POWER CONSUMPTION REDUCTION TECHNIQUES FOR H.264 VIDEO COMPRESSION HARDWARE

by

Yusuf Adıbelli

Submitted to the Graduate School of Engineering and Natural Sciences

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

Sabancı University

August 2012

POWER CONSUMPTION REDUCTION TECHNIQUES FOR H.264 VIDEO COMPRESSION HARDWARE

APPROVED BY:

Assist. Prof. Dr. İlker Hamzaoğlu …….
(Thesis Supervisor)

Prof. Dr. Onur Toker …….

Assist. Prof. Dr. Hakan Erdoğan …….

Assist. Prof. Dr. Müjdat Çetin …….

Assoc. Prof. Dr. Albert Levi …….

DATE OF APPROVAL: ……..

To my Mother, Father and Sisters

To my beloved wife Hümeyra

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 also like to thank my thesis committee members Dr. Onur Toker, Dr. Hakan Erdoğan, Dr. Müjdat Çetin and Dr. Albert Levi for participating in my thesis jury.

I like to convey my heartiest thanks to Mustafa Parlak and his wife Neslihan Parlak for their unlimited support, encouragement. It is very heartwarming to know that one has such friends.

My sincere thanks to System-on-Chip Design & Test group members, Mert Çetin, Merve Peyiç, Çağlar Kalaycıoğlu, Onur Can Ulusel, Aydın Aysu, Abdulkadir Akın, Zafer Tevfik Ozcan, Serkan Yalıman, Yusuf Akşehir, Kamil Erdayandı, Ercan Kalali and Erdem Özcan.

My sincere thanks to all my friends and colleagues in Sabancı University including Mehmet Özdemir, Alisher Kholmatov, Ünal Şen and İbrahim İnanç. I appreciate their friendship and help which made my life easier and more pleasant during my PhD study.

I also would like to express my deepest gratitude to my friends; Malik Sina, Zeynep and Ozgur for their unlimited support, encouragement. It is very heartwarming to know that one has such friends.

I am particularly grateful to my parents and my wife, Hümeyra, for their constant support, encouragement, assistance and patience. Without them, this study would never have been possible.

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

POWER CONSUMPTION REDUCTION TECHNIQUES FOR H.264 VIDEO COMPRESSION HARDWARE

Yusuf Adıbelli
Electronics, Ph.D. Dissertation, 2012

Thesis Supervisor: Asst. Prof. İlker HAMZAOĞLU

Keywords: H.264, Intra Prediction, Deblocking Filter, Mode Decision, Template Matching

ABSTRACT

Video compression systems are used in many commercial products such as digital camcorders, cellular phones and video teleconferencing systems. H.264 / MPEG4 Part 10, the recently developed international standard for video compression, offers significantly better compression efficiency than previous video compression standards. However, this compression efficiency comes with an increase in encoding complexity and therefore in power consumption. Since portable devices operate with battery, it is important to reduce power consumption so that battery life can be increased. In addition, consuming excessive power degrades the performance of integrated circuits, increases packaging and cooling costs, reduces reliability and may cause device failures.

In this thesis, we propose novel computational complexity and power reduction techniques for intra prediction, deblocking filter (DBF), and intra mode decision modules of an H.264 video encoder hardware, and intra prediction with template matching (TM)

hardware. We quantified the computation reductions achieved by these techniques using H.264 Joint Model reference software encoder. We designed efficient hardware architectures for these video compression algorithms and implemented them in Verilog HDL. We mapped these hardware implementations to Xilinx Virtex FPGAs and estimated their power consumptions using Xilinx XPower Analyzer tool. We integrated the proposed techniques to these hardware implementations and quantified their impact on the power consumptions of these hardware implementations on Xilinx Virtex FPGAs. The proposed techniques significantly reduced the power consumptions of these FPGA implementations in some cases with no PSNR loss and in some cases with very small PSNR loss.

H.264 VİDEO SIKIŞTIRMA DONANIMI İÇİN GÜÇ TÜKETİMİ AZALTMA TEKNİKLERİ

Yusuf Adıbelli
Elektronik Müh., Doktora Tezi, 2012

Tez Danışmanı: Yrd. Doç. Dr. İlker HAMZAOĞLU

Anahtar Kelimeler: H.264, Çerçeve İçi Öngörü, Blok Giderici Filtre, Kip Seçimi, Şablon Eşleştirme

ÖZET

Video sıkıştırma sistemleri, dijital kameralar, cep telefonları ve video telekonferans sistemleri gibi bir çok ticari üründe kullanılmaktadır. Yakın tarihte geliştirilmiş uluslararası bir standart olan H.264 / MPEG4 Part 10, kendinden önceki standartlara göre belirgin şekilde daha iyi sıkıştırma verimi sağlamaktadır. Ancak,bu kodlama kazancı hesaplama karmaşıklığı ve güç tüketimi artışını beraberinde getirmektedir. Taşınabilir cihazlar pil ile çalıştığı için, güç tüketimini azaltmak pil ömrünün uzamasını sağlayacaktır. Bunun yanında aşırı güç tüketimi, entegre devrelerin performansını düşürür, paketleme ve soğutma maliyetlerini arttırır, dayanıklılığını azaltır ve bozulmalarına sebep olabilir.

Bu tezde, H.264 video kodlayıcı donanımı modülleri olan çerçeve içi öngörü, blok giderici filtre, çerçeve içi kip seçimi algoritması ve şablon eşleştirmeli çerçeve içi öngörü algoritmaları için yeni hesaplama karmaşıklığı ve güç tüketimi azaltma teknikleri önerildi. Önerilen tekniklerin hesaplama miktarında yaptığı azalma H.264 referans yazılımı (JM) kullanılarak belirlendi. Bu video sıkıştırma algoritmaları için verimli donanım mimarileri tasarlandı ve donanım mimarileri Verilog HDL ile gerçeklendi. Ayrıca bu donanım

uygulamaları Xilinx Virtex FPGA'lerine sentezlendi ve Xilinx XPower Analyzer yazılımı kullanılarak bu donanımların FPGA gerçeklemelerinin detaylı güç tüketim analizleri yapıldı. Daha sonra, önerilen teknikleri bu donanım uygulamalarına entegre edilerek, bu donanımların Xilinx Virtex FPGA'lerindeki güç tüketimine olan etkisi belirlendi. Önerilen teknikler bu FPGA uygulamalarının güç tüketiminde bazen hiçbir PSNR kaybı olmaksızın, bazen de çok küçük PSNR kaybına sebep olarak önemli azalmalara sebep olmuştur.

TABLE OF CONTENTS

ACKNOWLEDGEMENT V

1 ABSTRACT VI

2 ÖZET VIII

3 TABLE OF CONTENTS X

LIST OF FIGURES XIII

LIST OF TABLES XV

1 CHAPTER I INTRODUCTION 1

1.1 H.264 Video Compression Standard 1

1.2 Low Power Hardware Design 4

1.3 Thesis Contributions 6

1.4 Thesis Organization 9

2 CHAPTER II PIXEL EQUALITY AND PIXEL SIMILARITY BASED COMPUTATION AND POWER REDUCTION TECHNIQUES FOR H.264 INTRA PREDICTION 10

2.1 H.264 Intra Prediction Algorithm 12

2.2 Proposed Computational Complexity and Power Reduction Techniques 25

2.3 Proposed Intra Prediction Hardware Architecture 41

2.4 Power Consumption Analysis 42

3 CHAPTER III DATA REUSE, PECR AND PSCR TECHNIQUES FOR COMPUTATION AND POWER REDUCTION IN H.264 INTRA PREDICTION 46

3.1 Proposed Computational Complexity and Power Reduction Techniques 47

3.2 Proposed Intra Prediction Hardware Architecture 56

3.3 Power Consumption Analysis 59

4 CHAPTER IV ENERGY REDUCTION TECHNIQUES FOR H.264 DEBLOCKING FILTER 62

4.1 H.264 Adaptive Deblocking Filter Algorithm 63

4.2 Proposed Energy Reduction Techniques 67

4.3 H.264 DBF Hardware and Its Energy Consumption 81

5 CHAPTER V A NOVEL ENERGY REDUCTION TECHNIQUE FOR H.264 INTRA MODE DECISION 89

5.1 Hadamard Transform 93

5.2 Proposed Computational Complexity Reduction Technique 93

5.2.1 HT of Predicted Blocks by Intra 4x4 Modes 95

5.2.2 HT of Predicted Blocks by Intra 16x16 and 8x8 Horizontal, Vertical and DC Modes 96

5.2.3 HT of Predicted Blocks by Intra 16x16 and 8x8 Plane Mode 108

5.3 Computation Reduction for Residue Calculations 109

5.4 Computation Reduction Results 111

5.5 Proposed 16x16 Intra Mode Decision Hardware Architectures 113

5.6 Energy Consumption Analysis 115

6 CHAPTER VI A NOVEL ENERGY REDUCTION TECHNIQUE FOR INTRA PREDICTION WITH TEMPLATE MATCHING 119

6.1 Proposed Computation and Energy Reduction Technique 122

6.2 Proposed Intra Prediction with Template Matching Hardware 131

6.2.1 PE Array Architectures 131

6.2.2 Memory Organization and Data Alignment 134

6.2.3 Implementation Results 134

6.3 Energy Consumption Analysis.....136
7 CHAPTER VII CONCLUSIONS AND FUTURE WORK 139
8 BIBLIOGRAPHY 142

LIST OF FIGURES

Figure 1.1 H.264 Encoder Block Diagram 2

Figure 1.2 H.264 Decoder Block Diagram 3

Figure 2.1 A 4x4 Luma Block and Neighboring Pixels 13

Figure 2.2 4x4 Luma Prediction Modes 13

Figure 2.3 Examples of Real Images for 4x4 Luma Prediction Modes 14

Figure 2.4 Prediction Equations for 4x4 Luma Prediction Modes 17

Figure 2.5 16x16 Luma Prediction Modes 18

Figure 2.6 Examples of Real Images for 16x16 Luma Prediction Modes 19

Figure 2.7 Prediction Equations for 16x16 Luma Prediction Modes 21

Figure 2.8 Chroma Component of a MB and its Neighboring Pixels 22

Figure 2.9 Prediction Equations for 8x8 Chroma Prediction Modes 25

Figure 2.10 Four Pixel Groups of Neighboring Pixels of a MB 31

Figure 2.11 Rate Distortion Curves of the Original 4x4 Intra Prediction Algorithm and 4x4 Intra Prediction Algorithm with Proposed Technique 40

Figure 2.12 4x4 Intra Prediction Hardware Architecture 42

Figure 3.1 Rate Distortion Curves of the Original 4x4 Intra Prediction Algorithm and a) 4x4 Intra Prediction Algorithm with PSCR Technique proposed in [9] b) 4x4 Intra Prediction Algorithm with Proposed PSCR Technique 55

Figure 3.2 Top-Level Block Diagram of 4x4 Intra Prediction Hardware Architecture 57

Figure 3.3 Datapath for The Prediction Equations Used in DDL, DDR, VR, VL, HD, HUP and DC Modes 58

Figure 4.1 Illustration of H.264 DBF Algorithm 65

Figure 4.2 Edge Filtering Order Specified in H.264 Standard 64

Figure 4.3 H.264 Deblocking Filter Algorithm 66

Figure 4.4 Rate Distortion Curves of the Original H.264 DBF Algorithm and H.264 DBF Algorithm with Proposed PSCR Technique 80
Figure 4.5 H.264 DBF Hardware Architecture 81
Figure 4.6 Processing Order of 4×4 Blocks 82
Figure 4.7 4x4 Blocks Stored in LUMA and CHRM SRAMs 83
Figure 4.8 H.264 DBF Datapath 85
Figure 4.9 Unfiltered video frame shown on the above and the same frame filtered by H.264 Deblocking Filter algorithm shown on the below 86
Figure 5.1 Formation of DC Block for Intra 16x16 Prediction Modes 91
Figure 5.2 SATD Calculation for Each 4x4 Block 91
Figure 5.3 Addition Operations Performed by Intra Prediction and Mode Decision 92
Figure 5.4 Fast HT Algorithm for a 4x4 Block 94
Figure 5.5 Hadamard Transform of Vertical, Horizontal and DC Modes 95
Figure 5.6 16x16 MB and its Neighboring Pixels 105
Figure 5.7 Rate Distortion Curves of Original SATD Mode Decision and SATD Mode Decision with Proposed Technique 113
Figure 5.8 Proposed Hardware for Original Intra 16x16 Mode Decision 116
Figure 5.9 Proposed Hardware for Intra 16x16 Mode Decision with Proposed Technique 117
Figure 6.1 Intra Prediction with Template Matching 120
Figure 6.2 Different Size Templates and Search Windows 123
Figure 6.3 Top Level Block Diagram of Proposed 4x4 Intra Prediction with Template Matching Hardware 129
Figure 6.4 Template Search PE Array and 16 Adder Tree 130
Figure 6.5 PE Architecture 132
Figure 6.6 SAD Calculation PE Array and Adder Tree 133
Figure 6.7 Memory Organization of 32x32 SW 135
Figure 6.8 Predicted by H.264 9 intra 4x4 modes video frame shown on the above and the same frame predicted by H.264 9 intra 4x4 modes with TM including proposed technique shown on the below 137

LIST OF TABLES

Table 2.1 Availability of 4x4 Luma Prediction Modes 17

Table 2.2 Availability of 16x16 Luma Prediction Modes 19

Table 2.3 Availability of 8x8 Luma Prediction Modes 22

Table 2.4 4x4 Intra Modes and Corresponding Neighboring Pixels 26

Table 2.5 Percentage of 4x4 Intra Prediction Modes with Equal Neighboring Pixels 28

Table 2.6 Percentage of 4x4 Intra Prediction Modes with Similar Neighboring Pixels..... 29

Table 2.7 Computation Amount of 4x4 Intra Modes 29

Table 2.8 Intra 4x4 Modes Computation Reduction Results by PECR Technique..... 30

Table 2.9 Intra 4x4 Modes Computation Reduction Results by PSCR Technique..... 30

Table 2.10 Percentage of 16x16 Intra Prediction Modes with Equal Neighboring Pixels 32

Table 2.11 Percentage of 8x8 Intra Prediction Modes (Chroma CB, CR) with Equal Neighboring Pixels..... 33

Table 2.12 Percentage of 16x16 Intra Prediction Modes with Similar Neighboring Pixels..... 34

Table 2.13 Percentage of 8x8 Intra Prediction Modes (Chroma CB, CR) with Similar Neighboring Pixels 35

Table 2.14 Computation Amount of Intra 16x16 and Intra 8x8 Modes 36

Table 2.15 Intra 16x16 Computation Reduction Results by PECR 37

Table 2.16 Intra 8x8 (Chroma CB, CR) Computation Reduction Results by PECR 37

Table 2.17 Intra 16x16 Computation Reduction Results by PSCR..... 38

Table 2.18 Intra 8x8 (Chroma CB, CR) Computation Reduction Results by PECR 39

Table 2.19 Average Psnr Comparison of the Proposed PSCR Technique 41

Table 2.20 Power Consumption Reduction (Q=28) by PSCR Technique..... 44

Table 2.21 Power Consumption Reduction (Q=35) by PSCR Technique..... 44

Table 2.22 Power Consumption Reduction (Q=42) by PSCR Technique..... 45

Table 3.1 Prediction Equations of 4x4 Intra Prediction Modes 49

Table 3.2 4x4 Intra Modes and Corresponding Neighboring Pixels 50
Table 3.3 Percentage of 4x4 Intra Prediction Blocks with Equal and Similar Prediction Equation Pixels 51
Table 3.4 Addition and Shift Operations Performed by 4x4 Intra Prediction for a CIF Frame with PECR Technique 52
Table 3.5 Addition and Shift Operations Performed by 4x4 Intra Prediction for a CIF Frame with PSCR Technique 52
Table 3.6 Computation Reduction by PECR and PSCR (4bT) Techniques for 4x4 Intra Prediction with Data Reuse 53
Table 3.7 Computation Reduction for 4x4 Intra Prediction by PECR Technique 53
Table 3.8 Computation Reduction for 4x4 Intra Prediction by PSCR Technique with 4bT 54
Table 3.9 Average PSNR Comparison of the PSCR Techniques 56
Table 3.10 Comparison of 4x4 Intra Prediction Hardware 58
Table 3.11 Power Consumption Reduction (QP = 28) 60
Table 3.12 Power Consumption Reduction (QP = 35) 61
Table 3.13 Power Consumption Reduction (QP = 42) 61
Table 4.1 Conditions that Determine BS 67
Table 4.2 DBF Modes 68
Table 4.3 Equations for Mode 6 and their Simplified Versions when p2=p1=p0=q0=q1=q2 69
Table 4.4 The Amount of Computation Required by DBF Mode 0 For Different Equal Pixel Combinations 70
Table 4.5 The Amount of Computation Required by DBF Mode 1 For Different Equal Pixel Combinations 70
Table 4.6 The Amount of Computation Required by DBF Mode 2 For Different Equal Pixel Combinations 70
Table 4.7 The Amount of Computation Required by DBF Mode 3 For Different Equal Pixel Combinations 71
Table 4.8 The Amount of Computation Required by DBF Mode 4 For Different Equal Pixel Combinations 71

Table 4.9 The Amount of Computation Required by DBF Mode 5 For Different Equal Pixel Combinations..... 72

Table 4.10 The Amount of Computation Required by DBF Mode 6 For Different Equal Pixel Combinations..... 72

Table 4.11 The Amount of Computation Required by DBF Mode 7 For Different Equal Pixel Combinations..... 73

Table 4.12 Amount of Operations Performed by All DBF Modes 73

Table 4.13 Filtering Units with All Equal or Similar Pixels for Luma Components 74

Table 4.14 Filtering Units with All Equal or Similar Pixels for Chroma (CbCr) Components ... 74

Table 4.15 Computation Reductions for Luma Components 76

Table 4.16 Computation Reductions for Chroma (CbCr) Components 77

Table 4.17 Comparison Overhead..... 79

Table 4.18 Average PSNR Comparison of PSCR Technique 79

Table 4.19 FPGA Resource Usage and Clock Frequency After P&R 84

Table 4.20 Energy Consumption Reduction By PECR Technique..... 87

Table 4.21 Energy Consumption Reduction By PSCR (1bT) Technique 87

Table 4.22 Energy Consumption Reduction By PSCR (2bT) Technique 88

Table 5.1 Pre-calculated Values for DDL Prediction Mode..... 100

Table 5.2 DDL Mode Prediction Calculations Using Pre-calculated Values 100

Table 5.3 Pre-calculated Values for DDR Prediction Mode..... 101

Table 5.4 DDR Mode Prediction Calculations Using Pre-calculated Values..... 101

Table 5.5 Pre-calculated Values for VR Prediction Mode 102

Table 5.6 VR Mode Prediction Calculations Using Pre-calculated Values..... 103

Table 5.7 Pre-calculated Values for HUP Prediction Mode 103

Table 5.8 HUP Mode Prediction Calculations Using Pre-calculated Values 104

Table 5.9 Computation Reductions for Intra Prediction Modes 112

Table 5.10 Average PSNR (dB) Comparison of Original SATD Mode Decision and SATD Mode Decision with Proposed Technique..... 112

Table 5.11 Energy Consumption Reduction..... 118

Table 6.1 PSNR Results (dB) of Different Size SWs and Templates 123

Table 6.2 PSNR Results (dB) of Intra Prediction with TM 125

Table 6.3 Number of TM Predictions Selected when Th_{SAD} Used 126

Table 6.4 Average PSNR (dB) Comparison of the Proposed Technique 126

Table 6.5 Average PSNR (dB) Comparison of the Proposed Technique for Higher Th_{SAD} 127

Table 6.6 Computation Reduction in Intra Prediction with TM Algorithm for Different Th_{SAD} values 128

Table 6.7 Energy Consumption Reduction when Th_{SAD} = 40 138

Table 6.8 Energy Consumption Reduction when Th_{SAD} = 50 138

Table 6.9 Energy Consumption Reduction when Th_{SAD} = 60 138

CHAPTER I

INTRODUCTION

1.1 H.264 Video Compression Standard

Video compression systems are used in many commercial products, from consumer electronic devices such as digital camcorders, cellular phones to video teleconferencing systems. H.264 / MPEG4 Part 10, the recently developed international standard for video compression, offers significantly better compression efficiency (capable of saving up to 50% bit rate at the same level of video quality) than previous video compression standards [1, 2, 3]. Because of its high coding efficiency and flexibility and robustness to different communication environments, H.264 is expected to be widely used in many applications such as digital TV, DVD, video transmission in wireless networks, and video conferencing over the internet.

The human visual system appears to distinguish scene content in terms of brightness and color information individually, and with greater sensitivity to the details of brightness

than color [3]. Same as the previous video compression standards, H.264 is designed to take advantage of this by using YCbCr color space. In YCbCr color space, each pixel is represented with three 8-bit components called Y, Cb, and Cr. Y, the luminance (luma) component, represents brightness. Cb and Cr, chrominance (chroma) components, represent the extent to which the color differs from gray toward blue and red, respectively. Since the human visual system is more sensitive to luma component than chroma components, H.264 standard uses 4:2:0 sampling. In 4:2:0 sampling, for every four luma samples, there are two chroma samples, one Cb and one Cr.

The top-level block diagram of an H.264 video encoder is shown in Figure 1.1. As shown in the figure, the video compression efficiency achieved in H.264 standard is not a result of any single feature but rather a combination of a number of encoding tools such as motion estimation, intra prediction and deblocking filter (DBF). Same as the previous video compression standards, H.264 standard does not specify all the algorithms that will be used in an encoder such as mode decision. Instead, it defines the syntax of the encoded bit stream and functionality of the decoder that can decode this bit stream.

As shown in Figure 1.1, an H.264 encoder has a forward path and a reconstruction path. The forward path is used to encode a video frame and create the bit stream by using intra and inter predictions. The reconstruction path is used to decode the encoded frame and 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 [1,3,4].

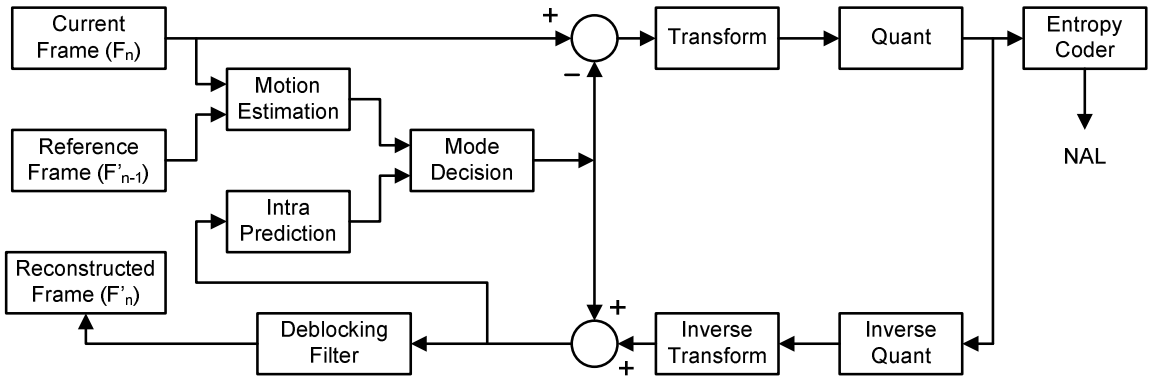


Figure 1.1 H.264 Encoder Block Diagram

Forward path starts with partitioning the input frame into macroblocks (MB). Each MB is encoded in intra or inter mode depending on the mode decision. In both intra and inter modes, the current MB is predicted from the reconstructed frame. Intra mode generates the predicted MB based on spatial redundancy, whereas inter mode, generates the predicted MB based on temporal redundancy. Mode decision compares the required amount of bits to encode a MB and the quality of the decoded MB 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 MB is subtracted from the current MB to generate the residual MB. Residual MB is transformed using 4x4 and 2x2 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 coded. The entropy-coded coefficients together with header information, such as MB prediction mode and quantization step size, form the compressed bit stream. The compressed bit stream is passed to network abstraction layer (NAL) for storage or transmission [1,3,4].

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. DBF is, then, applied to reduce the effects of blocking artifacts in the reconstructed frame [1,3,4].

