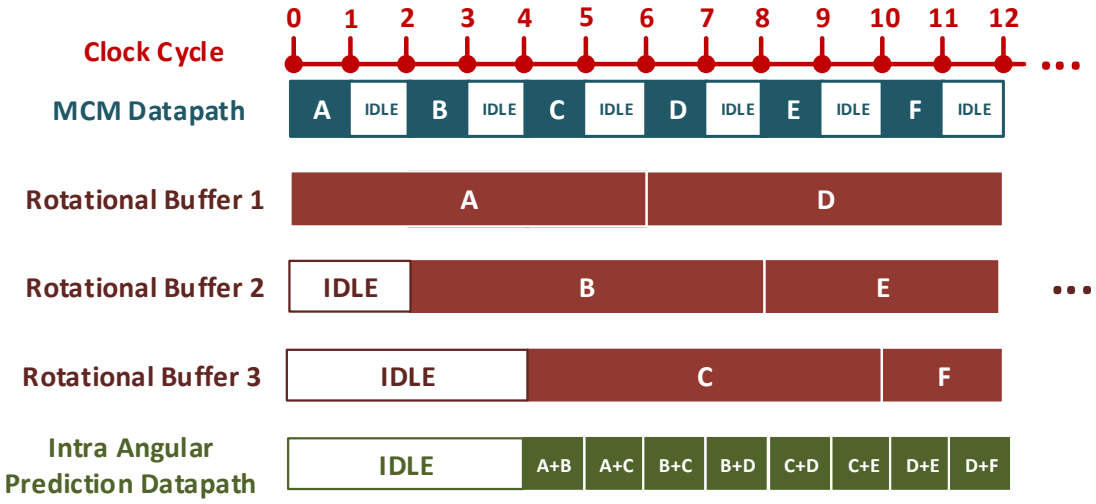

Figure 3.6 Scheduling of HEVC Intra Angular Prediction Hardware

The proposed approximate HEVC intra angular prediction hardware is implemented using Verilog HDL. The Verilog RTL implementation is verified with RTL simulations. RTL simulation results matched results of a software implementation of the proposed approximate intra angular prediction technique.

The Verilog RTL codes are synthesized and mapped to a Xilinx XC6VLX195T FF1156 FPGA with speed grade 3 using Xilinx ISE 14.7. The proposed approximate HEVC intra angular prediction hardware uses 318 LUTs, 1068 DFFs, and 8 BRAMs. The proposed FPGA implementation is verified to work at 200 MHz by post place and route simulations. Therefore, it can process 24 Quad Full HD (3840x2160) video frames per second.

FPGA implementations are also verified on a Xilinx ZYNQ ZC702 FPGA board as shown in Figure 3.7. The FPGA board has a 28 nm FPGA and dual-core ARM microprocessor. It also has 1GB DRAM and several interfaces such as UART and HDMI. Microprocessor reads video frames from SD card and sends them to FPGA using a high speed AXI bus. The proposed hardware performs intra prediction. Then, microprocessor displays intra predicted frames on HDMI monitor and stores them to SD card.

Verilog RTL code of the proposed approximate HEVC intra angular prediction hardware is also synthesized and place & routed to TSMC 90nm standard cell library. Gate count of resulting ASIC implementation is calculated as 3.2k, excluding on-chip memories, based on NAND (2x1) gate area.

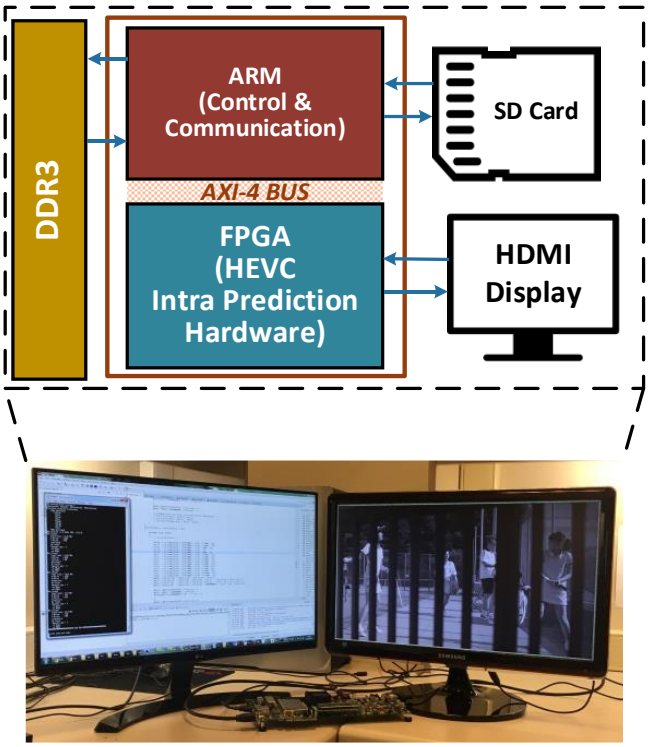

Figure 3.7 Implementation of Proposed Approximate HEVC Intra Prediction Hardware on an FPGA Board

Comparisons of the FPGA and ASIC implementations of proposed approximate HEVC intra angular prediction hardware with the FPGA and ASIC implementations of HEVC intra prediction hardware proposed in the literature are shown in Table 3.3 and Table 3.4, respectively [42]-[53]. The proposed approximate HEVC intra angular prediction hardware has the smallest area and the second best performance.

Table 3.3 Comparison of FPGA Implementations

	[43]	[44]	[45]	[46]	[42]	[52]	[53]	Proposed
FPGA	Xilinx Virtex 6	ZYNQ 7000	Xilinx Virtex 6	Altrea Stratix	Xilinx Virtex6	Xilinx Virtex6	Xilinx Virtex6	Xilinx Virtex6
DFF	5.5 K	22 K	110 K	6934	849	2006	1168	318
LUT	14 K	43 K	170 K	13409	2381	6013	4425	1068
BRAM	---	94	---	---	4	4	4	8
Max Freq. (MHz)	110	150	219	162	150	166	227	200
Fps	30 3840x2160	---	24 3840x2160	---	30 1920x1080	40 1920x1080	55 1920x1080	24 3840x2160
PU Size	4,8,16,32	4,8,16,32	4,8,16,32	4,8,16,32	4,8	4,8,16,32	4,8,16,32	4,8,16,32

Table 3.4 Comparison of ASIC Implementations

	[47]	[48]	[49]	[50]	[51]	[42]	[52]	Proposed
Tech.	90 nm	40 nm	90 nm	130 nm	90 nm	90 nm	90 nm	90 nm
Gate Count	127.3 K	27 K	76.8 K	324 K	712.2 K	5.4 K	16.1 K	3.2 K
Max Freq. (MHz)	200	200	270	400	357	150	250	333
Fps	30 3840x2160	---	---	60 1920x1080	46 2160x1600	30 1920x1080	60 1920x1080	40 3840x2160
Memory	6 KB	4.9 KB	5.6 KB	---	---	---	3 KB	3KB
Power Dissipatio	---	---	---	---	92.1 mW	23.2 mW	28.5 mW	---
PU Size	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8	4, 8, 16, 32	4, 8, 16, 32

Power consumption of the proposed approximate HEVC intra angular prediction hardware is estimated for Tennis and Kimono (1920 x 1080) videos [37] using Xilinx XPower Analyzer tool. Switching activities during post place & route timing simulation of the proposed hardware at 100 MHz clock frequency are stored to VCD files. Xilinx XPower Analyzer tool uses placed & routed netlist and these VCD files to estimate power consumption of the proposed FPGA implementation. Energy consumption comparison of the proposed FPGA implementation and the HEVC intra prediction hardware in the literature is shown in Figure 3.8.

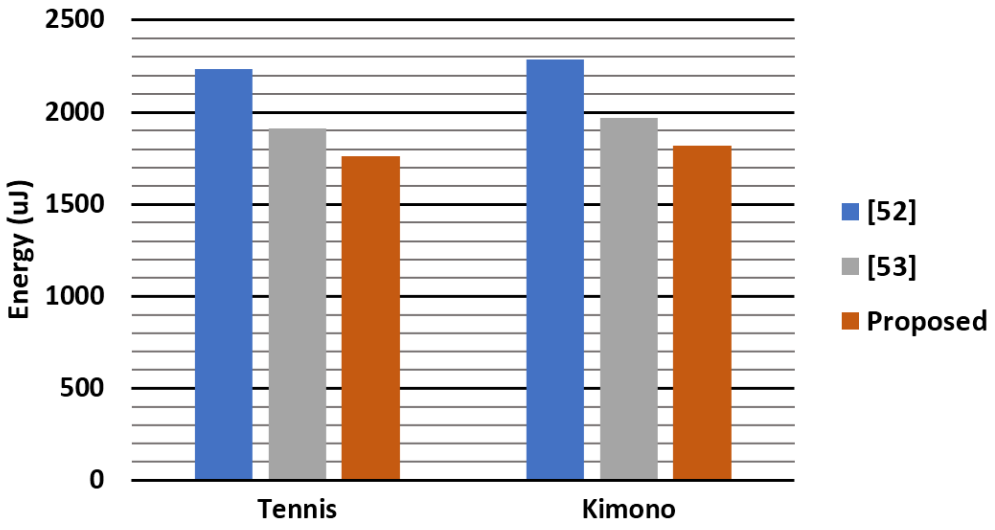

Figure 3.8 Energy Consumption Comparison

CHAPTER IV

LOW ENERGY HEVC FRACTIONAL INTERPOLATION HARDWARE

To increase the performance of integer pixel motion estimation, fractional pixel (half and quarter) accurate variable block size motion estimation is performed in HEVC. Fractional interpolation is one of the most computationally intensive parts of HEVC video encoder and decoder. On average, one fourth of the HEVC encoder complexity and 50% of the HEVC decoder complexity are caused by fractional interpolation [6].

In H. 264 standard, a 6-tap FIR filter is used for half-pixel interpolation and a bilinear filter is used for quarter-pixel interpolation [9]. In HEVC standard, one 8-tap and two different 7-tap FIR filters are used for both half-pixel and quarter-pixel interpolations. In H.264, 4×4 and 16×16 block sizes are used. However, in HEVC, prediction unit (PU) sizes can be from 4×4 to 64×64. Therefore, HEVC fractional interpolation is more complex than H.264 fractional interpolation.

Therefore, in this thesis, we proposed three different HEVC fractional interpolation hardware implementations for all PU sizes. In the first hardware implementation, two pixel correlation based computation and energy reduction techniques (pixel equality based computation reduction (PECR) and pixel similarity based computation reduction (PSCR)) are used. The second hardware implementation calculates common sub-expressions in different FIR filter equations used in HEVC

fractional interpolation algorithm once. It also uses Hcub multiplierless constant multiplication (MCM) algorithm [40] to reduce number and size of the adders and to minimize the adder tree depth. Two approximate HEVC fractional interpolation filters (F1 and F2) are proposed and used in the third hardware implementation.

4.1 HEVC Fractional Interpolation Algorithm

In HEVC standard, one 8-tap and two different 7-tap FIR filters are used for both half-pixel and quarter-pixel interpolations. These 3 FIR filters type A, type B and type C are shown in (4.1), (4.2), and (4.3), respectively. The symbol (>>) in the equations represents right shift operation which is used to reduce bit length of fractional pixels to 8 bits. The shift1 value is determined based on bit depth of the integer pixel [6].

$$a_{0,0} = \left(-A_{-3,0} + 4 * A_{-2,0} - 10 * A_{-1,0} + 58 * A_{0,0} + 17 * A_{1,0} - 5 * A_{2,0} + A_{3,0}\right) \gg shift1 \quad (4.1)$$

$$b_{0,0} = \left(-A_{-3,0} + 4 * A_{-2,0} - 11 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 11 * A_{2,0} + 4 * A_{3,0} - A_{4,0}\right) \gg shift1 \quad (4.2)$$

$$c_{0,0} = \left(-A_{-2,0} - 5 * A_{-1,0} + 17 * A_{-0,0} + 58 * A_{1,0} - 10 * A_{2,0} + 4 * A_{3,0} - A_{4,0}\right) \gg shift1 \quad (4.3)$$

Integer pixels ($A_{x,y}$), half pixels ($a_{x,y}$, $b_{x,y}$, $c_{x,y}$, $d_{x,y}$, $h_{x,y}$, $n_{x,y}$) and quarter pixels ($e_{x,y}$, $f_{x,y}$, $g_{x,y}$, $i_{x,y}$, $j_{x,y}$, $k_{x,y}$, $p_{x,y}$, $q_{x,y}$, $r_{x,y}$) in a PU are shown in Figure 4.1. The half pixels a, b, c are interpolated from nearest integer pixels in horizontal direction, and the half-pixels d, h, n are interpolated from nearest integer pixels in vertical direction. The quarter pixels e, f, g are interpolated from the nearest half pixels a, b, c respectively in vertical direction using type A filter. The quarter pixels i, j, k are interpolated similarly using type B filter, and the quarter pixels p, q, r are interpolated similarly using type C filter. All fractional pixels necessary for fractional motion estimation are calculated in HEVC fractional interpolation algorithm used in HEVC encoder.

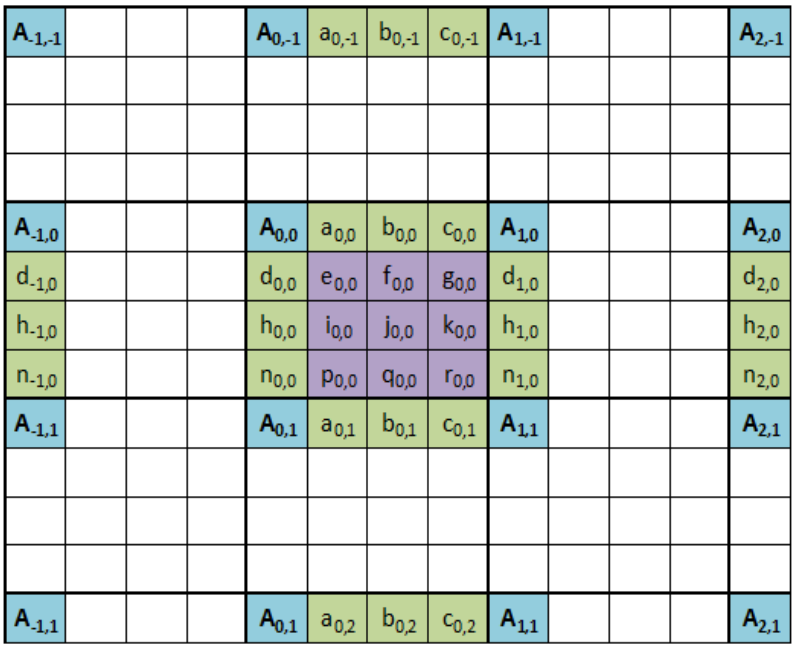

Figure 4.1 Integer, Half and Quarter Pixels

4.2 Proposed Pixel Correlation Based Computation and Energy Reduction Techniques and Their Hardware Implementations

Two pixel correlation based computation and energy reduction techniques (pixel equality based computation reduction (PECR) and pixel similarity based computation reduction (PSCR)) are proposed for HEVC intra prediction in [41, 42]. In this thesis, these techniques are applied to HEVC fractional interpolation. The proposed techniques compare the pixels at the inputs of HEVC fractional interpolation operation. If these pixels are equal or similar, interpolation operation is skipped and one of the input pixels is selected as output. Therefore, the computational complexity of HEVC fractional interpolation is reduced. The PECR technique does not affect the PSNR and bit-rate. The PSCR technique slightly decreases PSNR and increases bit-rate

In this thesis, a low energy HEVC fractional (half-pixel and quarter-pixel) interpolation hardware for all PU sizes including the proposed techniques is also designed and implemented using Verilog HDL. The Verilog RTL code is verified to work at 200 MHz in a Xilinx Virtex 6 FPGA. The proposed hardware, in the worst case, can process 30 quad full HD (3840x2160) video frames per second. The proposed PECR and PSCR techniques reduced the energy consumption of the proposed hardware up to 39.7% and 46.9%, respectively.

4.2.1 Proposed PECR and PSCR Techniques

In this thesis, two pixel correlation based computation and energy reduction techniques (PECR and PSCR) for HEVC fractional interpolation are proposed. The proposed PECR technique compares the input pixels of an FIR filter. If the input pixels are equal, the FIR filter output is equal to one of the input pixels. Therefore, the FIR filter calculation becomes unnecessary and it is skipped. If the input pixels are not equal, the FIR filter operation is performed.

The proposed PSCR technique compares the input pixels of an FIR filter. If the input pixels are similar, the FIR filter output is assumed to be equal to the input pixel multiplied with the largest coefficient in the FIR filter. Therefore, the FIR filter calculation becomes unnecessary and it is skipped. The PSCR technique checks the similarity of input pixels by truncating their least significant bits by specified amount (1, 2, 3 or 4 bits) and comparing the truncated pixels. If the input pixels are not similar, the FIR filter operation is performed.

Equality and similarity percentages of the input pixels of FIR filters vary from frame to frame. Therefore, one frame of Tennis, Kimono, Park Scene and BQ Terrace (1920x1080) videos [37] coded with quantization parameters (QP) 22, 27, 32 and 37 are analyzed to determine equality and similarity percentages using HEVC Test Model HM encoder software [39].

Table 4.1 shows the equality and 3-bit truncated similarity percentages for integer pixel inputs (Ax,y) and half-pixel inputs (ax,y, bx,y, cx,y) of FIR filters. As shown in Table 4.1, significant amount of FIR filter inputs are equal or similar. Therefore, the proposed PECR and PSCR techniques skip significant amount of FIR filter calculations.

Table 4.2 shows the addition and shift operation reductions achieved by the proposed PECR and PSCR for 3-bit truncated (3bT) techniques for one frame of each video sequence. As shown in Table 4.2, the proposed PECR and PSCR for 3bT techniques achieved up to 26.34% and 49.28% computation reductions, respectively. The proposed techniques have overhead of only 3628800 comparisons for a full HD (1920x1080) frame.

Table 4.1 Equality and Similarity Percentages

		HEVC Fractional Interpolation (Equal)				HEVC Fractional Interpolation (3bT)			
		A	a	b	c	A	a	b	c
Tennis	22	9.9	17.1	18.7	17.1	35.2	42.7	44.6	42.8
	27	13.8	24.8	25.5	24.7	37.4	45.4	47.4	45.5
	32	16.0	28.2	28.6	28.3	39.1	47.4	49.4	47.5
	37	18.9	31.3	31.2	31.4	40.5	50.0	52.1	50.1
Kimono	22	15.5	9.8	8.6	8.7	42.4	38.6	39.1	38.7
	27	17.2	11.1	10.3	10.1	45.7	41.5	42.1	41.5
	32	17.6	11.9	11.3	11.0	48.8	44.1	45.0	44.1
	37	19.5	12.6	12.0	11.7	52.3	46.9	47.9	47.0
Park Scene	22	4.8	2.4	2.0	2.3	30.8	28.8	30.0	28.8
	27	8.3	5.7	5.0	5.5	34.7	32.4	33.6	32.5
	32	10.2	7.7	6.8	7.5	37.9	35.5	36.9	35.6
	37	12.8	9.5	8.5	9.2	40.1	38.4	40.2	38.5
BQ Terrace	22	2.0	2.4	1.9	2.3	11.2	24.4	23.4	24.5
	27	7.3	6.0	5.3	5.9	21.2	34.2	32.8	34.3
	32	9.9	7.4	6.4	7.2	24.3	37.3	35.7	37.3
	37	11.9	9.5	8.4	9.3	26.6	39.3	37.4	39.4

Table 4.2 Computation Reductions by PECR and PSCR 3bT

	QP	**PECR**		**PSCR for 3bT**	
		Addition Reductio	**Shift Reduction**	**Addition Reductio**	**Shift Reduction**
Tennis	**22**	14.54 %	14.54 %	40.10 %	40.10 %
	37	26.34 %	26.34 %	46.64 %	46.64 %
Kimono	**22**	11.62 %	11.62 %	40.24 %	40.24 %
	37	15.06 %	15.06 %	49.28 %	49.28 %
Park Scene	**22**	3.26 %	3.26 %	29.84 %	29.84 %
	37	10.56 %	10.56 %	39.46 %	39.46 %
BQ Terrace	**22**	2.12 %	2.12 %	18.94 %	18.94 %
	37	10.20 %	10.20 %	33.86 %	33.86 %

The proposed PSCR technique is integrated into fractional interpolation performed by HEVC Test Model HM encoder software [39]. The impact of the proposed PSCR technique on rate-distortion performance is determined for Tennis, Kimono, Park Scene and BQ Terrace (1920x1080) videos [37]. Rate-distortion

performances of original HEVC and HEVC using PSCR technique for fractional interpolation are shown in Figure 4.2. The proposed PSCR technique slightly decreased PSNR and increased bit-rate.

