HEVC VIDEO COMPRESSION HARDWARE DESIGNS

by

ERDEM ÖZCAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University

August 2013

HEVC VIDEO COMPRESSION HARDWARE DESIGNS

APPROVED BY

Assoc. Prof. Dr. İlker HAMZAOĞLU .....
(Thesis Supervisor)

Assoc. Prof. Dr. Ayhan BOZKURT .....

Assoc. Prof. Dr. Erkay SAVAŞ .....

DATE OF APPROVAL: 02.08.2013

HEVC VIDEO COMPRESSION HARDWARE DESIGNS

Erdem ÖZCAN

EE, MS Thesis, 2013

Thesis Supervisor: Assoc. Prof. Dr. İlker HAMZAOĞLU

Keywords: HEVC, Deblocking Filter, Intra Mode Decision, Hadamard Transform

Abstract

High Efficiency Video Coding (HEVC), a recently developed international standard for video compression, offers significantly better video compression efficiency than previous international standards. However, this coding gain comes with an increase in computational complexity.

Therefore, in this thesis, we first designed a high performance hardware architecture for deblocking filter algorithm used in HEVC standard. Two parallel datapaths are used in the hardware to increase its performance. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code is mapped to a Xilinx XC6VLX240T FPGA, and it is verified to work correctly on a Xilinx ML605 FPGA board which includes a Xilinx XC6VLX240T FPGA. The FPGA implementation can work at 108 MHz, and it can code 30 full HD (1920x1080) video frames per second.

We then proposed an energy reduction technique for Sum of Absolute Transformed Difference (SATD) based HEVC intra mode decision algorithm. We designed an efficient hardware architecture for SATD based HEVC intra mode decision algorithm including the proposed technique. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code is mapped to a Xilinx XC6VLX365T FPGA, and it is verified with post place & route simulations. The FPGA implementation can work at 116 MHz, and it can code 21 HD (1280x720) video frames per second. The proposed technique reduced its energy consumption up to 64.6% on this FPGA without any PSNR loss.

YVVK VİDEO SIKIŞTIRMA DONANIM TASARIMLARI

Erdem ÖZCAN

EE, Yüksek Lisans Tezi, 2013

Tez Danışmanı: Doç. Dr. İlker HAMZAOĞLU

Anahtar Kelimeler: YVVK, blok giderici filtre, çerçeve içi kip seçimi, hadamard dönüşümü

ÖZET

Yakın tarihte geliştirilmiş uluslararası bir standard olan Yüksek Verimlilikli Video Kodlama (YVVK), kendinden önceki standartlara göre belirgin şekilde daha iyi sıkıştırma verimi sunmaktadır. Ancak bu kodlama kazancı beraberinde işlem miktarında önemli bir artış getirmektedir.

Bu tezde ilk olarak YVVK video standardında kullanılan blok giderici filtre (BGF) algoritması için yüksek performanslı bir donanım mimarisi tasarlandı. Donanımın performansını artırmak için iki paralel veriyolu kullanıldı. Önerilen donanım Verilog HDL kullanılarak gerçeklendi. Verilog RTL kodu Xilinx XC6VLX240T FPGA'ne yerleştirildi ve Xilinx XC6VLX240T FPGA içeren bir Xilinx ML605 FPGA kartında doğrulandı. FPGA gerçeklemesi 108 MHz hızla çalışabilmekte ve saniyede 30 tam HD (1920x1080) çerçevesini kodlayabilmektedir.

Daha sonra, Mutlak Dönüşüm Fark Toplamı (MDFT) tabanlı YVVK çerçeve içi kip seçimi için özgün bir enerji azaltma tekniği önerildi. Önerilen tekniği de içeren MDFT tabanlı YVVK çerçeve içi kip seçimi için verimli bir donanım mimarisi tasarlandı. Önerilen donanım Verilog HDL kullanılarak gerçeklendi. Verilog RTL kodu Xilinx XC6VLX365T FPGA'ne yerleştirildi ve yerleştirme sonrası RTL simulasyonları ile doğrulandı. FPGA gerçeklemesi 116 MHz hızla çalışabilmekte ve saniyede 21 HD (1280x720) çerçevesini kodlayabilmektedir. Önerilen teknik, donanımın enerji tüketimini bu FPGA'da herhangi bir PSNR kaybı olmaksızın %64.6 azaltmıştır.

Acknowledgements

First and foremost I would like to thank my advisor Dr. İlker Hamzaoğlu for his invaluable guidance and support throughout my study. I appreciate very much for his suggestions, detailed reviews, invaluable advices and life lessons. He has been a great mentor to me and I feel privileged to be his student.

I am sincerely grateful to my thesis committee members Dr. Ayhan Bozkurt and Dr. Erkay Savaş for their invaluable feedback.

I would like to thank to all members of System-on-Chip Design and Testing Lab, Yusuf Adıbelli, Ercan Kalalı, Tevfik Zafer Özcan, Serkan Yalıman, Yusuf Akşehir, Kamil Erdayandı and Hasan Azgın who have been greatly supportive during my study. I also would like to thank Tarık Edip Kurt and Özge Arabacı for their friendship.

I would also like to express my deepest gratitude for my beloved family who always believed in me, and always tried their best to make things easier for me.

Finally I would like to acknowledge Sabancı University and TÜBİTAK for supporting me throughout my graduate education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASIC	:	Application Specific Integrated Circuit
BRAM	:	Block Ram
BS	:	Boundary Strength
CU	:	Coding Unit
DBF	:	Deblocking Filter
DFF	:	D Flip Flop
HD	:	High Definition
HEVC	:	High Efficiency Video Coding
HM	:	HEVC Test Model
HT	:	Hadamard Transform
IBUF	:	Input Buffer
ISO	:	International Standards Organization
ITU	:	International Telecommunications Union
LCU	:	Largest Coding Unit
LUT	:	Look-Up Table
MB	:	Macro Block
OBUF	:	Output Buffer
QP	:	Quantization Parameter
PE	:	Processing Element
PSNR	:	Peak Signal-to-Noise Ratio
PU	:	Prediction Unit
SATD	:	Sum of Absolute Transformed Difference
SPAD	:	Scratch Pad
SSD	:	Sum of Squared Difference
TU	:	Transform Unit
VCD	:	Value Change Dump

Chapter 1

INTRODUCTION

1.1 Motivation

Video compression systems are used in many commercial products, from consumer electronic devices such as digital camcorders, cellular phones to video teleconferencing systems. These applications make the video compression hardware devices an inevitable part of many commercial products. Since better coding efficiency is required for high resolution videos, recently, a new international standard for video compression is developed with the collaboration of ITU and ISO standardization organizations. This new standard, called High Efficiency Video Coding (HEVC), provides 50% bit rate reduction for equal perceptual video quality in comparison to H.264/AVC standard [1]. The video compression efficiency achieved in HEVC standard is not a result of any single feature but rather a combination of a number of encoding tools, and this coding gain comes with an increase in computational complexity. Because of its high coding efficiency, HEVC is expected to be widely used in many applications such as digital TV, mobile phones, video transmission in wireless networks, and video conferencing over the Internet.

The top-level block diagram of a HEVC Encoder is shown in Figure 1.1. As shown in this figure, HEVC encoder has a forward path and a reconstruction path. The forward path is used to encode a video frame by using intra and inter predictions and to create the bit stream. The reconstruction path is used to decode the encoded frame and to reconstruct the decoded frame. Since a decoder never gets original images, but rather works on the decoded frames, reconstruction path in the encoder ensures that both encoder and decoder use identical reference frames for intra and inter prediction. This avoids possible encoder – decoder mismatches [1, 2].

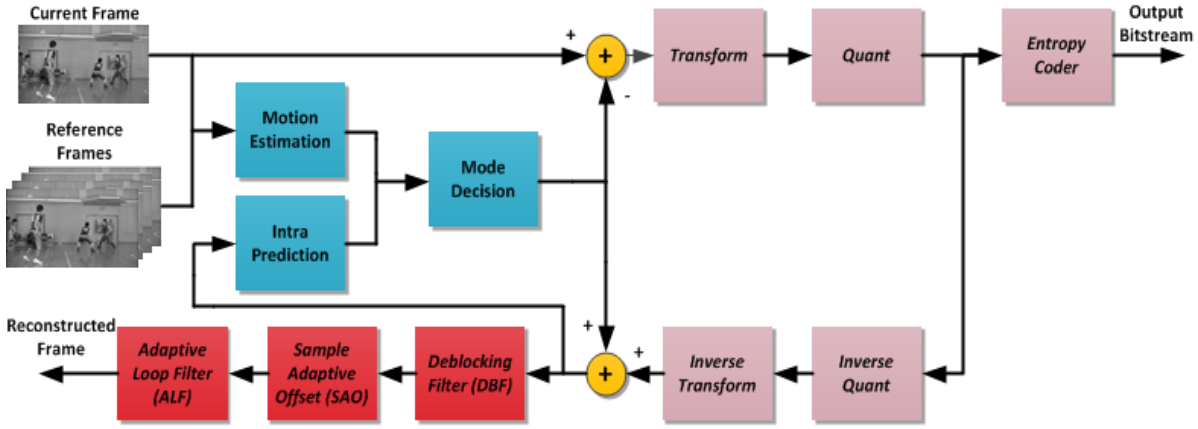

Figure 1.1: HEVC Encoder Block Diagram

In HEVC there is a quad tree structure which partitions the frame into Largest Coding Units (LCUs). LCUs can be recursively split into smaller Coding Units (CUs), which in turn can be split into small prediction units (PUs) and transform units (TUs) [3]. LCUs can be as large as 64x64 down to 16x16. LCU in HEVC is similar to that of a macroblock (MB) in the previous video coding standards.

Forward path starts with partitioning the input frame into LCUs. LCUs split into CUs. Each CU is encoded in intra or inter mode depending on the mode decision. In both intra and inter modes, the current CU is predicted from the reconstructed frame. Intra mode generates the predicted CU based on spatial redundancy, whereas inter mode, generates the predicted CU based on temporal redundancy. Mode decision compares the required amount of bits to encode a CU and the quality of the decoded CU for both of these modes and chooses the mode with better quality and bit-rate performance. In either case, intra or inter mode, the predicted CU is subtracted from the current CU to generate the residual CU. Residual CU is split into TUs and transformed using integer transforms. Transformed residual data is quantized and quantized transform coefficients are re-ordered in a zig-zag scan order. The reordered quantized transform coefficients are entropy encoded. The entropy-encoded coefficients together with header information, such as PU prediction mode and quantization step size, form the compressed bit stream.

Reconstruction path begins with inverse quantization and inverse transform operations. 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 in order to create the

reconstructed frame. The reconstructed frame is filtered by three in loop filters to smooth out artifacts induced by the block-wise processing and quantization.

Deblocking filter (DBF) is one of the in loop filters used in HEVC video encoder and decoder. In a coding scheme that uses block-based prediction and transform coding, discontinuities can occur in the reconstructed signal at block boundaries. Visible discontinuities at block boundaries are known as blocking artifacts. A major source of blocking artifacts is the block-transform coding of the prediction error followed by coarse quantization. Moreover, in the motion compensated prediction process, predictions for adjacent blocks in the current picture might not come from adjacent blocks in the previously coded pictures, which create discontinuities at the block boundaries of the prediction signal. Similarly, when applying intra prediction, the prediction process of adjacent blocks might be different causing discontinuities at the block boundaries of the prediction signal [4].

The main difficulty when designing a DBF algorithm is to decide whether or not to filter a particular block boundary, as well as to decide the strength of the filtering to be applied. Excessive filtering may lead to unnecessary smoothing of the picture details whereas lack of filtering may leave blocking artifacts which would reduce the subjective quality. Deciding whether to filter a block boundary should therefore depend on the characteristics of the reconstructed pixel values on both sides of that block boundary, and on coding parameters indicating whether it is likely that a blocking artifact has been created by coding process [4].

HEVC DBF algorithm is designed to improve both subjective and objective quality. Different from the H.264/AVC standard where DBF is applied on a 4x4 sample grid basis, HEVC applies DBF on an 8x8 sample grid which enables parallel processing by preventing cascading interactions between nearby filtering operations [1].

HEVC intra mode decision algorithm determines the best prediction mode for a block by using cost metrics such as Hadamard Transform (HT) based Sum of Absolute Transform Difference (SATD). In H.264, there are 9 intra prediction modes for 4x4 luminance (luma) blocks, and 4 intra prediction modes for 16x16 luma blocks [5], where as in HEVC, there are 18 modes for 4x4, 35 modes for 8x8, 35 modes for 16x16, 35 modes for 32x32 and 4 modes for 64x64 luma blocks [6]. Therefore, HEVC intra mode decision algorithm has much higher computational complexity than H.264/AVC intra mode decision algorithm.