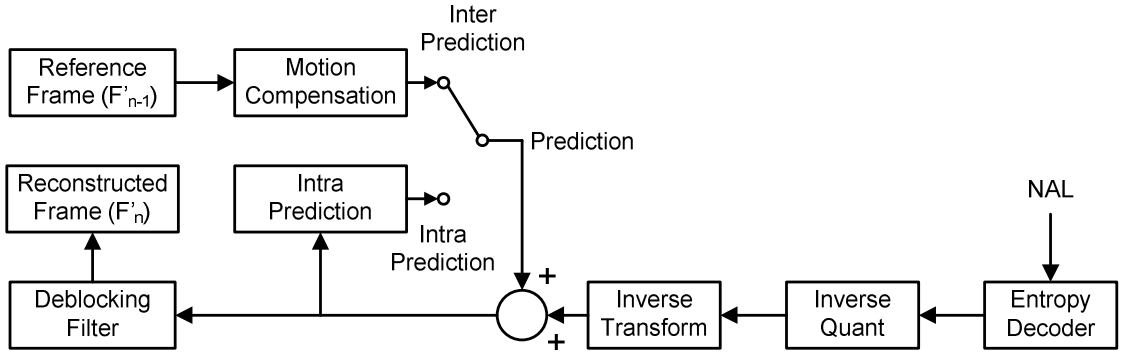


Figure 1.2 H.264 Decoder Block Diagram

The compression efficiency achieved by H.264 standard comes with an increase in encoding complexity and therefore in power consumption. H.264 intra prediction and mode decision algorithms have very high computational complexity. Because, in order to improve the compression efficiency, H.264 standard uses many intra prediction modes for a MB and selects the best mode for that MB using a mode decision algorithm. The DBF algorithm used in H.264 standard is more complex than the DBF algorithms used in previous video compression standards. First of all, H.264 DBF algorithm is highly adaptive and applied to each edge of all the 4×4 luma and chroma blocks in a MB. Second, it can update 3 pixels in each direction that the filtering takes place. Third, in order to decide whether the DBF will be applied to an edge, the related pixels in the current and neighboring 4×4 blocks must be read from memory and processed. Because of these complexities, the DBF algorithm can easily account for one-third of the computational complexity of an H.264 video decoder [4,5].

H.264 decoder is similar to the reconstruction path of H.264 encoder. It receives a compressed bit stream from the NAL as shown in Figure 1.2. The bit stream is decoded, inverse quantized and inverse transformed to get residual data. Using the header information decoded from the bit stream, the decoder creates a prediction block, identical to the prediction block generated in reconstruction path of H.264 encoder. The prediction block is added to the residual block to create the reconstructed block. Blocking artifacts are, then, removed from reconstructed block by applying DBF.

H.264 has three profiles; Baseline, Main, and Extended. A profile is a set of algorithmic features and a level shows encoding capability such as picture size and frame rate. In this thesis, we use Baseline profile. Baseline profile has lower latency than main and extended profiles, and it is used for wireless video applications and video conferencing. In Baseline profile, YCbCr color space with 4:2:0 sampling, I and P slices, and context-adaptive variable length entropy coding are supported [1,3].

1.2 Low Power Hardware Design

Multimedia applications running on portable devices have increased recently and this

trend is expected to continue in the future. Since portable devices operate with battery, it is important to reduce power consumption so that battery life can be increased. In addition, consuming excessive power for a long time causes chips to heat up and degrades performance, because transistors run faster when they are cool rather than hot. Excessive power consumption also increases packaging and cooling costs. Excessive power consumption also reduces reliability and may cause device failures [6, 7].

Field Programmable Gate Arrays (FPGA) consume more power than standard cell-based Application Specific Integrated Circuits (ASIC). FPGAs have look-up tables and programmable switches. Look-up table based logic implementation is inefficient in terms of power consumption and programmable switches have high power consumption because of large output capacitances. Therefore, reducing power consumption is even more important for FPGA implementations.

ICs have static and dynamic power consumption. Static power consumption is a result of leakage currents in an IC. Dynamic power consumption is a result of short circuit currents and charging and discharging of capacitances in an IC. Dynamic power consumption is proportional to the switching activity (α), total capacitance (C_L), supply voltage (V_{DD}), operating frequency (*f*) and short circuit current (I_{SC}) as shown in the following equation. The power consumption due to charging and discharging of capacitances is the dominant component of dynamic power consumption and it can be reduced either by decreasing switching activity, capacitance, supply voltage or frequency.

$$P_{dyn} \approx \alpha_{0\to 1} C_L V_{DD}^2 f + I_{SC} V_{DD} f \tag{1.1}$$

In this thesis, we focused on reducing the dynamic power consumptions of FPGA implementations of H.264 video compression hardware. The dynamic power consumption of a digital hardware implementation on a Xilinx FPGA is estimated using Xilinx XPower tool. Since the switching activity is input pattern dependent, in order to estimate the dynamic power consumption, timing simulation of the placed and routed netlist of that hardware implementation is done for several input patterns using Mentor Graphics ModelSim and the signal activities are stored in a Value Change Dump (VCD) file. This VCD file is used for estimating the dynamic power consumption of that hardware using Xilinx XPower tool.

1.3 Thesis Contributions

We propose pixel equality based computation reduction (PECR) technique for reducing the amount of computations performed by H.264 intra prediction algorithm and therefore reducing the power consumption of H.264 intra prediction hardware significantly without any PSNR and bit rate loss. The proposed technique performs a small number of comparisons among neighboring pixels of the current block before the intra prediction process. If the neighboring pixels of the current block are equal, the prediction equations of H.264 intra prediction modes simplify significantly for this block. By exploiting the equality of the neighboring pixels, the proposed technique reduces the amount of computations performed by 4x4 luminance, 16x16 luminance, and 8x8 chrominance prediction modes up to 60%, 28%, and 68% respectively with a small comparison overhead. We also implemented an efficient 4x4 intra prediction hardware including the proposed technique using Verilog HDL. We quantified the impact of the proposed technique on the power consumption of this hardware on a Xilinx Virtex II FPGA using Xilinx XPower, and it reduced the power consumption of this hardware up to 46% [8].

We also propose pixel similarity based computation reduction (PSCR) technique for reducing the amount of computations performed by H.264 intra prediction algorithm and therefore reducing the power consumption of H.264 intra prediction hardware significantly. The proposed technique performs a small number of comparisons among neighboring pixels of the current block before the intra prediction process. If the neighboring pixels of the current block are similar, the prediction equations of H.264 intra prediction modes are simplified for this block. The proposed technique reduces the amount of computations performed by 4x4 luminance, 16x16 luminance, and 8x8 chrominance prediction modes up to 68%, 39%, and 65% respectively with a small comparison overhead. The proposed technique does not change the PSNR for some video frames, it increases the PSNR slightly for some video frames and it decreases the PSNR slightly for some video frames. We also implemented an efficient 4x4 intra prediction hardware including the proposed technique using Verilog HDL. We quantified the impact of the proposed technique on the power

consumption of this hardware on a Xilinx Virtex II FPGA using Xilinx XPower. The proposed technique reduced the power consumption of this hardware up to 57% [9, 10].

We, then, propose to calculate the common prediction equations only once and to use the results for the corresponding 4x4 intra modes, and to apply the PECR and PSCR techniques for each intra prediction equation separately. These techniques exploit pixel equality and similarity in a video frame by performing a small number of comparisons among pixels used in prediction equations before the intra prediction process. If the pixels used in prediction equations are equal or similar, prediction equations simplify significantly. By exploiting the equality and similarity of the pixels used in prediction equations, the proposed PECR and PSCR techniques reduce the amount of computations performed by 4x4 intra prediction modes up to 78% and 89%, respectively, with a small comparison overhead. We also implemented an efficient 4x4 intra prediction hardware including the proposed techniques using Verilog HDL. We quantified the impact of the proposed techniques on the power consumption of this hardware on a Xilinx Virtex II FPGA using Xilinx XPower. The proposed PECR and PSCR techniques reduced the power consumption of this hardware up to 13.7% and 17.2%, respectively. The proposed PECR technique does not affect the PSNR and bitrate. The proposed PSCR technique increases the PSNR slightly for some videos frames and it decreases the PSNR slightly for some videos frames [11, 12].

We also propose pixel equality and pixel similarity based techniques for reducing the amount of computations performed by H.264 DBF algorithm, and therefore reducing the energy consumption of H.264 DBF hardware. These techniques avoid unnecessary calculations in H.264 DBF algorithm by exploiting the equality and similarity of the pixels used in DBF equations. The proposed techniques reduce the amount of addition and shift operations performed by H.264 DBF algorithm up to 52% and 67% respectively with a small comparison overhead. The pixel equality based technique does not affect PSNR. The pixel similarity based technique does not affect the PSNR for some video frames, but it decreases the PSNR slightly for some video frames. We also implemented an efficient H.264 DBF hardware including the proposed techniques using Verilog HDL. We quantified the impact of the proposed techniques on the energy consumption of this hardware on a Xilinx Virtex4 FPGA using Xilinx XPower. The proposed pixel equality and

pixel similarity based techniques reduced the energy consumption of this H.264 DBF hardware up to 35% and 39%, respectively [14, 15].

We propose a novel energy reduction technique for H.264 intra mode decision. The proposed technique reduces the number of additions performed by Sum of Absolute Transformed Difference based 4x4, 16x16 and 8x8 intra mode decision algorithms used in H.264 joint model reference software encoder by 46%, 43% and 42% respectively for a CIF size frame without any PSNR loss. In addition, it avoids the calculation of intra 16x16 and intra 8x8 plane prediction modes by slightly modifying SATD criterion used in H.264 Joint Model (JM) reference software encoder which slightly impacts the coding efficiency. It doesn't affect the PSNR for some videos, it increases the PSNR slightly for some videos and it decreases the PSNR slightly for some videos. Since plane mode is the most computationally intensive 16x16 and 8x8 prediction mode, avoiding plane mode calculations reduces the computational complexity of 16x16 and 8x8 intra prediction algorithm by 80%. We also implemented an efficient H.264 16x16 intra mode decision hardware including the proposed technique using Verilog HDL. We quantified the impact of the proposed technique on the energy consumption of this hardware on a Xilinx Virtex II FPGA using Xilinx XPower. The proposed technique reduced the energy consumption of this H.264 16x16 intra mode decision hardware up to 59.6% [16].

H.264 intra prediction algorithm is not well suited for processing complex textures at low bit rates. Therefore, intra prediction with Template Matching (TM) is proposed for improving H.264 intra prediction. However, intra prediction with TM has high computational complexity. Therefore, in this thesis, we propose a novel technique for reducing the amount of computations performed by intra prediction with TM, and therefore reducing the energy consumption of intra prediction with TM hardware. The proposed technique does not change the PSNR for some video frames, but it decreases the PSNR slightly for some video frames. We also designed and implemented a high performance 4x4 intra prediction with TM hardware including the proposed technique using Verilog HDL, and mapped it to a Xilinx Virtex 6 FPGA. The FPGA implementation is capable of processing 53 HD (1280x720) frames per second, and the proposed technique reduced its energy consumption up to 50% [13].

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter II, first, explains H.264 intra prediction algorithm. It, then, presents the proposed PECR and PSCR techniques for H.264 intra prediction. An efficient H.264 intra prediction hardware including these techniques and its power consumption analysis are also presented in this chapter.

Chapter III, presents the data reuse and the application of PECR and PSCR techniques for each intra prediction equation separately. An efficient H.264 intra prediction hardware including these techniques and its power consumption analysis are also presented in this chapter.

Chapter IV, first, explains H.264 DBF algorithm. It, then, presents pixel equality and pixel similarity based techniques for reducing the amount of computations performed by H.264 DBF algorithm. An efficient H.264 DBF hardware including the proposed technique and its energy consumption analysis are also presented in this chapter.

Chapter V, first, explains H.264 intra mode decision algorithm. It, then, presents a novel computational complexity and power reduction technique for H.264 intra mode decision. An efficient H.264 16x16 intra mode decision hardware including the proposed technique and its energy consumption analysis are also presented in this chapter.

Chapter VI, first, explains intra prediction with Template Matching (TM) algorithm. It, then, presents a novel technique for reducing the amount of computations performed by intra prediction with TM. A high performance 4x4 intra prediction with TM hardware including the proposed technique and its energy consumption analysis are also presented in this chapter.

Chapter VII presents conclusions and future work.

CHAPTER II

PIXEL EQUALITY AND PIXEL SIMILARITY BASED COMPUTATION AND POWER REDUCTION TECHNIQUES FOR H.264 INTRA PREDICTION

H.264 intra prediction algorithm achieves better coding results than the intra prediction algorithms used in previous video compression standards. However, this coding gain comes with a significant increase in computational complexity. Therefore, in this thesis, we propose pixel equality and pixel similarity based techniques for reducing the amount of computations performed by H.264 intra prediction algorithm and therefore reducing the power consumption of H.264 intra prediction hardware. Both techniques are applicable to 4x4 luminance, 16x16 luminance and 8x8 chrominance prediction modes. Both techniques perform a small number of comparisons among neighboring pixels of the current block before the intra prediction process.

Pixel equality based computation reduction (PECR) technique checks the equality of the neighboring pixels. If the neighboring pixels used for calculating the predicted pixels by an intra 4x4 prediction mode are equal, the predicted pixels by this mode are equal to one of these neighboring pixels. Therefore, the prediction equations simplify to a constant value and prediction calculations for this mode become unnecessary. Furthermore, if the neighboring pixels used for calculating the predicted pixels by an intra 16x16 or an intra

8x8 prediction mode are equal, the prediction equations used by this mode simplify significantly. In this way, the amount of computations performed by H.264 intra prediction algorithm is reduced significantly without any PSNR loss [8].

Pixel similarity based computation reduction (PSCR) technique checks the similarity of the neighboring pixels, and if the neighboring pixels used for calculating the predicted pixels by an intra 4x4 prediction mode are similar, the predicted pixels by this mode are assumed to be equal to one of these neighboring pixels. Therefore, the prediction equations are simplified to a constant value and prediction calculations for this mode become unnecessary. Furthermore, if the neighboring pixels used for calculating the predicted pixels by an intra 16x16 or an intra 8x8 prediction mode are similar, the prediction equations used by this mode are simplified significantly. In this way, the proposed technique reduces the amount of computations performed by H.264 intra prediction algorithm even further with a small PSNR loss [9, 10].

The simulation results obtained by H.264 reference software, JM 14.0 [17], for several video sequences showed that PECR technique reduces the amount of computations performed by H.264 intra 4x4, 16x16 and 8x8 prediction modes up to 60%, 28%, and 68% respectively and PSCR technique reduces the amount of computations performed by H.264 intra 4x4, 16x16 and 8x8 prediction modes up to 68%, 39%, and 65% respectively with a small comparison overhead. The proposed techniques, for each MB, requires 12 and 24 comparisons for intra 4x4 and intra 8x8 prediction modes respectively. Since intra 4x4 and intra 16x16 prediction modes operate on the same MB, the comparison results for intra 4x4 prediction modes are also used for intra 16x16 prediction modes. The simulation results also showed that the proposed PSCR technique does not change the PSNR for some video frames, it increases the PSNR slightly for some video frames and it decreases the PSNR slightly for some video frames.

Several techniques are reported in the literature for reducing the computational complexity of H.264 intra prediction algorithm [18, 19, 20, 21]. These techniques reduce the amount of computation for H.264 intra prediction algorithm by trying selected intra prediction modes rather than trying all intra prediction modes. However, the techniques proposed in this thesis try all intra prediction modes, and it can also be used together with the techniques proposed in [18, 19, 20, 21]. Several hardware architectures for H.264 4x4

intra prediction algorithm are reported in the literature [22, 23, 24, 25]. However, they do not report their power consumption and they do not implement the technique proposed in this thesis.

We also designed an efficient H.264 4x4 intra prediction hardware architecture including the proposed PECR and PSCR techniques. The hardware architecture is implemented in Verilog HDL. The Verilog RTL codes are verified to work at 50 MHz in a Xilinx Virtex II FPGA. The impacts of the proposed techniques on the power consumption of this hardware implementation on a Xilinx Virtex II FPGA are quantified using Xilinx XPower tool. The proposed PECR and PSCR techniques reduced the power consumption of this hardware on this FPGA up to 46% and 57%, respectively.

2.1 H.264 Intra Prediction Algorithm

Intra prediction algorithm predicts the pixels in a MB using the pixels in the available neighboring blocks. For the luma component of a MB, a 16x16 predicted luma block is formed by performing intra predictions for each 4x4 luma block in the MB and by performing intra prediction for the 16x16 MB. There are nine prediction modes for each 4x4 luma block and four prediction modes for a 16x16 luma block. A mode decision algorithm is then used to compare the 4x4 and 16x16 predictions and select the best luma prediction mode for the MB. 4x4 prediction modes are generally selected for highly textured regions while 16x16 prediction modes are selected for flat regions.

There are nine 4x4 luma prediction modes designed in a directional manner. A 4x4 luma block consisting of the pixels a to p is shown in Figure 2.1. The pixels A to M belong to the neighboring blocks and are assumed to be already encoded and reconstructed and are therefore available in the encoder and decoder to generate a prediction for the current MB. Each 4x4 luma prediction mode generates 16 predicted pixel values using some or all of the neighboring pixels A to M as shown in Figure 2.2. The examples of each 4x4 luma prediction mode for real images are given in Figure 2.3. The arrows indicate the direction of prediction in each mode. The predicted pixels are calculated by a weighted average of the neighboring pixels A-M for each mode except Vertical, Horizontal and DC modes.

The prediction equations used in each 4x4 luma prediction mode are shown in Figure 2.4 where [x, y] denotes the position of the pixel in a 4x4 block (the top left, top right, bottom left, and bottom right positions of a 4x4 block are denoted as [0, 0], [0, 3], [3, 0], and [3, 3], respectively) and pred[x, y] is the prediction for the pixel in the position [x, y].

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Figure 2.1 A 4x4 Luma Block and Neighboring Pixels

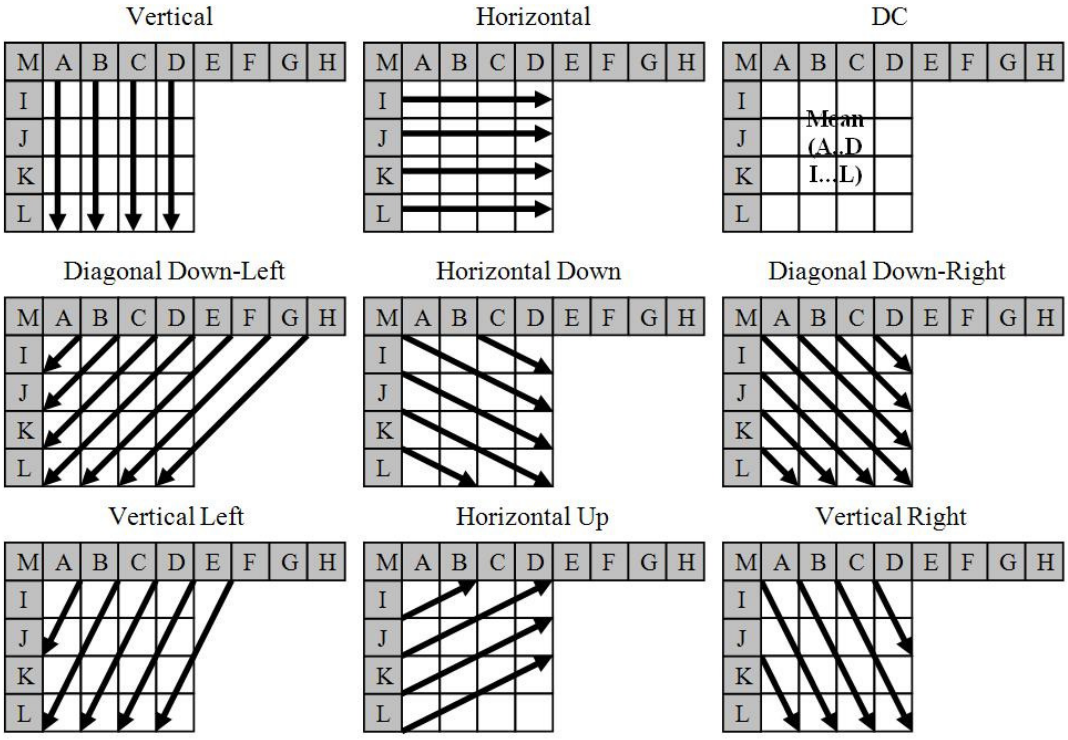


Figure 2.2 4x4 Luma Prediction Modes

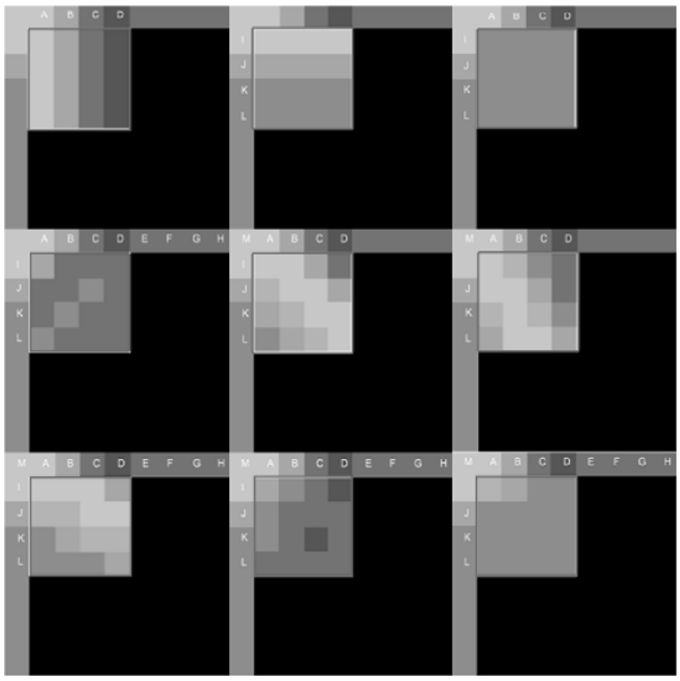


Figure 2.3 Examples of Real Images for 4x4 Luma Prediction Modes

DC mode is always used regardless of the availability of the neighboring pixels. However, it is adopted based on which neighboring pixels A-M are available. If pixels E, F, G and H have not yet been encoded and reconstructed, the value of pixel D is copied to these positions and they are marked as available for DC mode. The other prediction modes can only be used if all of the required neighboring pixels are available [1, 3]. Available 4x4 luma prediction modes for a 4x4 luma block depending on the availability of the neighboring 4x4 luma blocks are given in Table 2.1.

pred[0, 0] = A	pred[0, 0] = I
pred[0, 1] = B	pred[0, 1] = I
pred[0, 2] = C	pred[0, 2] = I
pred[0, 3] = D	pred[0, 3] = I
pred[1, 0] = A	pred[1, 0] = J
pred[1, 1] = B	pred[1, 1] = J
pred[1, 2] = C	pred[1, 2] = J

pred[1, 3] = D	pred[1, 3] = J
pred[2, 0] = A	pred[2, 0] = K
pred[2, 1] = B	pred[2, 1] = K
pred[2, 2] = C	pred[2, 2] = K
pred[2, 3] = D	pred[2, 3] = K
pred[3, 0] = A	pred[3, 0] = L
pred[3, 1] = B	pred[3, 1] = L
pred[3, 2] = C	pred[3, 2] = L
pred[3, 3] = D	pred[3, 3] = L

(a) 4x4 Vertical Mode *(b) 4x4 Horizontal Mode*

pred[x, y] = (A + B + C + D + I + J + K + L + 4) >> 3	If the left and the top neighboring pixels are available
pred[x, y] = (I + J + K + L + 2) >> 2	Else If only the left neighboring pixels are available
pred[x, y] = (A + B + C + D + 2) >> 2	Else If only the top neighboring pixels are available
pred[x, y] = 128	Else