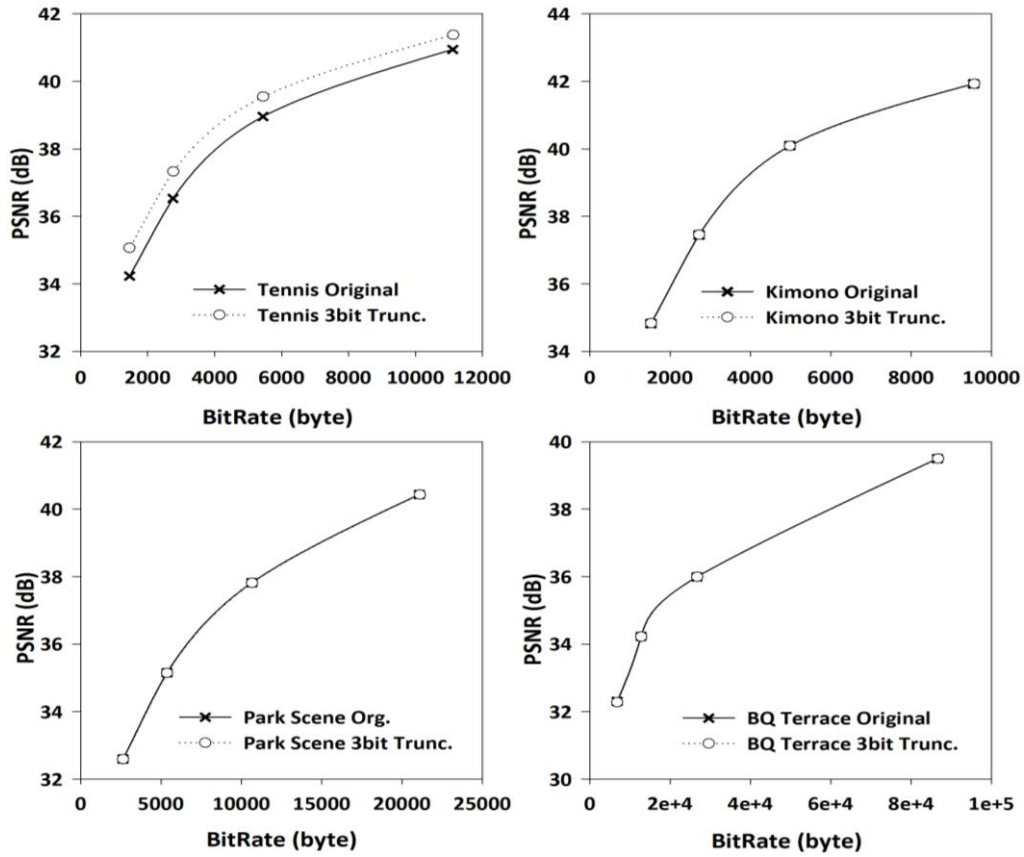

Figure 4.2 Rate-Distortion Performances of Original HEVC and HEVC Using PSCR Techniques for Fractional Interpolation

4.2.2 Proposed HEVC Fractional Interpolation Hardware (PECR and PSCR)

The proposed HEVC fractional interpolation hardware for all PU sizes including the proposed PECR and PSCR techniques is shown in Figure 4.3. The proposed hardware interpolates all the fractional (half-pixels and quarter-pixels) pixels for the luma component of a PU using integer or half pixels. Four buffers are used to store integer and half pixels necessary for interpolating the half and quarter pixels. The interpolated a, b and c half-pixels are stored in the filtered pixels buffers A, B and C, respectively. These on-chip buffers reduce the required off-chip memory bandwidth and power consumption.

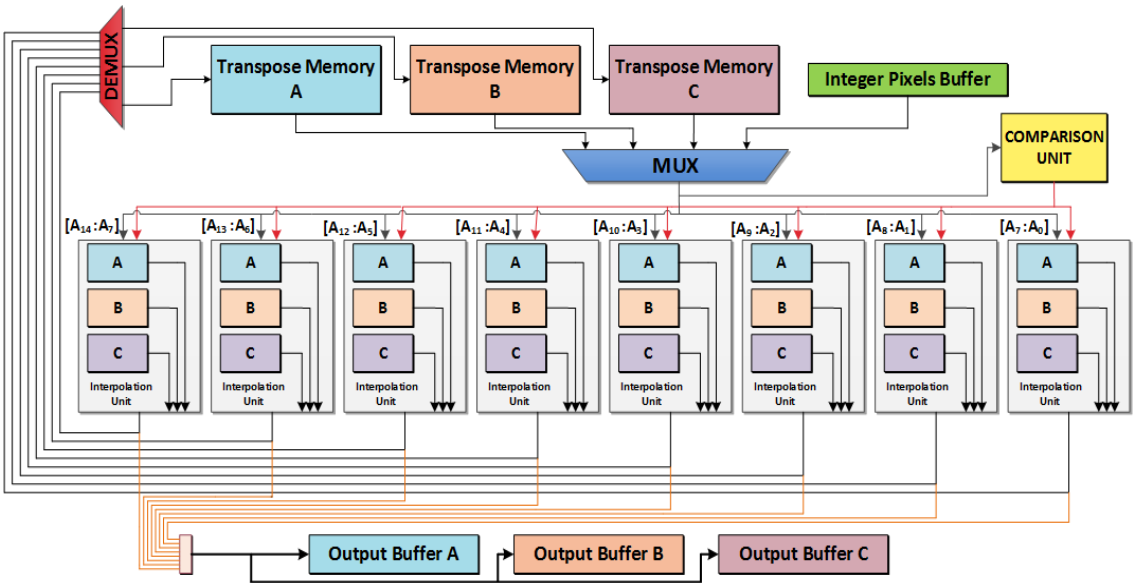

Figure 4.3 Proposed HEVC Fractional Interpolation Hardware (PECR and PSCR)

8 parallel interpolation units are used to interpolate the 8x3=24 fractional pixels of a PU in parallel. As shown in Figure 4.3, three FIR filters (type A, type B, type C) are implemented separately in an interpolation unit.

Since 15 fractional pixels should be interpolated for one integer pixel, 64x15 fractional pixels should be interpolated for an 8x8 PU. Also, 8x7 extra a, b, c half-pixels should be interpolated for the interpolation of quarter-pixels. First, integer pixels are loaded into integer pixel buffer in one clock cycle. Then, 8x8 d, h, n half-pixels are interpolated and stored in the output buffer in 8 clock cycles. After that 15x8 a, b, c half-pixels are interpolated and stored in the filtered pixel buffers A, B and C, respectively, in 15 clock cycles. Finally, 9x8x8 quarter-pixels are interpolated using a, b, c half-pixels and stored in the output buffer in 3x8=24 clock cycles. Therefore, the proposed hardware, in the worst case, interpolates the fractional pixels for an 8x8 PU in 48 clock cycles.

In this thesis, an original HEVC fractional interpolation hardware (FIHW) is also designed for energy consumption comparison. This hardware computes type A, B and C filters separately. The original HEVC fractional interpolation hardware (FIHW) does not have the comparison unit. In both the proposed HEVC fractional interpolation hardware including the PECR technique (FIHW+PECR) and the proposed HEVC fractional interpolation hardware including the PSCR technique (FIHW+PSCR), 14 comparators are used to check similarity of the input pixels of FIR filters. FIHW+PECR uses 8-bit comparators. FIHW+PSCR for 1bT uses 7-bit comparators. Similarly,

FIHW+PSCR for 4bT uses 4-bit comparators. Based on the comparison results, disable signals are generated for each FIR filter and sent to the interpolation units. If the input pixels of an FIR filter are equal or similar, input registers of the corresponding FIR filter hardware are not updated, and a multiplexer at the output of interpolation unit is used to select the input pixel multiplied with the largest coefficient in the FIR filter instead of interpolated pixel. This prevents unnecessary switching activities in the FIR filter hardware.

The proposed FIHW, FIHW+PECR and FIHW+PSCR hardware are implemented using Verilog HDL. The Verilog RTL codes are verified with RTL simulations. RTL simulation results matched the results of fractional interpolation implementation in HEVC HM encoder software [39].

The Verilog RTL codes are mapped to a Xilinx XC6VLX75T FF784 FPGA with speed grade 3 using Xilinx ISE 13.4. All FPGA implementations are verified to work at 200 MHz by post place and route simulations. Post place and route simulation results matched the results of fractional interpolation implementation in HEVC HM encoder software [39]. Therefore, they can process 30 quad full HD (3840x2160) video frames per second. FIHW FPGA implementation uses 4110 LUTs, 3448 DFFs and 6 BRAMs. FIHW+PECR FPGA implementation uses 4577 LUTs, 3408 DFFs, and 4 BRAMs. FIHW+PSCR for 3bT FPGA implementation uses 2381 LUTs, 849 DFFs, and 4 BRAMs.

Power consumptions of FIHW, FIHW+PECR and FIHW+PSCR for 3bT FPGA implementations are estimated using Xilinx XPower Analyzer tool. Post place and route timing simulations are performed for Tennis, Kimono, Park Scene and BQ Terrace (1920x1080) videos at 100 MHz [37], and signal activities are stored in VCD files. These VCD files are used for estimating the power consumptions of all FPGA implementations. Energy consumption results of FIHW, FIHW+PECR and FIHW+PSCR for 3bT for one frame of each video are shown in Figure 4.4. As shown in Figure 4.4, PECR and PSCR techniques reduced the energy consumption of FIHW FPGA implementation up to 39.7% and 46.9%, respectively.

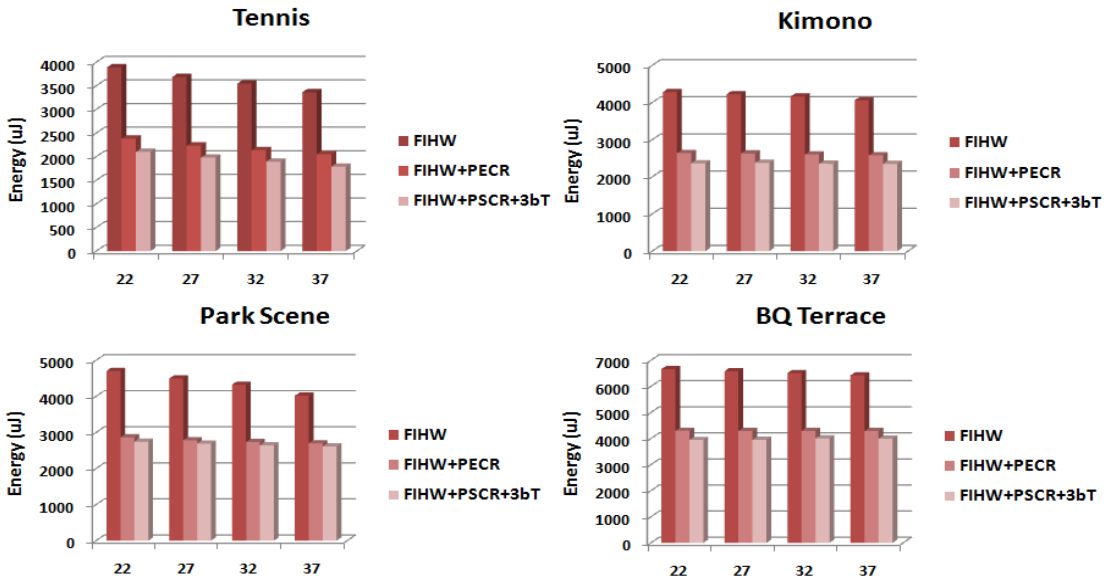

Figure 4.4 Energy Consumptions of HEVC Fractional Interpolation Hardware

4.3 Proposed HEVC Fractional Interpolation Hardware (MCM)

The proposed hardware calculates common sub-expressions in different FIR filter equations in HEVC fractional interpolation algorithm once. The proposed hardware also uses Hcub multiplierless constant multiplication (MCM) algorithm [40] in order to reduce number and size of the adders and to minimize the adder tree depth.

The type A and type B FIR filter equations for 8 half-pixels are shown in Figure 4.5. As shown in Figure 4.5, common sub expressions are calculated in different equations and same integer pixel is multiplied with different constant coefficients in different equations. Therefore, in the proposed hardware, common sub-expressions in different equations are calculated once, and the result is used in all the equations. The proposed hardware also uses Hcub MCM algorithm in order to reduce number and size of the adders, and to minimize the adder tree depth [40].

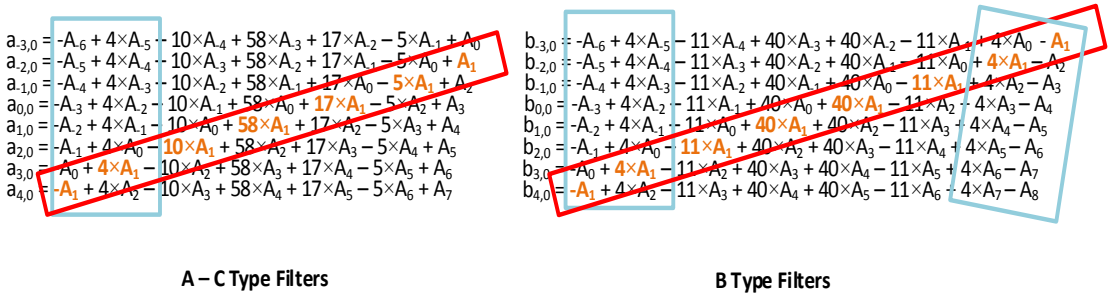

Figure 4.5 Type A and Type B Filters

Hcub algorithm tries to minimize number of adders, their bit size and adder tree depth in a multiplier block, which multiplies a single input with multiple constants. Hcub algorithm is used in this thesis, because it produces better results than other MCM algorithms [40]. Multiplier block creation tool from Spiral implementing Hcub algorithm is used [54]. This tool takes constants to be multiplied as input and produces all necessary shift and add operations in a multiplier block as output. A multiplier block hardware has only one input, and it outputs results of multiplications with all the constants.

The proposed HEVC fractional (half-pixel and quarter-pixel) interpolation hardware for all PU sizes is shown in Figure 4.6. The proposed hardware interpolates all the fractional pixels (half-pixels and quarter-pixels) for the luma component of a PU using integer or half pixels. Four buffers are used to store integer and half pixels necessary for interpolating the half and quarter pixels. The interpolated a, b, c half-pixels are stored in the filtered pixels buffers A, B, C. These on-chip buffers reduce the required off-chip memory bandwidth and power consumption.

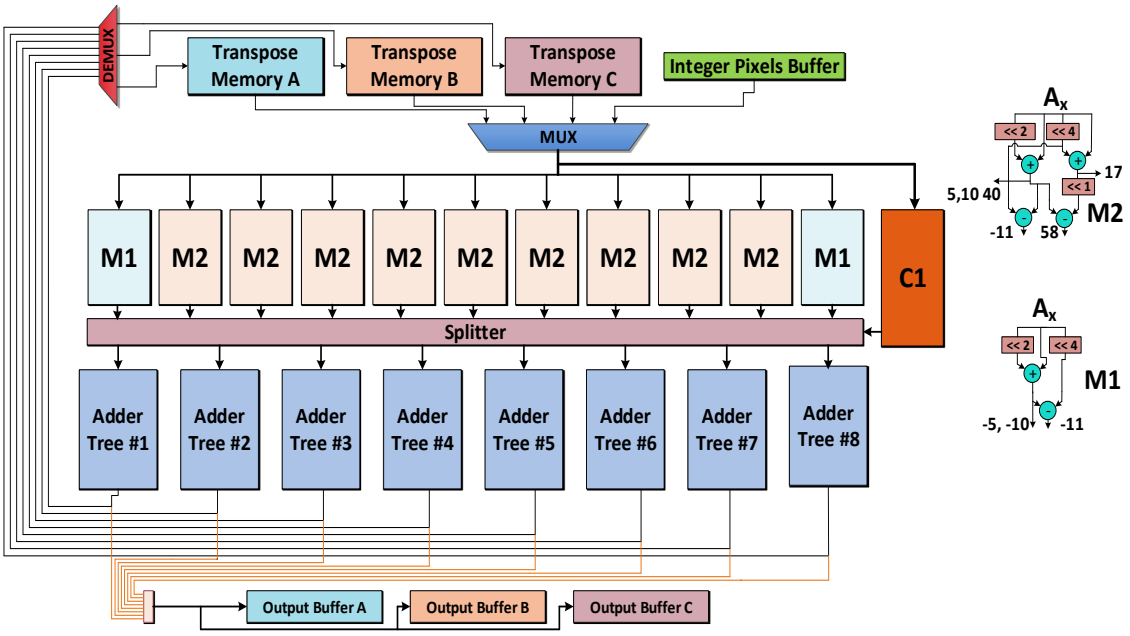

Figure 4.6 Proposed HEVC Fractional Interpolation Hardware (MCM)

8x3=24 fractional pixels are interpolated in parallel using type A, type B and type C FIR filter equations. Common 1 (C1) datapath calculates the common sub-expressions in the equations shown as blue boxes in Figure 4.5. Multiplier 1 (M1) and Multiplier 2 (M2) datapaths calculate the multiplications with multiple constant

coefficients shown as red boxes in Figure 4.5. As shown in Table 4.3, since constant coefficients of input pixels (A-4, A6) and (A-3-A5) are different, two different multiplier block hardware are used. Then, fractional pixels are calculated using adder trees.

Table 4.3 Common Coefficients of Input Pixels

Input Pixel	Coefficient	Datapath
A_{-6}	-1	C1
A_{-5}	-1,4	
A_{-4}	-1,4,-5,-10,-11	M1
A_{-3}	-1,4,-5,-10,-11,17,40,58	
A_{-2}	-1,4,-5,-10,-11,17,40,58	
A_{-1}	-1,4,-5,-10,-11,17,40,58	
A_0	-1,4,-5,-10,-11,17,40,58	
A_1	-1,4,-5,-10,-11,17,40,58	M2
A_2	-1,4,-5,-10,-11,17,40,58	
A_3	-1,4,-5,-10,-11,17,40,58	
A_4	-1,4,-5,-10,-11,17,40,58	
A_5	-1,4,-5,-10,-11,17,40,58	
A_6	-1,4,-5,-10,-11	M1
A_7	-1,4	C1
A_8	-1	

Since 15 fractional pixels should be interpolated for one integer pixel, 64x15 fractional pixels should be interpolated for an 8x8 PU. 8x7 extra a, b, c half-pixels are necessary for the interpolation of quarter pixels. Therefore, the proposed hardware, in the worst case, interpolates the fractional pixels for an 8x8 PU in 48 clock cycles.

First, integer pixels are loaded into integer pixels buffer in one clock cycle. Then, 8x8 d, h, n half-pixels are interpolated and stored in the output buffer. After that, 8x15 a, b and c half-pixels necessary for interpolating quarter pixels are interpolated in 15 clock cycles, and stored in the filtered pixel buffers A, B, and C. Finally, 9x8x8 quarter pixels are interpolated and stored in the output pixel buffers.

The proposed FIHW+MCM HEVC fractional interpolation hardware is implemented using Verilog HDL. The hardware implementation is verified with RTL simulations. The RTL simulation results matched the results of a software model of HEVC fractional interpolation algorithm. The Verilog RTL codes are synthesized and mapped to a Xilinx XC6VLX130T FF1156 FPGA with speed grade 3 using Xilinx ISE 13.4. FIHW+MCM FPGA implementation uses 3929 LUTs, 3422 DFFs, and 6 BRAMs. The proposed FPGA implementation is verified to work at 200 MHz by post place and route simulations. Therefore, it can process 30 quad HD (3840x2160) video frames per second.

The power consumptions of FIHW and FIHW+MCM FPGA implementations are estimated using Xilinx XPower Analyzer tool for Tennis (1920x1080) and Kimono (1920x1080) videos [37]. The energy consumptions of FIHW and FIHW+MCM FPGA implementations are shown in Figure 4.7 and Figure 4.8. As shown in these figures, the proposed HEVC fractional interpolation hardware (FIHW+MCM) has up to 48% less energy consumption than original HEVC fractional interpolation hardware (FIHW).

In order to estimate the power consumption of a fractional interpolation hardware, timing simulation of its placed and routed netlist is done at 100 MHz using Mentor Graphics Questa for encoding one frame of each video sequence. The signal activities of these timing simulations are stored in VCD files, and these VCD files are used for estimating the power consumption of that fractional interpolation hardware using Xilinx XPower Analyzer tool. Since fractional interpolation hardware will be used as part of a HEVC encoder or decoder, only internal power consumption is considered and input and output power consumptions are ignored.