1.2 Thesis Contribution

In this thesis, we first designed a high performance hardware architecture for deblocking filter algorithm used in HEVC standard. Two parallel datapaths are used in the hardware to increase its performance. The proposed hardware [31] is implemented in Verilog HDL. The Verilog RTL code is mapped to a Xilinx XC6VLX240T FPGA, and it is verified to work correctly on a Xilinx ML605 FPGA board which includes a Xilinx XC6VLX240T FPGA. The FPGA implementation can work at 108 MHz, and it can code 30 full HD (1920x1080) video frames per second.

We then proposed an energy reduction technique for Sum of Absolute Transformed Difference (SATD) based HEVC intra mode decision algorithm. We designed an efficient hardware architecture for SATD based HEVC intra mode decision algorithm including the proposed technique. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code is mapped to a Xilinx XC6VLX365T FPGA, and it is verified with post place & route simulations. The FPGA implementation can work at 116 MHz, and it can code 21 HD (1280x720) video frames per second. The proposed technique reduced its energy consumption up to 64.6% on this FPGA without any PSNR loss.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, first, introduces DBF algorithm used in HEVC standard. Then, it describes the proposed HEVC DBF hardware in detail and presents the implementation results.

Chapter 3, first, introduces intra prediction and intra mode decision algorithms used in HEVC standard. Then, it explains the proposed energy reduction technique. Finally, it describes the proposed HT based SATD hardware in detail and presents the implementation results.

Chapter 4 presents the conclusions and the future work.

Chapter 2

A HIGH PERFORMANCE DEBLOCKING FILTER HARDWARE FOR HIGH EFFICIENCY VIDEO CODING

HEVC, same as the previous video compression standards, divides video frames into blocks and performs transform and quantization for each block separately. This causes correlation loss between blocks and discontinuities on the edges of blocks. Therefore, reconstructed frames suffer from blocking artifacts. Deblocking filter (DBF) improves the visual quality of decoded frames by reducing visually disturbing blocking artifacts and discontinuities in a frame due to coarse quantization. Since the filtered frame is used as a reference frame for motion-compensated prediction of future frames, DBF also increases coding efficiency resulting in bit rate savings [4, 7, 8, 9].

HEVC DBF algorithm is applied to each edge of all luma and chroma blocks in a Largest Coding Unit (LCU), a 64x64 pixel array, after inverse quantization and inverse transform [4, 6]. In order to decide whether DBF will be applied to an edge or not, the related pixels in the current and neighboring 16x16 Coding Units (CU) must be read from memory and processed.

H.264 DBF algorithm has high computational complexity. H.264 DBF algorithm accounts for one-third of the computational complexity of an H.264 video decoder [7]. HEVC DBF algorithm also has high computational complexity. HEVC has higher computational complexity than H.264, and HEVC DBF algorithm accounts for one-fifth of the computational complexity of an HEVC video decoder [10].

Therefore, in this chapter, we propose the first HEVC DBF hardware in the literature. The proposed DBF hardware can be used as part of an HEVC video encoder or an HEVC video decoder. The proposed DBF hardware starts filtering the available edges after a new 64x64 LCU is ready. Two parallel datapaths are used in the hardware to increase its

performance. The proposed DBF hardware is implemented in Verilog HDL. The Verilog RTL code is verified to work at 108 MHz in a Xilinx Virtex 6 FPGA. The proposed HEVC DBF hardware can code 30 full HD (1920x1080) video frames per second.

The rest of the chapter is organized as follows. Section 2.1 presents a brief overview of HEVC DBF algorithm. Section 2.2 describes the proposed HEVC DBF hardware in detail. Section 2.3 presents the implementation results.

2.1 HEVC DBF Algorithm

HEVC DBF algorithm for an 8x8 block edge consisting of two segments is shown in Fig. 2.1. In HEVC, there is a quadtree structure [6]. Each video frame is divided into 64x64 LCUs in raster scan order, and each LCU is divided into 16x16 CUs as shown in Fig. 2.2. DBF is applied to edges of the 8x8 blocks in all 16x16 CUs. Each edge of an 8x8 block consists of 8 consecutive lines which are divided into two independent 4 line segments. Each line has 8 pixels along the edge. DBF can update up to 3 pixels in each direction that the filtering takes place.

First, vertical edges are filtered. Then, horizontal edges are filtered. There are several conditions that determine whether a segment will be filtered or not. There are additional conditions that determine the strength of the filtering for 16x16 CU edges that will be filtered. Strong or weak filtering can be applied to an edge depending on these conditions. Boundary strength (BS) parameter, quantization parameter (QP), β and tc threshold values and the values of the pixels in the edge determine the outcomes of these conditions, and the values of up to 3 pixels on both sides of an edge can be changed depending on the outcomes of these conditions.

Every edge is assigned a BS value depending on the coding modes and conditions of 16x16 CUs. The strength of the filtering done for an edge is proportional to its BS value. BS value can be 0, 1, or 2. No filtering is done for the edges with a BS value of 0, whereas strongest filtering is done for edges with a BS value of 2. BS decision is critical, since excessive filtering may lead to unnecessary smoothing of the picture details whereas lack of filtering may leave blocking artifacts which would reduce visual quality. The conditions used for determining the BS value for an edge between two neighboring 16x16 CUs are summarized in Table 2.1.

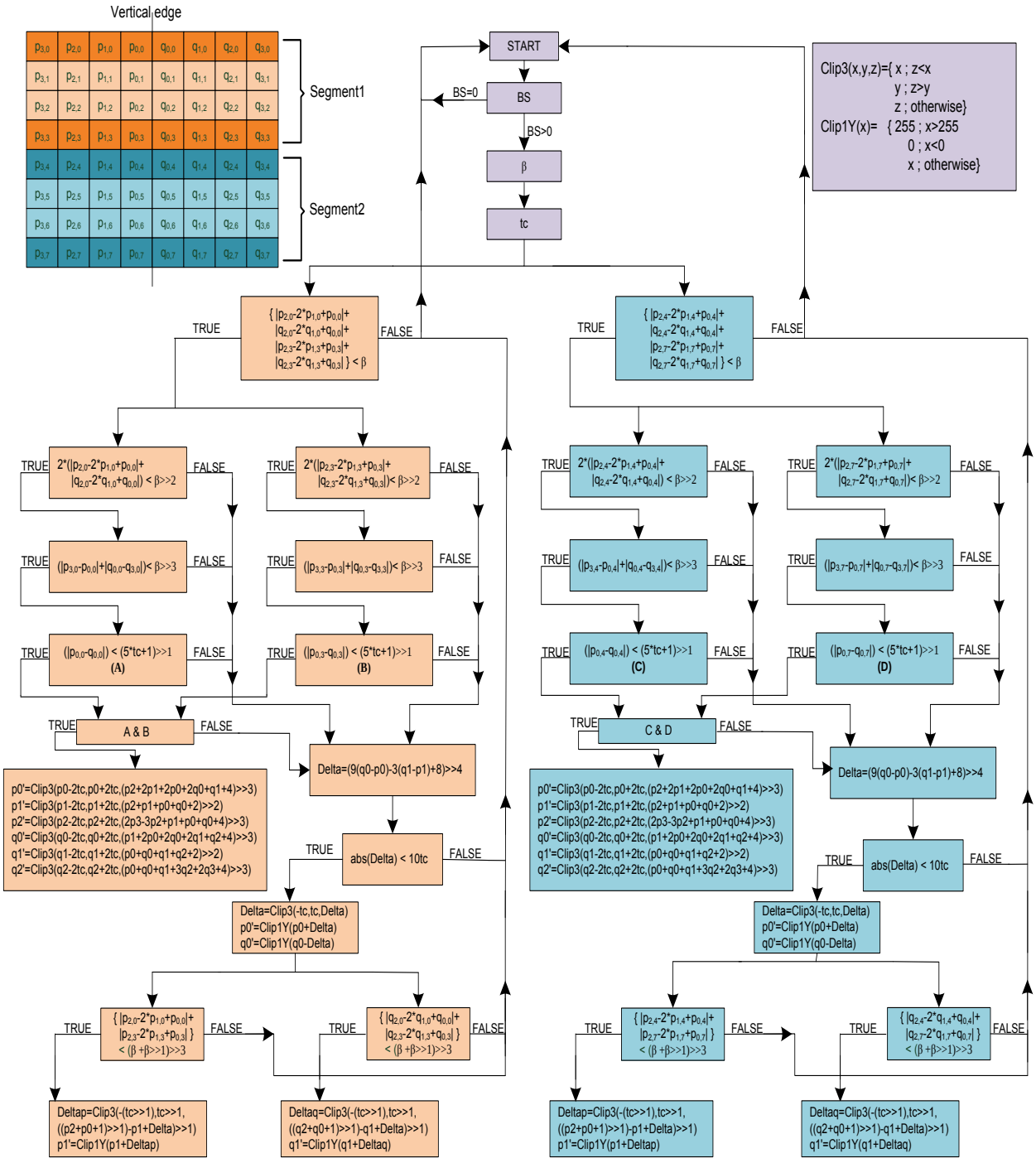

Figure 2.1: HEVC Deblocking Filter Algorithm

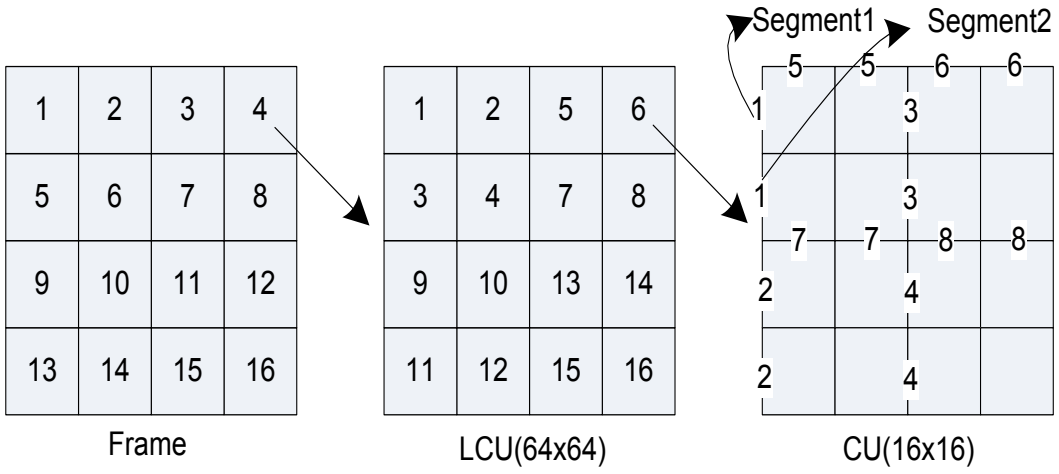

Figure 2.2: Edge Processing Order

Table 2.1: Conditions That Determine BS

Coding Modes and Conditions	BS
At least one of the blocks is Intra	2
At least one of the blocks has non-zero coded residual coefficient and boundary is a transform boundary	1
Absolute differences between corresponding spatial motion vector components of the two blocks are >= 1 in units of integer pixels	1
Motion compensated prediction for the two blocks refers to different reference pictures or the number of motion vectors is different for the two blocks	1
Otherwise	0

2.2 Proposed HEVC DBF Hardware

The proposed DBF hardware architecture is shown in Fig. 2.3. It includes two parallel datapaths, a control unit, a transpose memory, two input buffers to store the pixels in segment1 and segment2 of a CU, two dual port and four single port internal SRAMs to store partially filtered pixels, and two output buffers to store the filtered output pixels. In order to

process full HD video frames in real time, proposed DBF hardware reads 16 pixels in one clock cycle from external memory. Therefore, it fills the input pixel memory in 4 clock cycles. Since the decision process needs the first and fourth lines of each segment, input pixel memory is loaded with the pixels along the edge for subsequent filtering process.

DBF hardware starts filtering as soon as 64x64 LCU is ready. The two datapaths filter two segments, segment1 and segment2, in parallel. Transpose memory is used to transpose the filtered pixels before they are stored to intermediate or output SRAMs. This allows accessing 16 pixels in one clock cycle from transpose memory and simplifies reading the pixels from intermediate SRAMs.