(c) 4x4 DC Mode

pred[0, 0] = A + 2B + C + 2 >> 2	pred[0, 0] = A + 2M + I + 2 >> 2
pred[0, 1] = B + 2C + D + 2 >> 2	pred[0, 1] = M + 2A + B + 2 >> 2
pred[0, 2] = C + 2D + E + 2 >> 2	pred[0, 2] = A + 2B + C + 2 >> 2
pred[0, 3] = D + 2E + F + 2 >> 2	pred[0, 3] = B + 2C + D + 2 >> 2
pred[1, 0] = B + 2C + D + 2 >> 2	pred[1, 0] = M + 2I + J + 2 >> 2
pred[1, 1] = C + 2D + E + 2 >> 2	pred[1, 1] = A + 2M + I + 2 >> 2
pred[1, 2] = D + 2E + F + 2 >> 2	pred[1, 2] = M + 2A + B + 2 >> 2
pred[1, 3] = E + 2F + G + 2 >> 2	pred[1, 3] = A + 2B + C + 2 >> 2
pred[2, 0] = C + 2D + E + 2 >> 2	pred[2, 0] = I + 2J + K + 2 >> 2

pred[2, 1] = D + 2E + F + 2 >> 2
pred[2, 2] = E + 2F + G + 2 >> 2
pred[2, 3] = F + 2G + H + 2 >> 2
pred[3, 0] = D + 2E + F + 2 >> 2
pred[3, 1] = E + 2F + G + 2 >> 2
pred[3, 2] = F + 2G + H + 2 >> 2
pred[3, 3] = G + 3H + 2 >> 2

(d) 4x4 Diagonal Down Left Mode

pred[2, 1] = M + 2I + J + 2 >> 2
pred[2, 2] = A + 2M + I + 2 >> 2
pred[2, 3] = M + 2A + B + 2 >> 2
pred[3, 0] = J + 2K + L + 2 >> 2
pred[3, 1] = I + 2J + K + 2 >> 2
pred[3, 2] = M + 2I + J + 2 >> 2
pred[3, 3] = A + 2M + I + 2 >> 2

(e) 4x4 Diagonal Down Right Mode

pred[0, 0] = M + A + 1 >> 1
pred[0, 1] = A + B + 1 >> 1
pred[0, 2] = B + C + 1 >> 1
pred[0, 3] = C + D + 1 >> 1
pred[1, 0] = I + 2M + A + 2 >> 2
pred[1, 1] = M + 2A + B + 2 >> 2
pred[1, 2] = A + 2B + C + 2 >> 2
pred[1, 3] = B + 2C + D + 2 >> 2
pred[2, 0] = M + 2I + J + 2 >> 2
pred[2, 1] = M + A + 1 >> 1
pred[2, 2] = A + B + 1 >> 1
pred[2, 3] = B + C + 1 >> 1
pred[3, 0] = I + 2J + K + 2 >> 2
pred[3, 1] = I + 2M + A + 2 >> 2
pred[3, 2] = M + 2A + B + 2 >> 2
pred[3, 3] = A + 2B + C + 2 >> 2

(f) 4x4 Vertical Right Mode

pred[0, 0] = M + I + 1 >> 1
pred[0, 1] = I + 2M + A + 2 >> 2
pred[0, 2] = B + 2A + M + 2 >> 2
pred[0, 3] = C + 2B + A + 2 >> 2
pred[1, 0] = I + J + 1 >> 1
pred[1, 1] = M + 2I + J + 2 >> 2
pred[1, 2] = M + I + 1 >> 1
pred[1, 3] = I + 2M + A + 2 >> 2
pred[2, 0] = J + K + 1 >> 1
pred[2, 1] = I + 2J + K + 2 >> 2
pred[2, 2] = I + J + 1 >> 1
pred[2, 3] = M + 2I + J + 2 >> 2
pred[3, 0] = K + L + 1 >> 1
pred[3, 1] = J + 2K + L + 2 >> 2
pred[3, 2] = J + K + 1 >> 1
pred[3, 3] = I + 2J + K + 2 >> 2

(g) 4x4 Horizontal Down Mode

pred[0, 0] = A + B + 1 >> 1	pred[0, 0] = I + J + 1 >> 1
pred[0, 1] = B + C + 1 >> 1	pred[0, 1] = I + 2J + K + 2 >> 2
pred[0, 2] = C + D + 1 >> 1	pred[0, 2] = J + K+ 1 >> 1
pred[0, 3] = D + E + 1 >> 1	pred[0, 3] = J + 2K + L + 2 >> 2
pred[1, 0] = A + 2B + C + 2 >> 2	pred[1, 0] = J + K+ 1 >> 1
pred[1, 1] = B + 2C + D + 2 >> 2	pred[1, 1] = J + 2K + L + 2 >> 2
pred[1, 2] = C + 2D + E + 2 >> 2	pred[1, 2] = K + L + 1 >> 1
pred[1, 3] = D + 2E + F + 2 >> 2	pred[1, 3] = K + 3L + 2 >> 2
pred[2, 0] = B + C + 1 >> 1	pred[2, 0] = K + L + 1 >> 1
pred[2, 1] = C + D + 1 >> 1	pred[2, 1] = K + 3L + 2 >> 2
pred[2, 2] = D + E + 1 >> 1	pred[2, 2] = L
pred[2, 3] = E + F + 1 >> 1	pred[2, 3] = L
pred[3, 0] = B + 2C + D + 2 >> 2	pred[3, 0] = L
pred[3, 1] = C + 2D + E + 2 >> 2	pred[3, 1] = L
pred[3, 2] = D + 2E + F + 2 >> 2	pred[3, 2] = L
pred[3, 3] = E + 2F + G + 2 >> 2	pred[3, 3] = L
(h) 4x4 Vertical Left Mode	*(i) 4x4 Horizontal Up Mode*

Figure 2.4 Prediction Equations for 4x4 Luma Prediction Modes

Table 2.1 Availability of 4x4 Luma Prediction Modes

Availability of Neighboring 4x4 Luma Blocks	**Available 4x4 Luma Prediction Modes**
None available	DC
Left available, Top not available	Horizontal, DC, Horizontal Up
Top available, Left not available	Vertical, DC, Vertical Left, Diagonal Down-Left
Both available	All Modes

There are four 16x16 luma prediction modes designed in a directional manner. Each 16x16 luma prediction mode generates 256 predicted pixel values using some or all of the upper (H) and left-hand (V) neighboring pixels as shown in Figure 2.5. Vertical, Horizontal and DC modes are similar to 4x4 luma prediction modes. Plane mode is an approximation of bilinear transform with only integer arithmetic. The examples of each 16x16 luma prediction mode for real images are given in Figure 2.6. The prediction equations used in 16x16 luma prediction modes are shown in Figure 2.7 where [y, x] denotes the position of the pixel in a MB (the top left, top right, bottom left, and bottom right positions of a MB are denoted as [0,0], [0,15], [15,0], and [15,15], respectively), p represents the neighboring pixel values and Clip1 is to clip the result between 0 and 255.

DC mode is always used regardless of the availability of the neighboring pixels. However, it is adopted based on which neighboring pixels are available. The other prediction modes can only be used if all of the required neighboring pixels are available [1, 3]. Available 16x16 luma prediction modes for a MB depending on the availability of the neighboring MBs are given in Table 2.2.

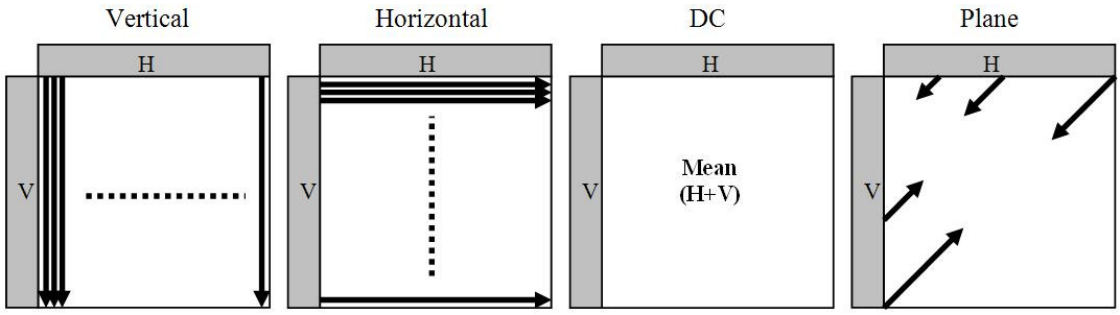


Figure 2.5 16x16 Luma Prediction Modes

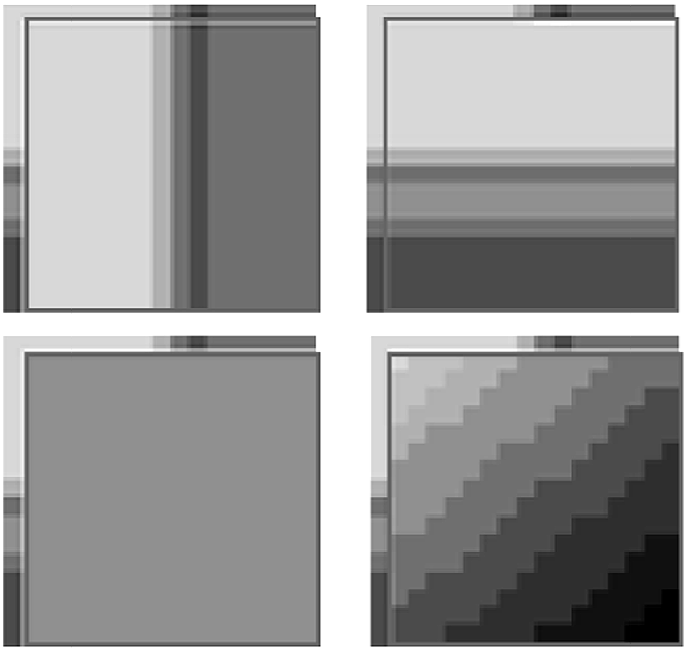


Figure 2.6 Examples of Real Images for 16x16 Luma Prediction Modes

Table 2.2 Availability of 16x16 Luma Prediction Modes

Availability of Neighboring 16x16 Luma Blocks	**Available 16x16 Luma Prediction Modes**
None available	DC
Left available, Top not available	Horizontal, DC
Top available, Left not available	Vertical, DC
Both available	All Modes

pred[x, 0] = p[-1, 0]	pred[0, y] = p[0, -1]
pred[x, 1] = p[-1, 1]	pred[1, y] = p[1, -1]
pred[x, 2] = p[-1, 2]	pred[2, y] = p[2, -1]
pred[x, 3] = p[-1, 3]	pred[3, y] = p[3, -1]
pred[x, 4] = p[-1, 4]	pred[4, y] = p[4, -1]
pred[x, 5] = p[-1, 5]	pred[5, y] = p[5, -1]

pred[x, 6] = p[-1, 6]	pred[6, y] = p[6, -1]
pred[x, 7] = p[-1, 7]	pred[7, y] = p[7, -1]
pred[x, 8] = p[-1, 8]	pred[8, y] = p[8, -1]
pred[x, 9] = p[-1, 9]	pred[9, y] = p[9, -1]
pred[x, 10] = p[-1, 10]	pred[10,y] = p[10, -1]
pred[x, 11] = p[-1, 11]	pred[11, y] = p[11, -1]
pred[x, 12] = p[-1, 12]	pred[12, y] = p[12, -1]
pred[x, 13] = p[-1, 13]	pred[13, y] = p[13, -1]
pred[x, 14] = p[-1, 14]	pred[14, y] = p[14, -1]
pred[x, 15] = p[-1, 15]	pred[15, y] = p[15, -1]
(a) 16x16 Vertical Mode	*(b) 16x16 Horizontal Mode*

$pred[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + \sum_{y'=0}^{15} p[-1, y'] + 16 \right) >> 5$	If the left and the top neighboring pixels are available
$pred[x, y] = \left(\sum_{y'=0}^{15} p[-1, y'] + 8 \right) >> 4$	Else If only the left neighboring pixels are available
$pred[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + 8 \right) >> 4$	Else If only the top neighboring pixels are available
$pred[x, y] = 128$	Else //If the left and the upper neighboring pixels are not available

(c) 16x16 DC Mode with x=0..15 and y=0..15

$$pred[x, y] = Clip1((a + b*(x-7) + c*(y-7) + 16) >> 5)$$
$$a = 16*(p[-1,15] + p[15,-1])$$
$$b = (5*H + 32) >> 6$$
$$c = (5*V + 32) >> 6$$
$$H = \sum_{x'=0}^{7} (x' + 1)*(p[8 + x', -1] - p[6 - x', -1])$$
$$V = \sum_{y'=0}^{7} (y' + 1)*(p[-1,8 + y'] - p[-1,6 - y'])$$

(d) 16x16 Plane Mode with x, y = 0..15

Figure 2.7 Prediction Equations for 16x16 Luma Prediction Modes

For the chroma components of a MB, a predicted 8x8 chroma block is formed for each 8x8 chroma component by performing intra prediction for the MB. The chroma component of a MB and its neighboring pixels are shown in Figure 2.8. There are four 8x8 chroma prediction modes which are similar to 16x16 luma prediction modes. A mode decision algorithm is used to compare the 8x8 predictions and select the best chroma prediction mode for each chroma component of the MB. Both chroma components of a MB always use the same prediction mode. The prediction equations used in 8x8 chroma prediction modes are shown in Figure 2.9 where [x, y] denotes the position of the pixel in a MB (the top left, top right, bottom left, and bottom right positions of a MB are denoted as [0,0], [0,7], [7,0], and [7,7], respectively), p represents the neighboring pixel values and Clip1 is to clip the result between 0 and 255.

DC mode is always used regardless of the availability of the neighboring pixels. However, it is adopted based on which neighboring pixels are available. The other prediction modes can only be used if all of the required neighboring pixels are available [1,3]. Available 8x8 chroma prediction modes for a MB depending on the availability of the neighboring MBs are given in Table 2.3.

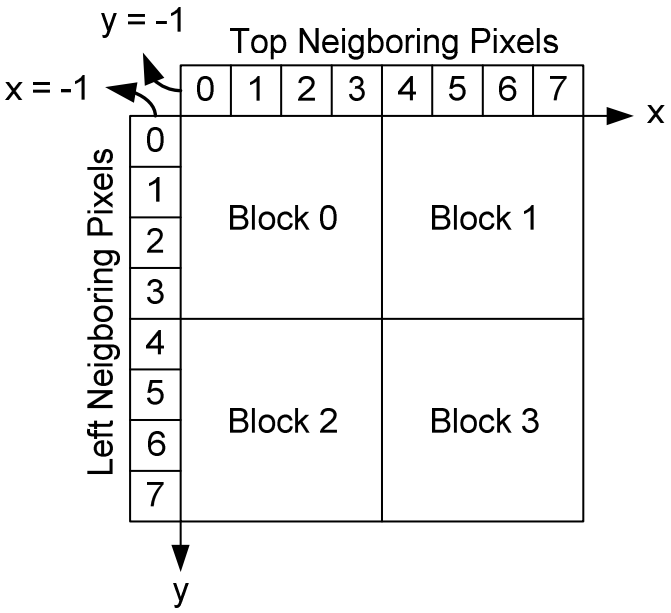


Figure 2.8 Chroma Component of a MB and its Neighboring Pixels

Table 2.3 Availability of 8x8 Luma Prediction Modes

Availability of Neighboring 8x8 Luma Blocks	**Available 8x8 Luma Prediction Modes**
None available	DC
Left available, Top not available	Horizontal, DC
Top available, Left not available	Vertical, DC
Both available	All Modes

predc[x, 0] = p[-1, 0]	predc[0, y] = p[0, -1]
predc[x, 1] = p[-1, 1]	predc[1, y] = p[1, -1]
predc[x, 2] = p[-1, 2]	predc[2, y] = p[2, -1]
predc[x, 3] = p[-1, 3]	predc[3, y] = p[3, -1]
predc[x, 4] = p[-1, 4]	predc[4, y] = p[4, -1]
predc[x, 5] = p[-1, 5]	predc[5, y] = p[5, -1]
predc[x, 6] = p[-1, 6]	predc[6, y] = p[6, -1]
predc[x, 7] = p[-1, 7]	predc[7, y] = p[7, -1]
(a) 8x8 Vertical Mode	*(b) 8x8 Horizontal Mode*

$predc[x, y] = \left(\sum_{x'=0}^{3} p[x', -1] + \sum_{y'=0}^{3} p[-1, y'] + 4 \right) >> 3$	If p[x, –1] with x = 0..3, and p[–1, y] with y = 0..3 are available
$predc[x, y] = \left(\sum_{y'=0}^{3} p[-1, y'] + 2 \right) >> 2$	Else If p[–1, y] with y = 0..3 are available and p[x, –1] with x = 0..3 are not available
$predc[x, y] = \left(\sum_{x'=0}^{3} p[x', -1] + 2 \right) >> 2$	Else If p[x, –1] with x = 0..3 are available and p[–1, y] with y = 0..3 are not available
$predc[x, y] = 128$	Else //If p[x, –1] with x = 0..3, and p[–1, y] with y = 0..3 are not available

(c) 8x8 DC Mode with x=0..3 and y=0..3 (Block 0 in Fig. 2.8)

$predc[x, y] = \left(\sum_{x'=4}^{7} p[x', -1] + 2 \right) >> 3$	If p[x, –1] with x = 4..7 are available
$predc[x, y] = \left(\sum_{y'=0}^{3} p[-1, y'] + 2 \right) >> 2$	Else If p[–1, y] with y = 0..3 are available

$predc[x, y] = 128$ Else //If p[x, –1] with x = 4..7, and p[–1, y] with y = 0..3 are not available

(c) 8x8 DC Mode with x=4..7 and y=0..3 (Block 1 in Fig. 2.8)

$$predc[x, y] = \left(\sum_{y'=4}^{7} p[-1, y'] + 2 \right) >> 2$$ If p[–1, y] with y = 4..7 are available

$$predc[x, y] = \left(\sum_{x'=0}^{3} p[x', -1] + 2 \right) >> 2$$ Else If p[x, –1] with x = 0..3 are available

$predc[x, y] = 128$ Else //If p[x, –1] with x = 0..3, and p[–1, y] with y = 4..7 are not available

(c) 8x8 DC Mode with x=0..3 and y=4..7 (Block 2 in Fig. 2.8)

$$predc[x, y] = \left(\sum_{x'=4}^{7} p[x', -1] + \sum_{y'=4}^{7} p[-1, y'] + 4 \right) >> 3$$ If p[x, –1] with x = 4..7, and p[–1, y] with y = 4..7 are available

$$predc[x, y] = \left(\sum_{y'=4}^{7} p[-1, y'] + 2 \right) >> 2$$ Else If p[–1, y] with y = 4..7 are available and p[x, –1] with x = 4..7 are not available

$$predc[x, y] = \left(\sum_{x'=4}^{7} p[x', -1] + 2 \right) >> 2$$ Else If p[x, –1] with x = 4..7 are available and p[–1, y] with y = 4..7 are not available

$predc[x, y] = 128$ Else //If p[x, –1] with x = 4..7, and p[–1, y] with y = 4..7 are not available

(c) 8x8 DC Mode with x=4..7 and y=4..7 (Block 3 in Fig. 2.8)

$$pred[x, y] = Clip1((a + b*(x-7) + c*(y-7) + 16) >> 5)$$
$$a = 16*(p[-1,7] + p[7,-1])$$
$$b = (17*H + 16) >> 5$$
$$c = (17*V + 16) >> 5$$
$$H = \sum_{x'=0}^{3} (x'+1)*(p[4+x',-1] - p[2-x',-1])$$
$$V = \sum_{y'=0}^{3} (y'+1)*(p[-1,4+y'] - p[-1,2-y'])$$

(d) 8x8 Plane Mode with x, y = 0..7

Figure 2.9 Prediction Equations for 8x8 Chroma Prediction Modes

2.2 Proposed Computational Complexity and Power Reduction Techniques

PECR technique exploits equality of neighboring pixels for simplifying the prediction calculations done by H.264 intra prediction modes. PSCR technique exploits similarity of neighboring pixels for simplifying the prediction calculations done by H.264 intra prediction modes. Both techniques are applied to H.264 4x4 luminance, 16x16 luminance and 8x8 chrominance prediction modes.

Intra 4x4 modes use 13 neighboring pixels for prediction calculations. PECR technique for intra 4x4 modes is based on the equality of the neighboring pixels A, B, C, D, E, F, G, H, I, J, K, L, M of the currently processed 4x4 block. Each intra 4x4 prediction mode uses some of these neighboring pixels to predict a 4x4 block. H.264 4x4 intra prediction modes and the neighboring pixels they use for prediction calculations are shown in Table 2.4. The prediction equations of a 4x4 intra prediction mode simplify to a constant value if the neighboring pixels used by this mode are all equal.

The prediction equation used by DC mode is given in equation (2.1). If the neighboring pixels A, B, C, D, I, J, K, L are equal, we can substitute one of the neighboring pixels, e.g. pixel A, in place of every neighboring pixel in equation (2.1). Therefore, the equation (2.1) simplifies to A as shown in (2.2).

Table 2.4 4x4 Intra Modes and Corresponding Neighboring Pixels

4x4 Intra Modes	Neighboring Pixels
Vertical	A, B, C, D
Horizontal	I, J, K, L
DC	A, B, C, D, I, J, K, L
Diagonal Down Left	A, B, C, D, E, F, G, H
Diagonal Down Right	A, B, C, D, I, J, K, L, M
Vertical Right	A, B, C, D, I, J, K, M
Horizontal Down	A, B, C, I, J, K, L, M
Vertical Left	A, B, C, D, E, F, G
Horizontal Up	I, J, K, L

$$pred[y,x] = [(A+B)+(C+D)+(I+J)+(K+L)+4] >> 3 \quad (2.1)$$

$$pred[y,x] = [8A+4] >>3 = A \quad (2.2)$$

This is the case for other prediction modes as well. For example, as shown in Figure 2.4, DDL mode uses A, B, C, D, E, F, G, H neighboring pixels in its prediction equations. The prediction equation for the pixel [0, 0] is given in equation (2.3). If neighboring pixels A, B, C, D, E, F, G, H are all equal, all prediction equations of DDL mode simplifies to a constant value as shown in (2.4).

$$pred[0, 0] = A + 2B + C + 2 >> 2 \quad (2.3)$$

$$pred[0,0] = [4A+2] >>2 = A \quad (2.4)$$

Since, in this case, all predicted pixels by DDL mode will be the same and equal to one of the neighboring pixels, the calculations done by DDL prediction mode become unnecessary. Therefore, during 4x4 intra prediction, the calculations done by DDL mode can be avoided by only comparing a few neighboring pixels at the beginning of the prediction process. During 4x4 intra prediction, the calculations done by the other prediction modes can be avoided in the same way by comparing the neighboring pixels used by the prediction equations of these modes.

PSCR technique for intra 4x4 modes is based on the similarity of the neighboring pixels of the currently processed 4x4 block. If the neighboring pixels used by the prediction equations of a 4x4 intra prediction mode are similar, the pixels predicted by this mode will

also be similar. PSCR technique determines the similarity of the neighboring pixels by truncating their least significant bits by the specified truncation amount (1, 2, 3, or 4 bits) and comparing the truncated pixels. If the truncated neighboring pixels of a prediction mode are all equal, one of the original neighboring pixels is substituted in place of every neighboring pixel in the prediction equations of this prediction mode. Therefore, prediction equations simplify to a constant value and prediction equation calculations become unnecessary.

The number of 4x4 intra prediction modes with equal and similar neighboring pixels in a frame varies from frame to frame. We analyzed CIF sized Foreman, Akiyo and Mother&Daughter frames at 28, 35 and 42 QP values using JM 14.0 to determine how many prediction modes have equal and similar neighboring pixels. The percentages of 4x4 modes that have equal neighboring pixels for each frame are given in Table 2.5. The percentage of prediction modes with equal neighboring pixels vary from 14% to 89%.

The percentages of 4x4 modes that have similar neighboring pixels for different truncation amounts for each frame are given in Table 2.6. The percentage of prediction modes with similar neighboring pixels vary from 11% to 94%. The percentage increases with higher QP values. Vert, Horz, Horz_up, DDL and Vert_left modes typically, on the average, have more than 50% similar neighboring pixels. DDR, Horz_down and Vert_right have relatively lower percentage with a typical value of more than 20%.