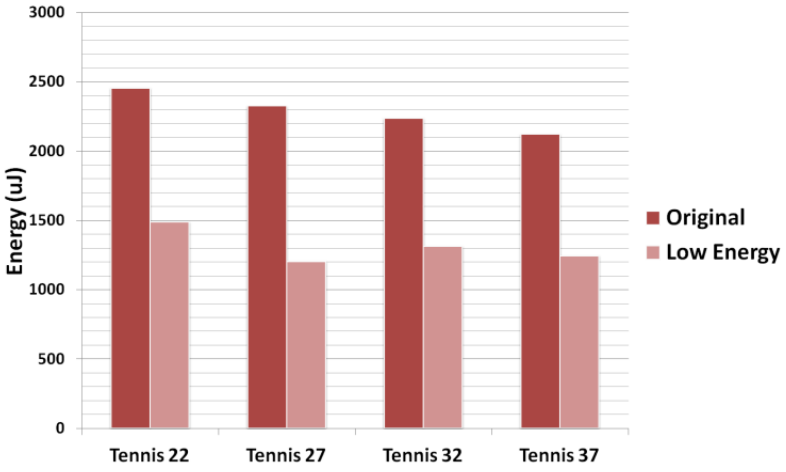

Figure 4.7 Energy Consumption of HEVC Fractional Interpolation Hardware for Tennis (1920x1080) with different QP Values

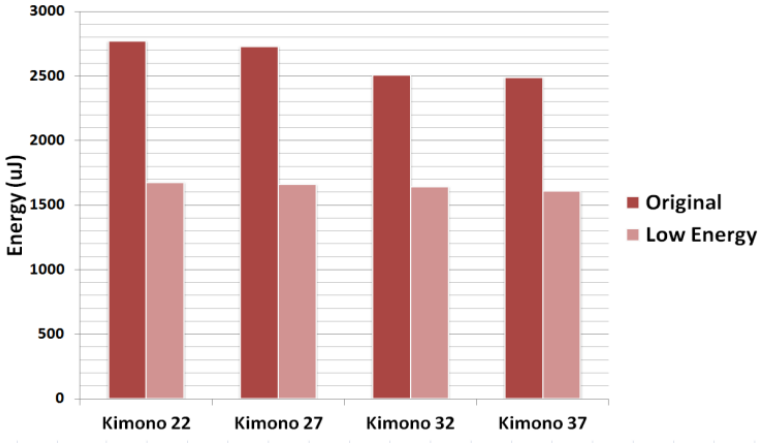

Figure 4.8 Energy Consumption of HEVC Fractional Interpolation Hardware for Kimono (1920x1080) with different QP Values

The Verilog RTL code of the proposed HEVC fractional interpolation hardware is also synthesized and place & routed to Synopsys 90nm standard cell library. The gate count of resulting ASIC implementation is calculated as 28.5k, excluding on-chip memories, based on NAND (2x1) gate area.

4.4 Proposed Approximate HEVC Fractional Interpolation Filters and Their Hardware Implementations

In this thesis, two approximate HEVC fractional interpolation filters (F1 and F2) are proposed. Both F1 and F2 use one 4-tap and two different 3-tap FIR filters instead of using one 8-tap and two different 7-tap FIR filters. The proposed interpolation filters significantly reduce computational complexity of HEVC fractional interpolation with a negligible PSNR loss and bit rate increase. F2 reduces computational complexity more than F1 with more PSNR loss and bit rate increase.

The proposed approximate fractional interpolation filters are used in fractional motion estimation stage of an HEVC encoder. After best fractional motion vector is determined, original HEVC fractional interpolation filter is used in coding stage of the HEVC encoder. Therefore, the proposed approximate fractional interpolation filters do not cause encoder-decoder mismatch.

In this thesis, two approximate HEVC fractional interpolation hardware for all PU sizes are designed and implemented using Verilog HDL for each proposed approximate fractional interpolation filter. The first hardware implements multiplications with constant coefficients using adders and shifters. The second hardware implements

addition and shift operations using Hcub multiplierless constant multiplication (MCM) algorithm. The second hardware for both F1 and F2, in the worst case, can process 45 quad full HD (QFHD) frames per second (fps). They consume up to 67.1% less energy than original HEVC fractional interpolation hardware. F2 fractional interpolation hardware has smaller area and lower energy consumption than F1 fractional interpolation hardware.

Approximate HEVC fractional interpolation filters are proposed in [55]-[56]. However, the approximate HEVC fractional interpolation filters proposed in this thesis have less computational complexity and better rate-distortion performance than the ones proposed in [55]-[56].

4.4.1 Proposed Approximate HEVC Fractional Interpolation Filters

In this thesis, two approximate HEVC fractional interpolation filters (F1 and F2) are proposed. Both F1 and F2 use one 4-tap and two different 3-tap FIR filters. But, they use different filter coefficients. The proposed approximate HEVC fractional interpolation filter equations for F1 and F2 are shown in (4.4)-(4.6) and (4.7)-(4.9), respectively.

$$a_{0,0} = \left(-7 * A_{-1,0} + 58 * A_{0,0} + 13 * A_{1,0}\right) \gg 6 \tag{4.4}$$

$$b_{0,0} = \left(-8 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 8 * A_{2,0}\right) \gg 6 \tag{4.5}$$

$$c_{0,0} = \left(13 * A_{-0,0} + 58 * A_{1,0} - 7 * A_{2,0}\right) \gg 6 \tag{4.6}$$

$$a_{0,0} = \left(-8 * A_{-1,0} + 64 * A_{0,0} + 8 * A_{1,0}\right) \gg 6 \tag{4.7}$$

$$b_{0,0} = \left(-8 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 8 * A_{2,0}\right) \gg 6 \tag{4.8}$$

$$c_{0,0} = \left(8 * A_{0,0} + 64 * A_{1,0} - 8 * A_{2,0}\right) \gg 6 \tag{4.9}$$

In original HEVC FIR filter A, if values of the pixels ($A_{-3,0}$, $A_{-2,0}$, $A_{-1,0}$) multiplied with first three coefficients (-1, 4, -10) are the same, multiplication and addition result can be calculated by multiplying one pixel with -7 (-1+4-10 = -7). In original HEVC fractional interpolation filters, small coefficients have less effect on the filter result. In addition, since the pixels multiplied with small coefficients are neighboring pixels, because of spatial correlation, their values will be very similar. Therefore, the coefficients of F1 are determined by assuming that values of the pixels multiplied with small coefficients are the same. The coefficients of F2 are determined by replacing the

coefficients of F1 with closest 2^n values. In this way, multiplications with coefficients of F2 are performed using shift operations. In addition, F1 and F2 have similar frequency responses with original HEVC fractional interpolation filters.

Table 4.4 shows the number of addition and shift operations necessary for calculating FIR filters used in HEVC fractional interpolation (Original), FIR filters used in the proposed approximate HEVC fractional interpolation (F1 and F2), and FIR filters used in the approximate HEVC fractional interpolation proposed in [55]-[56].

Table 4.4 Addition and Shift Reductions

Filter		**A**		**B**		**C**		**Avg. (%)**
		Add	**Shift**	**Add**	**Shift**	**Add**	**Shift**	
Original		11	8	13	10	11	8	
F1	**Num.**	7	6	5	6	7	6	
	Red. (%)	36.3	25.0	61.5	40.0	36.3	25.0	37.4
F2	**Num.**	2	3	5	6	2	3	
	Red. (%)	81.8	62.5	61.5	40.0	81.8	62.5	65.0
[55]	**Num.**	11	6	11	8	11	6	
	Red. (%)	0.0	25.0	15.4	20.0	0.0	25.0	14.2
[56]	**Num.**	9	6	9	10	9	6	
	Red. (%)	18.2	25.0	30.8	0.0	18.2	25.0	19.5

The proposed approximate HEVC fractional interpolation filters (F1 and F2) are integrated into fractional motion estimation in HEVC HM software encoder 15.0 [39]. First ten frames of some of the HEVC test videos [37] are coded with low delay P (LP) test configuration and with four different quantization parameters (QP) using HEVC HM 15.0 with original HEVC fractional interpolation filters, F1 and F2. The resulting rate-distortion performances are shown in Table 4.5.

Table 4.5 BD-Rate(%) and BD-PSNR(dB)

Video Sequence		F1		F2		[55]		[56]	
		BD-Rate	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**	**BD-Rate**	**BD-PSNR**
2560x1600	**People on Street**	-0.27	0.01	1.13	-0.05	---	---	---	---
	Traffic	0.51	-0.02	1.56	-0.06	---	---	---	---
1920x1080	**Tennis**	-0.01	0.01	0.76	-0.02	---	---	---	---
	Kimono	-0.31	0.01	0.31	-0.01	1.79	-0.06	1.05	-0.03
	Basketball Drive	0.76	-0.01	1.46	-0.03	1.22	-0.03	1.41	-0.03
	Park Scene	0.73	-0.03	1.77	-0.06	2.42	-0.08	3.77	-0.11
1280x720	**Vidyo1**	0.17	-0.01	0.60	-0.02	---	---	---	---
	Vidyo4	0.25	-0.01	0.49	-0.01	---	---	---	---
	Kristen and Sara	0.53	-0.02	1.14	-0.04	3.87	-0.12	4.12	-0.12
	Four People	0.08	0.00	0.48	-0.02	3.25	-0.11	3.02	-0.10
832x480	**Keiba**	0.16	-0.01	1.36	-0.05	---	---	---	---
	BQ Mall	0.79	-0.04	1.36	-0.06	1.69	-0.07	3.73	-0.14
	Race Horses	0.61	-0.03	1.91	-0.09	1.28	-0.05	2.21	-0.08
	Basketball Drill	1.56	-0.06	1.64	-0.07	0.35	-0.01	1.28	-0.05
	Average	**0.40**	**-0.01**	**1.14**	**-0.04**	**1.98**	**-0.07**	**2.57**	**-0.08**

The proposed F1 and F2 filters significantly reduce computational complexity of HEVC fractional interpolation with a negligible PSNR loss and bit rate increase. They have less computational complexity and better rate-distortion performance than the ones proposed in [55]-[56].

4.4.2 Proposed Approximate HEVC Fractional Interpolation Hardware

In this thesis, two approximate HEVC fractional interpolation hardware for all PU sizes are designed for each proposed approximate interpolation filter. The first hardware (AS) implements multiplications with constant coefficients using adders and shifters. In this hardware, three different datapaths are used for implementing A, B and C FIR filters. It interpolates 8x3=24 fractional pixels in parallel using 24 (8 A, 8 B, 8 C) parallel datapaths. The proposed AS approximate HEVC fractional interpolation hardware is shown in Figure 4.9.

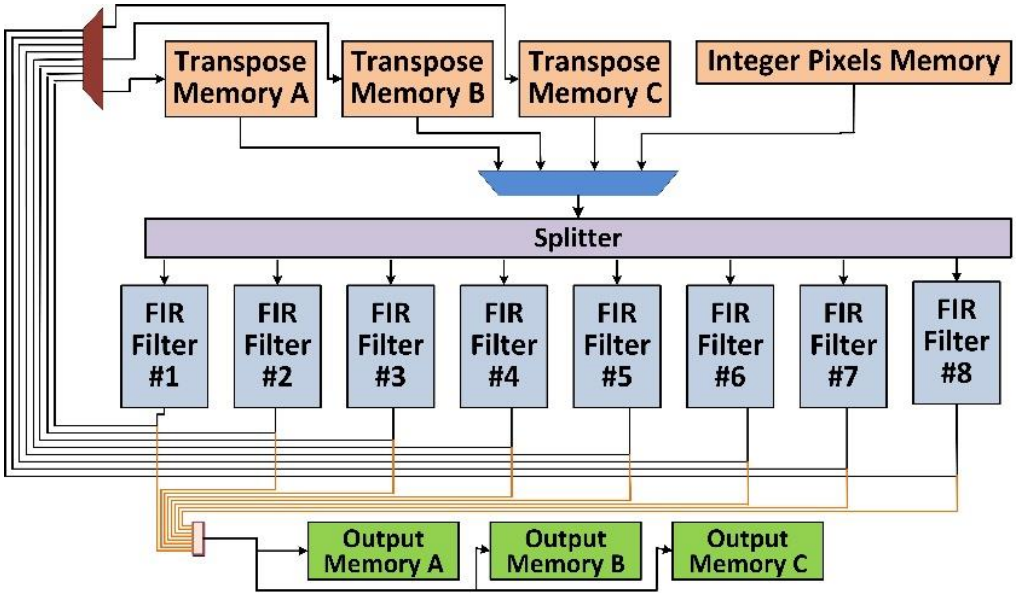

Figure 4.9 Proposed AS Approximate HEVC Fractional Interpolation Hardware

Since different fractional interpolation filter equations multiply same integer pixel with different constant coefficients, in the second hardware (MCM), Hcub MCM algorithm is used for reducing number and size of the adders. A multiplier block (MB) hardware is given one input. It outputs multiplications of this input with all the constants. The proposed MCM approximate HEVC fractional interpolation hardware is shown in Figure 4.10.

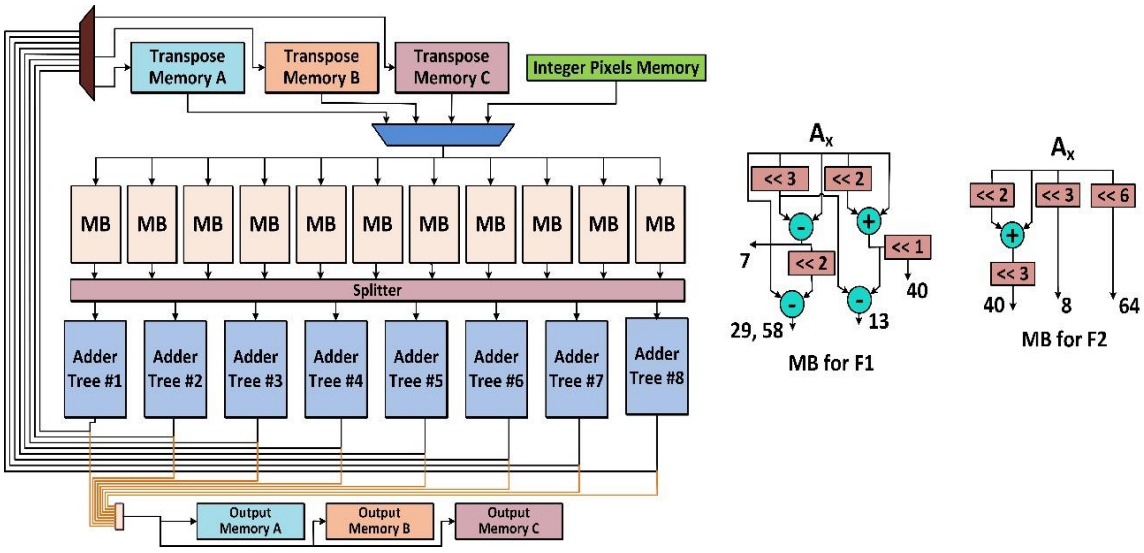

Figure 4.10 Proposed MCM Approximate HEVC Fractional Interpolation Hardware

Integer pixels are stored in one on-chip memory. Then, half pixels (a, b, c) that will be used for interpolating the quarter pixels are stored in three on chip memories. Since a, b, c half pixels are interpolated in horizontal direction and used in vertical direction for quarter pixel interpolations, transpose memory architecture is used to store a, b, c half pixels.

Both proposed MCM hardware implementing the proposed F1 fractional interpolation filter (F1 MCM hardware) and proposed MCM hardware implementing the proposed F2 fractional interpolation filter (F2 MCM hardware) interpolate 8x3=24 fractional pixels in parallel. First, multiplier blocks perform multiplications with constant coefficients. Then, fractional pixels are calculated using adder trees. Since different constant coefficients are used in F1 and F2 filters, different multiplier blocks are used in F1 MCM hardware and F2 MCM hardware.

Since the proposed approximate HEVC fractional interpolation filters F1 and F2 use FIR filters with less number of taps than the original HEVC fractional interpolation filter, the proposed F1 AS, F1 MCM, F2 AS and F2 MCM hardware need to access 11 pixels instead of 15 pixels in order to interpolate 8x3=24 fractional pixels. Therefore, they require less memory accesses than the original HEVC fractional interpolation hardware.

F1 AS, F2 AS, F1 MCM and F2 MCM hardware interpolate the fractional pixels for an 8x8 PU in 44 clock cycles. First, 8x8 half pixels are interpolated. Then, 8x11 half pixels that will be used for interpolating the quarter pixels are interpolated. Finally, 64x9 quarter pixels are interpolated. Scheduling of memory read and interpolation operations in F1 AS, F2 AS, F1 MCM and F2 MCM hardware are shown in Figure 4.11.

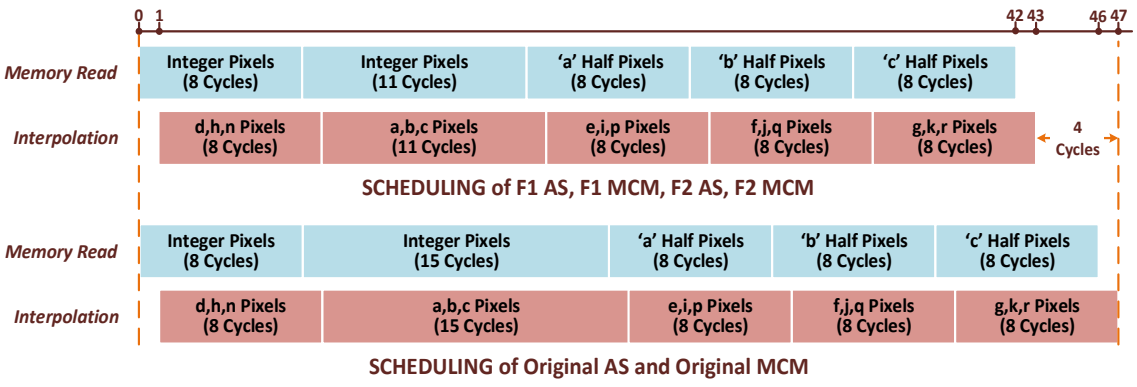

Figure 4.11 Scheduling of HEVC Fractional Interpolation Hardware

The proposed F1 AS, F1 MCM, F2 AS and F2 MCM hardware are implemented using Verilog HDL. The Verilog RTL codes are synthesized, placed and routed to a Xilinx XC6VLX130T FF1156 FPGA. FPGA implementations are verified with both RTL and post place & route timing simulations. The simulation results matched the results of HEVC HM software encoder [39].

FPGA implementations are also verified on an Xilinx ZYNQ ZC702 FPGA board as shown in Figure 4.12. The FPGA board has a 28 nm FPGA and dual-core ARM microprocessor. It also has 1GB DRAM and several interfaces such as UART and HDMI. Microprocessor reads video frames from SD card and sends them to FPGA using a high speed bus. The proposed hardware interpolates the video frames. Then, microprocessor displays interpolated frames on HDMI monitor and stores them to SD card.

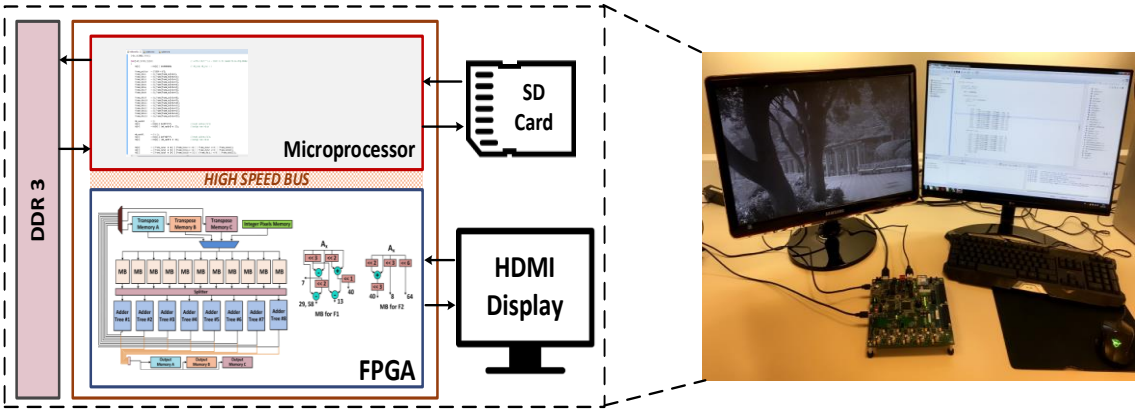

Figure 4.12 Implementation of Proposed Approximate HEVC Fractional Interpolation Hardware on an FPGA Board

FPGA implementation results are shown in Table 4.6. F1 AS implementation can work at 200 MHz, and it can process 33 QFHD (3840x2160) fps. F2 AS implementation can work at 250 MHz, and it can process 41 QFHD fps. F1 MCM and F2 MCM implementations can work at 278 MHz, and they can process 45 QFHD fps. The proposed F1 and F2 approximate HEVC fractional interpolation hardware are faster and smaller than the original HEVC fractional interpolation hardware proposed in [15].