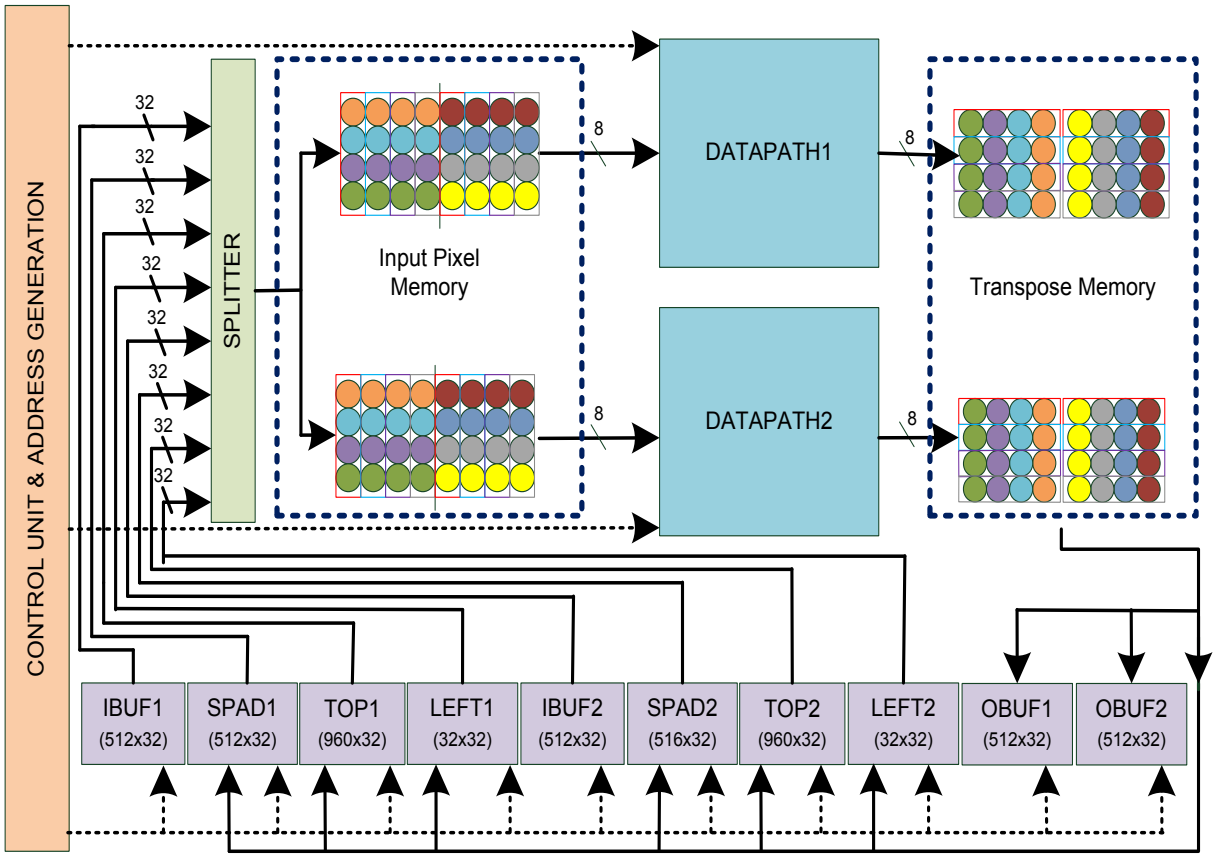

Figure 2.3: Proposed HEVC DBF Hardware

If an LCU is located in the left frame boundary, its left edges are not filtered. This causes an irregularity, and therefore increases the complexity of the control unit. In order to avoid this irregularity and therefore simplify the control unit, frame is extended at left boundary for 4 pixels as shown in Fig. 2.4. We assigned zero to these pixels and assigned zero to the BS values of these edges in order to avoid filtering these edges without causing an irregularity in the control unit.

Top and left memories are used to store the pixels in the leftmost and topmost edges of an LCU as shown in Fig. 2.4. In the MxN frame shown in Fig. 2.4, squares represent 64x64 LCUs and each LCU has sixteen 16x16 CUs. In order to filter an LCU, its top and left neighboring 4x64 and 64x4 blocks, shown as shaded small squares in Fig. 2.4, should be available. In order to reduce the amount of off-chip memory accesses and therefore reduce power consumption of the DBF hardware, top 64x4 blocks of all LCUs in a row of a frame, shown as lightly shaded small squares in Fig. 2.4, and left 4x64 blocks of the current LCU, shown as darkly shaded small squares in Fig. 2.4, are stored in on-chip SRAM memories. For full HD video frames, 1920x32/2 = 960x32 size 2 SRAM memories are used for storing top blocks, and 64x32/2 = 32x32 size 2 SRAM memories are used for storing left blocks.

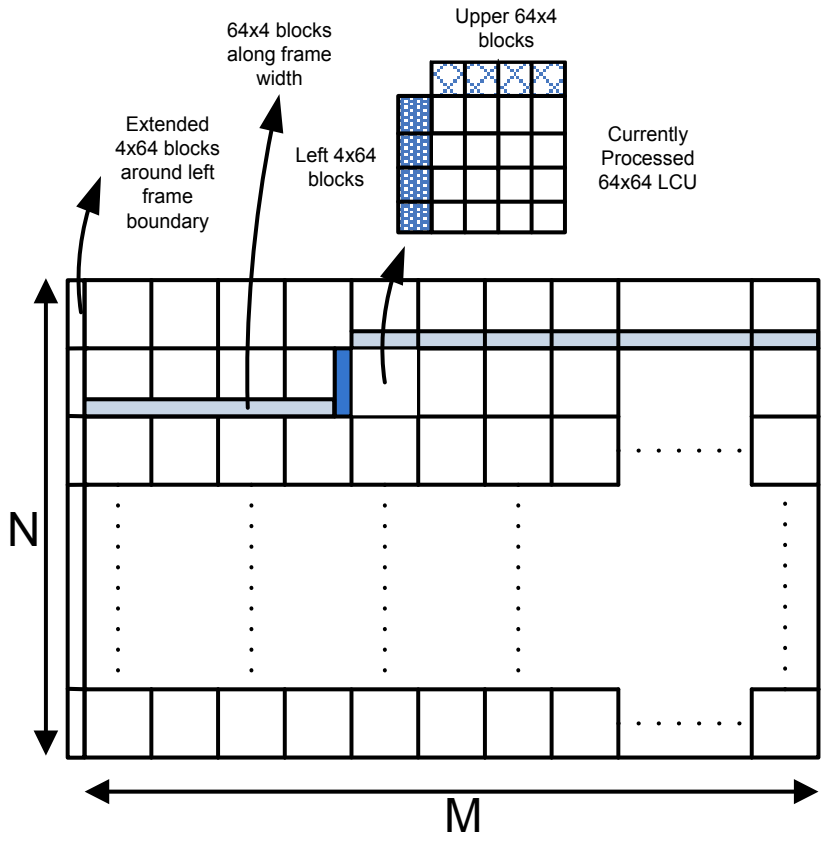

Figure 2.4: Pixels Stored in Top and Left Memories

The proposed DBF datapath is shown in Fig. 2.5. It can process 4 pixels, which are selected by the first four multiplexers, in parallel to increase the performance. The proposed datapath implements both the decision and filtering parts of HEVC DBF algorithm. Comparator1 is used for implementing the decision part. Comparator2 is used for implementing Clip3 function. Comparator3 and Comparator4 are used for implementing Clip1Y function. The filtered pixels are stored in outreg register.

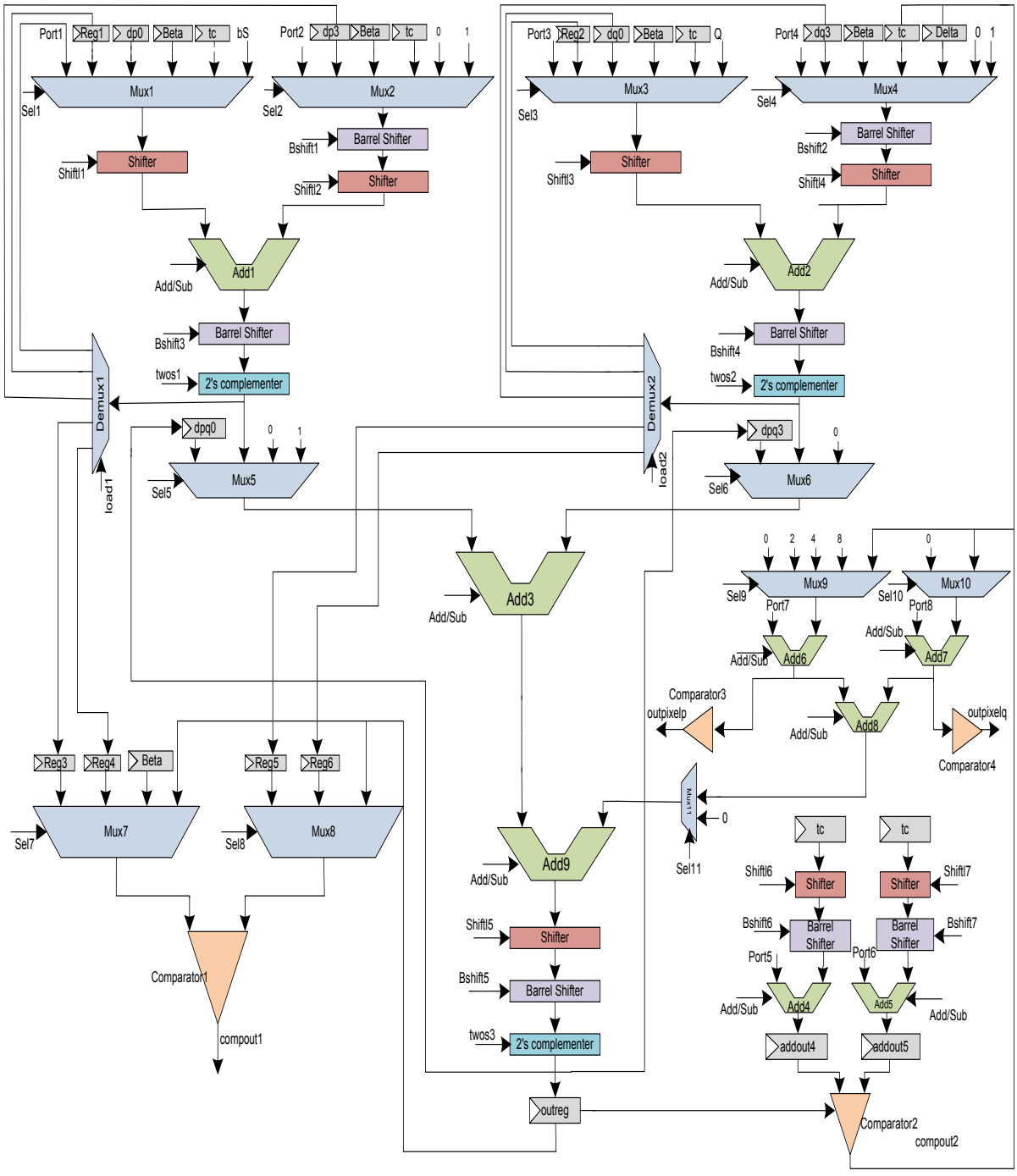

Figure 2.5: Proposed HEVC DBF Datapath

2.3 Implementation Results

The proposed HEVC DBF hardware is implemented in Verilog HDL. The implementation is verified with the RTL simulations using Mentor Graphics Modelsim SE. RTL simulation results matched the results of a software model of the HEVC DBF algorithm. The Verilog RTL code is synthesized and mapped to a XC6VLX130T-ff1156 Xilinx Virtex 6 FPGA with speed grade 3. The resulting netlist is placed and routed to the same FPGA using Xilinx ISE 11.5.

The FPGA implementation uses 5236 LUTs (6%), 1547 DFFs (1%) and 8 BRAMs (3%). BRAMs are implemented as dual-port block SelectRAMs. The FPGA implementation works at 108 MHz. It takes 7680 clock cycles in the worst-case to process an LCU. The FPGA implementation can process a full HD (1920x1080) video frame in 33.9 ms (480 LCUs x 7680 clock cycles per LCU x 9.2 ns clock cycle = 33.9 ms). Therefore, it can process 1000/33.9 = 30 full HD frames per second.

The FPGA implementation is verified to work correctly on a ML605 FPGA board which includes a Virtex 6 XC6VLX240T FPGA, 512 MB DDR RAM and 32 MB Flash memory, and interfaces such as UART and DVI. A software running on MicroBlaze processor is developed to transfer the inputs of the HEVC DBF hardware from a host computer in an appropriate order and to gather the outputs of the hardware for sending them back to the host computer and displaying the resulting frame on a monitor. HEVC DBF hardware is added as a peripheral to a bus where the MicroBlaze processor is the master. For this purpose HEVC DBF hardware is modified to be a slave peripheral for this data bus and 16 software accessible registers are added to the hardware. 11 of these registers are used by the software running on MicroBlaze for writing the inputs to the hardware and the other 5 are used for gathering the outputs and the status information from the hardware.

The software gets 1 blocky input frame from the host computer using the UART interface and writes it to a DDR RAM. Then, it loads the BRAMs of HEVC DBF hardware with the input pixels. After HEVC DBF hardware generates the done signal, the software reads the deblocked pixels by HEVC DBF hardware and writes them to the DDR RAM. This process is repeated for all the LCUs. After all the LCUs are processed, the deblocked frame is displayed on a monitor using the DVI interface of the FPGA board as shown in Fig. 2.6.