Table 2.7 shows the amount of computation performed by the prediction equations of each 4x4 intra mode in terms of number of addition and shift operations. Vertical and Horizontal modes require no computation. The prediction equations of the other modes include only addition and shift operations. Vertical right, Horizontal down and Vertical left modes have large amount of computation. A total of 882337 addition and 528045shift operations are performed by the H.264 4x4 intra prediction algorithm for a CIF (352x288) frame.

Based on this information and the information given in Tables 2.5, 2.6 and 2.7, we calculated the computation reduction achieved by the PECR and PSCR techniques for CIF size Foreman, Akiyo and Mother&Daughter frames. As shown in Table 2.8, the computation reduction ranges from 28% to 60% by PECR technique. As shown in Table 2.9, the computation reduction ranges from 18% to 68% by PSCR technique. The proposed

techniques, on the other hand, have an overhead of only 74882 comparisons for a CIF (352x288) frame.

Table 2.5 Percentage of 4x4 Intra Prediction Modes with Equal Neighboring Pixels

	4x4 Intra Modes	QP = 28	QP = 35	QP = 42
Foreman	VERT	50.17%	68.75%	84.31%
	HORZ/HORZ_UP	47.76%	65.74%	79.51%
	DC	29.34%	48.93%	68.77%
	DDL	40.94%	61.10%	80.26%
	DDR	14.08%	21.26%	24.61%
	VERT_RIGHT	14.55%	21.61%	25.02%
	HORZ_DOWN	14.47%	21.89%	24.78%
	VERT_LEFT	41.56%	61.58%	80.51%
Akiyo	VERT	65.01%	75.14%	85.89%
	HORZ/HORZ_UP	66.19%	78.82%	87.06%
	DC	48.94%	62.52%	76.69%
	DDL	56.66%	67.52%	81.06%
	DDR	28.54%	34.00%	35.05%
	VERT_RIGHT	28.88%	34.20%	35.31%
	HORZ_DOWN	29.25%	34.44%	35.50%
	VERT_LEFT	57.20%	67.93%	81.66%
Mother Daughter	VERT	57.58%	74.23%	87.58%
	HORZ/HORZ_UP	62.06%	77.90%	89.13%
	DC	43.62%	60.24%	78.31%
	DDL	48.33%	65.75%	82.04%
	DDR	29.20%	37.03%	37.50%
	VERT_RIGHT	29.34%	37.34%	37.86%
	HORZ_DOWN	30.59%	38.01%	38.19%
	VERT_LEFT	48.82%	66.16%	82.51%

Table 2.6 Percentage of 4x4 Intra Prediction Modes with Similar Neighboring Pixels

		Original (%)			1 bit Trunc. (%)			2 bit Trunc. (%)			3 bit Trunc. (%)			4 bit Trunc. (%)		
		QP			QP			QP			QP			QP		
	Modes	**28**	**35**	**42**	**28**	**35**	**42**	**28**	**35**	**42**	**28**	**35**	**42**	**28**	**35**	**42**
Foreman	V	50.1	70.9	85.1	52.1	70.7	85.6	54.0	72.3	85.6	59.5	74.0	87.6	68.3	78.4	89.0
	H/HU	46.4	65.6	79.9	48.5	66.9	80.4	50.5	67.3	80.0	55.9	70.4	82.2	65.1	75.4	84.7
	DDL	26.9	40.2	47.7	29.5	43.2	52.1	33.0	46.4	56.2	41.6	52.2	61.5	53.1	61.9	69.4
	VL	27.1	40.5	47.9	29.8	43.5	52.3	33.4	46.9	56.6	42.5	52.6	61.9	54.7	62.9	70.0
	HD	8.03	11.8	11.6	10.3	16.4	15.7	14.0	21.3	23.2	25.1	28.4	29.6	38.9	44.0	45.8
	VR	8.09	11.6	11.5	10.3	16.2	15.7	13.9	21.3	23.2	25.2	28.5	29.6	39.4	43.9	45.8
	DDR	7.92	11.5	11.4	10.1	15.9	15.5	13.5	21.0	22.8	24.4	27.9	29.1	37.2	42.8	45.2
	DC	9.91	13.4	13.6	13.2	18.4	18.4	18.0	25.0	26.1	27.9	32.8	33.8	40.3	46.6	50.1
Akiyo	V	65.4	78.1	87.1	67.0	78.0	87.2	68.0	79.3	89.1	70.7	80.2	90.2	74.7	83.0	91.4
	H/HU	66.6	81.0	90.2	68.6	80.4	90.6	70.8	82.1	91.6	74.1	83.3	92.3	80.2	86.7	93.6
	DDL	43.4	51.3	55.8	44.3	52.4	58.7	49.3	56.5	64.0	56.1	61.8	70.6	61.8	69.5	75.3
	VL	43.6	51.5	56.2	44.5	52.5	58.8	49.7	56.8	64.1	57.2	62.1	70.9	63.6	70.3	75.6
	HD	17.2	20.7	23.0	20.7	26.3	27.6	28.2	33.0	34.9	41.2	41.5	45.8	53.7	56.0	55.8
	VR	17.1	20.7	22.7	20.6	26.3	27.6	28.0	32.8	34.8	40.5	41.7	45.5	52.4	55.7	55.7
	DDR	16.9	20.5	22.6	20.5	26.2	27.5	27.8	32.7	34.7	40.0	41.3	45.4	51.1	55.0	55.5
	DC	20.2	24.0	25.8	25.1	30.1	31.5	33.1	37.6	39.0	44.3	46.5	50.2	54.5	58.9	60.8
M&D	V	59.5	77.4	90.2	61.2	78.5	91.0	63.2	79.6	91.7	66.3	81.2	92.9	72.8	84.1	93.4
	H/HU	62.2	78.7	90.6	62.5	79.2	91.0	65.3	81.0	92.1	69.7	82.7	92.9	75.9	85.7	93.9
	DDL	38.0	50.2	56.3	42.9	55.5	60.6	46.6	57.8	65.4	51.7	62.0	69.6	59.3	68.8	76.3
	VL	38.1	50.5	56.4	43.1	55.8	60.9	46.8	58.0	65.5	52.5	62.3	69.8	61.0	69.3	76.6
	HD	20.4	23.4	24.0	26.0	31.4	29.6	32.7	36.5	38.6	41.4	43.6	45.2	52.5	55.4	58.5
	VR	20.4	23.3	23.9	26.1	31.4	29.5	32.7	36.7	38.5	40.9	43.6	45.1	51.6	55.2	58.4
	DDR	20.3	23.1	23.9	25.8	31.1	29.4	32.5	36.3	38.4	40.5	43.3	44.9	50.3	54.7	58.2
	DC	23.1	25.7	26.2	29.3	34.0	32.5	36.8	40.2	42.3	44.3	48.3	50.0	53.2	59.3	63.4

Table 2.7 Computation Amount of 4x4 Intra Modes

Modes	**Number of Addition**	**Number of Shift**
DDL	21	14
DDR	21	14
VERT_RIGHT	26	16
HORZ_DOWN	26	16
VERT_LEFT	25	15
HORZ_UP	15	9
DC (Left Avail.)	4	1
DC (Top Avail.)	4	1
DC (Both Avail.)	8	1

Table 2.8 Intra 4x4 Modes Computation Reduction Results by PECR Technique

	QP	Addition Reduction		Shift Reduction	
		Number	Percent	Number	Percent
Foreman	28	246939	27.93%	146816	27.74%
	35	365863	41.38%	216263	40.87%
	42	459269	51.94%	269710	50.97%
Akiyo	28	386890	43.76%	229707	43.41%
	35	461728	52.22%	273099	51.61%
	42	521463	58.98%	306887	57.99%
Mother Daughter	28	359883	40.70%	214067	40.45%
	35	469840	53.14%	278673	52.66%
	42	539033	60.96%	317345	59.97%

Table 2.9 Intra 4x4 Modes Computation Reduction Results by PSCR Technique

		PSCR 1 bit Truncation		PSCR 2 bit Truncation		PSCR 3 bit Truncation		PSCR 4 bit Truncation	
	QP	Add. Red. Number Percent	Shift Red. Number Percent	Add. Red. Number Percent	Shift Red. Number Percent	Add. Red. Number Percent	Shift Red. Number Percent	Add. Red. Number Percent	Shift Red. Number Percent
Foreman	28	183794	111155	214423	129167	300422	180625	410869	246741
		%20.83	**%21.05**	**%24.30**	**%24.46**	**%34.05**	**%34.21**	**%46.57**	**%46.73**
	35	269853	163547	305898	184558	361121	217274	470764	282662
		%30.58	**%30.97**	**%34.67**	**%34.95**	**%40.93**	**%41.15**	**%53.35**	**%53.53**
	42	305594	185751	354949	214871	404623	244140	511344	307139
		%34.63	**%35.18**	**%40.23**	**%40.69**	**%45.86**	**%46.23**	**%57.95**	**%58.17**
Akiyo	28	297732	179575	351813	211561	437888	262795	519800	311536
		%33.74	**%34.01**	**%39.87**	**%40.06**	**%49.63**	**%49.77**	**%58.91**	**%59.00**
	35	359847	217158	407028	244912	466843	280299	562924	337540
		%40.78	**%41.12**	**%46.13**	**%46.38**	**%52.91**	**%53.08**	**%6380**	**%63.92**
	42	393978	238060	446557	269246	520684	313053	586565	351819
		%44.65	**%45.08**	**%50.61**	**%50.99**	**%59.01**	**%59.29**	**%66.48**	**%66.63**
M&D	28	314438	189134	361454	216771	422357	253059	503723	301789
		%35.64	**%35.82**	**%40.97**	**%41.05**	**%47.87**	**%47.92**	**%57.09**	**%57.15**
	35	392699	236762	427529	257091	477127	286213	557736	334094
		%44.51	**%44.84**	**%48.45**	**%48.69**	**%54.08**	**%54.20**	**%63.21**	**%63.27**
	42	409721	247572	469649	282807	516205	310142	603375	361596
		%46.44	**%46.88**	**%53.23**	**%53.56**	**%58.50**	**%58.73**	**%68.38**	**%68.48**

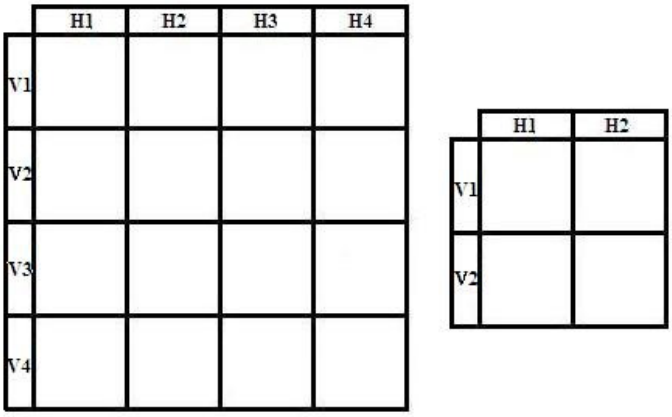


Figure 2.10 Four Pixel Groups of Neighboring Pixels of a MB

Intra 8x8 chrominance modes use 17 neighboring pixels for prediction calculations and intra 16x16 luminance modes use 33 neighboring pixels for prediction calculations. Therefore, the probability of all the neighboring pixels of an intra 8x8 or an intra 16x16 mode being equal is much smaller than that of an intra 4x4 mode. Therefore, as shown in Figure 2.10, we divide the neighboring pixels of intra 16x16 and intra 8x8 modes into four pixel groups (H1, H2, H3, H4, V1, V2, V3, V4) and check the equality of the neighboring pixels in each group separately.

We analyzed CIF sized Foreman, Akiyo and Mother Daughter frames at 28, 35 and 42 QP values respectively using JM 14.0 to determine how many 16x16 luminance and 8x8 chrominance four pixel groups have equal/similar pixels. The percentages of 16x16 luminance and 8x8 chrominance four pixel groups that have equal pixels for each frame are given in Tables 2.10 and 2.11 respectively. The percentages of 16x16 luminance and 8x8 chrominance four pixel groups that have similar pixels for each frame are given in Tables 2.12 and 2.13 respectively.

There are 396 MBs in CIF sized frame, but 378 MBs have horizontal groups H1, H2, H3, H4 and 374 MBs have vertical groups V1, V2, V3, V4. For intra 16x16 luminance modes, the percentage of four pixel groups with equal pixels ranges from 43% to 77% and it is typically greater than 50%. For intra 8x8 chrominance modes, the percentage ranges from 73% to 90% and it is typically more than 80%. For intra 16x16 luminance modes, the percentage of four pixel groups with similar pixels ranges from 42% to 90% and it is

typically greater than 50%. For intra 8x8 chrominance modes, the percentage ranges from 73% to 95% and it is typically more than 80%.

Table 2.14 shows the amount of computation performed by the prediction equations of each 16x16 and 8x8 intra mode in terms of number of addition and shift operations. Vertical and Horizontal modes require no computation. The prediction equations of the DC mode include only addition and shift operations. Plane mode have large amount of computation, and as shown in Figure 2.8, it uses multiplication in the prediction equations. But the multiplication operation can be replaced with addition and shift operations [22, 24]. Therefore, a total of 121631 addition and 106067 shift operations are performed by the H.264 16x16 intra prediction algorithm for a CIF (352x288) frame, and a total of 30778

Table 2.10 Percentage of 16x16 Intra Prediction Modes with Equal Neighboring Pixels

		QP = 28	**QP = 35**	**QP = 42**
Foreman	H1	45.96%	61.36%	68.43%
	H2	46.21%	65.15%	72.22%
	H3	47.22%	62.12%	71.97%
	H4	43.94%	62.12%	71.46%
	V1	46.97%	58.59%	65.15%
	V2	45.20%	58.33%	65.91%
	V3	45.71%	56.06%	67.17%
	V4	43.69%	56.57%	62.37%
Akiyo	H1	61.36%	63.38%	70.45%
	H2	58.33%	64.65%	71.46%
	H3	58.59%	66.67%	71.21%
	H4	57.83%	60.86%	70.71%
	V1	61.87%	65.91%	70.96%
	V2	58.59%	67.93%	71.46%
	V3	58.84%	67.93%	70.20%
	V4	61.11%	69.70%	71.21%
Mother Daughter	H1	48.99%	65.15%	74.49%
	H2	52.53%	67.93%	71.46%
	H3	49.24%	65.91%	74.49%
	H4	47.47%	64.39%	68.18%
	V1	51.77%	65.66%	74.49%
	V2	58.59%	71.97%	76.52%
	V3	57.32%	70.45%	76.77%
	V4	60.10%	74.24%	77.53%

Table 2.11 Percentage of 8x8 Intra Prediction Modes (Chroma CB, CR) with Equal Neighboring Pixels

			QP = 28	QP = 35	QP = 42
Foreman	Cb	H1	78.03%	84.85%	87.37%
		H2	78.79%	85.35%	87.12%
		V1	79.04%	85.86%	86.11%
		V2	78.54%	84.85%	86.87%
	Cr	H1	83.33%	85.35%	84.60%
		H2	84.60%	85.35%	87.12%
		V1	86.62%	85.10%	85.10%
		V2	83.59%	85.35%	85.10%
Akiyo	Cb	H1	73.23%	79.55%	82.07%
		H2	74.75%	81.31%	84.34%
		V1	75.51%	78.79%	84.09%
		V2	77.53%	79.55%	83.33%
	Cr	H1	78.03%	83.08%	85.10%
		H2	80.81%	83.59%	86.87%
		V1	80.05%	83.08%	87.37%
		V2	79.80%	81.57%	86.11%
Mother Daughter	Cb	H1	84.85%	84.09%	80.30%
		H2	80.05%	82.32%	84.09%
		V1	81.57%	86.36%	87.37%
		V2	83.59%	87.37%	87.63%
	Cr	H1	82.83%	85.10%	86.36%
		H2	85.10%	86.62%	88.38%
		V1	82.83%	86.62%	89.14%
		V2	85.61%	86.62%	89.90%

Table 2.12 Percentage of 16x16 Intra Prediction Modes with Similar Neighboring Pixels

		PSCR 1 bit Trunc. (%)			PSCR 2 bit Trunc. (%)			PSCR 3 bit Trunc. (%)			PSCR 4 bit Trunc. (%)		
		QP			QP			QP			QP		
		28	35	42	28	35	42	28	35	42	28	35	42
Foreman	H1	49.7	63.1	72.4	51.2	66.1	74.2	59.6	70.4	78.0	64.6	73.9	80.8
	H2	52.2	67.4	76.5	54.0	71.4	77.5	58.5	73.9	83.3	68.1	79.8	87.1
	H3	47.2	64.9	75.5	50.7	68.9	75.7	60.3	72.4	81.0	67.1	77.5	83.5
	H4	48.2	63.8	75.5	49.7	67.1	75.5	54.5	69.9	80.3	62.3	71.7	82.3
	V1	49.7	62.8	68.1	51.7	64.9	70.7	56.3	68.6	76.0	64.6	72.2	79.2
	V2	47.9	61.1	70.4	48.9	62.8	70.2	54.8	67.1	74.7	62.6	69.7	78.7
	V3	46.4	58.5	71.4	51.0	62.6	71.7	54.8	66.4	76.0	60.8	71.2	80.3
	V4	46.2	59.8	67.6	47.7	59.8	67.9	52.2	64.6	70.4	62.1	69.9	77.7
Akiyo	H1	62.6	70.2	78.2	64.9	72.2	80.0	66.9	73.9	83.3	69.4	77.7	84.8
	H2	60.1	68.1	75.7	62.3	71.4	80.3	63.3	73.2	83.8	69.1	76.7	84.8
	H3	63.8	71.2	78.0	62.8	74.4	83.8	68.4	75.7	85.6	72.2	80.0	85.6
	H4	62.6	67.4	78.0	62.3	69.1	79.8	65.4	71.2	83.5	70.7	76.0	85.6
	V1	64.1	69.9	80.5	69.4	73.9	81.8	70.7	77.2	86.1	77.2	82.8	88.6
	V2	63.6	72.2	81.3	67.6	75.5	83.0	69.7	78.2	86.8	75.7	82.5	87.3
	V3	64.6	73.4	79.5	66.1	76.2	83.8	69.1	80.0	85.3	73.9	82.5	87.8
	V4	65.4	74.7	81.3	65.6	77.7	85.3	71.9	77.7	87.1	78.7	83.5	88.3
M&D	H1	52.7	71.9	80.5	57.0	74.7	84.0	60.6	77.2	85.6	67.4	81.3	87.6
	H2	58.0	70.4	77.7	58.8	75.5	80.0	61.6	78.2	85.1	68.9	79.5	87.8
	H3	55.3	72.2	79.2	58.5	75.0	82.0	63.6	78.2	85.3	69.9	79.2	86.8
	H4	54.0	69.7	73.2	56.5	71.2	78.7	60.1	73.2	79.5	66.6	75.2	83.5
	V1	53.5	69.1	78.0	56.3	71.7	81.0	62.1	75.7	85.1	68.6	76.2	86.3
	V2	59.6	73.9	81.8	63.3	78.7	85.6	66.9	81.3	88.6	72.7	82.8	90.6
	V3	57.3	71.4	83.5	60.3	75.5	85.8	66.1	77.5	86.3	71.9	80.0	89.6
	V4	61.6	78.0	82.8	66.4	79.8	84.8	69.9	81.3	89.1	76.2	84.3	90.1

addition and 106067 shift operations are performed by the H.264 8x8 intra prediction algorithm for a CIF (352x288) frame.

The proposed PECR technique simplifies both 16x16 and 8x8 DC and plane mode prediction equations significantly. As shown in (2.5), 16x16 DC mode prediction equations add the upper and left neighboring pixels with a constant value and divide the result by 32. The part of the prediction equation using the neighboring pixels in H1 group is shown in equation (2.6). If the neighboring pixels in H1 group are equal, in this part of the prediction equation, instead of adding the four neighboring pixels in the H1 group, one of the neighboring pixels can be shifted by 2 as shown in (2.6). In this way, three addition operations are replaced with one shift operation. This is the case for the other four neighboring pixel groups as well. Whenever the four pixels in a group are equal, three

addition operations are avoided by doing one shift operation. A similar computation reduction is achieved for 16x16 plane mode as well.

Table 2.13 Percentage of 8x8 Intra Prediction Modes (Chroma CB, CR) with Similar Neighboring Pixels

			PSCR 1 bit Trunc. (%)			PSCR 2 bit Trunc. (%)			PSCR 3 bit Trunc. (%)			PSCR 4 bit Trunc. (%)		
			QP			QP			QP			QP		
			28	35	42	28	35	42	28	35	42	28	35	42
Foreman	Cb	H1	81.82	88.38	89.14	86.62	89.39	91.67	87.63	93.18	93.69	91.67	93.94	94.19
		H2	82.83	90.40	89.39	83.84	91.92	90.91	85.61	92.42	93.94	90.91	94.44	93.94
		V1	81.82	86.11	88.64	83.84	88.38	91.16	85.61	90.91	92.68	89.14	92.68	95.20
		V2	80.30	86.62	88.89	84.34	88.38	90.66	86.36	90.15	92.17	90.66	92.68	93.69
	Cr	H1	85.61	88.64	90.40	87.88	88.89	90.66	89.65	92.17	92.42	91.16	93.18	93.69
		H2	87.88	89.39	87.88	89.14	91.67	91.16	91.92	92.93	93.69	92.42	92.68	93.94
		V1	84.09	87.63	85.35	87.88	90.91	91.41	91.16	93.18	93.94	92.17	93.43	93.43
		V2	85.61	88.89	85.86	86.11	90.66	92.68	88.38	92.42	91.67	90.40	93.18	92.93
Akiyo	Cb	H1	76.77	82.83	82.07	78.54	83.84	83.84	79.80	84.34	84.34	83.33	85.35	86.62
		H2	77.27	84.09	84.34	79.04	86.11	86.36	81.57	87.88	88.38	86.62	88.38	89.14
		V1	76.52	81.31	85.10	79.80	84.34	86.11	84.85	86.36	86.36	86.36	87.12	88.13
		V2	79.04	82.58	83.84	81.57	84.85	86.87	84.09	86.62	87.63	85.10	87.63	88.64
	Cr	H1	79.04	83.33	83.59	81.06	86.36	86.62	83.84	87.37	89.14	87.63	88.64	90.66
		H2	82.83	86.36	85.61	84.34	85.35	86.36	86.11	88.64	89.39	89.65	88.89	90.15
		V1	83.84	85.61	89.14	84.34	87.88	89.65	86.62	90.91	93.18	90.91	92.42	94.95
		V2	80.05	84.60	87.63	84.34	88.64	88.13	85.61	89.90	90.15	89.39	90.15	92.17
M&D	Cb	H1	84.34	87.37	87.12	87.12	89.65	90.15	87.88	92.17	92.17	90.40	92.68	93.69
		H2	79.80	84.34	88.13	84.09	86.87	90.66	85.10	88.38	92.93	87.37	89.65	93.43
		V1	82.07	89.90	90.15	86.11	90.91	92.93	86.87	92.93	91.92	89.65	95.20	94.44
		V2	88.38	90.91	88.89	88.89	92.93	93.18	92.68	93.43	93.94	91.92	94.70	94.95
	Cr	H1	83.33	87.37	89.65	86.62	89.65	89.90	89.14	91.41	92.17	90.15	92.42	93.69
		H2	85.35	89.39	88.89	89.65	89.90	90.66	90.15	91.67	92.42	92.68	93.69	93.18
		V1	85.35	86.62	88.64	87.12	90.40	91.41	89.14	93.94	92.93	91.92	94.95	93.94
		V2	89.39	90.15	87.88	90.66	93.69	93.43	92.93	94.70	92.93	94.19	94.70	94.70

$$\text{pred}[y,x] = (\sum (p[x',-1]+p[-1, x'])+16) >> 5, \text{ with } x' = 0, 1, \ldots,15$$
$$= (p[0,-1]+ p[1,-1]+\ldots+p[15,-1] + p[-1,0]+ p[-1,1]+\ldots+p[-1,15]+16)>>5 \quad (2.5)$$

$$p[0,-1]+p[1,-1]+p[2,-1]+p[3,-1] = 4*p[0,-1] = p[0,-1]<<2 \quad (2.6)$$

Table 2.14 Computation Amount of Intra 16x16 and Intra 8x8 Modes

MODES	Intra 16x16		Intra 8x8	
	Number of Addition	Number of Shift	Number of Addition	Number of Shift
PLANE	307	296	89	82
DC (Left Available)	16	1	8	2
DC (Top Available)	16	1	8	2
DC (Both Available)	32	1	24	4

Plane mode prediction equations, however, are more complex than DC mode prediction equations. Plane mode has two calculation steps as shown in Figure 2.7. The first step calculates a, b, c parameters from the neighboring pixels of the current MB, and only 16% of the total plane mode calculations are performed in the first step. The second step calculates the predicted pixels from a, b, c parameters and 84% of the total plane mode calculations are performed in the second step. The predicted pixel values by the plane mode are the weighted sum of a, b and c parameters. If b or c or both are equal to zero, the plane mode prediction equations simplify significantly. Therefore, the proposed PECR and PSCR techniques also check whether b and c parameters are equal/similar to zero or not before the second step, and in this way, it avoids many additional unnecessary calculations with an additional small comparison overhead.