Table 4.6 FPGA Implementation Results

	Original [15]		Proposed F1		Proposed F2	
	AS	MCM	AS	MCM	AS	MCM
Slice	1669	1557	1144	834	963	731
LUT	4110	3929	2416	2008	1601	1567
DFF	3448	3422	2596	3034	1873	2762
BRAM	6	6	6	6	6	6
Freq. (MHz)	200	200	200	278	250	278
Fps	30 Quad Full HD	30 Quad Full HD	33 Quad Full HD	45 Quad Full HD	41 Quad Full HD	45 Quad Full HD
Power Dissip.	152 mW	93 mW	104 mW	88 mW	67 mW	80 mW

The Verilog RTL codes of the proposed F1 AS, F1 MCM, F2 AS and F2 MCM hardware are synthesized, placed and routed to a 90nm standard cell library as well. The gate counts of these ASIC implementations are calculated based on 2x1 NAND gate area. ASIC implementation results are shown in Table 4.7.

Table 4.7 ASIC Implementation Results

	Original [15]		Proposed F1		Proposed F2	
	AS	MCM	AS	MCM	AS	MCM
Tech.	90 nm	90 nm	90 nm	90 nm	90 nm	90 nm
Gate Count	29.5 K	28.5 K	13.2 K	12.8 K	10.6 K	11.2 K
Freq. (MHz)	250	250	300	300	300	300
Fps	37 Quad Full HD	37 Quad Full HD	49 Quad Full HD	49 Quad Full HD	49 Quad Full HD	49 Quad Full HD
Power Dissip.	27.3 mW	23.9 mW	16.4 mW	15.8 mW	14.8 mW	14.9 mW

Power consumptions of F1 AS, F2 AS, F1 MCM and F2 MCM are estimated for Tennis and Kimono (1920x1080) videos [37] using a Xilinx XPower Analyzer tool. Signal activities captured during post place & route timing simulations are used to estimate power consumptions. Energy consumptions of all FPGA implementations are

shown in Figure 4.13. The proposed approximate HEVC fractional interpolation hardware consume up to 67.1% less energy than the original HEVC fractional interpolation hardware proposed in [15].

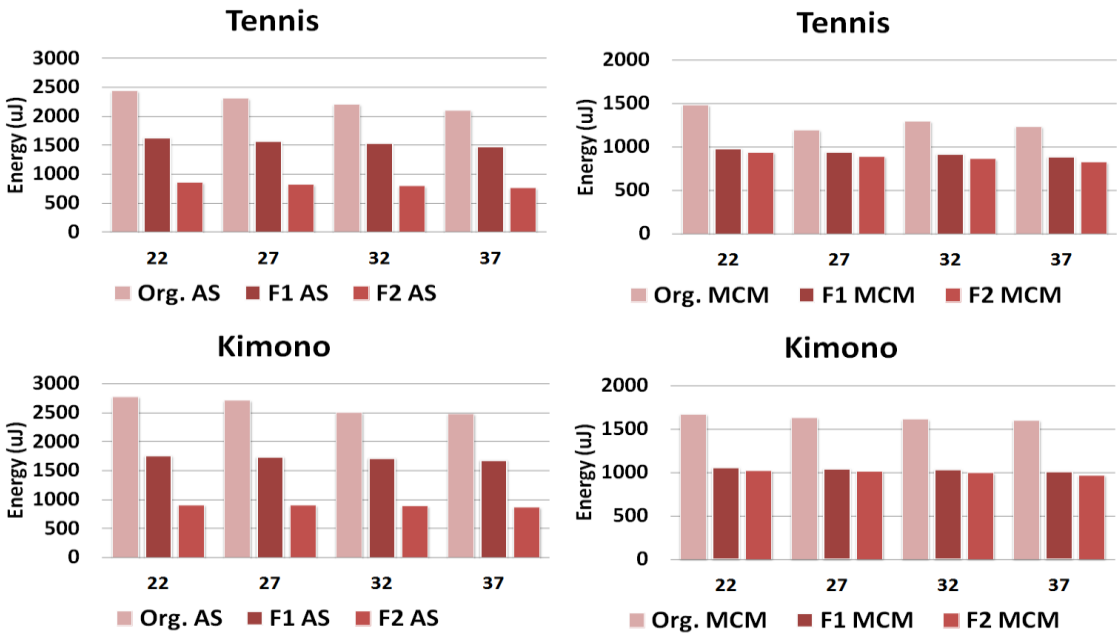

Figure 4.13 Energy Consumption Results

4.5 Hardware Comparison

The proposed FIHW, FIHW+PECR, FIHW+PSCR+3bT, FIHW+MCM, F1 AS, F1 MCM, F2 AS and F2 MCM FPGA implementations are compared in Table 4.8. The proposed approximate hardware implementations have higher performance than other hardware implementations because they need less clock cycles to interpolate one 8x8 PU. FIHW+PSCR+3bT has smaller area than other hardware implementations since it uses most significant 5-bits of integer and half pixels for interpolation. However, it has the worst rate distortion performance.

Table 4.8 Comparisons of The Proposed FPGA Implementations

	FIHW	FIHW+ PECR	FIHW+ PSCR+3bT	FIHW+ MCM	F1 AS	F1 MCM	F2 AS	F2 MCM
FPGA	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6
DFF Count	3448	3408	849	3422	2596	3034	1873	2762
LUT Count	4110	4577	2381	3929	2416	2008	1601	1567
Max. Freq. (MHz)	200	200	200	200	200	278	250	278
Fps	30 QFHD	30 QFHD	30 QFHD	30 QFHD	33 QFHD	45 QFHD	41 QFHD	45 QFHD

The proposed approximate HEVC fractional interpolation hardware are compared with the HEVC fractional interpolation hardware proposed in the literature [57]-[65]. The comparisons of ASIC and FPGA implementations are shown in Table 4.9 and Table 4.10, respectively. Some of the results are not given in Table 4.9 and Table 4.10, because they are not available in the literature [57]-[63].

Table 4.9 Comparisons of ASIC Implementations

	[15]	[57]	[58]	[59]	[60]	[61]	[62]	F1	F2
Tech.	90 nm	150 nm	90 nm	90 nm	90 nm	130 nm	40 nm	90 nm	90 nm
Gate Count	28.5 K	30.2 K	224 K	383 K	37.2 K	126.8 K	297.3 K	12.8 K	11.2 K
Max. Freq. (MHz)	200	312	333	192	240	208	342	300	300
Fps	30 QFHD	30 QFHD	30 FHD	60 QFHD	47 QFHD	86 QFHD	60 UHD	49 QFHD	49 QFHD
Power Dissip.	23.9 mW	---	---	---	---	---	48.1 mW	15.8 mW	14.9 mW

Table 4.10 Comparisons of FPGA Implementations

	[15]	[59]	[63]	[64]	[65]	F1	F2
FPGA	Xilinx Virtex 6	Xilinx Virtex 5	Altera Arria II	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6
Slice Count	1557	---	---	2181	1498	834	731
LUT Count	3929	28486	24202	5017	3806	2008	1567
Max. Freq. (MHz)	200	120	200	283	233	278	278
Fps	30 3840x2160	---	60 1920x1080	30 2560x1600	35 3840x2160	45 3840x2160	45 3840x2160
Power Dissipation	93 mW	---	171 mW	89 mW	---	88 mW	80 mW

A coarse grained reconfigurable datapath is proposed to reduce area and adaptive scheduling is proposed to increase throughput in [57]. A fractional interpolation hardware is proposed for HEVC encoder in [58]. Data-reuse technique is used to reduce memory accesses and highly-parallel architecture is used to increase throughput in [59]. Efficient memory organization and reuse of datapath are proposed in [60]. Resource sharing for common partial terms of the interpolation filters is proposed in [61]. A fractional interpolation hardware is proposed for motion compensation in [62]. Many parallel interpolation hardware are used in [63]. Reconfigurable interpolation datapaths are used to reduce area and power consumption in [64]. [65] uses memory based constant multiplication technique for implementing multiplication with constant coefficients.

The proposed approximate HEVC fractional interpolation hardware have much smaller hardware area and lower power consumption than the other hardware. The smallest hardware in the literature has more than two times larger area than the proposed hardware. Only the HEVC fractional interpolation hardware proposed in [59], [61]-[62] have higher throughput than them. However, they have more than ten times larger area than the proposed hardware. In addition, performance result of the hardware proposed in [62] is given for motion compensation. Performance results of the rest of the hardware including the ones proposed in this thesis are given for motion estimation.

CHAPTER V

A COMPUTATION AND ENERGY REDUCTION TECHNIQUE FOR HEVC DISCRETE COSINE TRANSFORM

HEVC standard uses Discrete Cosine Transform (DCT) / Inverse Discrete Cosine Transform (IDCT) same as the H.264 standard. However, H.264 standard uses only 4x4 and 8x8 Transform Unit (TU) sizes for DCT/IDCT. HEVC standard uses 4x4, 8x8, 16x16, and 32x32 TU sizes for DCT/IDCT. Larger TU sizes achieve better energy compaction. However, they increase the computational complexity exponentially. In addition, HEVC uses Discrete Sine Transform (DST) / Inverse Discrete Sine Transform (IDST) for 4x4 intra prediction in certain cases.

Transform operations (DCT/IDCT and DST/IDST) are heavily used in an HEVC encoder [11]. DCT and DST have high computational complexity. DCT and DST operations account for 11% of the computational complexity of an HEVC video encoder. They account for 25% of the computational complexity of an all intra HEVC video encoder.

HEVC uses DCT-II and DST-VII. It uses 4x4, 8x8, 16x16, 32x32 TU sizes for DCT. It also uses DST for 4x4 intra prediction in certain cases. HEVC performs 2D transform operation by applying 1D transforms in vertical and horizontal directions. The coefficients in HEVC 1D transform matrices are derived from DCT-II and DST-VII

basis functions. However, integer coefficients are used for simplicity. HEVC 1D DCT-II and DST-VII matrices for 4x4 TU size are shown in (5.1) and (5.2).

$$DCT - II_{4x4} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \tag{5.1}$$

$$DST - VII_{4x4} = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix} \tag{5.2}$$

In this thesis, a novel computation and energy reduction technique for HEVC DCT for all TU sizes is proposed. After forward transform and quantization, most of the forward transformed and quantized high frequency coefficients in a TU become zero. In addition, if the values of non-zero forward transformed and quantized low frequency coefficients in a TU are small, they have small impact on the inverse quantized and inverse transformed TU. Therefore, the proposed technique only calculates several pre-determined low frequency coefficients of TUs, and it assumes that the remaining coefficients are zero.

The proposed technique is used in both mode decision and coding stages of an HEVC encoder. Since the same DCT coefficients are used in both HEVC encoder and HEVC decoder, the proposed technique does not cause any encoder-decoder mismatch. The proposed technique does not require any modification in an HEVC decoder. The proposed technique reduces the computational complexity of HEVC DCT significantly at the expense of slight decrease in PSNR and slight increase in bit rate. It reduced the execution time of HEVC HM software encoder [39] up to 12.74%, and it reduced the execution time of the DCT operations in HEVC HM software encoder up to 37.27% on a workstation with 3.33 GHz dual-core processor and 64 GB DRAM.

In this thesis, a low energy HEVC 2D DCT hardware for all TU sizes is also designed and implemented using Verilog HDL. The proposed hardware calculates 4, 8, 16 and 32 DCT coefficients per clock cycle for 4x4, 8x8, 16x16 and 32x32 TU sizes, respectively. It, in the worst case, can process 48 Quad Full HD (3840x2160) video frames per second. In this thesis, another low energy HEVC 2D DCT hardware for all TU sizes with higher hardware utilization is also designed and implemented using

Verilog HDL. This hardware processes four 4x4 TUs or two 8x8 TUs in parallel. Therefore, it can calculate 16 DCT coefficients per clock cycle for 4x4, 8x8 and 16x16 TU sizes, and 32 DCT coefficients per clock cycle for 32x32 TU size. It, in the worst case, can process 53 Ultra HD (7680x4320) video frames per second.

Clock gating is used to reduce the energy consumptions of both hardware. Hcub Multiplierless Constant Multiplication (MCM) algorithm [40] is used to reduce number and size of the adders in both hardware. Hcub MCM algorithm reduced the energy consumption of the lower utilization (LU) hardware and the higher utilization (HU) hardware up to 5.9% and 13.1%, respectively. Finally, the proposed technique is used to reduce the energy consumptions of both hardware. It further reduced the energy consumption of the LU hardware and the HU hardware up to 17.9% and 18.9%, respectively.

Several zero quantized DCT coefficient detection techniques are proposed for H.264 and HEVC [66]-[69]. These techniques try to predict the blocks with zero forward transformed and quantized coefficients before DCT and quantization operations in the coding stage of an H.264 or HEVC encoder in order to avoid DCT and quantization operations. However, the technique proposed in this thesis avoids most of the DCT operations that have no impact or low impact on the transformed and quantized TUs in both mode decision and coding stages of an HEVC encoder. In addition, the zero quantized DCT coefficient detection techniques have much more computational overhead than the proposed technique which requires only one comparison for each TU.

Several HEVC DCT hardware are proposed in the literature [70]-[74]. In [70], 2D DCT hardware calculates all DCT outputs using multipliers. In [71], 2D DCT hardware reuses smaller TU hardware for DCT operations of larger TUs. In [72], 2D DCT hardware implementation uses two different 1D transform hardware for column and row transforms, and it can process 32 pixels per clock cycle for all TU sizes. In [73], 2D DCT hardware calculates all DCT outputs using multipliers, and it modifies the order of TU processing for optimizing transform buffer. In [74], the proposed hardware only performs 1D DCT transform, and it uses canonical signed digit representation and common sub-expression elimination technique to decrease number of adders and shifters. The low energy HEVC 2D DCT hardware proposed in this thesis is compared with these HEVC DCT hardware in Section 5.3.

5.1 Proposed Computation and Energy Reduction Technique

After forward transform and quantization, most of the forward transformed and quantized high frequency coefficients in a TU become zero. In addition, if the values of non-zero forward transformed and quantized low frequency coefficients in a TU are small, they have small impact on the inverse quantized and inverse transformed TU. Therefore, the proposed technique only calculates several pre-determined low frequency coefficients of TUs, and it assumes that the remaining coefficients are zero.

As shown in Figure 5.1, in this thesis, the impact of the proposed technique on the computational complexity and rate-distortion performance is determined for three different DCT coefficient sets. In the first two coefficient sets, the coefficients that will be calculated by the HEVC DCT for all TU sizes are pre-determined, and they are not changed during DCT operations. When the proposed technique is used with coefficient set 1, only 25% (1/4) of DCT coefficients are calculated for all TU sizes. When the proposed technique is used with coefficient set 2, 56.25% (9/16) of DCT coefficients are calculated for 4x4 and 16x16 TU sizes, and 14% (9/64) of DCT coefficients are calculated for 8x8 and 32x32 TU sizes. These DCT coefficient percentages are experimentally determined to reduce the computational complexity of HEVC DCT significantly with slight impact on distortion and bit rate.

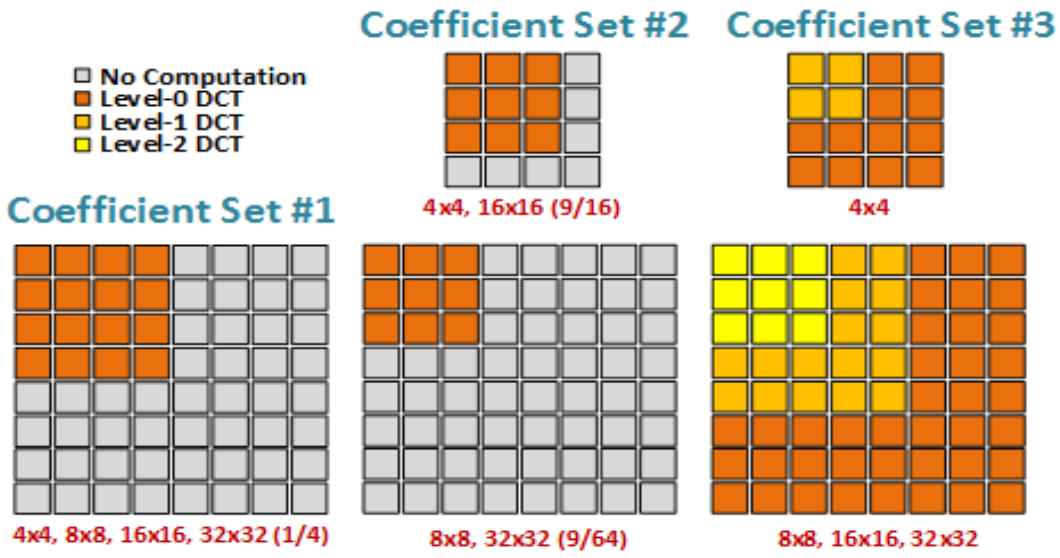

Figure 5.1 Proposed Computation and Energy Reduction Technique

In the coefficient set 3, the pre-determined coefficients that will be calculated by HEVC DCT for all TU sizes are adaptively changed during DCT operations. For 4x4 TUs, level-0 or level-1 DCT is performed. For the other TUs, level-0, level-1 or level-2 DCT is performed. In level-0 DCT, all DCT coefficients are calculated for all TUs. In level-1 DCT, 25% (1/4) of DCT coefficients are calculated for 4x4 TUs, and 39% (25/64) of DCT coefficients are calculated for 8x8, 16x16 and 32x32 TUs. In level-2 DCT, 14% (9/64) of DCT coefficients are calculated for 8x8, 16x16 and 32x32 TUs.

Initially, level-0 DCT is used for each TU size. As shown in Figure 5.2, if the distortion value for current TU obtained by the current DCT operation is smaller than 90% of the previous distortion value for the same TU or same size TU obtained by the previous DCT operation, DCT level for this TU size is increased. If the distortion value for current TU obtained by the current DCT operation is larger than 110% of the previous distortion value for the same TU or same size TU obtained by the previous DCT operation, DCT level for this TU size is decreased.

```
DCT(Residuals, Distortion)  {

   if (Distortion(curr_dct) is larger than 1.1*Distortion(prev_dct) and
       DCT_Level is larger than zero)
       DCT_Level  ← (DCT_Level - 1)

   else if (Distortion(curr_dct) is smaller than 0.9*Distortion(prev_dct)
            and DCT_Level is smaller than two)
       DCT_Level  ← (DCT_Level + 1)

   if (DCT_Level is zero)
       DCT Coefficients  ← DCT_L0(Residuals)
   else if (DCT_Level is one)
       DCT Coefficients  ← DCT_L1(Residuals)
   else if (DCT_Level is two)
       DCT Coefficients  ← DCT_L2(Residuals)
}
```

Figure 5.2 Pseudocode of HEVC DCT with The Proposed Technique

Since the distortion value for current TU is already calculated by an HEVC encoder, the proposed technique does not calculate the distortion value for current TU. When the proposed technique is used with coefficient set 3, the percentages of DCT levels used for all TUs for first 10 frames of three different full HD (1920x1080) videos are shown in Figure 5.3.

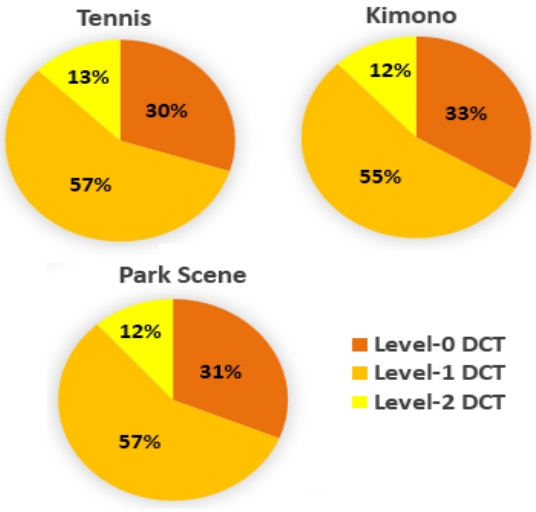

Figure 5.3 DCT Level Percentages

Table 5.1 shows the number of addition and shift operations required for calculating all DCT coefficients in a TU (Original) and for calculating the pre-determined DCT coefficients in a TU for three different DCT coefficient sets. Calculating only the pre-determined DCT coefficients in a TU significantly reduces the number of addition and shift operations.