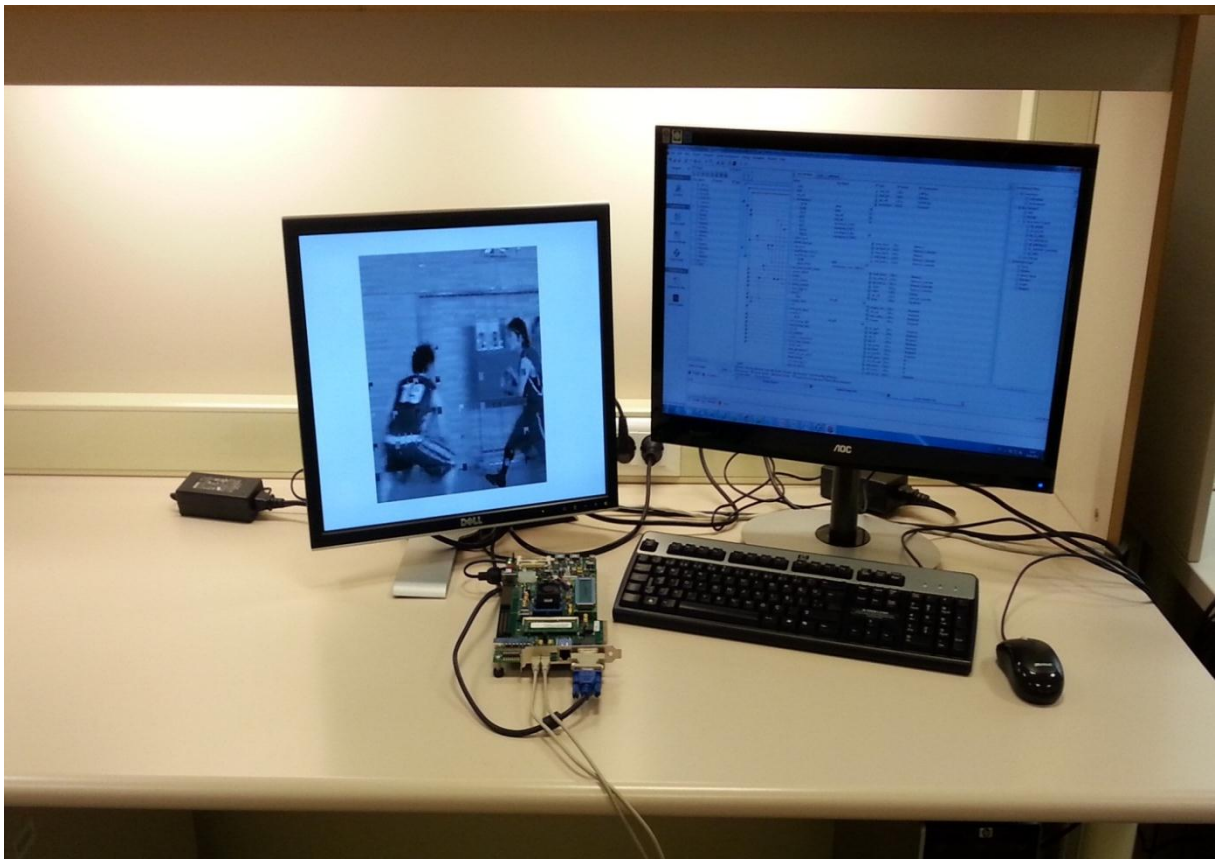

Figure 2.6: HEVC DBF Hardware FPGA Board Implementation

Since HEVC DBF algorithm is highly adaptive, amounts of strong and weak filtering operations performed for block edges differ from frame to frame. The amounts of strong and weak filtering operations performed for five different video sequences are shown in Fig. 2.7. All video sequences are intra coded and quantization parameter (QP) is 42. An example unfiltered video frame and the same frame filtered by HEVC DBF algorithm are shown in Fig. 2.8 and Fig. 2.9. As it can be seen from Fig. 2.9, some of the blocking artifacts are reduced and some of them are totally removed.

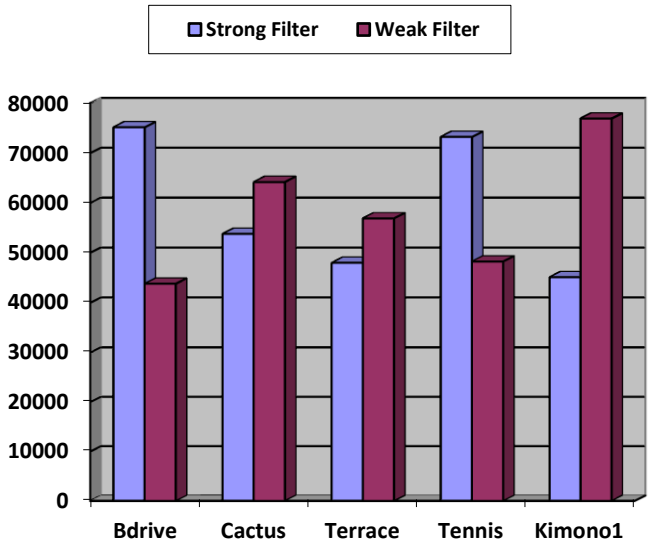

Figure 2.7: Strong and Weak Filter Amounts

Figure 2.8: Unfiltered Tennis (1920x1080) Video Frame

Figure 2.9: The Same Frame Filtered by HEVC DBF Algorithm

The power and energy consumptions of the FPGA implementation for several full HD (1920x1080) video frames are given in Table 2.2. The power consumption results are estimated using Xilinx XPower Analyzer tool. Post place & route timing simulations are performed for one frame of each video sequence at 50 MHz, and signal activities are stored in VCD files. These VCD files are used for estimating the power consumption of the FPGA implementation using Xilinx XPower Analyzer tool.

The Verilog RTL code of the proposed HEVC DBF hardware is also synthesized to Synopsys 90nm standard cell library using Synopsys Design Compiler and the resulting netlist is place & routed using Cadence SoC Encounter tool. The resulting ASIC layout is shown in Fig. 2.10. Gate count of the resulting ASIC implementation is calculated as 16.4k, excluding on-chip memories, based on NAND (2x1) gate area.

Table 2.2: Power and Energy Consumption Results

Category	Video Sequences				
	Basketball Drive	Cactus	Terrace	Tennis	Kimono1
Clock (mW)	7.63	7.61	7.63	7.62	7.62
Logic (mW)	11.44	11.72	11.55	11.15	11.86
Signal (mW)	25.44	26.26	25.83	25.03	26.72
BRAM (mW)	12.19	12.22	12.24	12.19	12.23
Total Power (mW)	56.70	57.81	57.25	55.99	58.43
Total Time (sec)	0.072	0.069	0.067	0.073	0.072
Energy (mJ)	4.082	3.988	3.835	4.087	4.206

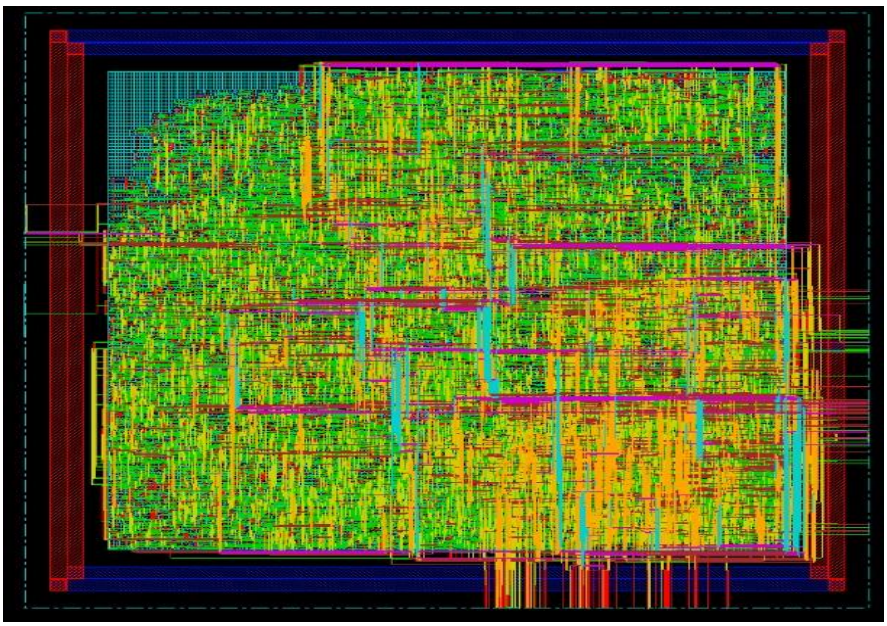

Figure 2.10: HEVC DBF ASIC Layout

In HEVC DBF algorithm, the pixels in the neighboring edges of 8x8 blocks do not overlap. Since the pixels in the neighboring edges can be filtered in parallel, depending on the application requirements, large number of parallel datapaths can be used in an HEVC DBF

hardware. The impact of parallel filtering on the proposed HEVC DBF hardware is shown in Table 2.3. The clock frequency for all cases is 108 MHz. As the number of parallel datapaths in HEVC DBF hardware increases, its performance increases significantly. However, this increases its gate count and on-chip memory usage. 640 byte on-chip memory is used for processing 16x16 CUs, and each parallel datapath uses 32 byte on-chip transpose memory.

Table 2.3: HEVC DBF Hardware Scalability Results

Parallel Datapaths	Cycles/CU (worst case)	Throughput (CU/sec)	1920x1080 fps	On-Chip Memory (Byte)	Gate Count
2	480	230k	30	640+64	16.4k
3	320	345k	43	640+96	21.5k
4	240	460k	57	640+128	26.6k
5	192	575k	72	640+160	31.7k
6	160	690k	86	640+192	36.8k

Since this is the first HEVC DBF hardware in the literature, we compared it with the H.264 DBF hardware in the literature. In order to make a fair comparison, we give its implementation results for processing 16x16 CUs. The comparison results are given in Table 2.4. However, this comparison is not perfect because of the following differences between HEVC and H.264 DBF algorithms.

Since the block sizes, conditions used to determine whether an edge will be filtered or not, conditions used to determine the strength of the filtering that will be applied to an edge, and the amount of computations performed in filtering operations are different, the amount of computations performed by HEVC DBF hardware and H.264 DBF hardware will be different for the same video frames. In HEVC DBF algorithm, 53% of the operations are performed in the decision part, and because of the data dependencies most of these operations are performed sequentially. However, this is not the case for H.264 DBF algorithm. Since the pixels in neighboring edges can be filtered in parallel in HEVC DBF algorithm, HEVC DBF hardware can use large number of parallel datapaths. However, this is not the case for H.264 DBF hardware. Because, the pixels in the neighboring edges of 4x4 blocks overlap in H.264 DBF algorithm.

Table 2.4: DBF Hardware Comparison

DBF Hardware	Technology	Memory Type	Cycles/MB (worst case)	Frequency (MHz)	Throughput (MB/sec)	Throughput (fps)	On-Chip Memory (Byte)	Gate Count
Proposed HEVC DBF Hardware	Xilinx Virtex 6 FPGA	dual port SRAM	480	108	230k	1920x1080 30 fps	640 + 64 = 704	16.4k (ASIC)
Huang [11]	0.25 um CMOS ASIC	two port SRAM	614	100	163k	1920x1080 20 fps	640	20.6k
Huang [11]	0.25 um CMOS ASIC	single port SRAM	878	100	114k	1920x1080 14 fps	640	18.9k
Sheng [12]	0.25 um CMOS ASIC	dual port SRAM	446	100	224k	1920x1080 28 fps	64x32 + 2x96x32 = 1024	24k
Parlak [13]	Xilinx Virtex 2 FPGA	dual port SRAM	5544	72	13k	352x288 33 fps	1792	5.3k
Shih [14]	0.25um CMOS ASIC	two port SRAM	646	100	154k	1920x1080 19 fps	160x32 + 32 = 672	18.7k
Liu [15]	0.18um CMOS ASIC	single port SRAM	250	100	400k	1920x1080 49 fps	96x32 + 2Nx32	19.6k
Chao [16]	0.18um CMOS ASIC	two port SRAM	228	100	369k	2048xl536 30 fps	144x32 + 2x16x32 = 704	16.6k
Shih [17]	0.18um CMOS ASIC	single port SRAM	246	100	406k	1920x1080 50 fps	512 + 12N	20.9k

Chapter 3

A COMPUTATION AND ENERGY REDUCTION TECHNIQUE FOR HEVC INTRA MODE DECISION

HEVC intra mode decision algorithm has a huge computational complexity. HEVC intra prediction algorithm predicts the pixels in prediction units (PU) of a coding unit (CU), which is similar to macroblock (MB) in H.264, from the pixels of its already coded and reconstructed neighboring PUs. In H.264, there are 9 intra prediction modes for 4x4 luminance blocks, and 4 intra prediction modes for 16x16 luminance blocks [5]. In HEVC, there are 18 modes for 4x4, 35 modes for 8x8, 35 modes for 16x16, 35 modes for 32x32 and 4 modes for 64x64 luminance PUs [6, 18]. The number of HEVC intra prediction modes for a 64x64 luminance CU is approximately 3.2 times larger than H.264. In order to determine the best HEVC intra prediction mode for the luminance component of a 64x64 CU, predictions for 7552 intra prediction modes should be calculated.