Based on the information given in Tables 2.10, 2.11 and 2.14, we calculated the computation reduction achieved by the PECR technique for intra 16x16 and intra 8x8 prediction modes for CIF-sized Foreman, Akiyo and Mother Daughter frames. As shown in Tables 2.15 and 2.16, the computation reduction ranges from 28% to 68% by PECR technique. Based on the information given in Tables 2.12, 2.13, and 2.14, we calculated the computation reduction achieved by the PSCR technique for intra 16x16 and intra 8x8 prediction modes for different truncation amounts for CIF-sized Foreman, Akiyo and Mother Daughter frames. As shown in Tables 2.17 and 2.18, the computation reduction ranges from 13% to 65% by PSCR technique.

Table 2.15 Intra 16x16 Computation Reduction Results by PECR

	QP	Addition Reduction		Shift Reduction	
		Number	Percent	Number	Percent
Foreman	28	16183	13.30%	9403	8.87%
	35	19662	16.17%	10764	10.15%
	42	21831	17.95%	11786	11.11%
Akiyo	28	28865	23.73%	20182	19.03%
	35	30204	24.83%	20608	19.43%
	42	30950	25.45%	20604	19.43%
Mother Daughter	28	25660	21.10%	17911	16.89%
	35	34543	28.40%	24566	23.16%
	42	33779	27.77%	22875	21.57%

Table 2.16 Intra 8x8 (Chroma CB, CR) Computation Reduction Results by PECR

		QP	Addition Reduction		Shift Reduction	
			Number	Percent	Number	Percent
Foreman	Cb	28	19347	47.60%	11293	36.69%
		35	24624	60.58%	15847	51.49%
		42	26018	64.01%	17055	55.41%
	Cr	28	23379	57.52%	14688	47.72%
		35	26925	66.24%	18113	58.85%
		42	27771	68.33%	18885	61.36%
Akiyo	Cb	28	19712	48.50%	12035	39.10%
		35	21863	53.79%	13650	44.35%
		42	24893	61.24%	16277	52.89%
	Cr	28	21746	53.50%	13557	44.05%
		35	23199	57.08%	14678	47.69%
		42	25498	62.73%	16591	53.91%
Mother Daughter	Cb	28	20844	51.28%	12318	40.02%
		35	23833	58.64%	15046	48.89%
		42	25751	63.36%	17022	55.31%
	Cr	28	21327	52.47%	12706	41.28%
		35	25386	62.46%	16496	53.60%
		42	27618	67.95%	18482	60.05%

Table 2.17 Intra 16x16 Computation Reduction Results by PSCR

		PSCR 1 bit Truncation		PSCR 2 bit Truncation		PSCR 3 bit Truncation		PSCR 4 bit Truncation	
	QP	Add. Red.	Shift Red.	Add. Red.	Shift Red.	Add. Red.	Shift Red.	Add. Red.	Shift Red.
		Number **Percent**	Number **Percent**	Number **Percent**	Number **Percent**	Number **Percent**	Number **Percent**	Number **Percent**	Number **Percent**
Foreman	**28**	16663 **%13.70**	9466 **%8.92**	16622 **%13.67**	9089 **%8.57**	16525 **%13.59**	8132 **%7.67**	18935 **%15.57**	9428 **%8.89**
	35	23891 **%19.64**	14594 **%13.76**	22441 **%18.45**	12723 **%12.00**	24890 **%20.46**	14614 **%13.78**	25477 **%20.95**	14601 **%13.77**
	42	27631 **%22.72**	16963 **%15.99**	27738 **%22.81**	16960 **%15.99**	28135 **%23.13**	16672 **%15.72**	30801 **%25.32**	18793 **%17.72**
Akiyo	**28**	27958 **%22.99**	18687 **%17.62**	28059 **%23.07**	18508 **%17.45**	25759 **%21.18**	15761 **%14.86**	26765 **%22.01**	15992 **%15.08**
	35	32819 **%26.98**	22446 **%21.16**	34101 **%28.04**	23287 **%21.95**	32756 **%26.93**	21627 **%20.39**	34397 **%28.28**	22622 **%21.33**
	42	41978 **%34.51**	30419 **%28.68**	43263 **%35.57**	31236 **%29.45**	48364 **%39.76**	35885 **%33.83**	43436 **%35.71**	30744 **%28.99**
M&D	**28**	28253 **%23.23**	20024 **%18.88**	29854 **%24.54**	21160 **%19.95**	30238 **%24.86**	20905 **%19.71**	33663 **%27.68**	23376 **%22.04**
	35	37207 **%30.59**	26633 **%25.11**	37820 **%31.09**	26755 **%25.22**	38265 **%31.46**	26814 **%25.28**	36790 **%30.25**	25030 **%23.60**
	42	41987 **%34.52**	30278 **%28.55**	42844 **%35.22**	30668 **%28.91**	43436 **%35.71**	30829 **%29.07**	43635 **%35.87**	30692 **%28.94**

H.264 intra 4x4 prediction equations and intra 16x16 prediction equations use the same neighboring pixels at MB boundaries. Since the proposed techniques checks the equality/similarity of these neighboring pixels for intra 4x4 modes, these equality/similarity results are re-used for checking the equality/similarity of four neighboring pixel groups for intra 16x16 modes, and therefore, 3008 1-bit comparisons are performed for intra 16x16 DC and plane modes. In addition, 714 comparisons are performed for checking the equality/similarity of parameters b and c to zero for 16x16 plane mode. The both proposed techniques, on the other hand, requires 3x4x2=24 comparison operations for checking the equality/similarity of four neighboring pixel groups for intra 8x8 prediction calculations of the current Cb and Cr chrominance blocks. Therefore, they have an overhead of 5226 comparisons for intra 8x8 prediction modes.

Table 2.18 Intra 8x8 (Chroma CB, CR) Computation Reduction Results by PSCR

			PSCR 1 bit Truncation		PSCR 2 bit Truncation		PSCR 3 bit Truncation		PSCR 4 bit Truncation	
		QP	Add. Red.	Shift Red.	Add. Red.	Shift Red.	Add. Red.	Shift Red.	Add. Red.	Shift Red.
			Number Percent	Number Percent	Number Percent	Number Percent	Number Percent	Number Percent	Number Percent	Number Percent
Foreman	Cb	28	16947 %45.03	10524 %35.05	17031 %45.25	10376 %34.56	17135 %45.53	10351 %34.47	17936 %47.66	10834 %36.08
		35	21674 %57.59	14731 %49.06	21613 %57.42	14544 %48.44	22410 %59.54	15161 %50.49	23525 %62.50	16123 %53.70
		42	23386 %62.14	16338 %54.41	23873 %63.43	16655 %55.47	23685 %62.93	16313 %54.33	23871 %63.42	16407 %54.64
	Cr	28	20416 %54.24	13646 %45.45	20643 %54.85	13717 %45.68	21148 %56.19	14017 %46.68	21080 %56.01	13851 %46.13
		35	23695 %62.96	16671 %55.52	24055 %63.91	16879 %56.21	23448 %62.30	16116 %53.67	23948 %63.63	16574 %55.20
		42	24768 %65.81	17821 %59.35	24654 %65.50	17389 %57.91	24157 %64.18	16787 %55.91	24637 %65.46	17217 %57.34
Akiyo	Cb	28	17704 %47.04	11640 %38.77	18141 %48.20	11889 %39.60	18233 %48.44	11757 %39.16	18410 %48.91	11717 %39.02
		35	19759 %52.50	13257 %44.15	19877 %52.81	13210 %44.00	20547 %54.59	13752 %45.80	20293 %53.92	13438 %44.75
		42	22107 %58.74	15479 %51.55	22046 %58.58	15267 %50.85	21795 %57.91	14950 %49.79	21967 %58.37	15005 %49.97
	Cr	28	19555 %51.96	13154 %43.81	19735 %52.44	13169 %43.86	19646 %52.20	12925 %43.05	19809 %52.63	12787 %42.59
		35	20883 %55.49	14187 %47.25	20523 %54.53	13671 %45.53	21142 %56.17	14113 %47.00	21510 %57.15	14411 %48.00
		42	21889 %58.16	15060 %50.16	22191 %58.96	15265 %50.84	22375 %59.45	15229 %50.72	22213 %59.02	14952 %49.80
M&D	Cb	28	18210 %48.38	11561 %38.50	18311 %48.65	11437 %38.09	18602 %49.42	11600 %38.63	18539 %49.26	11406 %37.99
		35	21034 %55.89	14021 %46.70	21443 %56.97	14271 %47.53	21687 %57.62	14383 %47.90	21371 %56.78	13969 %46.52
		42	23154 %61.52	16141 %53.76	23130 %61.46	15872 %52.86	23489 %62.41	16148 %53.78	23126 %61.44	15682 %52.23
	Cr	28	18340 %48.73	11620 %38.70	19296 %51.27	12354 %41.14	19273 %51.21	12189 %40.59	19634 %52.17	12396 %41.28
		35	22015 %58.49	15039 %50.09	22280 %59.20	15103 %50.30	22345 %59.37	15010 %49.99	22433 %59.60	15018 %50.02
		42	23822 %63.29	16788 %55.91	23934 %63.59	16697 %55.61	23649 %62.83	16318 %54.35	23548 %62.57	16120 %53.69

We also quantified the impact of the proposed PSCR technique on the rate-distortion performance of the 4x4 intra prediction algorithm by using H.264 JM reference software encoder version 14.0. The rate distortion curves and average PSNR comparison of the original 4x4 intra prediction algorithm and the 4x4 intra prediction algorithm with the proposed technique for several CIF size benchmark video frames and different neighboring

pixel truncation amounts are shown in Fig. 2.11 and Table 2.19 respectively. The average PSNR values shown in Table 2.19 are calculated using the technique described in [26]. The proposed technique does not change the PSNR for some video frames, it increases the PSNR slightly for some video frames and it decreases the PSNR slightly for some video frames.

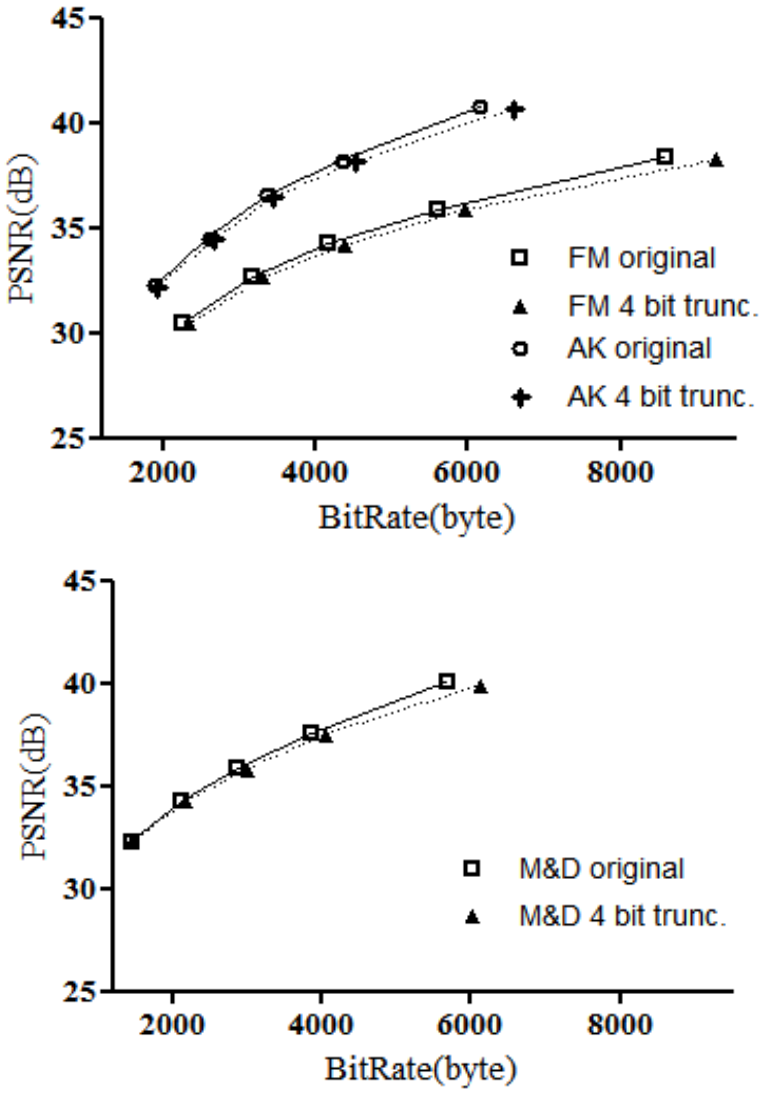


Figure 2.11 Rate Distortion Curves of the Original 4x4 Intra Prediction Algorithm and 4x4 Intra Prediction Algorithm with Proposed Technique

Table 2.19 Average PSNR Comparison of the Proposed PSCR Technique

Frame		Org. (dB)	PSCR 1bT (dB)	Diff. (dB)	PSCR 2bT (dB)	Diff. (dB)	PSCR 3bT (dB)	Diff. (dB)	PSCR 4bT (dB)	Diff. (dB)
FM	Y	35.28	35.26	**-0.02**	35.26	**-0.02**	35.21	**-0.08**	35.01	**-0.27**
	Cb	40.40	40.37	**-0.03**	40.41	**0.01**	40.36	**-0.04**	40.34	**-0.06**
	Cr	42.53	42.60	**0.07**	42.64	**0.11**	42.66	**0.13**	42.53	**0.00**
AK	Y	37.25	37.28	**0.03**	37.21	**-0.04**	37.17	**-0.08**	36.96	**-0.30**
	Cb	40.48	40.44	**-0.04**	40.48	**0.00**	40.46	**-0.03**	40.28	**-0.21**
	Cr	42.61	42.64	**0.03**	42.60	**-0.01**	42.52	**-0.09**	42.29	**-0.32**
M&D	Y	36.82	36.86	**0.03**	36.80	**-0.02**	36.74	**-0.08**	36.54	**-0.28**
	Cb	42.20	42.23	**0.02**	42.08	**-0.12**	42.10	**-0.10**	41.99	**-0.21**
	Cr	43.29	43.21	**-0.08**	43.16	**-0.14**	43.22	**-0.07**	43.07	**-0.23**

2.3 Proposed Intra Prediction Hardware Architecture

The proposed hardware architecture for implementing H.264 4x4 intra prediction algorithm is shown in Fig. 2.12.

Three local neighboring buffers, top neighboring buffer, left neighboring buffer and reconstructed pixel neighboring buffer are used to store the neighboring pixels in the previously coded and reconstructed neighboring 4x4 luma blocks in the current MB. After a 4x4 luma block in the current MB is coded and reconstructed, the neighboring pixels in this block are stored in the corresponding local buffers. 9 parallel datapaths are used to calculate the predicted pixels. Each datapath is used to calculate the predicted pixels by a different 4x4 intra prediction mode.

13 registers are used to store the neighboring pixels (A, B, C, D, E, F, G, H, I, J, K, L, M) for the current 4x4 block. When a new 4x4 block comes, neighboring pixel registers are loaded with the current neighboring pixels or the current neighboring pixels truncated by the corresponding truncation amount (1, 2, 3, or 4 bits) in four cycles. 12 8-bit comparators are used to check for the equality of these truncated neighboring pixels. Based on the comparison results, a disable signal is generated and sent to the datapaths used for implementing the prediction modes with similar neighboring pixels.

9 4x32 register files are used to store the predicted pixels for 9 4x4 intra prediction modes. When a datapath implementing a prediction mode is disabled, one of the neighboring pixels is taken as the predicted pixel, the clock signal for the corresponding predicted pixel register file is gated, and therefore this register file is not loaded.

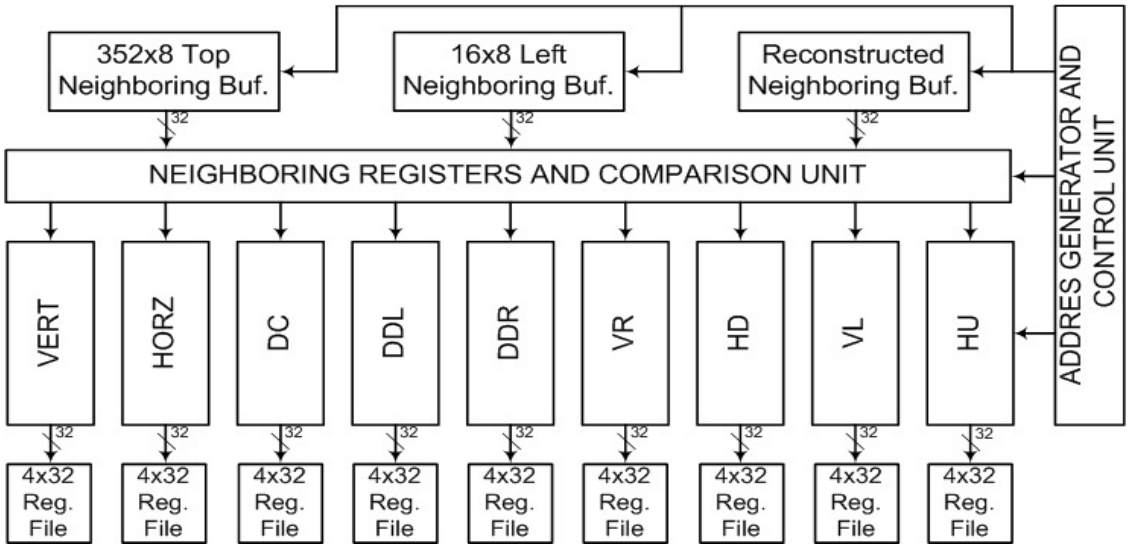


Figure 2.12 4x4 Intra Prediction Hardware Architecture

The proposed hardware architecture is implemented in Verilog HDL. The implementation is verified with RTL simulations. The Verilog RTL code is synthesized to 2V8000ff1157 Xilinx Virtex II FPGA with speed grade 5 using Mentor Graphics Precision RTL 2005b. The resulting netlist is placed and routed to the same FPGA at 50MHz using Xilinx ISE 8.2i. The resulting hardware uses 2448 4 input LUTs, 359 DFFs and 2 BlockRAMs.

2.4 Power Consumption Analysis

The power consumption of intra prediction hardware on a Xilinx Virtex II FPGA is estimated using Xilinx XPower tool. In order to estimate its power consumption, timing simulation of the placed and routed netlist of intra prediction hardware is done using Mentor Graphics ModelSim SE. Foreman, Akiyo and Mother&Daughter frames are used as inputs for timing simulations and the signal activities are stored in VCD files. These VCD files are used for estimating the power consumption of intra prediction hardware using Xilinx XPower tool.

The power consumptions of the proposed hardware implementations on a Xilinx Virtex II FPGA at 25 MHz are shown in Tables 2.20 - 2.22 for different QP values and video frames. As shown in the tables 2.20, 2.21 and 2.22, proposed PECR and PSCR power

reduction techniques reduce the power consumption of the intra 4x4 prediction hardware up to 46% and 57%, respectively.

Since intra prediction hardware will be used as part of an H.264 video encoder, only internal power consumption is considered and input and output power consumptions are ignored. Therefore, the power consumption of an intra prediction hardware can be divided into three main categories; signal power, logic power and clock power. Signal power is the power dissipated in routing tracks between logic blocks. Logic power is the amount of power dissipated in the parts where computations take place. Clock power is due to clock tree used in the FPGA.

There are several reasons for the differences between the computation reduction percentages shown in Tables 2.15 and 2.17 the power reduction percentages shown in Tables 2.20 - 2.22. The first reason is the power consumption overhead for the comparisons performed before the prediction process. For intra 4x4 prediction hardware, there are at most 12 comparisons among 13 neighboring pixels. These comparison operations add some power consumption overhead to 4x4 intra prediction hardware architecture.

The second reason is the clock power. Since we did not do clock gating in the FPGA for the disabled datapaths, the datapaths for all prediction modes are supplied with clock regardless of the equality of the neighboring pixels. Therefore, as shown in Tables 2.20 – 2.22, this implementation of the power reduction technique does not reduce the clock power.

The third reason is that even if the datapath of a prediction mode is disabled, address generator, control unit, neighboring registers and local neighboring buffers consume power for writing the predicted pixels for that prediction mode into the corresponding register file.

Table 2.20 Power Consumption Reduction (Q=28) by PSCR Technique

Frames	Caetgory	Power (mW)					
		Org.	PECR Tech.	PSCR Tech. (1bT)	PSCR Tech. (2bT)	PSCR Tech. (3bT)	PSCR Tech. (4bT)
Foreman	Clock	35	20	20	20	21	17
	Logic	29.32	12.44	12.28	12.22	11.55	10.2
	Signal	55.15	41.52	40.95	40.73	37.02	33.74
	Total	119.47	73.96	73.23	72.95	69.57	60.93
	Red. (%)		**38.09**	**38.70**	**38.94**	**41.77**	**49.00**
Akiyo	Clock	35	20	20	20	21	17
	Logic	28.55	10.81	10.45	10.27	9.55	8.46
	Signal	50.98	35.45	34.28	33.63	29.94	27.07
	Total	114.53	66.25	64.73	63.9	60.49	52.53
	Red. (%)		**42.15**	**43.48**	**44.21**	**47.18**	**54.13**
M&D	Clock	35	20	20	20	21	17
	Logic	28.37	10.76	10.45	10.28	9.68	8.7
	Signal	50.58	35.61	34.4	33.63	30.51	28
	Total	113.95	66.37	64.84	63.91	61.19	53.7
	Red. (%)		**41.76**	**43.10**	**43.91**	**46.30**	**52.87**

Table 2.21 Power Consumption Reduction (Q=35) by PSCR Technique

Frames	Caetgory	Power (mW)					
		Org.	PECR Tech.	PSCR Tech. (1bT)	PSCR Tech. (2bT)	PSCR Tech. (3bT)	PSCR Tech. (4bT)
Foreman	Clock	35	20	20	20	21	17
	Logic	28.8	11.3	11.12	11.07	10.44	9.28
	Signal	53.58	37.76	37.13	36.9	33.48	30.61
	Total	117.37	69.06	68.25	67.97	64.92	56.88
	Red. (%)		**41.16**	**41.85**	**42.09**	**44.69**	**51.54**
Akiyo	Clock	35	20	20	20	21	17
	Logic	28.32	10.13	9.85	9.82	9.13	8.08
	Signal	50.23	33.17	32.26	32.03	28.57	25.93
	Total	113.55	63.31	62.1	61.85	58.69	51.01
	Red. (%)		**44.24**	**45.31**	**45.53**	**48.31**	**55.08**
M&D	Clock	35	20	20	20	21	17
	Logic	28.1	9.78	9.47	9.34	8.76	7.85
	Signal	49.53	32.27	31.32	30.59	27.63	25.26
	Total	112.63	62.05	60.79	59.93	57.39	50.11
	Red. (%)		**44.91**	**46.03**	**46.79**	**49.05**	**55.51**

Table 2.22 Power Consumption Reduction (Q=42) by PSCR Technique

Frames	Caetgory	Power (mW)					
		Org.	**PECR Tech.**	**PSCR Tech. (1bT)**	**PSCR Tech. (2bT)**	**PSCR Tech. (3bT)**	**PSCR Tech. (4bT)**
Foreman	**Clock**	35	20	20	20	21	17
	Logic	28.5	10.61	10.31	10.21	9.68	8.55
	Signal	52.27	35.29	34.31	33.98	30.88	28.25
	Total	115.77	65.9	64.62	64.19	61.56	53.79
	Red. (%)		**43.08**	**44.18**	**44.55**	**46.83**	**53.54**
Akiyo	**Clock**	35	20	20	20	21	17
	Logic	28.07	9.54	9.24	9.12	8.57	7.46
	Signal	49.11	31.25	30.25	29.74	26.82	23.69
	Total	112.18	60.79	59.49	58.86	56.39	48.14
	Red. (%)		**45.81**	**46.97**	**47.53**	**49.73**	**57.09**
M&D	**Clock**	35	20	20	20	21	17
	Logic	27.77	9.3	8.96	8.83	8.35	7.4
	Signal	48.6	30.55	29.55	28.92	26.26	23.74
	Total	111.37	59.86	58.51	57.75	55.61	48.14
	Red. (%)		**46.25**	**47.46**	**48.15**	**50.07**	**56.77**

CHAPTER III
DATA REUSE, PECR AND PSCR TECHNIQUES FOR COMPUTATION AND POWER REDUCTION IN H.264 INTRA PREDICTION

H.264 4x4 intra prediction modes have identical equations and calculating these common equations for each mode is unnecessary. Therefore, we calculated the common prediction equations only once and used the results for the corresponding 4x4 intra modes, and we applied the PECR and PSCR techniques for each prediction equation separately [11, 12]. The proposed PECR technique reduces the amount of computations performed by H.264 intra prediction without any PSNR and bitrate loss. It compares the pixels used in a prediction equation. If the pixels used in this prediction equation are equal, the predicted pixel by this equation is equal to the pixels used in this equation. Therefore, this prediction equation simplifies to a constant value and prediction calculation for this equation becomes unnecessary. The proposed PSCR technique reduces the amount of computations performed by H.264 intra prediction even further with a small PSNR loss. It also compares the pixels used in a prediction equation. If the pixels used in this prediction equation are similar, the predicted pixel by this equation is assumed to be equal to one of the pixels used in this equation. Therefore, this prediction equation simplifies to a constant value and prediction calculation for this equation becomes unnecessary.