Table 5.1 Addition and Shift Reductions for All TU Sizes

TU Size		**Org.**	**C. Set #1**	**Red. (%)**	**C. Set #2**	**Red. (%)**	**Coefficient Set #3**			
							Level 1	**Red. (%)**	**Level 2**	**Red. (%)**
4x4	**Add.**	224	84	**62.5**	147	**34.4**	84	**62.5**	--	--
	Shift	224	84	**62.5**	147	**34.4**	84	**62.5**	--	--
8x8	**Add.**	2560	960	**62.5**	660	**74.2**	1300	**49.2**	660	**74.2**
	Shift	2304	864	**62.5**	594	**74.2**	1170	**49.2**	594	**74.2**
16x16	**Add.**	20992	7872	**62.5**	13776	**34.4**	10660	**49.2**	5412	**74.2**
	Shift	16896	6336	**62.5**	11088	**34.4**	8580	**49.2**	4356	**74.2**
32x32	**Add.**	182272	68352	**62.5**	46992	**74.2**	92560	**49.2**	46992	**74.2**
	Shift	153600	57600	**62.5**	39600	**74.2**	78000	**49.2**	39600	**74.2**
Average				**62.5**		**54.3**		**52.5**		**74.2**

The proposed technique is integrated into DCT operations performed by HEVC HM software encoder [39]. The pre-determined DCT coefficients are experimentally determined to achieve large computation reduction with slight decrease in PSNR and slight increase in bit rate using HEVC HM software encoder. The impact of the proposed technique on execution time and rate-distortion performance is determined for three different DCT coefficient sets on a workstation with 3.33 GHz dual-core processor

and 64 GB DRAM for People on Street, Traffic (2560x1600), Tennis, Kimono, Basketball Drive, Park Scene (1920x1080), Vidyo1, Vidyo4, Kristen and Sara, Four People (1280x720), Keiba, Party Scene, Race Horses, Basketball Drill (832x480) videos [37].

First 10 frames of all video sequences are coded with all intra (AI), low delay P (LP) (IPPPP) and random access (RA) (IBBBB) test configurations and with quantization parameters (QP) 22, 27, 32 and 37 using HEVC HM software encoder [39] with and without the proposed technique, and BD-Rate and BD-PSNR values are calculated. The results given in Tables 5.2, 5.3 and 5.4 show that the proposed technique reduces the computational complexity of HEVC DCT significantly at the expense of slight decrease in PSNR and slight increase in bit rate. Since it is used in mode decision stage of an HEVC encoder, it achieves different amount of execution time reductions for DCT operations and HEVC encoder.

Table 5.2 BD-Rate, BD-PSNR and Execution Time Results for HEVC All Intra (AI) Configuration

	Coefficient Set #1				**Coefficient Set #2**				**Coefficient Set #3**			
Video Sequence	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**
People on Street	1.32	-0.07	4.14	-22.05	2.09	-0.11	-4.10	-24.25	1.89	-0.10	-6.11	-23.81
Traffic	0.82	-0.04	-3.67	-20.78	1.68	-0.09	-3.68	-23.40	1.76	-0.09	7.21	-24.75
Tennis	3.18	-0.10	-4.48	-20.55	3.11	-0.09	-3.85	-22.45	2.32	-0.06	-8.07	-22.86
Kimono	2.06	-0.07	-4.94	-21.13	1.89	-0.06	-1.79	-23.05	1.24	-0.04	-7.21	-23.38
Basketball Drive	5.63	-0.19	-5.64	-21.05	4.17	-0.16	-4.07	-23.85	4.06	-0.13	-9.77	-24.22
Park Scene	2.88	-0.12	-6.02	-20.25	2.28	-0.09	-4.50	-22.40	2.52	-0.10	-8.83	-24.28
Vidyo1	2.73	-0.13	-3.53	-21.10	2.21	-0.10	-2.82	-24.13	2.09	-0.09	-7.67	-24.24
Vidyo4	3.28	-0.17	-4.32	-20.93	2.84	-0.12	-1.85	-23.55	2.85	-0.12	-8.44	-24.23
Kristen And Sara	3.37	-0.20	-5.18	-22.05	2.11	-0.10	-2.84	-23.38	2.25	-0.11	-10.14	-23.98
Four People	2.82	-0.16	-3.79	-21.33	2.56	-0.14	-2.64	-23.05	2.50	-0.14	-7.48	-24.17
Keiba	3.69	-0.18	-2.09	-21.23	3.20	-0.18	-3.60	-24.33	3.18	-0.15	-7.41	-22.99
Party Scene	-0.94	0.07	-13.48	-20.68	0.90	-0.07	-11.02	-23.15	0.61	-0.05	-11.21	-21.38
Race Horses	1.26	-0.08	-6.05	-20.63	2.31	-0.14	-5.22	-22.80	1.58	-0.10	-9.29	-24.25
Basketball Drill	-1.63	0.08	-6.42	-21.53	-0.10	0.01	-5.84	-23.15	0.44	-0.02	-9.34	-24.20
Average	**2.17**	**-0.09**	**-5.27**	**-21.09**	**2.23**	**-0.10**	**-4.13**	**-23.35**	**2.09**	**-0.09**	**-8.44**	**-23.77**

Table 5.3 BD-Rate, BD-PSNR and Execution Time Results for HEVC Low Delay P (LP) Configuration

	Coefficient Set #1				**Coefficient Set #2**				**Coefficient Set #3**			
Video Sequence	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**
People on Street	1.61	-0.07	-3.10	-28.35	1.75	-0.08	-3.23	-40.59	1.94	-0.09	-7.25	-32.08
Traffic	1.54	-0.06	-4.30	-24.45	2.10	-0.08	-4.08	-45.72	2.53	-0.10	-8.24	-30.65
Tennis	1.77	-0.05	-3.30	-32.13	1.59	-0.05	-3.36	-44.54	1.66	-0.05	-7.23	-35.58
Kimono	1.51	-0.05	-4.11	-31.60	0.98	-0.03	-2.95	-37.41	0.58	-0.02	-7.64	-36.33
Basketball Drive	4.48	-0.16	-5.51	-29.65	3.79	-0.12	-5.13	-38.68	3.01	-0.10	-8.87	-37.27
Park Scene	2.80	-0.09	-6.67	-27.48	2.09	-0.07	-3.51	-42.40	2.56	-0.08	-9.36	-30.13
Vidyo1	2.97	-0.12	-5.56	-20.38	1.86	-0.07	-5.30	-52.03	2.39	-0.09	-7.59	-31.11
Vidyo4	3.93	-0.14	-5.81	-20.85	3.43	-0.11	-2.38	-51.46	3.20	-0.09	-7.61	-32.62
Kristen And Sara	4.07	-0.16	-5.23	-19.90	2.60	-0.11	-2.87	-51.91	2.62	-0.10	-7.44	-30.84
Four People	2.89	-0.14	-4.03	-20.63	2.48	-0.13	-3.50	-53.30	2.52	-0.11	-7.82	-29.37
Keiba	6.01	-0.37	-7.87	-21.10	5.54	-0.31	-4.93	-36.66	3.08	-0.17	-9.80	-32.41
Party Scene	1.31	-0.09	-12.62	-20.28	2.01	-0.13	-10.21	-44.13	1.34	-0.08	-12.74	-25.84
Race Horses	4.57	-0.22	-8.32	-20.08	3.84	-0.20	-5.35	-25.58	2.27	-0.14	-10.04	-31.42
Basketball Drill	-0.20	0.01	-7.87	-20.48	1.04	-0.04	-5.01	-42.88	1.95	-0.08	-8.44	-32.83
Average	**2.80**	**-0.12**	**-6.02**	**-24.10**	**2.50**	**-0.10**	**-4.41**	**-43.37**	**2.26**	**-0.09**	**-8.58**	**-32.04**

Table 5.4 BD-Rate, BD-PSNR and Execution Time Results for HEVC Random Access (RA) Configuration

	Coefficient Set #1				Coefficient Set #2				Coefficient Set #3			
Video Sequence	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**	**BD-Rate (%)**	**BD-PSNR (dB)**	**ΔTime (%) (Enc.)**	**ΔTime (%) (DCT)**
People on Street	1.33	-0.06	-3.93	-39.29	1.54	-0.07	-3.27	-33.77	1.64	-0.07	-4.47	-31.02
Traffic	0.84	-0.03	-4.04	-16.32	1.70	-0.07	-4.30	-27.30	1.82	-0.08	-2.01	-18.47
Tennis	2.19	-0.07	-4.50	-40.03	1.77	-0.05	-3.03	-38.20	1.58	-0.04	-6.32	-32.47
Kimono	1.58	-0.05	-3.22	-44.48	1.31	-0.04	-2.07	-35.88	0.77	-0.02	-6.86	-33.21
Basketball Drive	4.94	-0.16	-4.35	-42.48	4.37	-0.13	-1.88	-33.48	4.17	-0.09	-7.40	-33.17
Park Scene	2.81	-0.09	-5.76	-13.97	2.27	-0.07	-3.93	-27.25	2.60	-0.09	-7.59	-27.14
Vidyo1	3.11	-0.13	-4.19	-22.16	2.85	-0.10	-3.64	-25.58	2.72	-0.09	-6.50	-26.46
Vidyo4	3.38	-0.12	-3.52	-21.70	2.55	-0.08	-3.88	-26.52	2.75	-0.09	-6.78	-29.24
Kristen And Sara	3.11	-0.14	-3.95	-22.93	2.00	-0.07	-1.17	-23.60	1.58	-0.06	-6.85	-25.32
Four People	2.79	-0.14	-4.24	-23.99	2.35	-0.11	-2.52	-23.53	2.60	-0.11	-6.90	-24.70
Keiba	8.76	-0.39	-5.12	-15.52	6.07	-0.22	-5.72	-37.00	5.92	-0.21	-6.45	-34.72
Party Scene	0.03	0.01	-10.86	-20.76	1.30	-0.08	-8.34	-29.88	0.92	-0.06	-9.65	-24.54
Race Horses	3.56	-0.15	-6.34	-12.86	3.14	-0.14	-5.17	-34.53	2.02	-0.09	-7.28	-29.96
Basketball Drill	-0.97	0.05	-4.07	-23.24	0.23	-0.01	-2.72	-31.28	1.11	-0.05	-6.64	-31.17
Average	**2.67**	**-0.10**	**-4.86**	**-25.70**	**2.38**	**-0.08**	**-3.69**	**-30.56**	**2.30**	**-0.08**	**-6.70**	**-28.69**

Since the proposed technique with coefficient set 3 achieved the best execution time, BD-PSNR and BD-Rate results, it is selected for hardware implementation. The proposed technique with coefficient set 3 reduced the execution time of HEVC HM software encoder, on the average, 8.44%, 8.58%, 6.70% for AI, LP, RA configurations, respectively. It reduced the execution time of DCT operations in HEVC HM software encoder, on the average, 23.77%, 32.04%, 28.69% for AI, LP, RA configurations, respectively.

5.2 Proposed HEVC 2D DCT Hardware

5.2.1 Proposed HEVC 2D DCT Lower Utilization Hardware

The proposed HEVC 2D DCT lower utilization (LU) hardware for all TU sizes including clock gating, Hcub MCM algorithm, and the proposed technique with coefficient set 3 is shown in Figure 5.4. Input splitter is used to select the proper DCT inputs for each TU size. Output multiplexers are used to select the proper DCT outputs for each TU size. Column and row clip modules are used to scale the outputs of 1D column DCT and 1D row DCT to 16 bits, respectively. Column clip shifts 1D column DCT outputs right by 1, 2, 3 and 4 for 4x4, 8x8, 16x16 and 32x32 TU sizes, respectively. Row clip shifts 1D row DCT outputs right by 8, 9, 10 and 11 for 4x4, 8x8, 16x16 and 32x32 TU sizes, respectively.

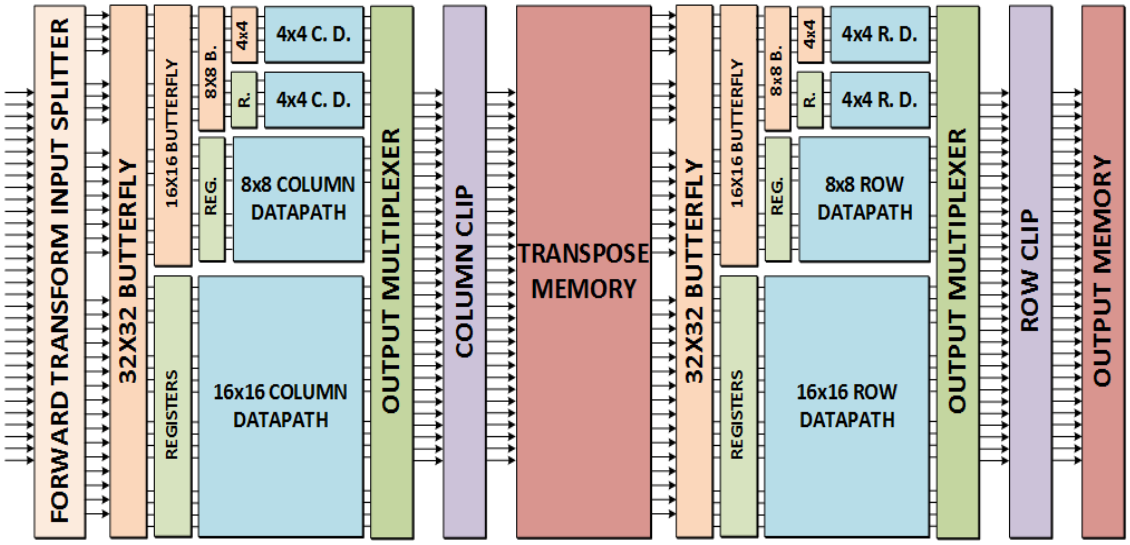

Figure 5.4 Proposed HEVC 2D DCT Lower Utilization Hardware

Since HEVC DCT algorithm allows performing an N-point 1D DCT by performing two N/2-point 1D DCTs with some preprocessing, the proposed hardware performs N-point 1D DCT transforms by performing two N/2-point 1D DCT transforms with an efficient butterfly structure. It performs 2D DCT by first performing 1D DCT transform on the columns of a TU, and then performing 1D DCT transform on the rows of the TU. After 1D column DCT, the resulting coefficients are stored in a transpose memory, and they are used as input for 1D row DCT.

The butterfly structure used for column transforms is shown in Figure 5.5. For 4x4 TUs, only 4x4 butterfly operation is used. For 8x8 TUs, 8x8 and 4x4 butterfly operations are used. For 16x16 TUs, 16x16, 8x8 and 4x4 butterfly operations are used. For 32x32 TUs, all butterfly operations (32x32, 16x16, 8x8, 4x4) are used.

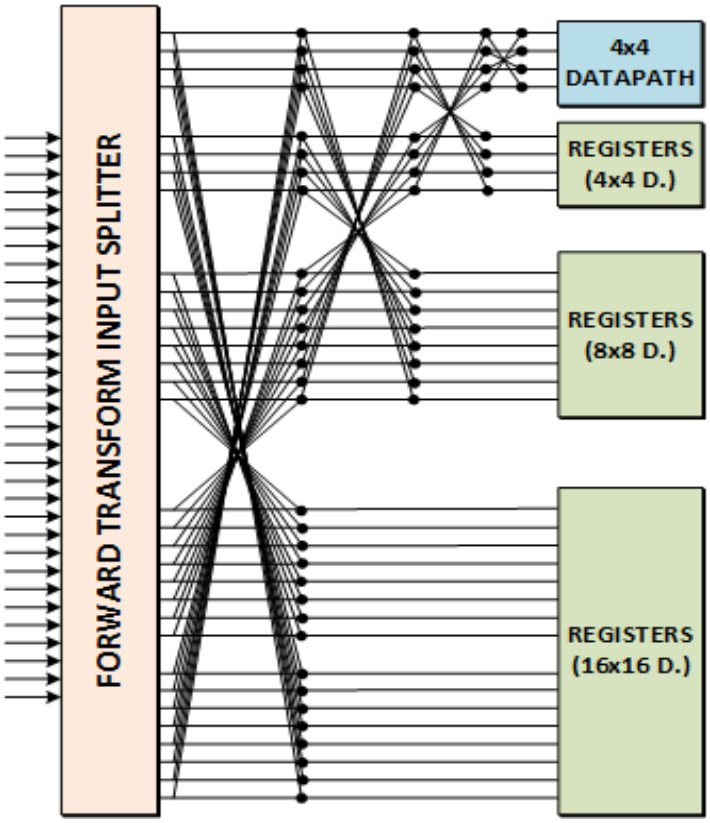

Figure 5.5 Column Butterfly Structure

One 4x4 datapath is used for 4x4 TU size. Two 4x4 datapaths are used for 8x8 TU size. Two 4x4 datapaths and one 8x8 datapath are used for 16x16 TU size. All datapaths (two 4x4, one 8x8 and one 16x16) are used for 32x32 TU size. In order to reduce the power consumption of proposed hardware, data gating is used for the inputs of 4x4, 8x8 and 16x16 column and row datapaths. The inputs of these datapaths are stored into registers. If a datapath is not used for a TU, its input registers are not updated. This prevents unnecessary switching activities in this datapath.

DCT multiplications are performed in the datapaths using only adders and shifters. In order to reduce number and size of the adders in the proposed hardware, Hcub MCM algorithm [40] is used for implementing multiplications with constants. Hcub algorithm tries to minimize number and size of the adders in a multiplier block

which multiplies a single input with multiple constants using shift and addition operations. Hcub algorithm determines necessary shift and addition operations in a multiplier block.

Since different constants are used in 2D DCT for 4x4, 8x8, 16x16 and 32x32 TU sizes, four different multiplier blocks are used in the proposed hardware. Multiplier block for second 4x4 column datapath is shown in Figure 5.6. Multiplier blocks in the first 4x4, second 4x4, 8x8 and 16x16 datapaths multiply a single input with 3, 4, 8 and 16 different constants, respectively. There are 4, 8 and 16 multiplier blocks in 4x4, 8x8 and 16x16 datapaths, respectively. When level-1 or level-2 DCT is performed for a TU, multiplier block outputs used for calculating the DCT coefficients that are assumed as zero by the proposed technique are assigned to zero.

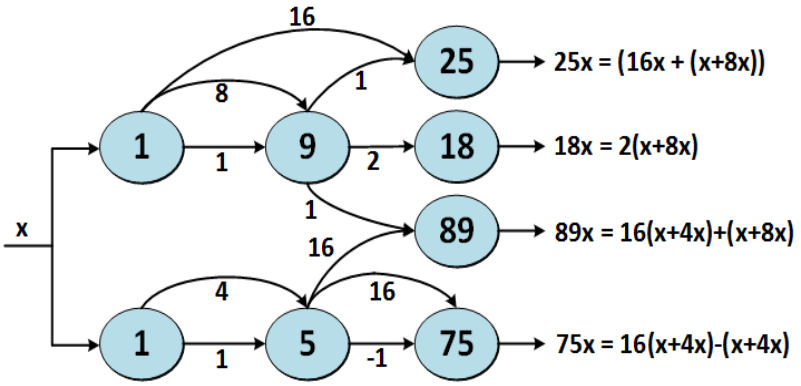

Figure 5.6 Multiplier Block in HEVC 2D DCT Lower Utilization Hardware

In order to calculate each output of 1D DCT for 4x4 TU size, an output from each multiplier block in a 4x4 datapath is selected, and these outputs are added or subtracted. In order to calculate each output of 1D DCT for 8x8 TU size, an output from each multiplier block in both 4x4 datapaths is selected, and these outputs are added or subtracted. Similarly, in order to calculate each output of 1D DCT for 16x16 TU size, 16 outputs from 16 multiplier blocks in two 4x4 datapaths and one 8x8 datapath are added or subtracted. Similarly, in order to calculate each output of 1D DCT for 32x32 TU size, 32 outputs from 32 multiplier blocks in all datapaths (two 4x4, one 8x8 and one 16x16) are added or subtracted.

As shown in Figure 5.7, the transpose memory is implemented using 32 Block RAMs (BRAM). 4, 8, 16 and 32 BRAMs are used for 4x4, 8x8, 16x16 and 32x32 TU sizes, respectively. In the figure, the numbers in each box show the BRAM that

coefficient is stored. The results of 1D column DCT are generated column by column. For 32x32 TU size, first, the coefficients in column 0 (C0) are generated in a clock cycle and stored in 32 different BRAMs. Then, the coefficients in column 1 (C1) are generated in the next clock cycle and stored in 32 different BRAMs using a rotating addressing scheme. This continuous until the coefficients in column 31 (C31) are generated and stored in 32 different BRAMs using the rotating addressing scheme.