The intra mode decision algorithm implemented in HEVC HM reference software encoder [19] uses Sum of Absolute Transformed Difference (SATD) based cost function. Fig. 3.1 shows the amount of addition operations performed by SATD calculations in HEVC and H.264 intra mode decisions. Because of the larger PU sizes and more intra prediction modes, 24 times more addition operations are performed for SATD calculation in HEVC intra mode decision than SATD calculation in H.264 intra mode decision. Therefore, in this thesis, we proposed a computation and energy reduction technique for SATD calculation in HEVC intra mode decision.

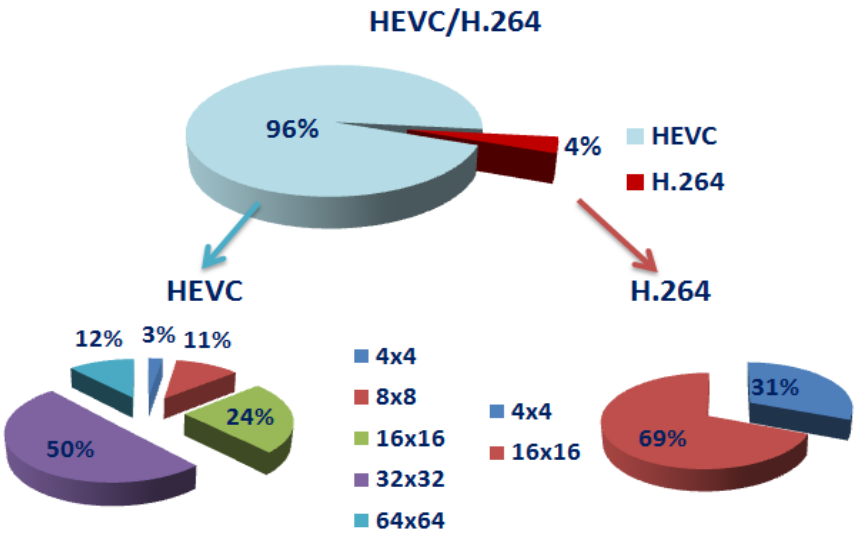

Figure 3.1: Addition Amounts in HEVC and H.264 SATD Calculations

The proposed technique reduces the number of additions performed by SATD calculations in HEVC intra mode decision algorithm used in HEVC HM reference software encoder [19] for 4x4 and 8x8 luminance intra prediction modes by 54% and 70% respectively without any PSNR loss. Since 94% of intra predicted blocks are predicted by 4x4 and 8x8 PU sizes [21], we showed the impact of the proposed technique for 4x4 and 8x8 PUs. But, it can also be used for 16x16, 32x32 and 64x64 PUs.

We designed efficient hardware architectures for both the original HEVC SATD calculation and HEVC SATD calculation with the proposed technique for 4x4 and 8x8 PUs. The proposed hardware architectures are implemented in Verilog HDL. The proposed technique reduced the energy consumption of the original HEVC SATD calculation hardware up to 64.6%.

A similar energy reduction technique is proposed for H.264 intra mode decision in [22]. However, the proposed technique is applied to HEVC intra mode decision and it includes an additional optimization to further reduce the energy consumption. There are several H.264 intra prediction and intra mode decision hardware implementations in the literature [23, 24, 25, 26]. There are a few HEVC intra prediction hardware implementations in the literature [21, 28]. A HEVC intra mode decision hardware only for 4x4 PU size is presented in [27]. However, no energy reduction technique is used in this hardware, and its power consumption is not reported.

3.1 HEVC Intra Prediction and Mode Decision Algorithms

HEVC intra prediction algorithm predicts the pixels in PUs of a CU using the pixels in the available neighboring PUs. For the luminance component of a frame, 4x4, 8x8, 16x16, 32x32 and 64x64 PU sizes are available. There are 16 angular prediction modes for 4x4 PU size, 33 angular prediction modes for 8x8, 16x16 and 32x32 PU sizes, and 2 angular prediction modes for 64x64 PU size. In addition to angular prediction modes shown in Fig. 3.2, there are DC and planar prediction modes for all PU sizes [6]. Fig. 3.2 shows the intra prediction angles and intra prediction modes corresponding to these intra prediction angles. Angles 0, 5, 13, 21 and 32 are used to predict 4x4 PUs. Angles 0, 2, 5, 9, 13, 17, 21, 26 and 32 are used to predict 8x8, 16x16 and 32x32 PUs. 64x64 PUs are predicted only with angle 0.

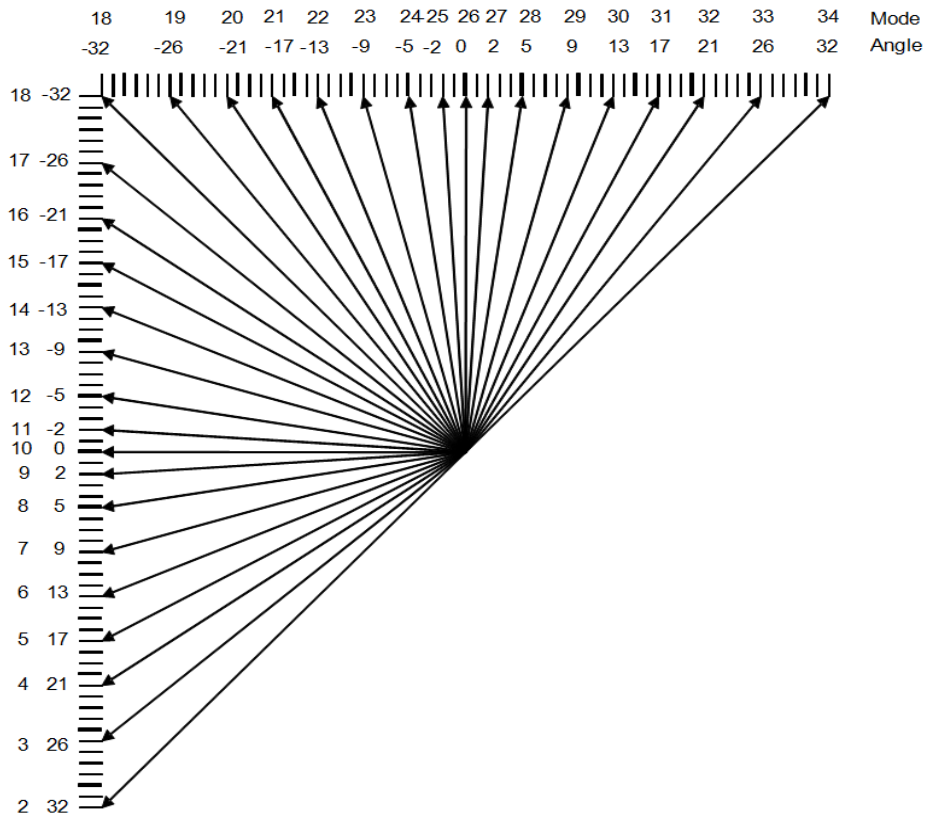

Figure 3.2: HEVC Intra Prediction Mode Directions

HEVC intra mode decision algorithm implemented in HEVC HM reference software encoder is shown in Fig. 3.3 [29]. This mode decision algorithm uses two cost functions; Sum of Absolute Transformed Difference (SATD) based Hadamard cost function shown in (3.1), and Sum of Squared Difference (SSD) based Rate Distortion (RD) cost function shown in (3.2). Hadamard cost function estimates distortion as SATD and rate as the number of bits used for encoding the prediction mode. RD cost function calculates the actual distortion after coding based on SSD and the actual bit rate used after coding. λ is calculated based on Quantization Parameter (QP).

$$J_{HAD} = SATD + \lambda R \quad (3.1)$$

$$J_{RDO} = SSD + \lambda R \quad (3.2)$$

This mode decision algorithm determines the best PU size, transform unit (TU) size and intra prediction mode of a CU as follows. First, SATD values for each intra prediction mode of each PU for the largest PU size are calculated as follows. Find residue block by subtracting intra predicted block from current block, apply Hadamard Transform (HT) to the residue block, and add the absolute values of the transformed residues. Then, 8 candidate modes for 4x4 and 8x8 PUs and 3 candidate modes for 16x16, 32x32 and 64x64 PUs with minimum Hadamard cost function value are selected as candidate modes for each PU. After that, for each PU, the most selected candidate modes for neighboring PUs are compared with the candidate modes selected for the current PU and up to 3 additional modes from neighboring PUs are added to the candidate modes of the current PU. Then, RD costs of each candidate mode of each PU are calculated using the cost function in (3.2) and the best mode with minimum RD cost is selected. After that, for each PU, RD cost of its best mode is calculated with TU sizes from 4x4 to 32x32 and best TU size with minimum RD cost is also selected. This process is repeated for each PU size of the CU from largest to smallest, and the best PU size, TU size and intra prediction mode for the CU with minimum RD cost are selected.

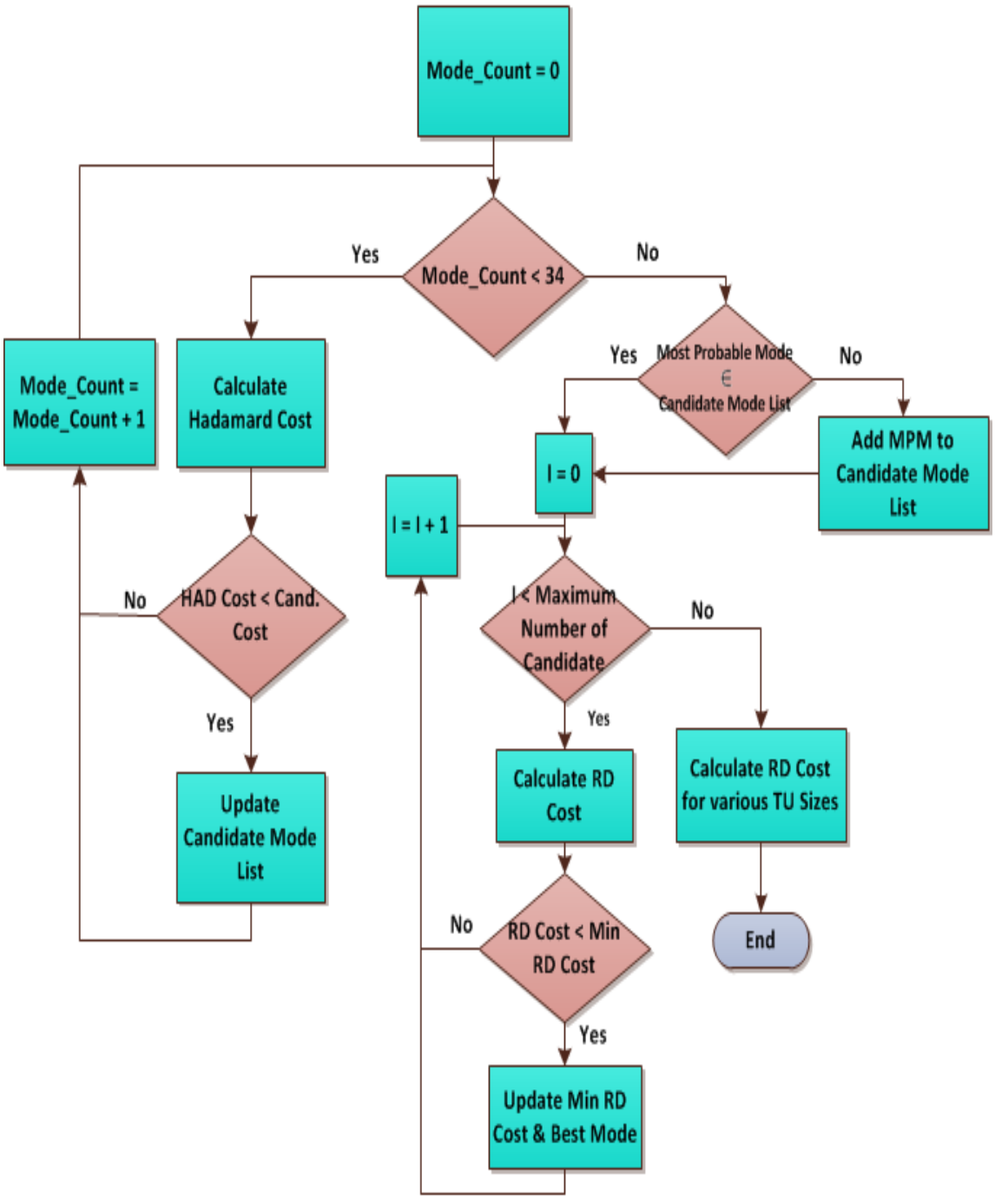

Figure 3.3: Intra Mode Decision Algorithm in HEVC HM Software Encoder

3.2 Proposed Computation and Energy Reduction Technique

HT is a linear operation and it can be applied before subtraction operation as shown in (3.3). H, C and P shown in (3.3) are Hadamard matrix, current block, and predicted block, respectively. 8x8 Hadamard matrix is shown in (3.4). Instead of applying HT after subtraction operation, we applied HT before subtraction operation. Applying HT before subtraction requires performing two HTs instead of one. However, this decreases the computational complexity of SATD based HEVC intra mode decision. Since the intra predicted blocks have regular patterns, HTs of the predicted blocks (H*P*H') can be calculated with a small amount of computation. In addition, since HT of the current block (H*C*H') is common to all intra prediction modes, it can be calculated only once.