The simulation results obtained by H.264 reference software, JM 14.0 [17], for several video sequences showed that PECR technique reduces the amount of computations performed by H.264 4x4 intra prediction modes up to 78%, more than the technique proposed in [8], with a small comparison overhead. PSCR technique reduces the amount of computations performed by H.264 4x4 intra prediction modes up to 89%, more than the technique proposed in [9], with a small comparison overhead. For each 4x4 block, both techniques require 12 comparisons for 4x4 intra prediction modes. PSCR technique increases the PSNR slightly for some video frames and it decreases the PSNR slightly for some video frames, and its PSNR loss is less than the PSNR loss of the technique proposed in [9].

We also designed an efficient H.264 4x4 intra prediction hardware including the proposed PECR and PSCR techniques. The hardware architecture is implemented in Verilog HDL. The Verilog RTL code is verified to work at 50 MHz in a Virtex II FPGA. The proposed PECR and PSCR techniques reduced the power consumption of this hardware on this FPGA up to 13.7% and 17.2%, respectively.

Data reuse techniques, similar to the one used in this thesis, are proposed for reducing the computational complexity of H.264 intra prediction algorithm in [24, 27] . Several other techniques are proposed for reducing the computational complexity of H.264 intra prediction algorithm in [18, 19]. These techniques reduce the amount of computations performed by H.264 intra prediction algorithm by trying selected intra prediction modes rather than trying all intra prediction modes and they require significant amount of pre-computation. However, the techniques proposed in this thesis try all intra prediction modes and they are applicable to computation reduction techniques proposed in literature [18, 19, 24, 27]. Several hardware architectures for H.264 4x4 intra prediction algorithm are reported in literature [22, 23, 24, 25, 28]. However, they do not report their power consumption and they do not implement the techniques proposed in this thesis.

3.1 Proposed Computational Complexity and Power Reduction Techniques

H.264 4x4 intra prediction modes have identical equations and calculating these common equations for each mode is unnecessary. Therefore, in this thesis, we calculated

the common prediction equations for all 4x4 intra prediction modes only once and used the results for the corresponding prediction modes instead of calculating the same equations again.

As it can be seen from Fig. 2.5, Eq. (3.1) is common in Diagonal Down-Left and Diagonal Down-Right prediction modes, and Diagonal Down-Right mode prediction equations for pred[0, 2] and pred[1, 3] are identical. Vertical Right mode prediction equations for pred[1, 2] and pred[3, 3], Vertical Left mode prediction equation for pred[1, 0], and Horizontal Down mode prediction equation for pred[0, 3] are also identical to the following equation.

$$pred[0, 0] = A + 2B + C + 2 >> 2 \quad (3.1)$$

There are 96 (6x16) prediction equations in H.264 4x4 intra prediction modes Diagonal Down-Left (DDL), Diagonal Down-Right (DDR), Vertical Right (VR), Vertical Left (VL), Horizontal Down (HD), Horizontal Up (HUP). Twenty three of these equations are distinct. Vertical and Horizontal prediction modes require no computation. These 23 prediction equations, the pixels used in these equations, number of modes these equations are used, number of pixels predicted by these equations and number of addition and shift operations performed by these prediction equations are shown in Table 3.1.

H.264 4x4 intra prediction performs 884183 addition and 529181 shift operations for a CIF (352x288) frame. When these 23 prediction equations are calculated only once, 417997 addition and 230839 shift operations are performed which corresponds to 53% and 56% reduction in addition and shift operations respectively.

The proposed PECR technique in [8] compares the pixels used in all prediction equations of a 4x4 intra prediction mode. If the pixels used in all equations of a prediction mode are equal, the predicted pixels by this mode are equal to these pixels. Therefore, the prediction equations for this mode simplify to a constant value and prediction calculations for this mode become unnecessary.

The proposed PSCR technique in [9] compares the pixels used in all prediction equations of a 4x4 intra prediction mode. If the pixels used in all equations of a prediction mode are similar, the predicted pixels by this mode are assumed to be equal to one of these

pixels. Therefore, the prediction equations for this mode simplify to a constant value and prediction calculations for this mode become unnecessary.

The neighboring pixels used in the prediction equations of each 4x4 intra prediction mode are shown in Table 3.2. For example, as shown in Fig. 2.4, the neighboring pixels A–H are used in the prediction equations of DDL mode. If all of these neighboring pixels are equal, all prediction equations of DDL mode simplify to a constant value as shown in the following equation.

$$pred[y,x] = [4A+2] >>2 = A \tag{3.2}$$

Table 3.1 Prediction Equations of 4x4 Intra Prediction Modes

Pixels	Equations	Used Modes	Predicted Pixels	# Add.	# Shift
A,B,C	[(A + 2B + C) + 2] >> 2	5	7	3	2
B,C,D	[(B + 2C + D) + 2] >> 2	4	6	3	2
C,D,E	[(C + 2D + E) + 2] >> 2	2	5	3	2
D,E,F	[(D + 2E + F) + 2] >> 2	2	6	3	2
E,F,G	[(E + 2F + G) + 2] >> 2	2	4	3	2
F,G,H	[(F + 2G + H) + 2] >> 2	1	2	3	2
J,K,L	[(J + 2K+ L) + 2] >> 2	3	4	3	2
I,J,K	[(I + 2J + K) + 2] >> 2	4	6	3	2
I,J,M	[(M + 2I + J) + 2] >> 2	3	6	3	2
A,I,M	[(A + 2M + I) + 2] >> 2	3	8	3	2
A,B,M	[(B + 2A + M) + 2] >> 2	3	6	3	2
A,B	[(A + B) + 1] >> 1	2	3	2	1
A,M	[(M + A) + 1] >> 1	1	2	2	1
B,C	[(B + C) + 1] >> 1	2	4	2	1
C,D	[(C + D) + 1] >> 1	2	3	2	1
D,E	[(D + E) + 1] >> 1	1	2	2	1
E,F	[(E + F) + 1] >> 1	1	1	2	1
G,H	[(G + 2H + H) + 2] >> 2	1	1	3	2
I,J	[(I + J) + 1] >> 1	2	3	2	1
I,M	[(M + I) + 1] >> 1	1	2	2	1
J,K	[(J + K) + 1] >> 1	2	4	2	1
K,L	[(K + L) + 1] >> 1	2	3	2	1
K,L	[(K + 2L + L) + 2)] >>2	1	2	3	2
L	[L]	1	6	0	0

Table 3.2 4x4 Intra Modes and Corresponding Neighboring Pixels

4x4 Intra Modes	Neighboring Pixels
Vertical	A, B, C, D
Horizontal	I, J, K, L
DC	A, B, C, D, I, J, K, L
Diagonal Down Left	A, B, C, D, E, F, G, H
Diagonal Down Right	A, B, C, D, I, J, K, L, M
Vertical Right	A, B, C, D, I, J, K, M
Horizontal Down	A, B, C, I, J, K, L, M
Vertical Left	A, B, C, D, E, F, G
Horizontal Up	I, J, K, L

In this thesis, we applied the PECR technique for each prediction equation separately. The proposed technique compares the pixels used in a prediction equation. If the pixels used in this prediction equation are equal, the predicted pixel by this equation is equal to the pixels used in this equation. Therefore, this prediction equation simplifies to a constant value and prediction calculation for this equation becomes unnecessary. For example, the prediction Eq. (3.1) is used in DDL, DDR, VR, VL and HD modes. If pixels A, B, and C are equal, Eq. (3.1) simplifies to a constant value as shown in (3.2).

We also applied the PSCR technique for each prediction equation separately. The proposed PSCR technique determines the similarity of the pixels used in a prediction equation by truncating their least significant bits by the specified truncation amount (1–4 bits) and comparing the truncated pixels. If these truncated pixels are all equal, one of the original pixels is substituted in place of every pixel used in this prediction equation. Therefore, this prediction equation simplifies to a constant value and prediction calculation for this equation becomes unnecessary.

The number of 4x4 intra prediction equations with equal and similar pixels in a frame varies from frame to frame. We analyzed CIF sized Foreman, Akiyo and Mother & Daughter frames coded with Quantization Parameters (QP) 28, 35 and 42 using JM 14.0 to determine how many prediction equations have equal and similar pixels. For each 4x4 intra prediction equation, the percentages of 4x4 blocks that have equal pixels and the percentages of 4x4 blocks that have similar pixels for 4 bits truncation (4bT) in these frames are given in Table 3.3. The percentages of 4x4 blocks with equal pixels vary from 10% to 94%, and the percentages of 4x4 blocks with similar pixels vary from 50% to 97%.

The percentages increase with higher QP values. Half of the prediction equations have equal pixels in more than 50% of the 4x4 blocks in these frames.

Based on the results given in Tables 3.1 and 3.3, we calculated the computation reductions achieved by the proposed PECR and PSCR techniques for CIF size Foreman, Akiyo and Mother & Daughter frames. The amount of computations performed by 4x4 intra prediction, 4x4 intra prediction with the PECR technique proposed in [8], 4x4 intra prediction with data reuse, and 4x4 intra prediction with both data reuse and the PECR technique proposed in this chapter are shown in Table 3.4.

Table 3.3 Percentage of 4x4 Intra Prediction Blocks with Equal and Similar Prediction Equation Pixels

	PECR									**PSCR (4bT)**								
	Foreman			**Akiyo**			**M&D**			**Foreman**			**Akiyo**			**M&D**		
4x4 Intra Equations	**QP 28**	**QP 35**	**QP 42**	**QP 28**	**QP 35**	**QP 42**	**QP 28**	**QP 35**	**QP 42**	**QP 28**	**QP 35**	**QP 42**	**QP 28**	**QP 35**	**QP 42**	**QP 28**	**QP 35**	**QP 42**
A,B,C	51.7	72.3	85.9	66.5	78.8	87.8	60.3	78.2	90.8	73.7	81.5	90.9	79.4	84.8	92.3	78.0	85.8	94.4
B,C,D	52.1	72.2	86.1	66.6	79.4	88.2	62.0	79.8	91.5	74.0	81.0	90.4	78.8	84.4	92.3	77.4	85.9	94.3
C,D,E	31.0	43.5	50.0	46.4	53.9	58.5	42.6	53.9	58.5	71.4	72.3	73.6	77.4	77.5	78.2	76.0	79.2	79.6
D,E,F	46.2	53.0	53.5	57.3	60.6	61.7	54.3	60.8	60.7	76.2	76.5	76.0	82.2	81.4	80.2	80.4	81.1	81.3
E,F,G	67.5	81.4	90.4	76.8	84.7	91.1	72.2	84.5	92.2	82.8	88.0	94.4	85.7	89.5	94.7	85.0	89.6	95.6
F,G,H	67.8	81.4	90.5	76.7	84.9	91.2	73.4	85.2	92.6	83.1	87.6	94.1	85.2	89.1	94.6	84.3	89.9	95.4
J,K,L	49.0	67.4	81.0	69.0	82.2	91.3	63.9	81.0	91.8	71.3	78.5	87.3	83.7	88.3	94.8	80.7	87.4	94.9
I,J,K	48.1	66.4	80.4	67.8	81.6	90.7	62.9	79.3	91.0	70.8	78.7	86.4	83.6	88.1	94.3	79.7	87.5	94.4
I,J,M	20.9	27.9	27.9	35.3	39.5	41.5	37.8	43.0	40.1	63.6	64.3	63.6	74.5	74.2	73.3	73.1	73.2	74.8
A,I,M	10.4	13.7	12.4	19.8	22.0	24.3	22.2	25.2	24.9	58.3	54.4	50.4	68.4	63.8	58.4	66.8	63.4	61.3
A,B,M	20.6	29.7	29.5	37.3	40.2	40.5	31.5	40.4	39.6	64.9	65.5	64.8	73.7	73.1	70.2	71.5	71.9	72.7
A,B	55.3	74.9	88.1	68.8	80.3	88.9	63.4	80.8	91.9	83.2	87.8	93.5	87.0	89.8	93.6	86.5	91.7	95.9
A,M	24.0	31.6	30.4	39.8	41.8	41.8	34.3	42.4	40.6	72.2	69.7	66.5	79.6	76.8	71.8	77.9	75.6	74.2
B,C	57.1	74.6	87.3	70.2	81.1	89.4	65.2	81.8	92.4	82.5	86.4	92.6	86.0	88.5	94.1	85.6	89.8	95.6
C,D	55.4	73.8	87.1	68.6	81.0	89.4	65.2	81.5	92.6	83.8	87.4	92.6	86.7	89.5	93.8	87.0	92.4	95.7
D,E	48.5	54.3	54.2	59.2	61.7	62.7	56.2	62.2	61.6	81.2	79.2	77.2	86.2	84.0	81.4	84.7	83.8	82.5
E,F	70.1	83.2	92.1	78.3	85.9	92.0	74.6	86.6	93.2	89.2	92.4	96.2	91.2	92.8	95.6	90.6	93.8	96.8
G,H	70.1	82.4	91.3	78.4	86.3	92.2	75.7	86.6	93.7	89.6	92.0	95.7	90.6	92.5	95.7	90.9	94.5	96.5
I,J	51.6	69.8	82.5	69.3	82.7	91.5	65.8	81.0	91.9	81.7	86.0	89.5	90.5	92.5	95.7	87.6	92.2	95.8
I,M	24.6	29.8	29.1	37.8	40.3	42.3	40.2	44.7	41.0	72.3	69.3	66.1	79.2	77.2	74.4	78.8	76.2	75.8
J,K	53.5	69.4	82.0	71.3	83.4	92.2	66.4	82.2	92.5	80.8	84.3	90.0	89.0	91.5	95.8	86.8	91.1	95.5
K,L	53.4	71.7	83.5	72.1	84.3	92.9	67.9	84.2	93.3	81.2	86.1	90.3	90.9	93.3	96.5	88.5	92.1	96.3

Table 3.4 Addition and Shift Operations Performed by 4x4 Intra Prediction for a CIF Frame with PECR Technique

	4x4 Intra Prediction		4x4 Intra Prediction with PECR Technique Proposed in [8]		4x4 Intra Prediction with Data Reuse		4x4 Intra Prediction with Data Reuse and PECR Technique	
QP	**Add.**	**Shift**	**Add.**	**Shift**	**Add.**	**Shift**	**Add.**	**Shift**
28	884183	529181	652779	400120	417997	230839	186593	101778
35	884183	529181	602186	372580	417997	230839	136000	74238
42	884183	529181	571858	356330	417997	230839	105672	57988

Table 3.5 Addition and Shift Operations Performed by 4x4 Intra Prediction for a CIF Frame with PSCR Technique

	4x4 Intra Prediction		4x4 Intra Prediction with PSCR Technique (4bT) Proposed in [9]		4x4 Intra Prediction with Data Reuse		4x4 Intra Prediction with Data Reuse and PSCR Technique (4bT)	
QP	**Add.**	**Shift**	**Add.**	**Shift**	**Add.**	**Shift**	**Add.**	**Shift**
28	884183	529181	419857	250334	417997	230839	87142	46413
35	884183	529181	327753	195549	417997	230839	63142	33788
42	884183	529181	329238	196688	417997	230839	63669	34103

The amount of computations performed by 4x4 intra prediction, 4x4 intra prediction with the PSCR technique with 4bT proposed in [9], 4x4 intra prediction with data reuse, and 4x4 intra prediction with both data reuse and the PSCR technique with 4bT proposed in this chapter are shown in Table 3.5. The average number of addition and shift operations performed for Foreman, Akiyo and Mother & Daughter frames coded with QP values 28, 35 and 42 are given in Tables 3.4 and 3.5.

The computation reductions achieved by the proposed PECR technique and PSCR technique with 4bT are shown in Table 3.6. The computation reduction ranges from 47% to 78% for PECR technique, and it ranges from 75% to 89% for PSCR technique with 4bT. PSCR technique achieves more computation reduction than PECR technique at the expense of a small PSNR loss. The computation reductions achieved by the PECR technique proposed in [8] and by the data reuse and PECR technique proposed in this chapter are shown in Table 3.7. The computation reductions achieved by the PSCR technique with 4bT proposed in [9] and by the data reuse and PSCR technique with 4bT proposed in this chapter are shown in Table 3.8. The data reuse and PECR technique together achieved 90%

computation reduction, and the data reuse and PSCR technique with 4bT together achieved 95% computation reduction.

The PECR and PSCR techniques proposed in this chapter, the PECR technique proposed in [8] and the PSCR technique proposed in [9] have an overhead of only 74882 comparisons for a CIF (352x288) frame. However, the PECR and PSCR techniques proposed in this chapter achieve more computation reduction than the PECR technique proposed in [8] and the PSCR technique proposed in [9], respectively.

Table 3.6 Computation Reduction by PECR and PSCR (4bT) Techniques for 4x4 Intra Prediction with Data Reuse

		PECR Technique				**PSCR Technique with 4bT**			
		Addition Reduction		**Shift Reduction**		**Addition Reduction**		**Shift Reduction**	
	QP	**#**	**%**	**#**	**%**	**#**	**%**	**#**	**%**
FM	**28**	196164	**46.9**	109839	**47.6**	314058	**75.1**	176142	**76.3**
	35	256231	**61.3**	142592	**61.8**	330313	**79.0**	184022	**79.7**
	42	292404	**70.0**	162008	**70.2**	348193	**83.3**	193115	**83.7**
Akiyo	**28**	256640	**61.4**	142943	**61.9**	344143	**82.3**	191945	**83.2**
	35	294605	**70.5**	163398	**70.8**	353911	**84.7**	196438	**85.1**
	42	320555	**76.7**	177422	**76.9**	366512	**87.7**	202769	**87.8**
M&D	**28**	241409	**57.8**	134401	**58.2**	336906	**80.6**	188069	**81.5**
	35	295154	**70.6**	163814	**71.0**	354610	**84.8**	196753	**85.2**
	42	324017	**77.5**	179124	**77.6**	371468	**88.9**	205386	**89.0**

Table 3.7 Computation Reduction for 4x4 Intra Prediction by PECR Technique

		Reduction by PECR Tech. Proposed in [8]				**Reduction by Data Reuse and PECR Technique**			
		Addition Reduction		**Shift Reduction**		**Addition Reduction**		**Shift Reduction**	
	QP	**#**	**%**	**#**	**%**	**#**	**%**	**#**	**%**
FM	**28**	246939	**27.9**	146816	**27.7**	662350	**74.9**	408181	**77.1**
	35	365863	**41.4**	216263	**40.9**	722417	**81.7**	440934	**83.3**
	42	459269	**51.9**	269710	**50.9**	758590	**85.8**	460350	**87.0**
Akiyo	**28**	386890	**43.8**	229707	**43.4**	722826	**81.8**	441285	**83.4**
	35	461728	**52.2**	273099	**51.6**	760791	**86.0**	461740	**87.3**
	42	521463	**58.9**	306887	**57.9**	786741	**89.0**	475764	**89.9**
M&D	**28**	359883	**40.7**	214067	**40.5**	707595	**80.0**	432743	**81.8**
	35	469840	**53.1**	278673	**52.6**	761340	**86.1**	462156	**87.3**
	42	539033	**60.9**	317345	**59.9**	790203	**89.4**	477466	**90.2**

Table 3.8 Computation Reduction for 4x4 Intra Prediction by PSCR Technique with 4bT

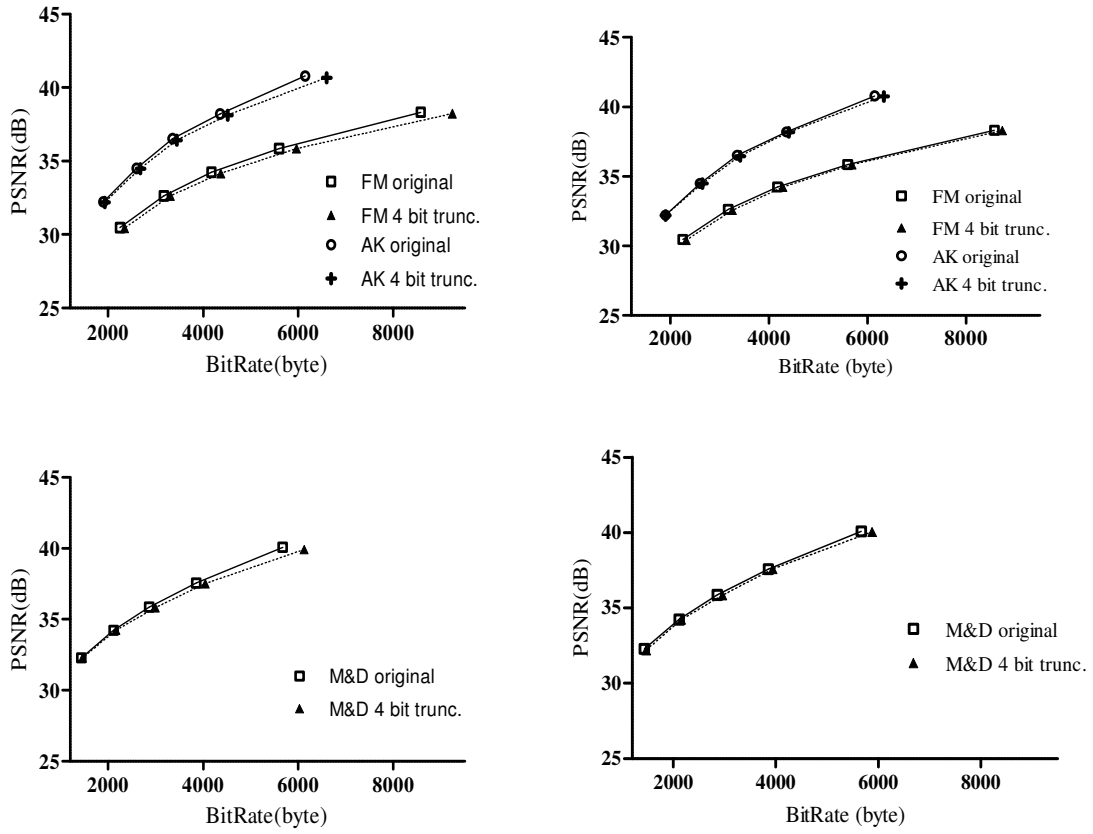
		Reduction by PSCR Technique with 4bT Proposed in [9]				Reduction by Data Reuse and PSCR Technique with 4bT			
		Addition Reduction		Shift Reduction		Addition Reduction		Shift Reduction	
	QP	#	%	#	%	#	%	#	%
FM	28	410869	**46.6**	246741	**46.7**	780244	**88.24**	474484	**89.7**
	35	470764	**53.3**	282662	**53.5**	796499	**90.08**	482364	**91.2**
	42	511344	**57.9**	307139	**58.2**	814379	**92.11**	491457	**92.9**
Akiyo	28	519800	**58.9**	311536	**59.0**	810329	**91.65**	490287	**92.7**
	35	562924	**63.8**	337540	**63.9**	820097	**92.75**	494780	**93.5**
	42	586565	**66.5**	351819	**66.6**	832698	**94.18**	501111	**94.7**
M&D	28	503723	**57.1**	301789	**57.2**	803092	**90.83**	486411	**91.9**
	35	557736	**63.2**	334094	**63.3**	820796	**92.83**	495095	**93.6**
	42	603375	**68.4**	361596	**68.5**	837654	**94.74**	503728	**95.2**

This is because the PECR technique proposed in [8] and the PSCR technique proposed in [9] achieve computation reduction for a 4x4 intra prediction mode only if the pixels used in all of its prediction equations are equal and similar, respectively. The probability of the pixels used in all prediction equations of a 4x4 intra prediction mode being equal/similar is less than the probability of the pixels used in a prediction equation being equal/similar. For example, the PECR technique proposed in [8] achieves computation reduction for DDL mode only if all the pixels used in DDL mode (A–H) are equal. But, the PECR technique proposed in this chapter can achieve computation reduction for DDL mode even if all of these pixels are not equal. For example, if the pixels A–C are equal, it achieves a computation reduction for a prediction equation of DDL mode.