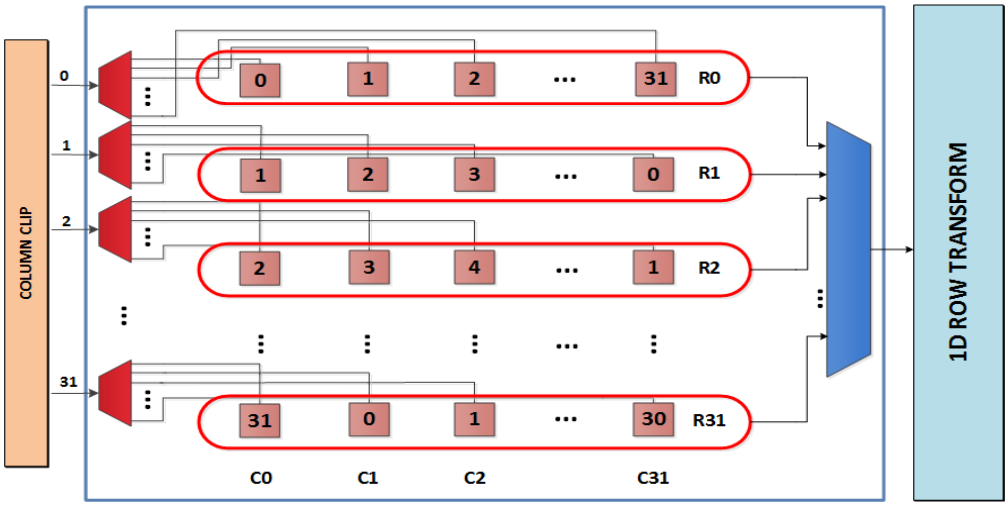

Figure 5.7 Transpose Memory

This ensures that the 32 coefficients necessary for 1D row DCT in a clock cycle can always be read in one clock cycle from 32 different BRAMs. Because of the input data loading and pipeline stages, the proposed hardware starts generating the results of 1D row DCT in 42 clock cycles. It then continues generating the results row by row in every clock cycle until the end of last TU in the frame without any stalls.

The proposed hardware performs 1D DCT transform for 4x4, 8x8, 16x16 and 32x32 TU sizes in 4, 8, 16 and 32 clock cycles, respectively. The 1D row DCT and 1D column DCT operations are pipelined. While 1D row DCT for current TU is performed, 1D column DCT for next TU is also performed.

5.2.2 Proposed HEVC 2D DCT Higher Utilization Hardware

The proposed HEVC 2D DCT higher utilization (HU) hardware processes four 4x4 TUs or two 8x8 TUs in parallel. Same as the LU hardware, it uses two 4x4 datapaths and one 8x8 datapath for 16x16 TU size, and it uses all datapaths (two 4x4, one 8x8 and one 16x16) for 32x32 TU size. However, the HU hardware uses two 4x4

datapaths and one 8x8 datapath for 4x4 and 8x8 TU sizes. Since 4x4 and 8x8 column and row datapaths are used for all TU sizes, data gating is used only for the inputs of 16x16 column and row datapaths.

Same as the LU hardware, multiplier blocks in the first 4x4 datapath and 16x16 datapath multiply a single input with 3 and 16 different constants, respectively. However, in the HU hardware, multiplier blocks in the second 4x4 datapath and 8x8 datapath multiply a single input with 7 and 15 different constants, respectively. Because, in the HU hardware, the second 4x4 datapath and 8x8 datapath are used for all TU sizes. Multiplier block for second 4x4 column datapath is shown in Figure 5.8.

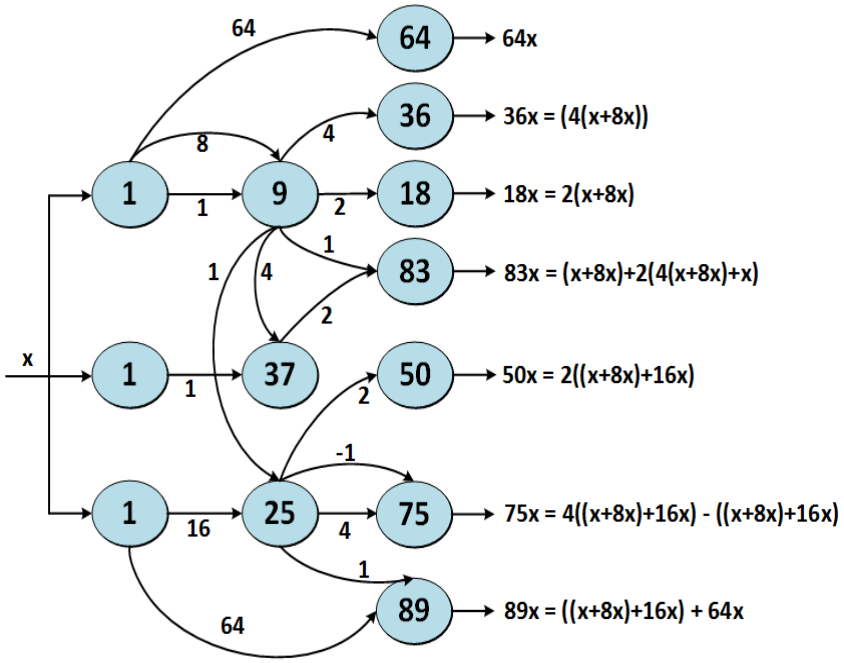

Figure 5.8 Multiplier Block in HEVC 2D DCT Higher Utilization Hardware

In order to calculate each output of 1D DCT for 4x4, 8x8 and 16x16 TU sizes, an output from each multiplier block in both 4x4 datapaths and 8x8 datapath is selected, and these outputs are added or subtracted. Similarly, in order to calculate each output of 1D DCT for 32x32 TU size, 32 outputs from 32 multiplier blocks in all datapaths (two 4x4, one 8x8 and one 16x16) are added or subtracted.

Same as the LU hardware, transpose memory is implemented using 32 BRAMs. However, in the HU hardware, 8, 8, 16 and 32 BRAMs are used for 4x4, 8x8, 16x16 and 32x32 TU sizes, respectively.

5.3 Implementation Results

The proposed low energy HEVC 2D DCT LU and HU hardware for all TU sizes including clock gating (original hardware), including clock gating and Hcub MCM algorithm (MCM hardware), and including clock gating, Hcub MCM algorithm and the proposed technique with coefficient set 3 (proposed hardware) are implemented in Verilog HDL. The Verilog RTL implementations are verified with RTL simulations. RTL simulation results matched the results of 2D DCT implementation in HEVC HM software encoder [39].

The Verilog RTL codes are synthesized and mapped to an Xilinx XC6VLX550T FF1156 FPGA. The FPGA implementations are verified with post place & route simulations. Post place & route simulation results matched the results of 2D DCT implementation in HEVC HM software encoder [39]. The FPGA implementation results given in Table 5.5 show that Hcub MCM algorithm considerably decreased area, and the proposed technique slightly increased area.

Table 5.5 FPGA Implementations Results

	LU Hardware			**HU Hardware**		
	Orig.	**MCM**	**Prop.**	**Orig.**	**MCM**	**Prop.**
Slice	12944	9797	10080	14981	11279	12712
LUT	39829	33376	35555	47737	38006	41905
DFF	11196	11110	11230	11964	12025	12200
BRAM	32	32	32	32	32	32
Freq. (MHz)	102	116	100	111	117	111
Fps	49 Quad Full HD	56 Quad Full HD	48 Quad Full HD	53 Ultra HD	56 Ultra HD	53 Ultra HD

Power consumptions of the FPGA implementations are estimated using a Xilinx XPower Analyzer. Post place & route timing simulations are performed for Tennis, Kimono and ParkScene (1920x1080) videos at 100 MHz [37] and signal activities are stored in VCD files. These VCD files are used for estimating power consumptions of the FPGA implementations.

The energy consumption results for the LU hardware and the HU hardware for one frame of each video are shown in Figure 5.9 and Figure 5.10, respectively. Hcub MCM algorithm reduced the energy consumption of the LU hardware and the HU

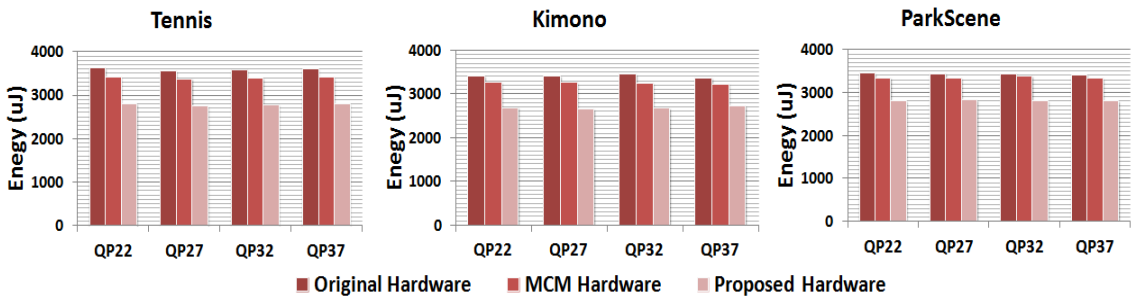

Figure 5.9 Energy Consumptions of HEVC 2D LU Hardware for Full HD (1920x1080) Video Frames

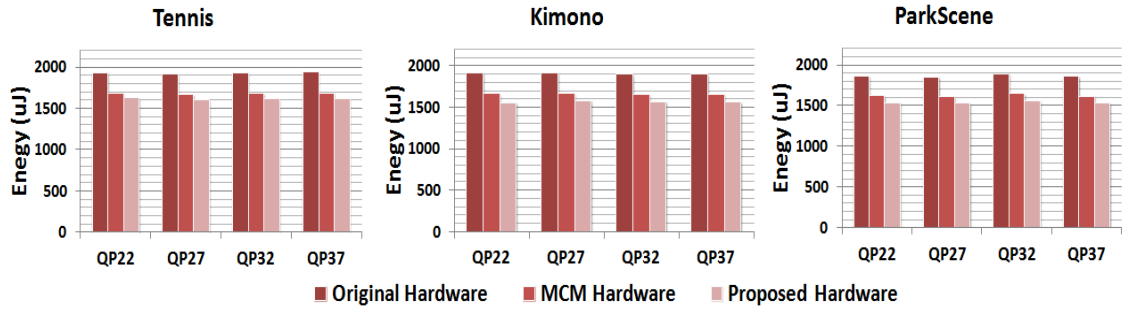

Figure 5.10 Energy Consumptions of HEVC 2D HU Hardware for Full HD (1920x1080) Video Frames

hardware up to 5.9% and 13.1%, respectively. The proposed energy reduction technique further reduced the energy consumption of the LU hardware and the HU hardware up to 17.9% and 18.9%, respectively.

In order to compare the LU hardware and the HU hardware with the HEVC DCT hardware in the literature, their Verilog RTL codes are also synthesized to a 90nm standard cell library and the resulting netlists are placed and routed. The resulting ASIC implementations of the LU hardware and the HU hardware work at 140 MHz and 130 MHz, respectively. Gate counts of the LU hardware and the HU hardware are calculated as 175K and 197K, respectively, according to NAND (3x1) gate area excluding on-chip memory. The comparison of the LU hardware and the HU hardware with the HEVC DCT hardware in the literature is shown in Table 5.6.

The proposed 2D DCT hardware has smaller area and power consumption than the 2D DCT hardware proposed in [70]-[74]. The DCT hardware proposed in [74] only performs 1D DCT, and its performance is not given. Since the 2D DCT hardware proposed in [70] and [73] use multipliers, they have larger area than the proposed 2D DCT hardware. Since the 2D DCT hardware proposed in [72] performs DCT operations

for several TUs in parallel for smaller TU sizes, it achieves higher performance than the proposed 2D DCT LU hardware at the expense of much larger area and power consumption. It has same performance as the proposed 2D DCT HU hardware with larger area.

Table 5.6 Hardware Comparison

	[70]	**[71]**	**[72]**	**[73]**	**[74]**	**LU Hardware**	**HU Hardware**
Technology	90 nm	45 nm	90 nm	90 nm	90 nm	90 nm	90 nm
Gate Count	343.5 K	205.5 K	347 K	328.2 K	149 K	175 K	197 K
Max Freq. (MHz)	311	333	187	400	100	140	130
Frames per Sec.	30 4096x2048	30 4096x2048	60 7680x4320	30 3840x2160	----	60 3840x2160	60 7680x4320
Throughput (pixels/cycle)	4/8/16/32	4/8/16/32	32	8/16/32/32	4/8/16/32	4/8/16/32	16/16/16/32
Power Dissipation	85.3 mW	----	67.6 mW	76.9 mW	25.0 mW	13.1 mW	65.8 mW
Transform Size	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32
Transform	2D	2D	2D	2D	1D	2D	2D

CHAPTER VI

A LOW ENERGY HEVC INVERSE TRANSFORM HARDWARE

HEVC standard uses Discrete Cosine Transform (DCT) / Inverse Discrete Cosine Transform (IDCT) same as the H.264 standard. However, H.264 standard uses only 4x4 and 8x8 Transform Unit (TU) sizes for DCT/IDCT. HEVC standard uses 4x4, 8x8, 16x16, and 32x32 TU sizes for DCT/IDCT. Larger TU sizes achieve better energy compaction. However, they increase the computational complexity exponentially. In addition, HEVC uses Discrete Sine Transform (DST) / Inverse Discrete Sine Transform (IDST) for 4x4 intra prediction in certain cases.

Transform operations (DCT/IDCT and DST/IDST) are heavily used in an HEVC encoder [11], [75, 76]. IDCT and IDST have high computational complexity. IDCT and IDST operations account for 11% of the computational complexity of an HEVC video encoder. They account for 25% of the computational complexity of an all intra HEVC video encoder.

In this thesis, a novel energy reduction technique for HEVC IDCT and IDST for all TU sizes is proposed. After forward transform and quantization, most of the forward transformed and quantized high frequency coefficients in a TU become zero. In addition, if the values of non-zero forward transformed and quantized low frequency coefficients in a TU are small, they have small impact on the inverse quantized and inverse transformed TU. Therefore, the proposed technique calculates IDCT and IDST only for DC coefficient if the values of several predetermined forward transformed low

frequency coefficients in a TU are smaller than a threshold. Otherwise, it calculates IDCT and IDST for all coefficients in the TU.

Since the proposed technique is used in mode decision stage of an HEVC encoder and it is not used in coding stage of an HEVC encoder, it does not cause any encoder-decoder mismatch. The proposed technique reduces the computational complexity of IDCT and IDST operations in an HEVC encoder significantly. It increases the bit rate slightly for most video frames. It decreases the PSNR slightly for some video frames, and it increases the PSNR slightly for some video frames. In addition, it can easily be used in HEVC encoders.

In this thesis, a low energy HEVC 2D inverse transform (IDCT and IDST) hardware for all TU sizes is also designed and implemented using Verilog HDL. Clock gating technique is used to reduce the energy consumption of the proposed hardware. Then, in order to reduce number and size of the adders in the proposed hardware, Hcub Multiplierless Constant Multiplication (MCM) algorithm [40] is used for calculating 8, 16 and 32 point IDCT. Hcub MCM algorithm reduced the energy consumption of the proposed hardware up to 56%. Finally, the proposed energy reduction technique is used to reduce the energy consumption of the proposed hardware. It reduced the energy consumption of the proposed hardware up to 31%. The proposed HEVC 2D inverse transform hardware can process 48 Quad HD (3840x2160) video frames per second. Therefore, it can be used in portable consumer electronics products that require a real-time HEVC encoder.

Several zero quantized DCT coefficient detection techniques are proposed for H.264 and HEVC [66]-[69]. These techniques try to predict the blocks with zero forward transformed and quantized coefficients before DCT and quantization operations in the coding stage of an H.264 or HEVC encoder in order to avoid DCT and quantization operations. However, the technique proposed in this thesis avoids the inverse transform (IDCT and IDST) operations that have no impact or low impact on the inverse quantized and inverse transformed TU in mode decision stage of an HEVC encoder.

Several HEVC IDCT hardware are proposed in the literature [70], [77]-[79]. In [77], only 1D IDCT is implemented for all TU sizes, and all IDCT outputs are calculated using multipliers. In [78], 2D IDCT is implemented only for 16x16 and 32x32 TU sizes, and processing elements are implemented using shifters, adders and multiplexers to reduce hardware area. In [79], 1D 8x8 IDCT for several video

compression standards (H.264, VC-1, AVS and HEVC) is implemented. In [70], 2D IDCT is implemented for all TU sizes, and the proposed hardware also calculates DCT and Hadamard Transform. The low energy HEVC 2D inverse transform hardware proposed in this thesis is compared with these HEVC IDCT hardware in Section 6.2.

6.1 Proposed Energy Reduction Technique

After forward transform and quantization, most of the forward transformed and quantized high frequency coefficients in a TU become zero. In addition, if the values of non-zero forward transformed and quantized low frequency coefficients in a TU are small, they have small impact on the inverse quantized and inverse transformed TU. Therefore, the proposed energy reduction technique calculates IDCT and IDST only for DC coefficient if the values of several predetermined forward transformed low frequency coefficients in a TU are smaller than a threshold. Otherwise, it calculates IDCT and IDST for all coefficients in the TU.

The proposed energy reduction technique for HEVC IDCT for all TU sizes is shown in Figure 6.1. The proposed technique checks the DC coefficient and three low frequency AC coefficients in the predetermined positions in a TU. If DC coefficient is not zero and all three low frequency AC coefficients are smaller than a threshold value, the proposed technique performs IDCT only for DC coefficient in the TU. Otherwise, it performs IDCT for all coefficients in the TU.

The proposed technique reduces the computational complexity of IDCT and IDST significantly by performing IDCT and IDST only for DC coefficient in a TU. Table 6.1 shows the number of addition and shift operations required for performing IDCT for all coefficients in a TU and for only DC coefficient in a TU for all TU sizes. Performing IDCT only for DC coefficient in a TU, on the average, achieves 98.87% reduction in addition and 98.70% reduction in shift operations. It achieves more computation reduction for larger TU sizes.

```
IDCT(Transform Coefficients)  {
  if (DC coefficient is not zero and
    predetermined AC coefficients are smaller than threshold)
      Residual ← IDCT(DC Coefficient)
  Else
      Residual ← IDCT(Transform Coefficients)
  end if   }
```

Figure 6.1 Pseudocode of HEVC IDCT with The Proposed Technique

Table 6.1 Addition and Shift Reductions for All TU Sizes

TU Size	IDCT for All Coefficients		IDCT for DC Coefficient		Reduction (%)	
	Add.	Shift	Add.	Shift	Add.	Shift
4x4	256	256	16	18	93.7	92.97
8x8	2688	2432	64	66	97.6	97.29
16x16	24576	2099	256	258	98.9	98.77
32x32	204800	1884	1024	1026	99.5	99.46
Total	362496	3276	4096	4266	98.8	98.70

The proposed technique is integrated into IDCT operations performed for rate distortion cost calculation in intra mode decision stage of HEVC reference software encoder (HM) version 10.0 [80]. The threshold value is experimentally determined as 64 to achieve large computation reduction with negligible bit rate increase and PSNR loss using this HEVC software encoder.

5 different low frequency AC coefficient sets shown in Figure 6.2 are evaluated using this HEVC software encoder for Class A and B video sequences [37]. The same AC coefficients are used for all TU sizes. For example, for coefficient set 1, the proposed technique checks the three low frequency AC coefficients in positions [0, 1], [0, 2] and [2, 0] for all TU sizes. The bit rate and PSNR results for three different quantization parameters (QP) are shown in Table 6.2. These results show that the proposed technique increases the bit rate slightly for most video frames. It decreases the PSNR slightly for some video frames, and it increases the PSNR slightly for some video frames. Since the proposed technique performs well for all video sequences with coefficient set 1, coefficient set 1 is selected for hardware implementation.