$$T = H * (C - P) * H' = (H * C * H') - (H * P * H') \tag{3.3}$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{3.4}$$

The predicted block pattern of horizontal mode and the result of performing HT for this predicted block pattern are shown in Fig. 3.4 for 8x8 PU size. SATD of an 8x8 block including HT can be calculated with 959 additions. However, SATD of an 8x8 block predicted by horizontal mode including HT can be calculated with 95 additions and 8 shifts as shown in Fig. 3.4. Similarly, SATD of an 8x8 block predicted by vertical mode and all angle 2 modes including HT can be calculated with 95 additions and 8 shifts. Therefore, the proposed technique significantly reduces the number of additions performed by SATD calculation.

$$\begin{bmatrix} a & a & a & a & a & a & a & a \\ b & b & b & b & b & b & b & b \\ c & c & c & c & c & c & c & c \\ d & d & d & d & d & d & d & d \\ e & e & e & e & e & e & e & e \\ f & f & f & f & f & f & f & f \\ g & g & g & g & g & g & g & g \\ h & h & h & h & h & h & h & h \end{bmatrix} \rightarrow \begin{bmatrix} 8(a+b+c+d+e+f+g+h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a-b+c-d+e-f+g-h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a+b-c-d+e+f-g-h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a-b-c+d+e-f-g+h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a+b+c+d-e-f-g-h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a-b+c-d-e+f-g+h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a+b-c-d-e-f+g+h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8(a-b-c+d-e+f+g-h) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3.4: Hadamard Transform of Horizontal Mode

We applied the proposed technique to all 4x4 intra prediction modes except planar and DC modes, and all 8x8 intra prediction modes of angles 2, 5, 13, 17 and vertical and horizontal modes. Therefore, we applied the proposed technique to 16 4x4 modes and 18 8x8 modes. Since the other modes have relatively irregular prediction patterns, the proposed technique achieves small amount of computation reduction for these modes. In order to have a less complex and smaller SATD calculation hardware, we did not apply the proposed technique to these prediction modes. Instead, for these prediction modes, we used the original HT operation which is applying HT after subtraction operation.

We determined the computation reductions achieved by the proposed technique and presented the results in Table 3.1. The columns labeled I show the amount of computations performed by the original HT operation and the columns labeled II show the amount of computations performed by the HT operation using the proposed technique. The proposed technique reduced the number of additions performed by HT operation for 4x4 and 8x8 luminance intra prediction modes by 54% and 70% respectively without any PSNR loss. The results show that the proposed technique significantly reduces the computational complexity of SATD based HEVC intra mode decision.

The proposed technique reduces the amount of computations because of two reasons. First, as shown in Fig. 3.4, most of the values in HT of intra predicted blocks are zero. Therefore, there is no need to calculate these values. Second, since intra predicted blocks have regular patterns, some of the values in HT of intra predicted blocks are the same. Therefore, these values are calculated only once. For example, the values in sixth row of HT of an 8x8 block predicted by an 8x8 intra prediction mode of angle 17 is shown in Fig 3.5. The first line gives the first value in the row, and so on. Since some of the values are the same, they are calculated only once.

Table 3.1: Computation Reductions for Intra Prediction Modes

	Prediction Angles	Hadamard Transform				Residue	
		Addition		Shift		Subtraction	
		I	II	I	II	I	II
Intra 4x4	**5(4 modes)**	444	108	0	16	64	64
	13(4 modes)	444	188	0	128	64	64
	21(4 modes)	444	332	0	64	64	64
	32(2 modes)	222	134	0	44	32	32
	Vertical	111	27	0	4	16	16
	Horizontal	111	27	0	4	16	16
	Total	**1776**	**816**	**0**	**260**	**256**	**256**
Intra 8x8	**2(4 modes)**	3836	160	0	32	256	32
	5(4 modes)	3836	628	0	128	256	256
	13(4 modes)	3836	1740	0	688	256	256
	17(4 modes)	3836	2372	0	680	256	256
	Vertical	959	95	0	32	64	64
	Horizontal	959	95	0	32	64	64
	Total	**17262**	**5090**	**0**	**1592**	**1152**	**928**

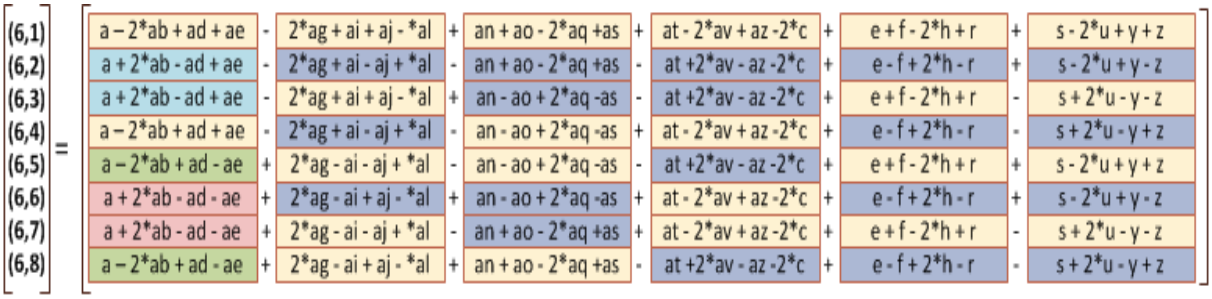

Figure 3.5: Sixth Row of HT of 8x8 Intra Predicted Block

3.3 Proposed Hardware Architecture

We designed two different hardware architectures for SATD calculation in HEVC intra mode decision for 4x4 and 8x8 PU sizes. The first hardware implements the original SATD calculation. Therefore, it first subtracts predicted block from current block, and then performs HT. The second hardware implements the SATD calculation with the proposed technique. Therefore, it first performs HT for predicted block and current block, and then performs subtraction.

The hardware architecture implementing the original SATD calculation has two 8 parallel datapaths in order to increase its throughput. The hardware architecture with one 8 parallel datapaths is shown in Fig. 3.6. One of these datapaths is shown in Fig. 3.7. Input pixels are stored in IBUF input buffer. First, predicted block pixels are subtracted from current block pixels. Then, addition or subtraction operation is performed depending on HT matrix. Since HT matrix is multiplied with the residue block both from left and right side as shown in (3.3), the results of the left side multiplication are stored in transpose registers as shown in Fig. 3.6. For 8x8 PU size, in each clock cycle, the values in one column of H*(C-P) are calculated by 8 parallel datapaths. Therefore, H*(C-P) is calculated in 8 clock cycles. Then, right side multiplication is performed. In each clock cycle, the values in one row of H*(C-P)*H´ are calculated by the same 8 parallel datapaths. Therefore, H*(C-P)*H´ is calculated in 8 clock cycles using the same 8 parallel datapaths. Then, absolute values are calculated and stored in transpose memory. Finally, SATD value is calculated by adding the absolute values using the last datapath. The original SATD calculation hardware calculates SATD values of all 4x4 and 8x8 intra prediction modes in 879 clock cycles.

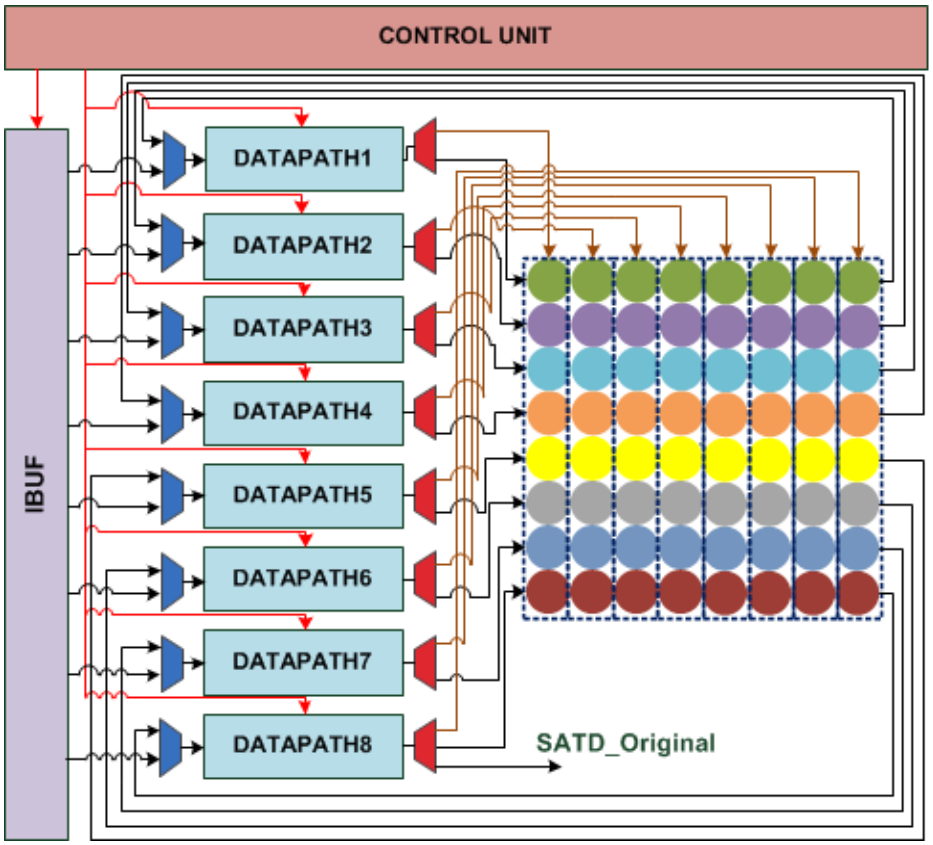

Figure 3.6: Original SATD Calculation Hardware

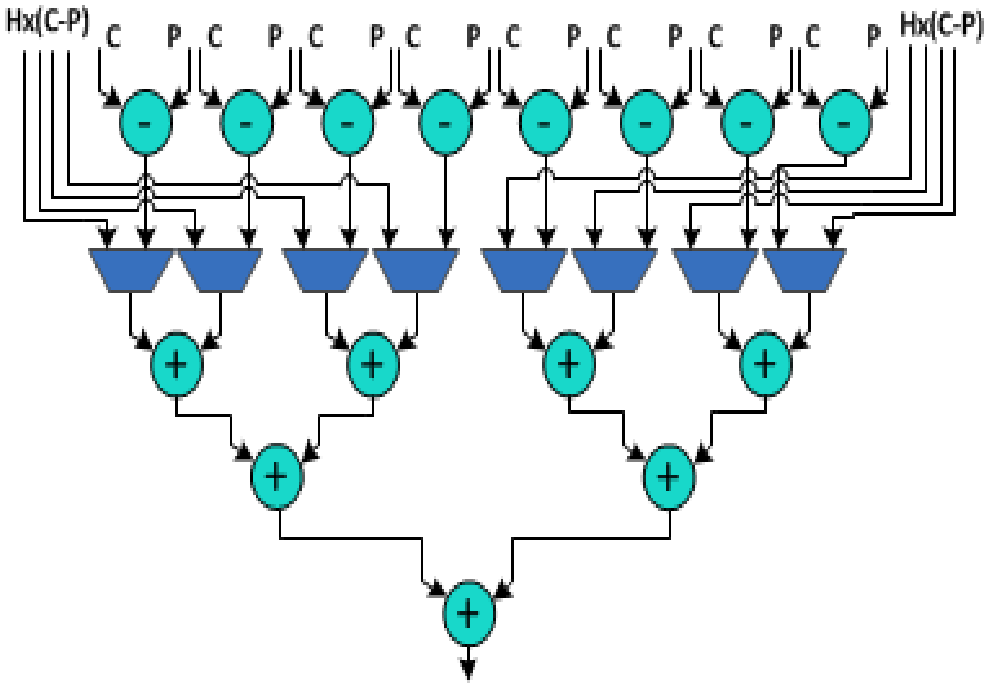

Figure 3.7: Datapath for Original SATD Calculation Hardware

The hardware architecture implementing the SATD calculation with the proposed technique is shown in Fig. 3.8. Parallel processing elements (PEs) are used in the hardware in order to increase its throughput. As it is shown in Fig. 3.9, each PE only has 3 adders and 4 multiplexers. HTs for 4x4 intra prediction modes are calculated using 4 PEs. HTs for 8x8 intra prediction modes of angles 2, 5 and vertical and horizontal modes are calculated using one PE. HTs for 8x8 intra prediction modes of angles 13 and 17 are calculated using 4 PEs. Since the proposed technique is not applied to some intra prediction modes, as it is shown in Fig. 3.8, the hardware also includes original SATD calculation hardware with 8 parallel datapaths in order to calculate the SATDs for these intra prediction modes.