Since there is only one prediction equation in DC prediction mode, the computation reduction achieved by the PECR and PSCR techniques proposed in this chapter for DC mode is same as the computation reduction achieved by the PECR and PSCR techniques proposed in [8, 9], respectively.

Since the PSCR technique achieves more computation reduction than PECR technique at the expense of a PSNR loss, we quantified the impact of the proposed PSCR technique on the rate-distortion performance of the 4x4 intra prediction algorithm by using H.264 JM reference software encoder version 14.0. The rate distortion curves and average PSNR comparison of the original 4x4 intra prediction algorithm, 4x4 intra prediction algorithm with the PSCR technique proposed in [9] and 4x4 intra prediction algorithm with

the PSCR technique proposed in this chapter for several CIF size video frames are shown in Fig. 3.1 and Table 3.9, respectively. The average PSNR values shown in Table 3.9 are calculated using the technique described in [26].



a) **b)**

Figure 3.1 Rate Distortion Curves of the Original 4x4 Intra Prediction Algorithm and
a) 4x4 Intra Prediction Algorithm with PSCR Technique proposed in [9]
b) 4x4 Intra Prediction Algorithm with Proposed PSCR Technique

Table 3.9 Average PSNR Comparison of the PSCR Techniques

Frame		Proposed PSCR Technique									PSCR Tech. Proposed in [9]	
		Orig. (dB)	1 bit Trunc. (dB)	ΔPSNR (dB)	2 bits Trunc. (dB)	ΔPSNR (dB)	3 bits Trunc. (dB)	ΔPSNR (dB)	4 bits Trunc. (dB)	ΔPSNR (dB)	4 bits Trunc. (dB)	ΔPSNR (dB)
Foreman	**Y**	35.28	35.28	**0.00**	35.27	**0.00**	35.27	**-0.01**	35.19	**-0.09**	35.01	**-0.27**
	Cb	40.40	40.38	**-0.02**	40.43	**0.03**	40.40	**0.00**	40.45	**0.05**	40.34	**-0.06**
	Cr	42.49	42.62	**0.13**	42.66	**0.17**	42.72	**0.23**	42.68	**0.19**	42.53	**0.04**
Akiyo	**Y**	37.25	37.30	**0.06**	37.24	**-0.01**	37.21	**-0.04**	37.12	**-0.13**	36.96	**-0.29**
	Cb	40.48	40.46	**-0.02**	40.50	**0.03**	40.49	**0.01**	40.37	**-0.11**	40.28	**-0.20**
	Cr	42.60	42.66	**0.06**	42.62	**0.02**	42.54	**-0.06**	42.36	**-0.24**	42.29	**-0.32**
Mother& Daughter	**Y**	36.82	36.87	**0.05**	36.87	**0.05**	36.80	**-0.02**	36.72	**-0.10**	36.54	**-0.28**
	Cb	42.20	42.24	**0.04**	42.11	**-0.09**	42.12	**-0.08**	42.07	**-0.14**	41.99	**-0.21**
	Cr	43.29	43.22	**-0.07**	43.18	**-0.11**	43.24	**-0.05**	43.14	**-0.15**	43.07	**-0.22**

The results show that the proposed PSCR technique increases the PSNR slightly for some video frames and it decreases the PSNR slightly for some video frames. The results also show that the proposed PSCR technique with 4bT has less PSNR loss than the PSCR technique with 4bT proposed in [9]. This is because the PSCR technique proposed in [9] substitutes one of the original pixels in place of every pixel used in all prediction equations of a 4x4 intra prediction mode if the pixels used in all of its prediction equations are similar. However, PSCR technique proposed in this chapter substitutes one of the original pixels in place of every pixel used in a prediction equation if the pixels used in that prediction equation are similar.

3.2 Proposed Intra Prediction Hardware Architecture

The top-level block diagram of the proposed hardware architecture for implementing H.264 4x4 intra prediction algorithm is shown in Fig. 3.2.

Three local neighboring buffers, top neighboring buffer, left neighboring buffer and reconstructed pixel neighboring buffer are used to store the neighboring pixels in the previously coded and reconstructed neighboring 4x4 blocks in the current MB. After a 4x4 block in the current MB is coded and reconstructed, the neighboring pixels in this block are stored in the corresponding local buffers.

Three parallel datapaths are used to calculate the predicted pixels. The first datapath calculates the pixels predicted by vertical mode, the second datapath calculates the pixels predicted by horizontal mode and the third datapath calculates the pixels predicted by DDL, DDR, VR, VL, HD, HUP and DC modes. As shown in Fig. 3.3, the third datapath calculates two predicted pixels in parallel. The predicted pixels are stored in the register files for the corresponding prediction modes.

Thirteen registers are used to store the neighboring pixels (A–M) for the current 4x4 block. When a new 4x4 block comes, neighboring pixel registers are loaded with the current neighboring pixels (A–M) in four cycles. Twelve 8-bit comparators are used to check the equality or similarity of the neighboring pixels. Based on the comparison results, disable signals are generated and sent to the datapaths implementing the prediction equations with equal or similar pixels.

Nine 4x32 register files are used to store the predicted pixels for 9 4x4 intra prediction modes. Based on the comparison results, disable signals are also generated and sent to these register files. When a disable signal is generated for a predicted pixel register file, this register file is loaded with one of the neighboring pixels.

The proposed hardware architecture is implemented in Verilog HDL. The implementation is verified with RTL simulations. RTL simulation results matched the results of a software model of the H.264 4x4 intra prediction algorithm.

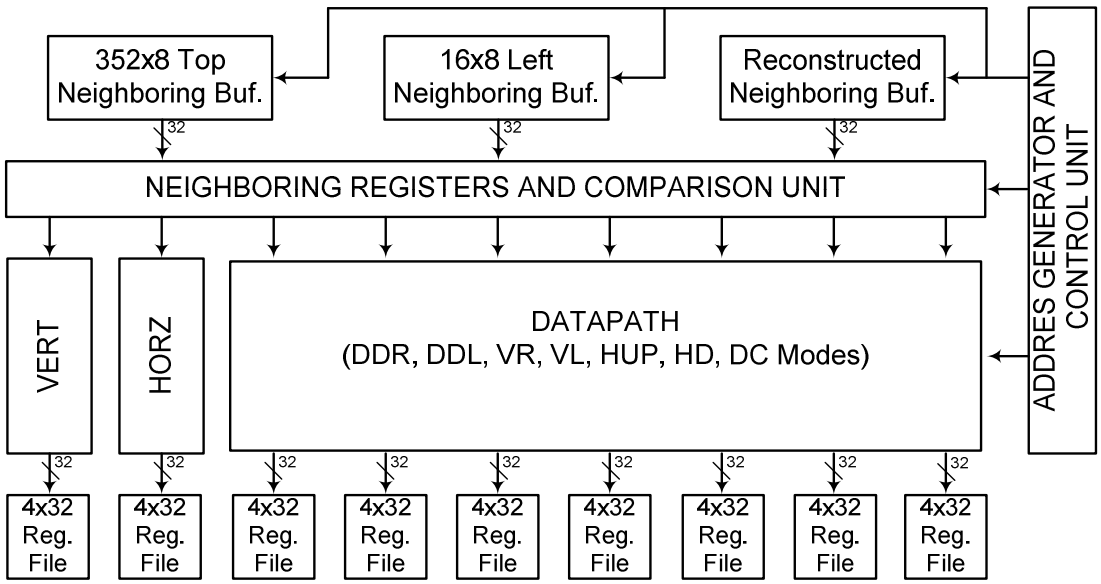


Figure 3.2 Top-Level Block Diagram of 4x4 Intra Prediction Hardware Architecture

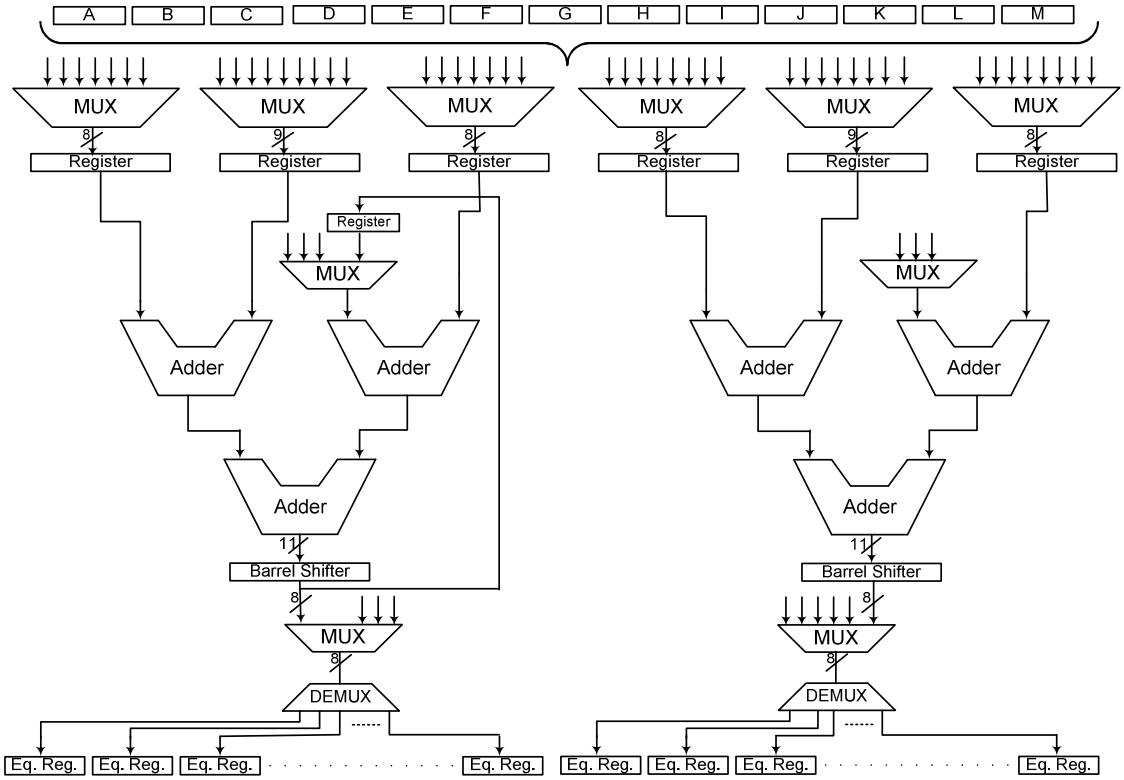


Figure 3.3 Datapath for The Prediction Equations Used in DDL, DDR, VR, VL, HD, HUP and DC Modes

Table 3.10 Comparison of 4x4 Intra Prediction Hardware

	Hardware Proposed in [8, 9]	**Proposed Hardware**
On-Chip Memory (bits)	6144	6144
Area	2448 LUTs 359 DFFs	1070 LUTs 497 DFFs
Maximum Frequency (MHz)	89.97	94.47
Technology	FPGA	FPGA
Average Clock Cycles for a 4x4 block	18.10	22.17

The comparison of the proposed 4x4 intra prediction hardware and 4x4 intra prediction hardware presented in [8, 9] is shown in Table 3.10 . Both hardware

architectures are implemented in Verilog HDL, and the Verilog RTL codes are synthesized to a 2V8000ff1157 Xilinx Virtex II FPGA with speed grade 5 using Mentor Graphics Precision RTL 2005b. The resulting netlists are placed and routed to the same FPGA at 50 MHz using Xilinx ISE 8.2i. Both 4x4 intra prediction hardware use 2 BlockRAMs. The proposed hardware uses 1070 LUTs and 497 DFFs. The hardware presented in [8, 9] uses 2448 LUTs and 359 DFFs. The proposed intra prediction hardware has 57% less LUTs and 38% more DFFs than the intra prediction hardware presented in [8, 9]. Because the proposed hardware uses smaller number of parallel datapaths, but it uses additional registers to store the results of common prediction equations.

The hardware architecture presented in [8, 9] has nine parallel datapaths and 4x4 intra prediction for a 4x4 block takes 18.10 clock cycles on the average. Since the hardware architecture proposed in this thesis has three parallel datapaths, 4x4 intra prediction for a 4x4 block takes 22.17 clock cycles on the average.

3.3 Power Consumption Analysis

The power consumption of the proposed 4x4 intra prediction hardware on a Xilinx Virtex II FPGA is estimated using Xilinx XPower tool. In order to estimate its power consumption, timing simulation of the placed and routed netlist of 4x4 intra prediction hardware is done using Mentor Graphics ModelSim SE. Foreman, Akiyo and Mother & Daughter frames are used as inputs for timing simulations and the signal activities are stored in VCD files. These VCD files are used for estimating the power consumption of the proposed 4x4 intra prediction hardware using Xilinx XPower tool.

The power consumptions (mW) of the proposed 4x4 intra prediction hardware implementation and 4x4 intra prediction hardware implementation presented in [8, 9] on the same FPGA at 25 MHz are shown in Tables 3.11–3.13 for different pixel truncation amounts, QP values and video frames. Since these intra prediction hardware implementations will be used as part of an H.264 video encoder, only internal power consumption is considered, and input and output power consumptions are ignored.

The internal power consumption is divided into three main categories; signal power, logic power and clock power. Signal power is the power dissipated in routing tracks between logic blocks. Logic power is the amount of power dissipated in the parts where

computations take place. Clock power is due to clock tree used in the FPGA. As shown in Table 3.11, power consumption of the original 4x4 intra prediction hardware implementation proposed in [8, 9] is 119.5 mW for a CIF size Foreman frame for QP 28. PECR technique proposed in [8] reduced the power consumption of this hardware to 74 mW, and PSCR technique with 4bT proposed in [9] reduced the power consumption of this hardware to 61 mW.

The power consumption of the original 4x4 intra prediction hardware implementation proposed in this thesis is 55.2 mW for the same CIF size Foreman frame for QP 28. PECR technique proposed in this thesis reduced the power consumption of this hardware to 49.8 mW, and PSCR technique with 4bT proposed in this thesis reduced the power consumption of this hardware to 46.5 mW.

As shown in Tables 3.11–3.13, even though the power consumption of the 4x4 intra prediction hardware proposed in this thesis is significantly less than the power consumption of the 4x4 intra prediction hardware proposed in [8, 9], the proposed PECR technique reduced its power consumption up to 13.7%, while the proposed PSCR technique reduced its power up to 17.2%.

Table 3.11 Power Consumption Reduction (QP = 28)

Frame	Category	Intra Prediction Hardware proposed in [8, 9]						Proposed Intra Prediction Hardware					
		Org.	PECR	PSCR (1bT)	PSCR (2bT)	PSCR (3bT)	PSCR (4bT)	Org.	PECR	PSCR (1bT)	PSCR (2bT)	PSCR (3bT)	PSCR (4bT)
Foreman	**Clock**	35	20	20	20	21	17	24	21	21.83	21.94	20.79	21
	Logic	29.32	12.44	12.28	12.22	11.55	10.2	7.29	6.15	6.35	6.18	5.67	5.48
	Signal	55.15	41.52	40.95	40.73	37.02	33.74	23.90	22.66	23.00	23.14	20.80	20.02
	Total	119.47	73.96	73.23	72.95	69.57	60.93	55.19	49.81	51.18	51.26	47.26	46.50
	Red. (%)		**38.09**	**38.70**	**38.94**	**41.77**	**49.00**		**9.76**	**7.27**	**7.12**	**14.37**	**15.75**
Akiyo	**Clock**	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.55	10.81	10.45	10.27	9.55	8.46	7.06	5.62	5.96	5.74	5.37	5.22
	Signal	50.98	35.45	34.28	33.63	29.94	27.07	22.87	20.85	21.22	21.19	19.32	18.56
	Total	114.53	66.25	64.73	63.9	60.49	52.53	53.93	47.47	49.18	48.93	45.69	44.78
	Red. (%)		**42.15**	**43.48**	**44.21**	**47.18**	**54.13**		**11.99**	**8.81**	**9.28**	**15.29**	**16.97**
Mother Daughter	**Clock**	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.37	10.76	10.45	10.28	9.68	8.7	7.06	5.71	6.01	5.84	5.5	5.27
	Signal	50.58	35.61	34.4	33.63	30.51	28	22.85	20.85	21.22	21.2	19.54	18.66
	Total	113.95	66.37	64.84	63.91	61.19	53.7	53.91	47.55	49.23	49.04	46.04	44.93
	Red. (%)		**41.76**	**43.10**	**43.91**	**46.30**	**52.87**		**11.79**	**8.68**	**9.03**	**14.59**	**16.65**

Table 3.12 Power Consumption Reduction (QP = 35)

Frame	Category	Intra Prediction Hardware proposed in [8, 9]						Proposed Intra Prediction Hardware					
		Org.	PECR	PSCR (1bT)	PSCR (2bT)	PSCR (3bT)	PSCR (4bT)	Org.	PECR	PSCR (1bT)	PSCR (2bT)	PSCR (3bT)	PSCR (4bT)
Foreman	Clock	35	20	20	20	21	17	24	21	22	22	21	20.95
	Logic	28.8	11.3	11.12	11.07	10.44	9.28	7.16	5.75	6.03	5.85	5.54	5.33
	Signal	53.58	37.76	37.13	36.9	33.48	30.61	23.44	21.68	22.04	22.13	20.29	19.52
	Total	117.37	69.06	68.25	67.97	64.92	56.88	54.60	48.43	50.07	49.98	46.83	45.80
	Red.		**41.16**	**41.85**	**42.09**	**44.69**	**51.54**		**11.30**	**8.30**	**8.48**	**14.23**	**16.13**
Akiyo	Clock	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.32	10.13	9.85	9.82	9.13	8.08	6.98	5.49	5.79	5.57	5.3	5.18
	Signal	50.23	33.17	32.26	32.03	28.57	25.93	22.68	20.31	20.61	20.69	19.05	18.39
	Total	113.55	63.31	62.1	61.85	58.69	51.01	53.65	46.80	48.4	48.26	45.35	44.57
	Red.		**44.24**	**45.31**	**45.53**	**48.31**	**55.08**		**12.77**	**9.79**	**10.05**	**15.48**	**16.93**
Mother Daughter	Clock	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.1	9.78	9.47	9.34	8.76	7.85	6.89	5.42	5.71	5.53	5.31	5.21
	Signal	49.53	32.27	31.32	30.59	27.63	25.26	22.51	20.07	20.45	20.45	18.98	18.28
	Total	112.63	62.05	60.79	59.93	57.39	50.11	53.40	46.49	48.16	47.98	45.29	44.49
	Red.		**44.91**	**46.03**	**46.79**	**49.05**	**55.51**		**12.94**	**9.82**	**10.15**	**15.19**	**16.69**

Table 3.13 Power Consumption Reduction (QP = 42)

Frame	Category	Intra Prediction Hardware proposed in [8, 9]						Proposed Intra Prediction Hardware					
		Org.	PECR	PSCR (1bT)	PSCR (2bT)	PSCR (3bT)	PSCR (4bT)	Org.	PECR	PSCR (1bT)	PSCR (2bT)	PSCR (3bT)	PSCR (4bT)
Foreman	Clock	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.8	11.3	11.12	11.07	10.44	9.28	7.04	5.59	5.79	5.61	5.38	5.29
	Signal	53.58	37.76	37.13	36.9	33.48	30.61	23.11	21.05	21.34	21.32	19.67	19.13
	Total	117.37	69.06	68.25	67.97	64.92	56.88	54.15	47.64	49.130	48.93	46.05	45.42
	Red.		**41.16**	**41.85**	**42.09**	**44.69**	**51.54**		**12.02**	**9.27**	**9.64**	**14.96**	**16.12**
Akiyo	Clock	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.32	10.13	9.85	9.82	9.13	8.08	6.90	5.34	5.61	5.41	5.18	5.11
	Signal	50.23	33.17	32.26	32.03	28.57	25.93	22.33	19.71	20	20.02	18.53	17.99
	Total	113.55	63.31	62.1	61.85	58.69	51.01	53.23	46.05	47.61	47.43	44.71	44.10
	Red.		**44.24**	**45.31**	**45.53**	**48.31**	**55.08**		**13.49**	**10.56**	**10.90**	**16.01**	**17.15**
Mother Daughter	Clock	35	20	20	20	21	17	24	21	22	22	21	21
	Logic	28.1	9.78	9.47	9.34	8.76	7.85	6.83	5.22	5.55	5.38	5.2	5.11
	Signal	49.53	32.27	31.32	30.59	27.63	25.26	22.18	19.53	19.8	19.76	18.44	17.81
	Total	112.63	62.05	60.79	59.93	57.39	50.11	53.02	45.75	47.35	47.14	44.64	43.92
	Red.		**44.91**	**46.03**	**46.79**	**49.05**	**55.51**		**13.71**	**10.69**	**11.09**	**15.80**	**17.16**

CHAPTER IV

ENERGY REDUCTION TECHNIQUES FOR H.264 DEBLOCKING FILTER

The DBF algorithm used in H.264 standard is more complex than the DBF algorithms used in previous video compression standards. The H.264 DBF algorithm can easily account for one-third of the computational complexity of an H.264 video decoder [36]. Therefore, in this thesis, we propose pixel equality and pixel similarity based techniques for reducing the amount of computations performed by H.264 DBF algorithm, and therefore reducing the energy consumption of H.264 DBF hardware. These techniques avoid unnecessary calculations in H.264 DBF algorithm by exploiting the equality and similarity of the pixels used in DBF equations.

PECR technique compares the pixels in the current edge before the filtering process. If some or all of these pixels are equal, H.264 DBF equations simplify significantly. PECR technique reduces the amount of computations performed by H.264 DBF algorithm with no PSNR loss. PSCR technique also compares the pixels in the current edge before the filtering process. If some or all of these pixels are similar, H.264 DBF equations are assumed to simplify significantly. PSCR technique reduces the amount of computations performed by H.264 DBF algorithm even further with a small PSNR loss.

The simulation results obtained by H.264 joint model (JM) reference software version 14.0 [17] for several video sequences showed that the amount of addition and shift operations performed by H.264 DBF algorithm are reduced up to 43% and 55% respectively by PECR technique, and up to 52% and 67% respectively by PSCR technique with a small comparison overhead. The simulation results also showed that the proposed PSCR technique does not affect the PSNR for some video frames, but it decreases the PSNR slightly for some video frames.