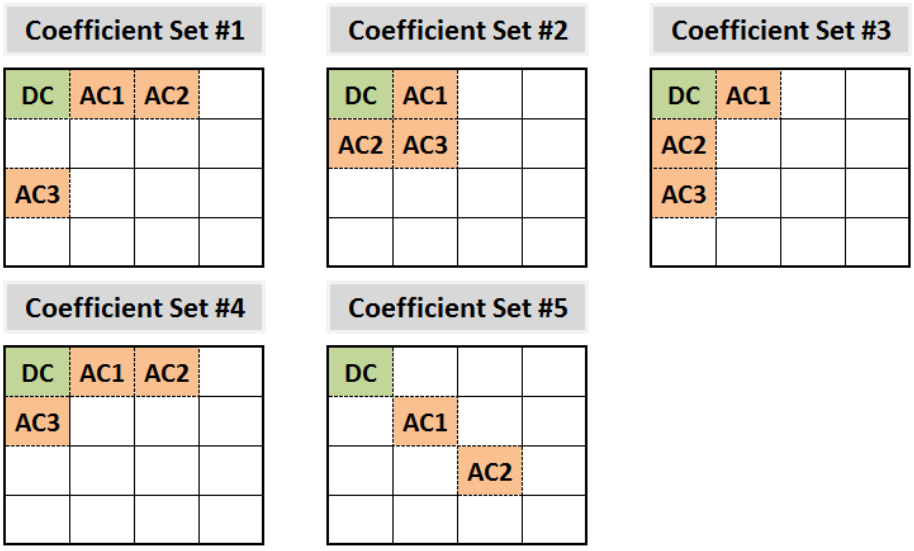

Figure 6.2 DC and Pre-Determined Coefficient Sets

Table 6.2 Bitrate and PSNR Values

	Frame	QP	Coefficient Set 1		Coefficient Set 2		Coefficient Set 3		Coefficient Set 4		Coefficient Set 5	
			ΔBRate (%)	ΔPSNR (dB)	ΔBRate (%)	ΔPSNR (dB)	ΔBRate (%)	ΔPSNR (dB)	ΔBRate (%)	ΔPSNR (dB)	ΔBRate (%)	ΔPSNR (dB)
Class A (2560x1600)	Steam Loco.	22	0.49	0.003	0.41	-0.001	0.42	-0.001	0.40	0.000	0.95	0.002
		27	0.53	-0.001	0.48	-0.007	0.47	-0.005	0.47	-0.004	0.40	-0.002
		32	0.64	-0.007	0.31	-0.009	0.39	-0.012	0.35	-0.013	0.80	-0.020
	Traffic	22	0.70	0.015	0.39	-0.016	0.25	-0.013	0.38	-0.018	4.03	-0.130
		27	1.25	0.016	0.60	-0.014	0.53	-0.011	0.68	-0.013	4.78	-0.107
		32	3.41	0.059	2.52	-0.043	2.34	-0.041	2.63	-0.040	7.43	-0.179
	People on Street	22	0.77	-0.005	0.07	-0.033	-0.03	0.011	-0.06	0.009	3.72	-0.072
		27	0.90	-0.019	0.17	-0.019	1.12	-0.028	1.18	-0.030	5.99	-0.104
		32	3.05	-0.054	3.97	-0.040	3.66	-0.131	3.79	-0.136	10.78	-0.231
Class B (1920x1080)	Park Scene	22	0.39	-0.010	0.43	-0.006	0.34	-0.008	0.39	-0.009	2.04	-0.058
		27	0.68	-0.017	0.44	-0.016	0.41	-0.019	0.47	-0.016	2.26	-0.081
		32	0.57	-0.085	0.36	-0.081	0.49	-0.073	0.52	-0.070	1.92	-0.172
	Kimono	22	0.40	-0.004	0.04	-0.003	0.01	-0.004	-0.09	-0.001	1.82	-0.011
		27	0.63	-0.002	0.27	-0.004	0.23	0.004	0.28	-0.005	2.52	-0.023
		32	0.95	-0.009	0.29	0.003	0.13	-0.004	0.17	-0.007	2.68	-0.042
	Cactus	22	-0.04	-0.039	0.36	-0.035	0.37	-0.033	0.30	-0.040	2.45	-0.108
		27	0.86	-0.016	0.33	-0.012	1.01	-0.014	1.00	-0.017	5.09	-0.063
		32	2.59	-0.046	2.84	-0.044	3.07	-0.049	3.07	-0.044	9.51	-0.136

The percentages of TU size selections (PTU) and the percentages of times the proposed technique with coefficient set 1 performs IDCT only for DC coefficient for the selected TU (PDC) are determined using this HEVC software encoder for Class A and B video sequences for different QPs, and they are shown in Table 6.3. The results in Table 6.1 and Table 6.3 show that the proposed technique reduces the computational complexity of inverse transform operations in an HEVC encoder significantly.

The percentages of TU size selections changes from frame to frame. But, the most selected TU size is 4x4 and the percentages of TU size selections get smaller with larger TU sizes. The percentage of times the proposed technique performs IDCT only for DC coefficient is highest for 4x4 TU size, and the percentage gets smaller with larger TU sizes. This is because DCT produces larger low frequency AC coefficients for larger TU sizes. Therefore, the three low frequency AC coefficients in the predetermined positions in a TU become smaller than the threshold value less often for larger TU sizes.

The percentage of times the proposed technique performs IDCT only for DC coefficient gets larger with larger QPs. This is because DCT produces more zero low frequency AC coefficients with larger QPs. Therefore, the three low frequency AC coefficients in the predetermined positions in a TU become smaller than the threshold value more often for larger QPs.

Table 6.3 Percentages (%) of TU Sizes and IDCT for DC Coefficient

Frame	QP		4x4	8x8	16x16	32x32	Total
Steam Loco.	22	PTU	74.36	20.40	4.71	0.53	100.0
		PDC	16.44	3.99	1.97	3.27	**13.15**
	27	PTU	71.76	22.26	5.36	0.62	100.00
		PDC	27.95	8.54	4.20	7.44	**22.23**
	32	PTU	67.52	25.15	6.55	0.78	100.00
		PDC	40.81	15.38	8.75	3.30	**32.03**
Traffic	22	PTU	69.23	19.28	4.65	6.84	100.00
		PDC	39.27	11.22	2.64	2.37	**25.28**
	27	PTU	66.32	25.97	6.86	0.85	100.00
		PDC	43.19	18.87	7.86	7.52	**34.15**
	32	PTU	60.77	29.42	8.67	1.14	100.00
		PDC	54.39	27.38	14.54	4.02	**42.42**
People on Street	22	PTU	71.50	22.52	5.33	0.65	100.00
		PDC	27.52	5.36	0.93	1.82	**20.95**
	27	PTU	66.60	25.84	6.72	0.84	100.00
		PDC	39.79	13.82	4.76	6.12	**30.44**
	32	PTU	61.04	29.04	8.74	1.18	100.00
		PDC	49.55	22.08	11.18	3.29	**37.67**
Park Scene	22	PTU	71.48	22.32	5.54	0.66	100.00
		PDC	23.29	10.75	5.63	7.58	**19.41**
	27	PTU	68.32	24.43	6.42	0.83	100.00
		PDC	33.67	15.72	9.22	17.08	**27.58**
	32	PTU	63.05	27.85	8.04	1.07	100.00
		PDC	48.56	22.47	13.34	6.85	**38.02**
Kimono	22	PTU	67.20	25.79	6.28	0.73	100.00
		PDC	59.20	13.14	3.68	3.28	**43.43**
	27	PTU	60.86	30.17	8.00	0.97	100.00
		PDC	77.84	25.50	6.54	7.25	**55.66**
	32	PTU	50.39	36.95	11.24	1.42	100.00
		PDC	89.07	43.60	11.64	2.83	**62.34**
Cactus	22	PTU	71.55	22.34	5.45	0.66	100.00
		PDC	21.68	11.41	4.55	4.44	**18.34**
	27	PTU	66.03	25.85	7.20	0.92	100.00
		PDC	34.03	18.65	9.50	8.91	**28.06**
	32	PTU	59.70	29.72	9.31	1.27	100.00
		PDC	44.88	25.28	14.45	3.80	**35.70**

6.2 Proposed HEVC 2D IDCT and IDST Hardware

The proposed low energy HEVC 2D inverse transform (IDCT and IDST) hardware for all TU sizes including clock gating, Hcub MCM algorithm, and the proposed energy reduction technique is shown in Figure 6.3. The proposed hardware uses an efficient butterfly structure for column and row transforms. The butterfly structure used for column transforms is shown in Figure 6.4. IDCT inputs are selected depending on size of the IDCT operation (4, 8, 16 or 32 point). Then, IDCT and IDST multiplications are performed in the datapaths using only adders and shifters. As shown in Figure 6.5, 4x4 datapaths perform both 4x4 IDCT and 4x4 IDST operations, and the result of one of these inverse transforms is selected based on a control signal.

In order to reduce number and size of the adders in the proposed hardware, Hcub MCM algorithm [40] is used for calculating 8, 16 and 32 point IDCT. Hcub algorithm tries to minimize number and size of the adders in a multiplier block which takes a single input, multiplies this input with multiple constants using shift and addition operations, and outputs the results of these multiplications. Hcub algorithm determines necessary shift and addition operations in a multiplier block. Hcub algorithm is used for 8, 16 and 32 point IDCT in the proposed hardware, because it did not achieve additional optimization for 4 point IDCT and 4 point IDST hardware.

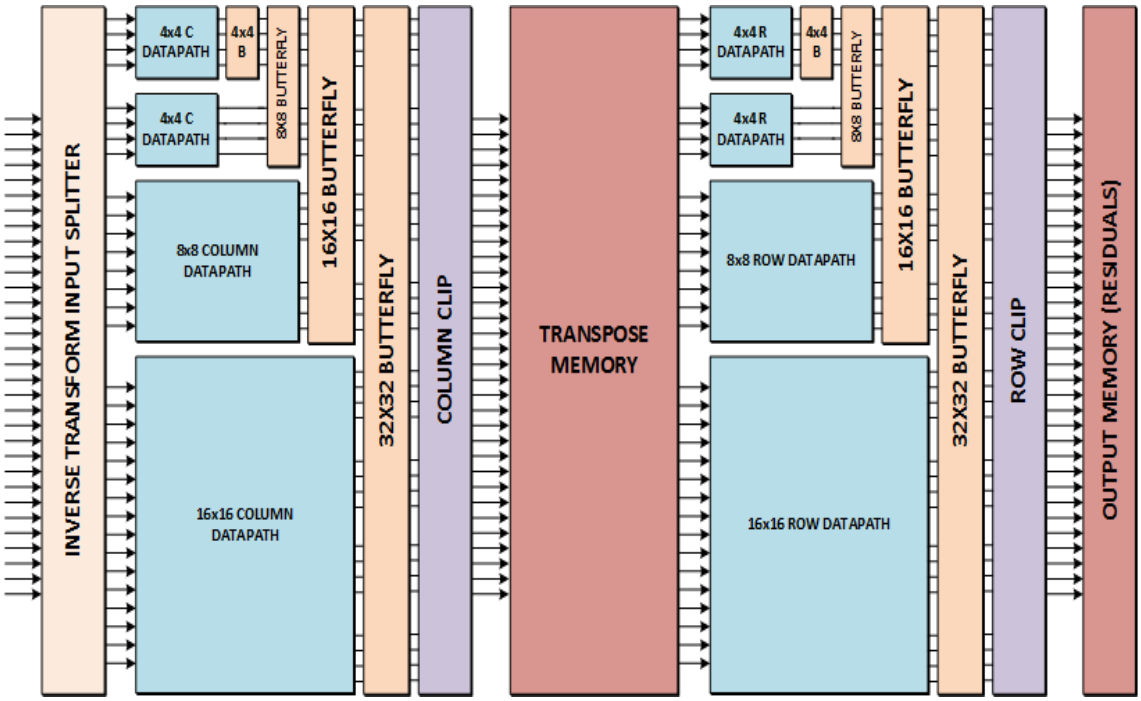

Figure 6.3 Proposed HEVC 2D IDCT and IDST Hardware

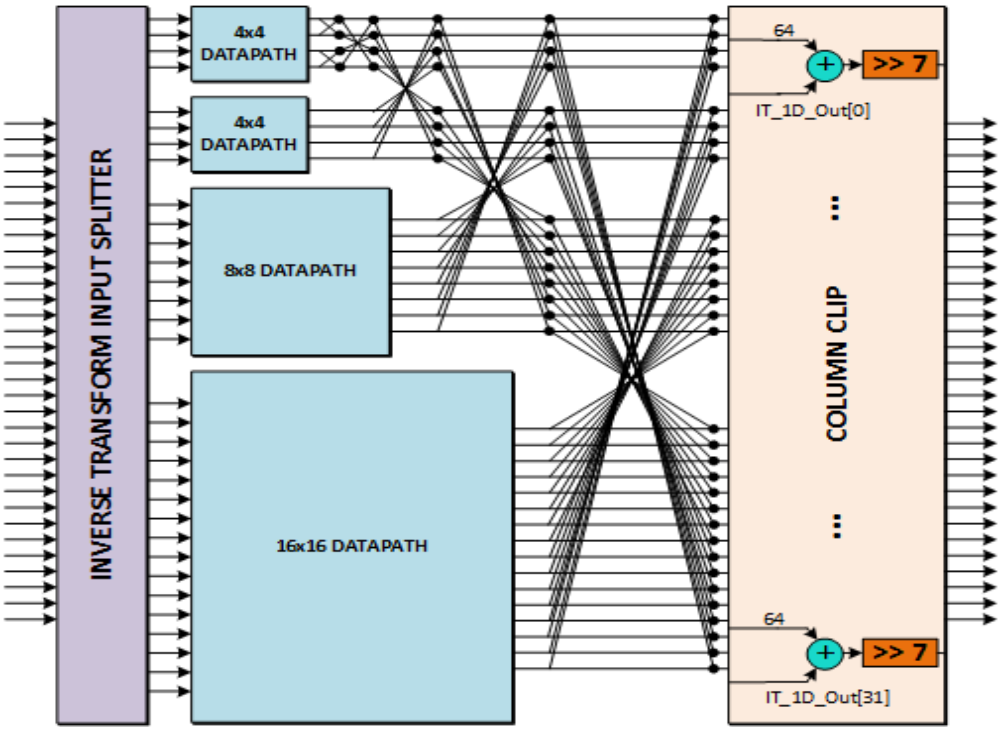

Figure 6.4 Column Butterfly Structure

Since different constants are used in 8, 16 and 32 point IDCT, three different multiplier blocks are used in the proposed hardware. Multiplier block used for 8 point IDCT is shown in Figure 6.6. Multiplier block for 8 point IDCT multiplies a single input with four different constants, multiplier block for 16 point IDCT multiplies a single input with eight different constants, and multiplier block for 32 point IDCT multiplies a single input with sixteen different constants. There are 4 multiplier blocks in 8x8 datapath, 8 multiplier blocks in 16x16 datapath and 16 multiplier blocks in 32x32 datapath.

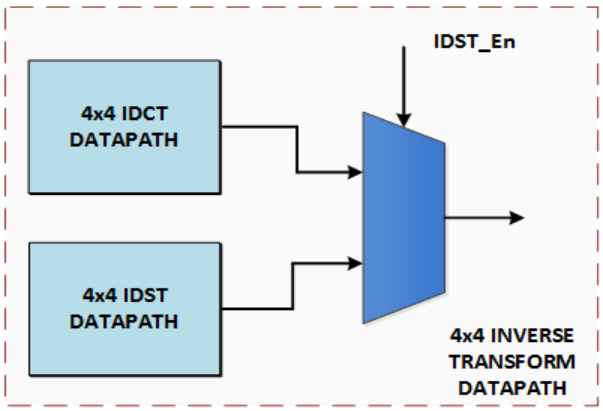

Figure 6.5 4x4 Datapath

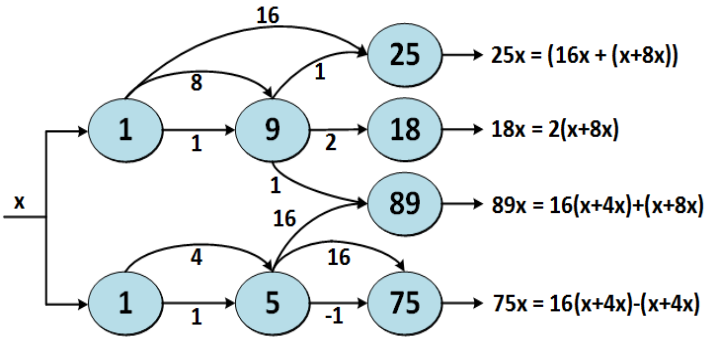

Figure 6.6 Multiplier Block in 8x8 Datapath

In order to calculate each output of 8 point IDCT, an output from each multiplier block is selected, and these outputs are added or subtracted. Similarly, in order to calculate each output of 16 point IDCT, eight outputs from eight multiplier blocks are added. Similarly, in order to calculate each output of 32 point IDCT, sixteen outputs from sixteen multiplier blocks are added.

In the proposed hardware, after 1D column IDCT, the resulting coefficients are stored in a transpose memory, and they are used as input for 1D row IDCT. As shown in Figure 6.7, the transpose memory is implemented using 32 Block RAMs (BRAM). 4, 8, 16 and 32 BRAMs are used for 4 point, 8 point, 16 point and 32 point IDCT, respectively. In the figure, the numbers in each box show the BRAM that coefficient is stored.

The results of 1D column IDCT are generated column by column. For 32 point IDCT, first, the coefficients in column 0 (C0) are generated in a clock cycle and stored in 32 different BRAMs. Then, the coefficients in column 1 (C1) are generated in the next clock cycle and stored in 32 different BRAMs using a rotating addressing scheme.

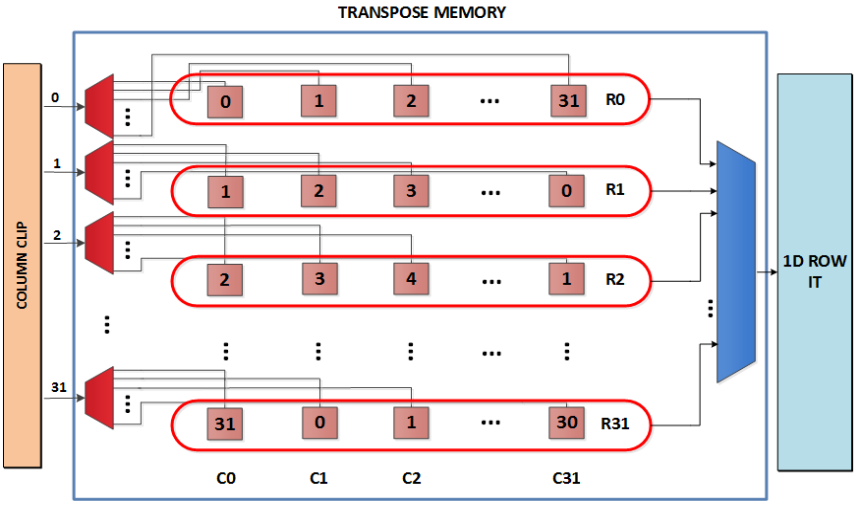

Figure 6.7 Transpose Memory

This continuous until the coefficients in column 31 (C31) are generated and stored in 32 different BRAMs using the rotating addressing scheme. This ensures that the 32 coefficients necessary for 1D row IDCT in a clock cycle can always be read in one clock cycle from 32 different BRAMs.

Because of the input data loading and pipeline stages, the proposed hardware starts generating the results of 1D row IDCT in 40 clock cycles. It then continues generating the results row by row in every clock cycle until the end of the last TU in the video frame without any stalls. The proposed HEVC 2D IDCT hardware finishes 4, 8, 16 and 32 point IDCT operations in 4, 8, 16 and 32 clock cycles, respectively.

6.3 Implementation Results

The proposed low energy HEVC 2D inverse transform (IDCT and IDST) hardware for all TU sizes including clock gating (original hardware), including clock gating and Hcub MCM algorithm (MCM hardware), and including clock gating, Hcub MCM algorithm and the proposed energy reduction technique (proposed hardware) are implemented in Verilog HDL.

The Verilog RTL implementations are verified with RTL simulations. RTL simulation results matched the results of inverse transform implementation in HEVC reference software encoder (HM) version 10.0 [80]. The Verilog RTL codes are synthesized and mapped to a Xilinx XC6VLX130T FF1156 FPGA. The FPGA implementations are verified with post place & route simulations. Post place & route simulation results matched the results of inverse transform implementation in HEVC reference software encoder (HM) version 10.0 [80].

All three FPGA implementations work at 150 MHz. Therefore, in the worst case (when all TU sizes in a video frame are 32x32), they can process 48 Quad HD (3840x2160) video frames per second. FPGA implementation of the original hardware uses 15101 slices, 45698 LUTs, 12187 DFFs, and 32 BRAMs. FPGA implementation of the MCM hardware uses 11343 slices, 38790 LUTs, 11762 DFFs, and 32 BRAMs. FPGA implementation of the proposed hardware uses 11397 slices, 38821 LUTs, 11763 DFFs, and 32 BRAMs. BRAMs are implemented as dual-port Select RAMs. These results show that Hcub MCM algorithm considerably decreased the area, and the proposed technique slightly increased the area.

The power consumptions of original hardware, MCM hardware, and proposed hardware are estimated using a Xilinx XPower Analyzer. Post place & route timing

simulations are performed for Cactus and Kimono (1920x1080) videos at 50 MHz [37] and signal activities are stored in VCD files. These VCD files are used for estimating the power consumptions of all three FPGA implementations. The power and energy consumption results for one frame of each video are shown in Tables 6.4 and 6.5. Hcub MCM algorithm reduced the energy consumption of the proposed hardware up to 56%. The proposed energy reduction technique further reduced the energy consumption of the proposed hardware up to 31%.