Architecture of 4 PEs is shown in Fig. 3.10. Since there is no matrix multiplication in the proposed technique, there is no transpose memory in this hardware. First, predicted pixels are stored in IBUF input buffer. Then, each PE reads 4 pixels from IBUF and performs operations of HT. The outputs of PEs are stored either in SPAD for performing further operations of HT or in OBUF output buffer. IBUF, SPAD and OBUF are implemented as BlockRAMs.

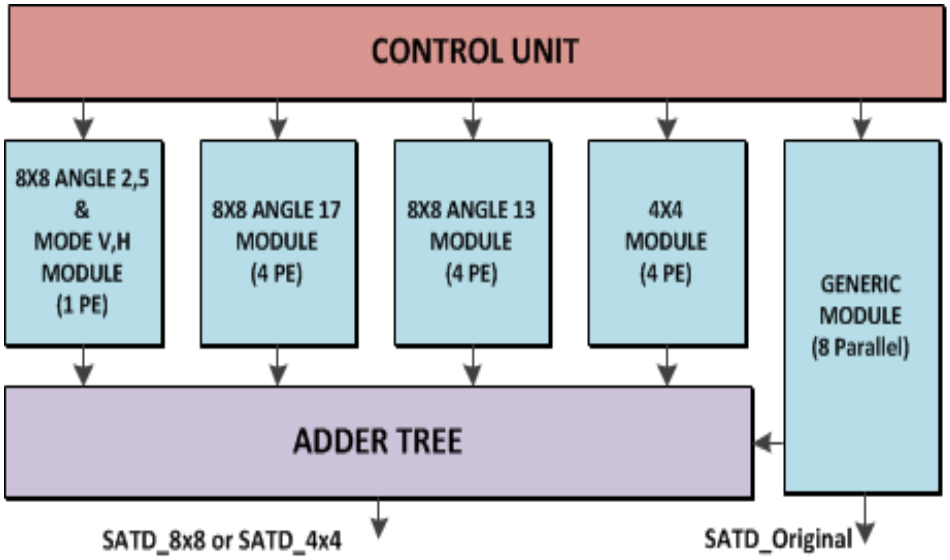

Figure 3.8: SATD Calculation Hardware with Proposed Technique

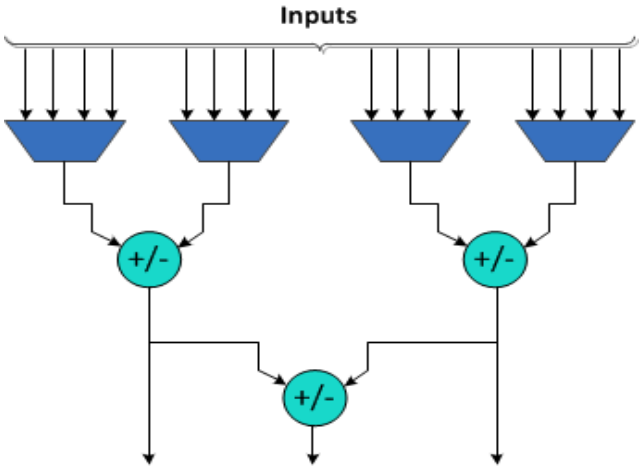

Figure 3.9: Processing Element (PE) Architecture

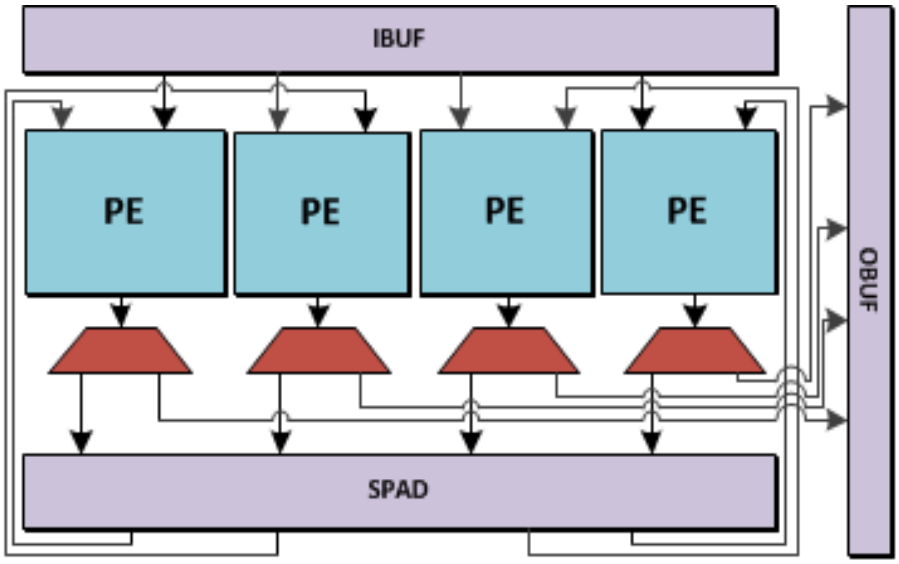

Figure 3.10: Architecture of 4 PEs

4 PEs used for performing HTs of 4x4 intra predicted blocks perform HTs of four 4x4 blocks in an 8x8 block sequentially. These 4 PEs are divided into two groups. Each group has 2 PEs and the PEs in a group perform HTs of 4x4 blocks predicted by the same intra prediction modes. HTs of 4x4 and 8x8 current blocks are calculated once in 8 parallel original SATD calculation hardware and stored. Then, SATD values are calculated by subtracting HT of intra predicted blocks from HT of current block and adding absolute values of the results using the adder tree shown in Fig. 3.11. Since 56 of 64 values in the HT of 8x8 blocks predicted by 8x8 intra prediction modes of angle 2 and horizontal mode are zero, these zero values are not subtracted from HT of current block in order to reduce the power consumption.

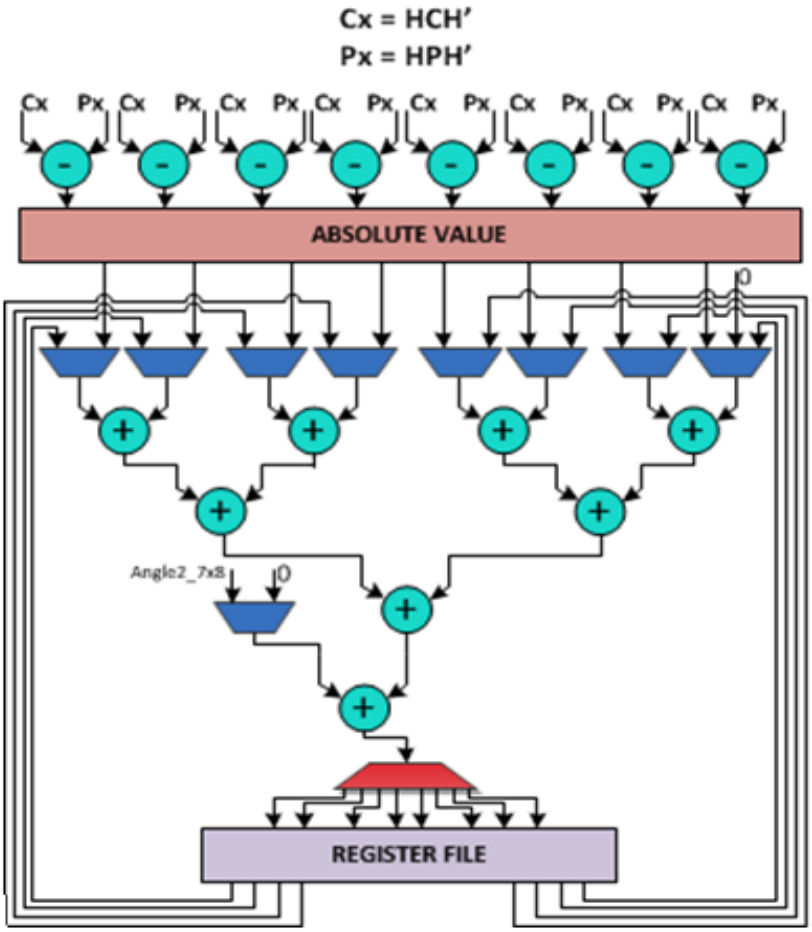

Figure 3.11: Adder Tree Architecture

Since, only one adder tree is used to reduce hardware area, the adder tree operations are scheduled to use this adder tree hardware efficiently. HT flow and adder tree scheduling for an 8x8 PU for 4x4 intra prediction modes and 8x8 intra prediction modes of angles 2, 5, 13, 17 and vertical and horizontal modes are shown in Fig. 3.12. Adder tree calculates SATD value for each 4x4 intra prediction mode and 8x8 intra prediction mode in 5 and 9 clock cycles respectively. Therefore, it takes 330 clock cycles to calculate SATD values of 4x4 and 8x8 intra prediction modes for which the proposed technique is applied. It takes 400 clock cycles to calculate SATD values of the intra prediction modes for which the proposed technique is not applied. Therefore, SATD calculation hardware with the proposed technique calculates SATD values of all 4x4 and 8x8 intra prediction modes in 400 clock cycles. Since PEs and adder tree has to wait for 70 clock cycles before processing the next 8x8 block, they are clock gated in order to reduce power consumption.

3.4 Implementation Results

Both the original HEVC SATD calculation hardware and HEVC SATD calculation hardware with the proposed technique are implemented in Verilog HDL. The implementations are verified with RTL simulations using Mentor Graphics Questa. RTL simulation results matched the SATD values calculated by HEVC HM reference software encoder 7.0. The Verilog RTL codes are synthesized to a XC6VLX365T-ff1759 Xilinx Virtex 6 FPGA with speed grade 3. The resulting netlists are placed and routed to the same FPGA using Xilinx ISE 13.4. Both FPGA implementations are verified with post place&route simulations as well.

Both FPGA implementations work at 116 MHz. There are 14080 8x8 blocks in an HD (1280x720) frame. FPGA implementation of the original HEVC SATD calculation hardware can process one HD frame (1280x720) in 106.4 msec. (14080 8x8 blocks x 879 clock cycles per 8x8 block x 8.6 ns clock cycle = 106.4 msec). Therefore, it can process 1000/106.4 = 9 HD frames per second. HEVC SATD calculation hardware with the proposed technique calculates SATD values of all 4x4 and 8x8 intra prediction modes in 400 clock cycles. Its FPGA implementation can process one HD (1280x720) frame in 48.4 msec. (14080 8x8 blocks x 400 clock cycles per 8x8 block x 8.6 ns clock cycle = 48.4 msec). Therefore, it can process 1000/48.4 = 21 HD frames per second. Therefore, the proposed technique significantly increases the performance of SATD calculation hardware.