In order to compare the proposed hardware with the HEVC IDCT hardware in the literature, its Verilog RTL code is also synthesized to a 90nm standard cell library and the resulting netlist is placed & routed. The resulting ASIC implementation works at 150 MHz, and its gate count is calculated as 142K according to NAND (3x1) gate

Table 6.4 Energy Consumption Reductions for Cactus (1920x1080)

QP	22			27			32		
	Original	**MCM**	**Proposed**	**Original**	**MCM**	**Proposed**	**Original**	**MCM**	**Proposed**
Clock (mW)	84	66	67	84	66	67	84	66	67
Logic (mW)	83	35	35	93	36	38	81	34	35
Signal (mW)	68	17	17	76	17	19	67	16	17
BRAM (mW)	56	16	16	56	17	18	55	18	19
Total Power (mW)	291	134	135	309	136	142	287	134	138
Time (ms)	5.159	5.159	4.254	5.422	5.422	4.523	5.862	5.862	4.556
Energy (uJ)	1501.27	691.31	574.29	1675.40	737.39	642.27	1682.40	785.51	628.73
Energy Red.		**53.95%**	**61.75%**		**55.99%**	**61.66%**		**53.31**	**62.63**

Table 6.5 Energy Consumption Reductions for Kimono (1920x1080)

QP	22			27			32		
	Original	**MCM**	**Proposed**	**Original**	**MCM**	**Proposed**	**Original**	**MCM**	**Proposed**
Clock (mW)	84	66	67	84	66	67	84	66	67
Logic (mW)	89	36	34	91	38	35	81	37	34
Signal (mW)	51	17	16	52	17	17	46	17	17
BRAM (mW)	54	15	15	53	16	17	53	18	18
Total Power (mW)	278	134	132	280	137	136	264	138	136
Time (ms)	5.153	5.153	4.085	5.524	5.524	4.080	5.895	5.895	4.027
Energy (uJ)	1432.53	690.50	539.22	1546.72	756.79	554.96	1556.28	813.51	547.67
Energy Red.		**51.80%**	**62.36%**		**51.07%**	**64.12%**		**47.72%**	**64.80%**

area excluding on-chip memory. The comparison of the proposed hardware with the HEVC IDCT hardware in the literature is shown in Table 6.6. Only the proposed hardware implements 4x4 IDST.

Since the IDCT hardware proposed in [77] only implements 1D IDCT, it has lower gate count than the proposed hardware. But, it is slower than the proposed hardware. Although the IDCT hardware proposed in [78] only implements 16 and 32 point 2D IDCT, it has higher gate count than the proposed hardware and it is slower than the proposed hardware. Since the IDCT hardware proposed in [79] only implements 8 point 1D IDCT, it has lower gate count than the proposed hardware. But, it is slower than the proposed hardware. The IDCT hardware proposed in [70] has higher gate count than the proposed hardware and it is slower than the proposed hardware.

Table 6.6 Hardware Comparison

	[77]	**[78]**	**[79]**	**[70]**	**Proposed**
Technology	0.13 um ASIC	0.18 um ASIC	0.18 um ASIC	90 nm ASIC	90 nm ASIC
Gate Count	109.2 K	287 K	12.3 K	235.4 K	142 K
Max Speed (MHz)	350	300	211	311	150
Frames per Second	30 4096x2048	30 3840x2160	67 1920x1080	30 4096x2048	48 3840x2160
Transform Size	4, 8, 16, 32	16, 32	8	4, 8, 16, 32	4, 8, 16, 32
Transform	1D	2D	1D	2D	2D

CHAPTER VII

CONCLUSIONS AND FUTURE WORKS

In this thesis, we proposed a novel adaptive 2D digital image processing algorithm for 2D median filter, Gaussian blur and image sharpening. We designed low energy 2D median filter, Gaussian blur and image sharpening hardware using the proposed algorithm. We proposed approximate HEVC intra prediction and HEVC fractional interpolation algorithms. We designed low energy approximate HEVC intra prediction and HEVC fractional interpolation hardware. We also proposed several HEVC fractional interpolation hardware architectures. We proposed novel computational complexity and energy reduction techniques for HEVC DCT and inverse DCT/DST. We designed high performance and low energy hardware for HEVC DCT and inverse DCT/DST including the proposed techniques. We quantified computation reductions achieved and video quality loss caused by the proposed algorithms and techniques. We implemented the proposed hardware architectures in Verilog HDL. We mapped the Verilog RTL codes to Xilinx Virtex 6 and Xilinx ZYNQ FPGAs, and estimated their power consumptions using Xilinx XPower Analyzer tool. The proposed algorithms and techniques significantly reduced the power and energy consumptions of these FPGA implementations in some cases with no PSNR loss and in some cases with very small PSNR loss.

As future work, application-specific approximate video processing and compression algorithms can be proposed. An HEVC video encoder and decoder can be

implemented by implementing the HEVC video encoder and decoder modules which are not implemented in this thesis and by integrating them with the ones implemented in this thesis. Versatile Video Coding (VVC) is a new video compression standard which will have much higher computational complexity than HEVC. Therefore, energy reduction techniques for VVC standard and low energy VVC hardware implementations can be proposed.

BIBLIOGRAPHY

[1] C. H. Huang, C. Y. Chang, “An Area and Power Efficient Adder-Based Stepwise Linear Interpolation for Digital Signal Processing,” *IEEE Transactions on Consumer Electronics*, vol. 61, no. 1, pp. 69-75, Feb. 2016.

[2] S. Li, X. Kang, “Fast Multi-Exposure Image Fusion with Median Filter and Recursive Filter,” *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 626-632, May 2012.

[3] Y. Yang, “Three-Dimensional Image Processing VLSI System with Network-on-Chip System and Reconfigurable Memory Architecture,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1345- 1353, Aug. 2011.

[4] J. O. Cadenas, R. S. Sherratt, P. Huerta, W. C. Kao, “Parallel Pipelined Array Architectures for Real-Time Histogram Computation in Consumer Devices,” IEEE Transactions on Consumer Electronics, vol. 57, no. 4, pp. 1460-1464, Nov. 2011.

[5] Cisco, "Cisco visual networking index: Forecast and methodology, 2016 - 2021,” Sep. 2017 [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[6] High Efficiency Video Coding, ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC, Apr. 2013.

[7] G.J.Sullivan, J.R. Ohm, W.J. Han, T. Wiegand, “ Overview of the High Efficiency Video Coding (HEVC) Standard, ”*IEEE Trans. on Circuits and Systems for Video Technology,* vol.22, no.12, pp.1649-1668, Dec. 2012.

[8] F. Pescador, M. Chavarrias, M. J. Garrido, E. Juarez, C. Sanz, “Complexity Analysis of an HEVC Decoder Based on a Digital Signal Processor”, *IEEE Trans. on Consumer Electronics*, vol.59, no.2, pp. 391-399, May 2013.

[9] Advanced Video Coding, ITU-T Rec. H.264 and ISO/IEC 14496-10 (H.264), ITU-T and ISO/IEC, Apr. 2017.

[10] V. Sze, M. Bugadavi, G. J. Sullivan, High Efficiency Video Coding (HEVC) Algorithms and Architectures, Springer, 2014.

[11] J. Vanne, M. Viitanen, T.D. Hämäläinen and A. Hallapuro, "Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs", *IEEE Trans. on Circuits and Systems for Video Technology*, vol.22, no.12, pp.1885-1898, Dec. 2012.

[12] E. Kalali, I Hamzaoglu, "A Low Energy 2D Adaptive Median Filter Hardware," *Design, Automation and Test in Europe (DATE) Conference*, pp. 725-729, March 2015.

[13] E. Kalali, I. Hamzaoglu, "Low Complexity 2D Adaptive Image Processing Algorithm and Its Hardware Implementation," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 3, pp. 277-284, Aug. 2017.

[14] E. Kalali, A. C. Mert, I. Hamzaoglu, "Pixel Correlation Based Computation and Energy Reduction Techniques for HEVC Fractional Interpolation," *IEEE. Int. Conf. on Consumer Electronics – Berlin*, Sep. 2017.

[15] E. Kalali, I. Hamzaoglu, "A Low Energy Sub-Pixel Interpolation Hardware," *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 1218-1222, Oct. 2014.

[16] E. Kalali, I. Hamzaoglu, "Approximate HEVC Fractional Interpolation Filters and Their Hardware Implementations," *IEEE Trans. on Consumer Electronics*, vol. 64, no. 3, Aug. 2018.

[17] E. Kalali, I. Hamzaoglu, "A Computation and Energy Reduction Technique for HEVC Discrete Cosine Transform," *IEEE Trans. on Consumer Electronics*, vol. 62, no. 2, pp. 166-174, May 2016.

[18] E. Kalali, E. Ozcan, O. M. Yalcinkaya, I. Hamzaoglu, "A Low Energy HEVC Inverse Transform Hardware," *IEEE Trans. on Consumer Electronics*, vol. 60, no. 4, pp. 754-761, Oct. 2014.

[19] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Prentice Hall, 2002.

[20] C. Chakrabarti, "Sorting network based architecture for median filters," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol.40, no. 11, pp. 723-727, Nov. 1993.

[21] J. Scott, M. Pusateri, M. U. Mushtaq, "Comparison of 2D median filter hardware implementations for real time stereo video," *37th IEEE Applied Imagery Pattern Recognition Workshop*, Oct. 2008.

[22] S. Esakkirajan, T. Veerakumar, A. N. Subramanyan, C. H. PremChand, "Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter," *IEEE Signal Processing Letters*, vol. 18, no. 5, pp. 287-290, March 2011.

[23] S. Akkoul, L. Roger, R. Leconge, R. Harba, "A new adaptive switching median filter," *IEEE Signal Processing Letters*, vol. 17, no. 6, pp. 587- 590, June 2010.

[24] Z. Vasicek, L. Sekanina, "Novel hardware implementation of adaptive median filters," *11th IEEE Workshop on Design and Diagnostics of Electronics Circuits and Systems*, Apr. 2008.

[25] V. G. Moshnyaga, K. Hashimoto, "An efficient implementation of 1-D median filter," *52nd IEEE Int. Midwest Symp. on Circuits and Systems*, Aug. 2009.

[26] S. A. Fahmy, P. Y. K. Cheung, W. Luk, "High-throughput onedimensional median and weighted median filters on FPGA," *IET Computers & Digital Techniques*, vol. 3, no. 4, pp. 384-394, June 2009.

[27] D. Prokin, M. Prokin, "Low hardware complexity pipelined rank filter," *IEEE Trans. on Circuits and Systems II: Express Brief*, vol. 57, no. 6, pp. 446-450, May 2010.

[28] A. Sanny, V. K. Prasanna, "Energy-efficient median filter on FPGA," *Int. Conf. on Reconfigurable Computing and FPGAs*, Dec. 2013.

[29] Z. Zhang, E. Klassen, A. Srivastava, "Gaussian Blurring-Invariant Comparison of Signals and Images," *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3145-3157, Aug. 2013.

[30] T. Popkin, A. Cavallaro, D. Hands, "Accurate and Efficient Method for Smoothly Space-Variant Gaussian Blurring," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1362-1370, May 2010.

[31] S. Song, S. Lee, J. P. Ko, J. W. Jeon, "A Hardware Architecture Design for Real-Time Gaussian Filter," *IEEE Int. Conf. on Industrial Technology*, Feb. 2014.

[32] A. Jaiswal, B. Garg, Vi Kaushal, G. K. Sharma, "SPAA-Aware 2D Gaussian Smoothing Filter Design Using Efficient Approximation Techniques," *28th Int. Conf. on VLSI Design*, pp. 333-338, Jan. 2015.

[33] H. Luo, X. Gai, Z. Chang, B. Hui, "A Real-Time Multi-Scale 2-D Gaussian Filter Based on FPGA," *SPIE Int. Symp. on Optoelectronic Technology and Application: Image Processing and Pattern Recognition*, Nov. 2014.

[34] S. Khorbotly, F. Hassan, "A Modified Approximation of 2D Gaussian Smoothing Filters for Fixed-Point Platforms," *IEEE Southeastern Symp. on System Theory*, pp. 151-159, March 2011.

[35] S. L. Chen, “VLSI Implementation of a Low-Cost High-Quality Image Scaling Processor,” *IEEE Transactions on Circuits and SystemsII:Express Briefs*, vol. 60, no. 1, pp. 31-35, Jan. 2013.

[36] S. L. Chen, “VLSI Implementation of an Adaptive Edge-Enhanced Image Scalar for Real-Time Multimedia Applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 9, pp. 1510-1522, Sep. 2013.

[37] F. Bossen, “Common test conditions and software reference configurations”, *JCTVC-I1100*, May 2012.

[38] Benchmark Images [Online]. Available: http://www.dcs.qmul.ac.uk/~phao/CIP/Images

[39] K. McCann, B. Bross, W.J. Han, I.K. Kim, K. Sugimoto, G. J. Sullivan, “High Efficiency Video Coding (HEVC) Test Model 15 (HM 15) Encoder Description”, *JCTVC-Q1002*, June 2014.

[40] Y. Voronenko, M. Püschel, “Multiplierless Constant Multiplication,” *ACM Trans. on Algorithms*, vol. 3, no. 2, May 2007.

[41] E. Kalali, Y. Adibelli, I. Hamzaoglu, “A High Performance and Low Energy Intra Prediction Hardware for High Efficiency Video Coding,” *22nd Int. Conf. on Field Programmable Logic and Applications (FPL)*, pp. 719-722, Aug. 2012.

[42] E. Kalali, Y. Adibelli, I. Hamzaoglu, “A Low Energy Intra Prediction Hardware for High Efficiency Video Coding,” *Journal of Real-Time Image Processing*, vol. 15, no. 2, pp. 221-234, Aug. 2018.

[43] B. Min, Z. Xu, R. C. C. Cheung, “A Fully Pipelined Hardware Architecture for Intra Prediction of HEVC”, *IEEE Trans. on Circuits and Systems for Video Technology*, July 2016.

[44] M. Abeydeera, M. Karunaratne, G. Karunaratne, K. De Silva, A. Pasqual, “4K Real Time HEVC Decoder on FPGA”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 236-249, Jan. 2016.

[45] F. Amish, E. B. Bourennane, “Fully Pipelined Real Time Hardware Solution for High Efficiency Video Coding (HEVC) Intra Prediction”, *Journal of System Architecture*, vol. 64, pp. 133-147, March 2016.

[46] M. U. K. Khan, M. Shafique, M. Grellert, J. Henkel, “HardwareSoftware Collaborative Complexity Reduction Scheme for The Emerging HEVC Intra Encoder,” *Design, Automation and Test in Europe (DATE) Conference*, pp. 125-128, March 2013.

[47] G. Pastuszak, A. Abramowski, “Algorithm and Architecture Design of The H.265/HEVC Intra Encoder”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 210-222, Jan. 2016.

[48] C.T. Huang, M. Tikekar, A. Chandrakasan, “Memory-Hierarchical and Mode-Adaptive HEVC Intra Prediction Architecture for Quad Full HD Video Decoding”, *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 7, pp. 1515-1525, July 2014.

[49] P. Chiang, Y. Ting, H. Chen, S. Jou, I. Chen, H. Fang, T. Chang, “A QFHD 30 fps HEVC Decoder Design”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 26, no. 4, pp. 724-735, April 2016.

[50] N. Zhou, D. Ding, L. Yu, “On Hardware Architecture and Processing Order of HEVC Intra Prediction Module”, *Picture Coding Symposium*, pp. 101-104, Dec. 2013.

[51] Z. Liu, D. Wang, H. Zhu, X. Huang, “41.7BN-pixels/s Reconfigurable Intra Prediction Architecture for HEVC 2560x1600 Encoder”, *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 2634-2638, May 2013.

[52] H. Azgin, E. Kalali, I. Hamzaoglu, “A Computation and Energy Reduction Technique for HEVC Intra Prediction,” *IEEE Trans. on Consumer Electronics*, vol. 63, no. 1, pp. 36-43, Feb. 2017.

[53] H. Azgin, A. C. Mert, E. Kalali, I. Hamzaoglu, “An Efficient FPGA Implementation of HEVC Intra Prediction,” *IEEE Int. Conf. on Consumer Electronics*, pp. 1-5, Jan. 2018.

[54] Spiral Multiplier Block Generator, http://spiral.ece.cmu.edu/mcm/gen.html.

[55] A. Diefy, A. Shalaby, and M. S. Sayed, “Efficient architectures for HEVC luma interpolation filters,” *Int. Conf. on Microelectronics*, 2015.

[56] A. Diefy, A. Shalaby, and M. S. Sayed, “Low cost luma interpolation filter for motion compensation in HEVC,” *Int. Symp. on Midwest Circuits and Systems*, 2016.

[57] C. M. Diniz, M. Shafique, S. Bampi, and J. Henkel, “High-throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for HEVC,” *IEEE Int. Conf. on Image Processing*, Oct. 2013.

[58] G. Pastuszak and M. Trochimiuk, “Architecture design and efficiency evaluation for the high-throughput interpolation in the HEVC encoder,” *Euromicro Conference on Digital System Design*, Sep. 2013.

[59] C. Y. Lung and C. A. Shen, “A high-throughput interpolator for fractional motion estimation in high efficient video coding (HEVC) systems,” *IEEE Asia Pacific Conference on CAS*, 2014.

[60] W. Zhou, X. Zhou, and X. Lian “An efficient interpolation filter VLSI architecture for HEVC,” *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2015.

[61] D. Kang, Y. Kang, and Y. Hong, “VLSI implementation of fractional motion estimation interpolation for high efficiency video coding,” *Electronic Letters*, vol. 51, no. 5, pp. 1163-1165, Jul. 2015.

[62] S. Wang, D. Zhou, J. Zhou, T. Yoshimura, and S. Goto, “VLSI implementation of HEVC motion compensation with distance biased direct cache mapping for 8K UHDTV applications,” *IEEE Trans. on Circuits and Systems for Video Technology*, Dec. 2015.

[63] G. Pastuszak and M. Trochimiuk, “Algorithm and architecture design of the motion estimation for the H.265/HEVC 4K-UHD encoder,” *Journal of Real-Time Image Processing*, vol. 12, no. 2, pp. 517-529, Aug. 2016.

[64] C. M. Diniz, M. Shafique, S. Bampi, and J. Henkel, “A reconfigurable hardware architecture for fractional pixel interpolation in high efficiency video coding,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 238-251, Feb. 2015.

[65] A. C. Mert, E. Kalali, I. Hamzaoglu, “An HEVC Fractional Interpolation Hardware Using Memory Based Constant Multiplication,” IEEE Int. Conf. on Consumer Electronics, Jan. 2018.

[66] Y. H. Moon, G. Y. Kim, J. H. Kim, "An Improved Early Detection Algorithm for All-Zero Blocks in H.264 Video Encoding", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 8, pp. 1053-1057, Aug. 2005.

[67] M. Zhang, T. Zhou, W. Wang, "Adaptive Method for Early Detecting Zero Quantized DCT Coefficients in H.264/AVC Video Encoding", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 19, no. 1, pp. 103-107, Jan. 2009.

[68] K. Lee, H. J. Lee, J. Kim, Y. Choi, "A Novel Algorithm for Zero Block Detection in High Efficiency Video Coding", *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1124-1134, Dec. 2013.

[69] J. Li, J. Takala, M. Gabbouj, H. Chen, "A Detection Algorithm for Zero-Quantized DCT Coefficients in JPEG", *IEEE Int. Conf. on Acoustics Speech and Signal Processing (ICASP)*, pp. 1189-1192, Apr. 2008.

[70] J. Zhu, Z. Liu, D. Wang, "Fully Pipelined DCT/IDCT/Hadamard Unified Transform Architecture for HEVC Codecs", *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pp. 677-680, May 2013.

[71] W. Zhao, T. Onoye, T. Song, "High-Performance Multiplierless Transform Architecture for HEVC", *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pp. 1668-1671, May 2013.

[72] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, C. Yeo, "Efficient Integer DCT Architectures for HEVC", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 168-178, Jan. 2014.

[73] G. Pastuszak, "Hardware Architecture for the H.265/HEVC Discrete Cosine Transform", *IET Image Processing*, vol. 9, no. 6, pp. 468-477, June 2015.

[74] A. D. Darji, R. P. Makwana, "High-Performance Multiplierless DCT Architecture for HEVC", *Int. Symp. on VLSI Design and Test*, pp. 1-5, June 2015.

[75] Y. J. Ahn, W. J. Han, D. G. Sim, "Study of Decoder Complexity for HEVC and AVC Standarts Based on Tool-by-Tool Comparison", *SPIE Applications of Digital Image Processing XXXV*, vol. 8499, Aug. 2012.

[76] F. Bossen, B. Bross, K. Suhring, D. Flynn, "HEVC Complexity and Implementation Analysis", *IEEE Trans. on Circuits and Systems for Video Technology*, vol.22, no.12, pp.1685-1696, Dec. 2012.

[77] S. Shen, W. Shen, Y. Fan, X. Zeng, "A Unified 4/8/16/32-Point Integer IDCT Architecture for Multiple Video Coding Standards", *IEEE Int. Conf. on Multimedia and Expo (ICME)*, pp. 788-793, July 2012.

[78] J. S. Park, W. J. Nam, S. M. Han, 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, pp. 203-211, June 2012.

[79] M. Martuza, K. A. Wahid, "Low Cost Design of a Hybrid Architecture of Integer Inverse DCT for H.264, VC-1, AVS, and HEVC", *Journal of VLSI Design*, vol. 2012, no. 242989, March 2012.

[80] K. McCann, B. Bross, W.J. Han, I.K. Kim, K. Sugimoto, G. J. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 10 (HM 10) Encoder Description", *JCTVC-L1002*, March 2013.