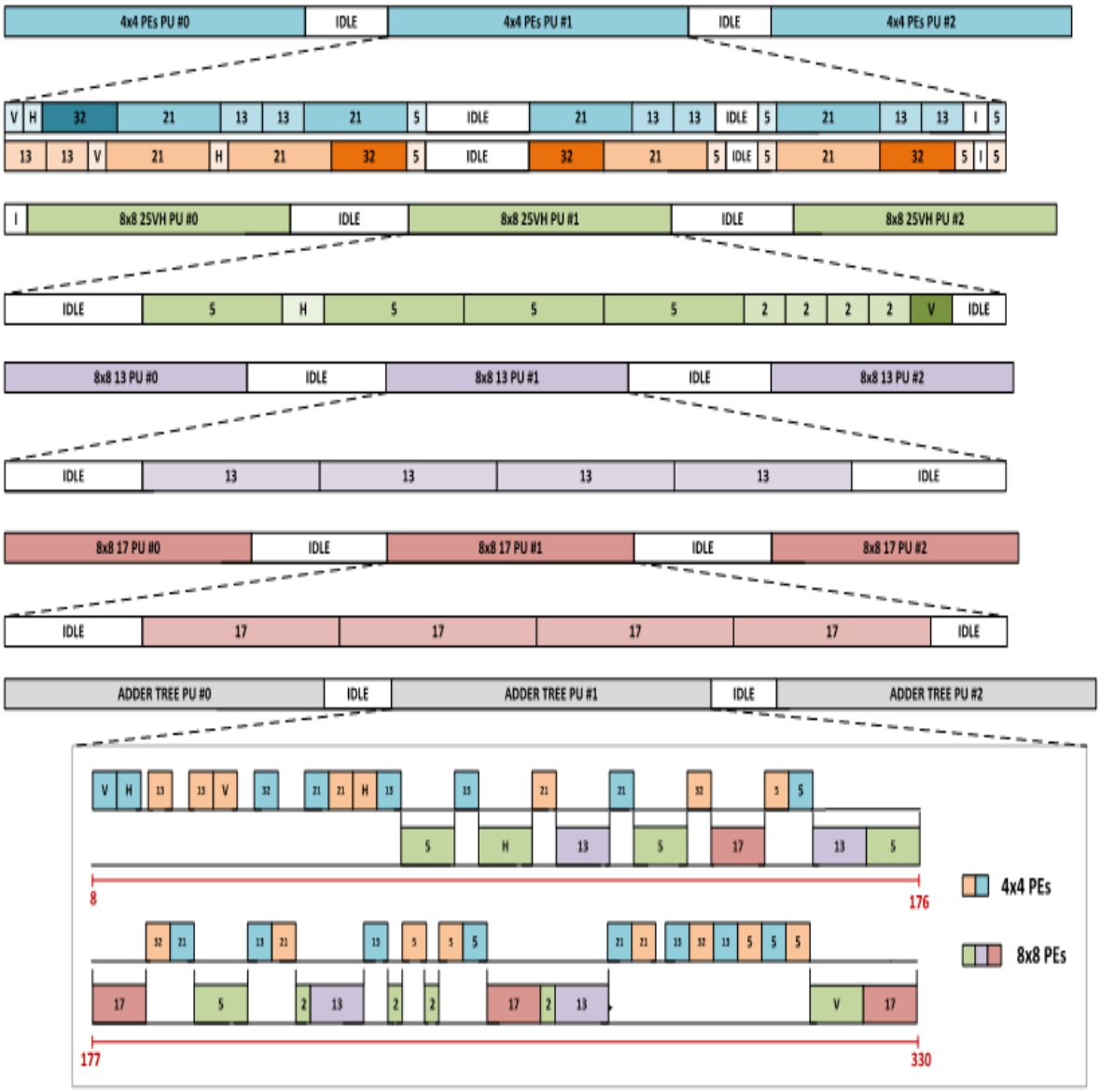

Figure 3.12: HT Flow and Adder Tree Scheduling

FPGA implementation of the original HEVC SATD calculation hardware uses 6909 Slices (12%), 20473 LUTs (8%), 3504 DFFs (1%) and 8 BRAMs (1%). FPGA implementation of the HEVC SATD calculation hardware with the proposed technique uses 6247 Slices (10%), 19227 LUTs (8%), 2184 DFFs (1%) and 40 BRAMs (9%). BRAMs are implemented as dual-port block SelectRAMs. Therefore, the proposed technique reduces the FPGA resources used by SATD calculation hardware except BRAMs.

As shown in Table 3.2, in order to increase the performance of the original HEVC SATD calculation hardware, the number of parallel datapaths can be increased at the expense of using more FPGA resources. For example, 48 parallel datapaths can be used to process 27 HD frames per second.

The power consumptions of both FPGA implementations on a Xilinx Virtex 6 FPGA are estimated using Xilinx XPower Analyzer tool. Post place&route timing simulations are performed for Vidyo1 (1280x720), Vidyo3 (1280x720), Johnny (1280x720), and KristenAndSara (1280x720) video sequences [30] at 100 MHz and signal activities are stored in VCD files. These VCD files are used for estimating the power consumptions of both FPGA implementations using Xilinx XPower Analyzer tool. The power and energy consumption results for one frame of each video sequence are shown in Table 3.3. The results show that the proposed technique reduced the power and energy consumptions of the original SATD calculation hardware up to 24.2% and 64.6% respectively. Since HEVC SATD calculation hardware is used as part of a HEVC video encoder, only internal power consumption is considered and input and output power consumptions are ignored. Therefore, power consumption of HEVC SATD hardware can be divided into four main categories; clock power, logic power, signal power and BRAM power.

Table 3.2: Performance and Area Results

	LUTs	FlipFlops	Slices	BRAMs	Performance (HD fps)
Original 16 Parallel	20473	3504	6909	8	9
Original 32 Parallel	40946	7008	13818	16	18
Original 48 Parallel	61419	10512	20727	24	27
Proposed Technique	19227	2184	6247	40	21

Table 3.3: Energy Consumption Reductions for 1280x720 Video Frames

Frames	Vidyo1		Vidyo3		Johnny		KristenAndSara	
	Org.	Low Energy	Org.	Low Energy	Org.	Low Energy	Org.	Low Energy
Time (ms)	132	61.6	132	61.6	132	61.6	132	61.6
Clock (mW)	50	45	50	45	50	45	50	45
Logic (mW)	157	43	158	41	157	43	157	38
Signal (mW)	273	154	273	145	273	154	272	131
BRAM (mW)	17	163	17	162	17	163	17	162
Total Power (mW)	497	405	498	393	497	405	496	376
Energy (uJ)	65604	24948	65736	24208	65604	24948	65472	23161
Power Red.	**18.5%**		**21.0%**		**18.5%**		**24.2%**	
Energy Red.	**61.2%**		**63.1%**		**61.2%**		**64.6%**	

Chapter 4

CONCLUSION AND FUTURE WORK

In this thesis, we first designed a high performance hardware architecture for deblocking filter algorithm used in HEVC standard. Two parallel datapaths are used in the hardware to increase its performance. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code is mapped to a Xilinx XC6VLX240T FPGA, and it is verified to work correctly on a Xilinx ML605 FPGA board which includes a Xilinx XC6VLX240T FPGA. The FPGA implementation can work at 108 MHz, and it can code 30 full HD (1920x1080) video frames per second.

We then proposed an energy reduction technique for Sum of Absolute Transformed Difference (SATD) based HEVC intra mode decision algorithm. We designed an efficient hardware architecture for SATD based HEVC intra mode decision algorithm including the proposed technique. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code is mapped to a Xilinx XC6VLX365T FPGA, and it is verified with post place & route simulations. The FPGA implementation can work at 116 MHz, and it can code 21 HD (1280x720) video frames per second. The proposed technique reduced its energy consumption up to 64.6% on this FPGA without any PSNR loss.

As future work, a complete HEVC video encoder hardware can be designed and it can be implemented on the Xilinx ML605 FPGA board.

BIBLIOGRAPHY

[1] G. Sullivan, J. Ohm, Woo-Jin 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.

[2] E.Sahin, "An Efficient H.264 Intra Frame Coder Hardware Design", Master Thesis, Sabancı University, Spring 2006.

[3] M. T. Pourazad, C. Doutre, M. Azimi, P. Nasiopoulos, "HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC?," *IEEE Consumer Electronics Magazine,* vol.1, no.3, pp.36,46, July 2012.

[4] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, Minhua Zhou, G. Van der Auwera, "HEVC Deblocking Filter," *IEEE Trans. on Circuits and Systems for Video Technology,* vol.22, no.12, pp.1746,1754, Dec. 2012.

[5] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard", *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 13, no. 7. pp. 560-576, July 2003.

[6] B. Bross, W.J. Han, J.R. Ohm, G.J. Sullivan, T. Wiegand, "High Efficiency Video Coding (HEVC) Text Specification Draft 6," *JCTVC-H1003,* Nov. 2011.

[7] P. List, A. Joch, J. Lainema, G. Bjøntegaard, M. Karczewicz, "Adaptive Deblocking Filter", *IEEE Trans. on CAS for Video Technology*, vol. 13, pp. 614-619, July 2003.

[8] Y. Adibelli, M. Parlak, I. Hamzaoglu, "Energy Reduction Techniques for H.264 Deblocking Filter Hardware", *IEEE Trans. on Consumer Electronics,* vol. 57, no. 3, Aug. 2011.

[9] A. Otero et al., "Run-Time Scalable Architecture for Deblocking Filtering in H.264/AVC-SVC Video Codecs", *Int. Conf. on Field Programmable Logic and Applications*, Sept. 2011.

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

[11] Y.W. Huang, T.W. Chen, B.Y. Hsieh, T.C. Wang, T.H. Chang, L.G. Chen, "Architecture design for deblocking filter in H.264/JVT/AVC", *IEEE Int. Conf. on Multimedia and Expo.*, pp. 693-696, July 2003.

[12] B. Sheng, W. Gao, D. Wu, "An Implemented Architecture of Deblocking Filter for H.264/AVC", *IEEE Int. Conf. on Image Processing*, pp. 665-668, 2004.

[13] M. Parlak, I. Hamzaoglu, "Low Power H.264 Deblocking Filter Hardware Implementations", *IEEE Trans. on Consumer Electronics*, vol. 54, no. 2, May 2008.

[14] S. Y. Shih, C. R. Chang, Y. L. Lin, “An AMBA-compliant deblocking filter IP for H.264/AVC”, *IEEE Int. Symp. on CAS*, pp. 4529-4532, May 2005.

[15] T. M. Liu, W. P. Lee, T. A. Lin, C. Y. Lee, “A memory-efficient deblocking filter for H.264/AVC video coding”, *IEEE Int. Symp. on CAS*, pp. 2140-2143, May 2005.

[16] Y C. Chao, J. K. Lin, J. F. Yang, B. D. Liu, "A high throughput and data reuse architecture for H.264/AVC deblocking filter", *IEEE Asia South Pacific Conf. on CAS*, pp. 1262-1265, Dec. 2006.

[17] S. Y Shih, C. R. Chang, Y L. Lin, "A near optimal deblocking filter for H.264 advanced video coding", *IEEE Asia South Pacific DAC*, pp.170-175, Jan. 2006.

[18] J. Lainema, F. Bossen, W.J. Han, J. Min and K. Ugur, “Intra Coding of the HEVC Standard”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol.22, no.12, pp.1792-1801, Dec. 2012.

[19] I.K. Kim, K. McCann, K. Sugimoto, B. Bross, and W.J. Han, “High Efficiency Video Coding (HEVC) Test Model 7 (HM 7) Encoder Description”, *JCTVC-I1002,* May 2012.

[20] H. Sun, D. Zhou, S. Goto, “A Low Complexity HEVC Intra Prediction Algorithm Based on Level and Mode Filtering,” *IEEE International Conference on Multimedia and Expo,* pp.1085-1090, July 2012.

[21] F. Li, G. Shi, F. Wu, “An Efficient VLSI Architecture for 4x4 Intra Prediction in High Efficiency Video Coding Standard”, *IEEE Int. Conf. on Image Processing*, Sep. 2011.

[22] Y. Adibelli, M. Parlak, I. Hamzaoglu, “A Novel Energy Reduction Technique for H.264 Intra Mode Decision,” *IEEE Int. Conf. on Image Processing*, pp.385-388, Sept. 2011.

[23] Y.W. Huang, B.Y. Hsieh, T.C. Chen, and L.G. Chen, “Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol.15, no.3, pp.378-401, Mar. 2005.

[24] J.C. Wang, J.F. Wang, J.F. Yang, and J.T.Chen, “A Fast Mode Decision Algorithm and Its VLSI Design for H.264/AVC Intra Prediction”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol.17, no.10, pp.1414-1422, Oct. 2007.

[25] H.Y. Lin, K.H. Wu, B.D. Liu, J.F. Yang, “An Efficient VLSI Architecture for Transform-Based Intra Prediction in H.264/AVC”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol.20, no.6, pp.894-906, Jun. 2010.

[26] C.W. Ku, C.C. Cheng, G.S. Yu, M.C. Tsai, and T.S. Chang, “A High-Definition H.264/AVC Intra-Frame Codec IP for Digital Video and Still Camera Applications”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol.16, no.8, pp.917-928, Aug. 2006.

[27] F. Li, G. Shi, “A Pipelined Architecture for 4x4 Intra Frame Mode Decision in The High Efficiency Video Coding”, *IEEE 13th Int. Workshop on Multimedia Signal Processing*, pp. 1-5, Oct. 2011.

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

[29] Y. Kim, D. Jun, S. Jung, and J. Choi, “A Fast Intra Prediction Method Using Hadamard Transform in High Efficiency Video Coding,” *Proc. SPIE Visual Information Processing and Communication III*, vol. 8305, Feb. 2012.

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

[31] E. Ozcan, Y. Adibelli, I. Hamzaoglu, “A High Performance Deblocking Filter Hardware for High Efficiency Video Coding”, *Int. Conference on Field Programmable Logic and Applications*, Sept. 2013.