We also applied the proposed PECR and PSCR techniques separately to the H.264 DBF hardware architecture proposed in [37]. The DBF hardware architectures are implemented in Verilog HDL. The Verilog RTL codes are verified to work at 98 MHz in a Virtex 4 FPGA. The FPGA implementations can code 44 CIF (288x352) frames per second. The power consumptions of the DBF hardware implementations on the same FPGA are estimated using a gate level power estimation tool. The proposed PECR and PSCR techniques reduced the energy consumption of this H.264 DBF hardware on this FPGA up to 35% and 39% respectively.

Several hardware architectures for real-time implementation of H.264 adaptive DBF algorithm are presented in the literature [38, 39, 40, 41, 42]. In order to increase the throughput, different memory organizations are proposed in [38, 39, 40], and an efficient four-stage pipelined hardware is proposed in [41]. The proposed PECR and PSCR techniques can be used in these DBF hardware. In [42], an efficient five-stage pipelined DBF hardware with clock gating is proposed. The proposed PECR and PSCR techniques can be used in Pre-Computation stage of this DBF hardware to achieve additional power reduction. In [43], the order of the branch operations is changed based on the probability of occurrence of certain conditions in order to reduce the amount of computation. The proposed PECR and PSCR techniques can be used in this algorithm to further reduce the amount of computation.

4.1 H.264 Adaptive Deblocking Filter Algorithm

H.264 DBF algorithm removes visually disturbing blocking artifacts and discontinuities in a frame created by coarse quantization of MBs and motion compensated

prediction. Filtering is applied to each edge of all the 4x4 luma and chroma blocks in a MB as shown in Figure 4.1. The vertical 4×4 block edges in a MB are filtered before the horizontal 4×4 block edges in the order shown in Fig. 4.2 [1].

H.264 DBF algorithm for one row/column of a vertical/horizontal edge, which is called a filtering unit, is shown in Fig. 4.3 [36]. There are several conditions that determine whether a 4×4 block edge will be filtered or not. There are additional conditions that determine the strength of the filtering for the 4x4 block edges that will be filtered. As shown in Fig. 4.3, H.264 DBF algorithm can be divided into eight modes based on the outcomes of these conditions. Boundary strength (BS) parameter, α and β threshold values and the pixels in the edge determine the outcomes of these conditions, and up to 3 pixels on both sides of an edge can be changed depending on the outcomes of these conditions. α, β, c0 and c1 values are determined by quantization parameter (QP), BS, OffsetA and OffsetB parameters.

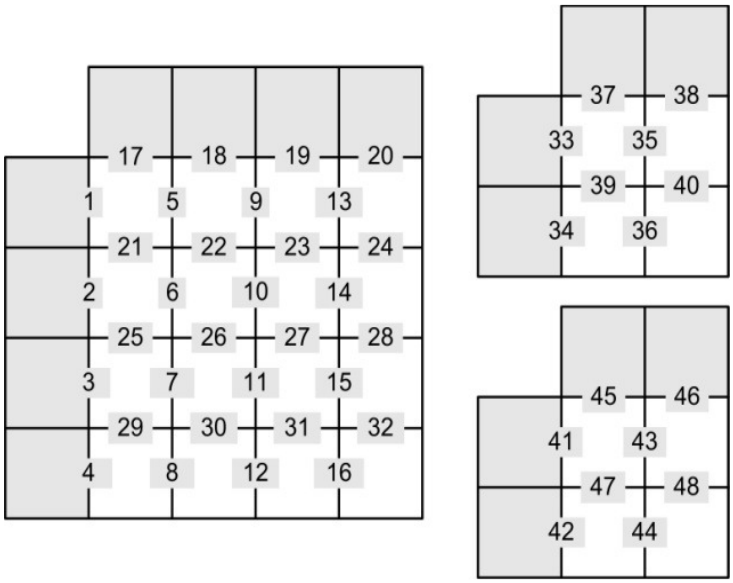


Figure 4.1 Edge Filtering Order Specified in H.264 Standard

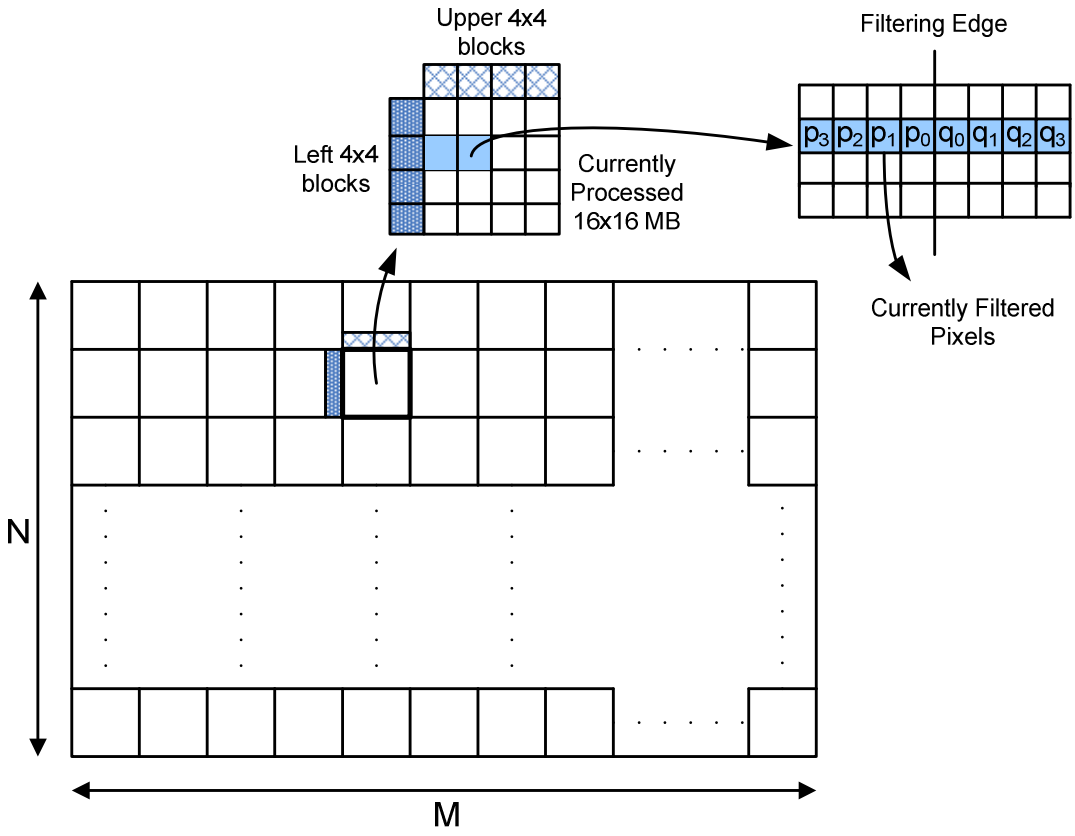


Figure 4.2 Illustration of H.264 DBF Algorithm

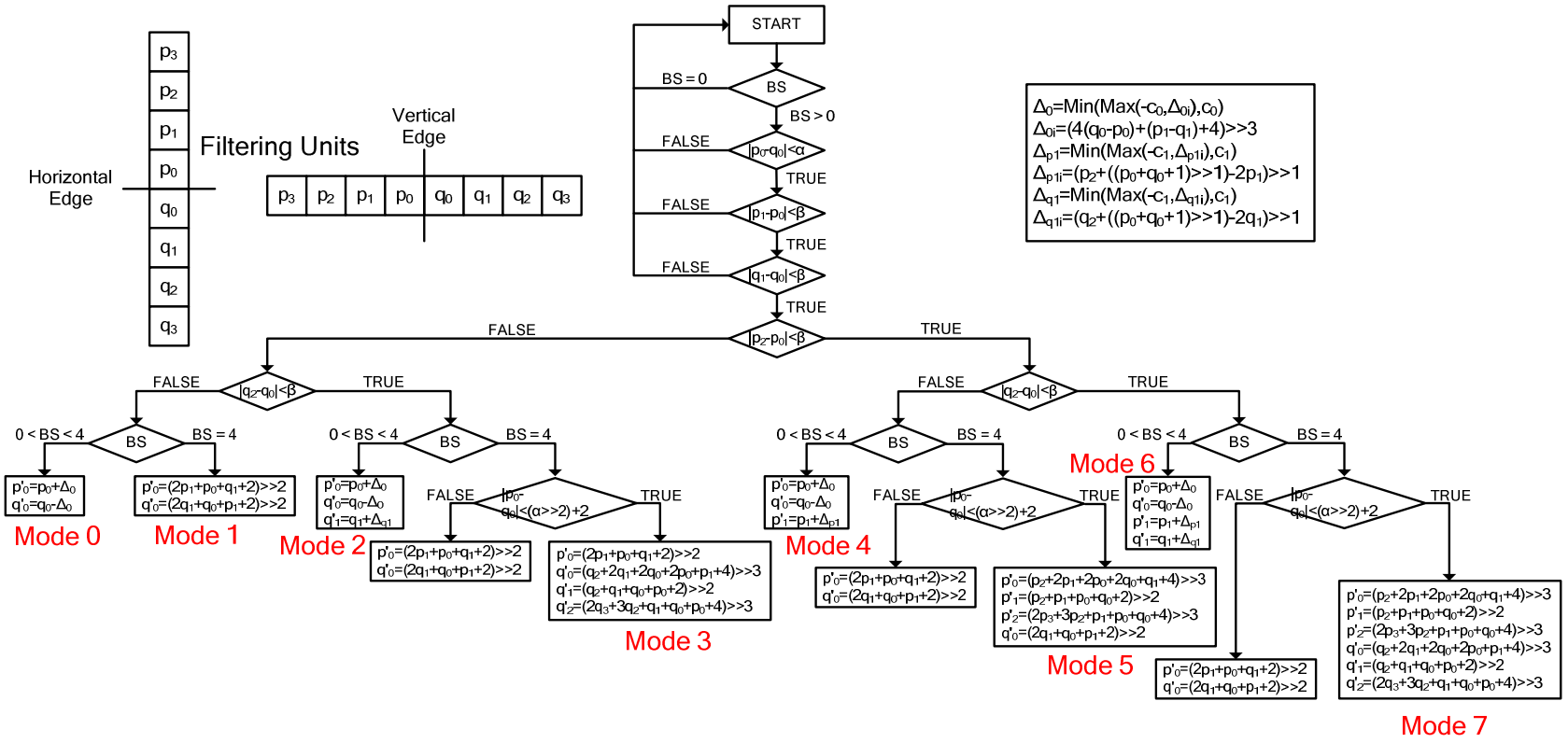


Figure 4.3 H.264 Deblocking Filter Algorithm

H.264 DBF algorithm is adaptive in three levels; slice level, edge level and sample level [1, 36]. Slice level adaptivity is used to adjust the filtering strength in a slice to the characteristics of the slice data. The filtering strength in a slice is adjusted by encoder using OffsetA and OffsetB parameters. The α and β threshold values that determine whether a 4x4 block edge will be filtered or not and how strong the filtering will be for an edge are a function of quantization parameter (QP) and these two offset parameters.

Edge level adaptivity is used to adjust the filtering strength for an edge to the characteristics of that edge. The filtering strength for an edge is adjusted using the BS parameter. Every edge is assigned a BS value depending on the coding modes and conditions of the 4x4 blocks. The conditions used for determining the BS value for an edge between two neighboring 4x4 blocks are summarized in Table 4.1 [36]. The strength of the filtering done for an edge is proportional to its BS value. No filtering is done for the edges with a BS value of 0, whereas strongest filtering is done for the edges with a BS value of 4.

4.2 Proposed Energy Reduction Techniques

The eight H.264 DBF modes and the pixels used in filtering equations of these modes are listed in Table 4.2. The filtering equations used in each mode are given in Fig. 4.3. As it can be seen from these filtering equations, H.264 DBF algorithm can be implemented using only addition and shift operations.

Table 4.1 Conditions that Determine BS

Coding Modes and Conditions	**BS**
One of the blocks is intra and the edge is a macroblock edge	4
One of the blocks is intra	3
One of the blocks has coded residuals	2
Difference of block motion ≥ 1 luma sample distance and Motion compensation from different reference frames	1
Else	0

Table 4.2 DBF Modes

BS	|p2-p0|<β	|q2-q0|<β	Pixels used in filtering equations	Mode
BS=4	False	False	p1, p0, q0, q1	1
	False	True	p1, p0, q0, q1, q2, q3	3
	True	False	p3, p2, p1, p0, q0, q1,	5
	True	True	p3, p2, p1, p0, q0, q1, q2, q3	7
4>BS>0	False	False	p1, p0, q0, q1	0
	False	True	p1, p0, q0, q1, q2	2
	True	False	p2, p1, p0, q0, q1	4
	True	True	p2, p1, p0, q0, q1, q2	6

PECR technique exploits equality of the pixels used in the filtering equations of DBF modes for reducing the amount of computation performed by H.264 DBF algorithm. If all the pixels used in the equations of a mode are equal, then these equations simplify significantly. PSCR technique exploits similarity of the pixels used in the filtering equations of DBF modes for reducing the amount of computation performed by H.264 DBF algorithm. PSCR technique determines the similarity of the pixels by truncating their least significant bits by the specified truncation amount (1 or 2 bits) and comparing the truncated pixels. If the truncated pixels used in the equations of a mode are all equal, then these equations are assumed to simplify significantly.

In order to reduce the overhead for determining the equality of the pixels used in the equations of a mode, we propose to use the subtraction operations performed in conditional branches of H.264 DBF algorithm. As shown in Fig. 4.3, these conditional branches include the five subtraction operations shown in (4.1) - (4.5). If two pixels are equal, their difference is equal to zero. Therefore, by checking the results of these five subtraction operations, we can determine the equality of the pixels used in the equations of a mode without performing additional comparison operations.

$$p0 - q0 \quad (4.1)$$
$$p1 - p0 \quad (4.2)$$
$$q1 - q0 \quad (4.3)$$
$$p2 - p0 \quad (4.4)$$
$$q2 - q0 \quad (4.5)$$

PECR technique compares 8-bit pixels. PSCR technique compares truncated pixels. If the results of equations (4.1) - (4.5) are zero, then the pixels p2, p1, p0, q0, q1, and q2 are equal. If the most significant 6 or 7 bits of the results of equations (4.1) - (4.5) are zero, then the pixels p2, p1, p0, q0, q1, and q2 are similar. If the results of equations (4.1), (4.2), and (4.4) are zero, then the pixels p2,p1,p0, and q0 are equal. If the most significant 6 or 7 bits of the results of equations (4.1), (4.2), and (4.4) are zero, then the pixels p2, p1, p0, and q0 are similar.

The equations of mode 6 are given in Table 4.3. p'0, q'0, p'1, q'1 are filtered values of p0, q0, p1, q1 pixels, respectively. If the pixels used in the equations of mode 6 are all equal, Δ0, Δp1 and Δq1 are zero, and all the filtered pixels are equal to one of the input pixels (p'0 = q'0 = p'1 = q'1 = p0). If the pixels used in the equations of mode 6 are all similar, Δ0, Δp1 and Δq1 are assumed to be zero, and therefore all the filtered pixels are assumed to be equal to their corresponding input pixels (p'0 = p0, q'0 = q0, p'1 = p1, q'1 = q1). Therefore, if the pixels used in the equations of mode 6 are all equal or similar, all the addition and shift operations performed by the equations of mode 6 are avoided.

Table 4.3 Equations for Mode 6 and their Simplified Versions when p2=p1=p0=q0=q1=q2

Equations for mode 6	**Simplified equations for mode 6 when p2=p1=p0=q0=q1=q2**
p'0 = p0+Δ0	p'0 = p0
q'0 = q0-Δ0	q'0 = q0
p'1 = p1+Δp1	p'1 = p1
q'1 = q1+Δq1	q'1 = q1
Δ0i = (4(q0-p0)+(p1-q1)+4)>>3	Δ0i = 0
Δ0 = Min(Max(-c0, Δ0i),c0)	Δ0 = 0
Δp1i = (p2+((p0+q0+1)>>1)-2p1)>>1	Δp1i = 0
Δp1 = Min(Max(-c1, Δp1i),c1)	Δp1 = 0
Δq1i = (q2+((p0+q0+1)>>1)-2q1)>>1	Δq1i = 0
Δq1 = Min(Max(-c1, Δq1i),c1)	Δq1 = 0

We calculated the amount of addition and shift operations performed by each mode, and the amount of addition and shift operations performed by each mode when the pixels used in the equations of this mode are all equal or similar. The amount of addition and shift operations performed by all DBF modes are shown in Tables 4.4 – 4.12 respectively. In the CB row, the subtraction operations performed before the filtering are shown.

Table 4.4 The Amount of Computation Required by DBF Mode 0 For Different Equal Pixel Combinations

Category	Original		$p_1=p_0=q_0=q_1$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
Δoi	4	2	0	0
p'0, q'0	2	0	0	0
Total	**11**	**2**	**5**	**0**

Table 4.5 The Amount of Computation Required by DBF Mode 1 For Different Equal Pixel Combinations

Category	Original		$p_1=p_0=q_0=q_1$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
p'0, q'0	6	4	0	0
Total	**11**	**4**	**5**	**0**

Table 4.6 The Amount of Computation Required by DBF Mode 2 For Different Equal Pixel Combinations

Category	Original		$p_1=p_0=q_0=q_1=q_2$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
Δoi	4	2	0	0
Δq1i	4	3	0	0
p'0, q'0, q'1	3	0	0	0
c0	1	0	0	0
Total	**17**	**5**	**5**	**0**

Table 4.7 The Amount of Computation Required by DBF Mode 3 For Different Equal Pixel Combinations

Category	Original		$p_1=p_0=q_0=q_1=q_2=q_3$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
p'0	3	2	0	0
q'0	2	2	0	0
q'1	2	1	0	0
q'2	3	2	0	0
(α>>2)+2	1	1	0	0
Total	**16**	**8**	**5**	**0**

Table 4.8 The Amount of Computation Required by DBF Mode 4 For Different Equal Pixel Combinations

Category	Original		$p_2= p_1=p_0=q_0=q_1$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
Δoi	4	2	0	0
Δp1i	4	3	0	0
p'0, q'0, p'1	3	0	0	0
c0	1	0	0	0
Total	**17**	**5**	**5**	**0**

Table 4.9 The Amount of Computation Required by DBF Mode 5 For Different Equal Pixel Combinations

Category	Original		$p_3=p_2=p_1=p_0=q_0=q_1$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
p'0	2	2	0	0
p'1	2	1	0	0
p'2	3	2	0	0
q'0	3	2	0	0
(α>>2)+2	1	1	0	0
Total	**16**	**8**	**5**	**0**

Table 4.10 The Amount of Computation Required by DBF Mode 6 For Different Equal Pixel Combinations

Category	Original		$p_2=p_1=p_0=q_0=q_1=q_2$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
Δoi	4	2	0	0
Δp1i	4	3	0	0
Δq1i	4	3	0	0
p'0, q'0, p'1, q'1	4	0	0	0
c0	1	0	0	0
Total	**22**	**8**	**5**	**0**

Table 4.11 The Amount of Computation Required by DBF Mode 7 For Different Equal Pixel Combinations

Category	Original		$p_3=p_2=p_1=p_0=q_0=q_1=q_2=q_3$	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
CB	5	0	5	0
p'0	2	2	0	0
p'1	4	1	0	0
p'2	3	2	0	0
q'0	2	2	0	0
q'1	2	1	0	0
q'2	3	2	0	0
(α>>2)+2	1	1	0	0
Total	**22**	**11**	**5**	**0**

Table 4.12 Amount of Operations Performed by All DBF Modes

Modes	Original		Pixel Equality / Similarity	
	# of Add.	**# of Shifts**	**# of Add.**	**# of Shifts**
Mode 0	11	2	5	0
Mode 1	11	4	5	0
Mode 2	17	5	5	0
Mode 3	16	8	5	0
Mode 4	17	5	5	0
Mode 5	16	8	5	0
Mode 6	22	8	5	0
Mode 7	22	11	5	0

The amount of computation reductions achieved by PECR and PSCR techniques depend on how many filtering units in a frame have all equal or similar pixels. Therefore, we determined how many filtering units in CIF (352x288) size Foreman, Akiyo, Mother&Daughter and Ice video frames (one frame from each video) at 28, 35 and 42 QP values have all equal or similar (with 1 bit truncation (1bT) and 2 bit truncation (2bT)) pixels using H.264 JM reference software version 14.0, and presented the results for luma and chroma components in Tables 4.13 and 4.14 respectively.

Table 4.13 Filtering Units with All Equal or Similar Pixels for Luma Components

		PECR			PSCR (1bT)			PSCR (2bT)		
Fram	QP	Total	Equal		Total	Equal		Total	Equal	
		#	#	%	#	#	%	#	#	%
Foreman	28	37902	4767	**12.6**	38458	6531	**17.0**	36539	10492	**28.7**
	35	42334	8365	**19.8**	41993	10609	**25.3**	42519	14135	**33.2**
	42	45584	11666	**25.6**	45215	13746	**30.4**	45511	17294	**38.0**
Akiyo	28	42716	12515	**29.3**	42961	15720	**36.6**	42011	20721	**49.3**
	35	45057	15967	**35.4**	44906	18631	**41.5**	45213	22071	**48.8**
	42	47243	18505	**39.2**	46858	20995	**44.8**	47139	24052	**51.0**
M&D	28	40952	12694	**31.0**	41421	15410	**37.2**	39936	19559	**49.0**
	35	45658	15424	**33.8**	45388	18224	**40.2**	45768	21518	**47.0**
	42	48231	18229	**37.8**	47714	20511	**43.0**	48134	24177	**50.2**
Ice	28	40842	17573	**43.0**	41081	21375	**52.0**	40272	25976	**64.5**
	35	42653	21862	**51.3**	42471	24905	**58.6**	42846	28465	**66.4**
	42	44926	24922	**55.5**	44507	27572	**62.0**	44905	30195	**67.2**

Table 4.14 Filtering Units with All Equal or Similar Pixels for Chroma (CbCr) Components

		PECR			PSCR (1bT)			PSCR (2bT)		
Frame		Total	Equal		Total	Equal		Total	Equal	
	QP	#	#	%	#	#	%	#	#	%
Foreman	28	24197	12497	**51.7**	24237	14502	**59.8**	24156	17550	**72.7**
	35	24542	16370	**66.7**	24551	17782	**72.4**	24572	19548	**79.6**
	42	24652	16613	**67.4**	24652	18063	**73.3**	24651	20192	**81.9**
Akiyo	28	23554	12624	**53.6**	23602	13975	**59.2**	23423	16602	**70.9**
	35	23764	13932	**58.6**	23845	15317	**64.2**	23931	17289	**72.3**
	42	23986	16126	**67.2**	23990	17000	**70.9**	23972	18174	**75.8**
M&D	28	24574	12517	**50.9**	24575	14262	**58.0**	24563	17377	**70.7**
	35	24704	15500	**62.7**	24704	17036	**69.0**	24704	18456	**74.7**
	42	24704	17077	**69.1**	24704	18262	**73.9**	24704	19570	**79.2**
Ice	28	24277	17018	**70.1**	24277	18233	**75.1**	24205	20100	**83.0**
	35	24404	17469	**71.6**	24421	19056	**78.0**	24450	20420	**83.5**
	42	24546	20910	**85.2**	24548	21273	**86.7**	24534	21850	**89.1**

Luma and chroma components of a CIF size frame have 50048 and 24704 filtering units, respectively. The column Total shows the number of filtering units that are filtered. The column Equal shows how many of these filtering units have all equal or similar pixels. The percentages of filtering units which have all equal or similar pixels vary from 12% to 67% for luma components, and 51% to 89% for chroma components. The percentages increase with higher QP values and truncation amounts.

We calculated the computation reduction achieved by the proposed PECR and PSCR techniques for H.264 DBF algorithm using H.264 JM reference software version 14.0 for Foreman, Akiyo, Mother&Daughter and Ice video frames (one frame from each video) at 28, 35 and 42 QP values. As shown in Tables 4.15 and 4.16, the amount of reductions achieved in addition and shift operations ranges from 10% to 52% and 14% to 67% respectively for luma components. The amount of reductions achieved in addition and shift operations ranges from 28% to 48% and 50% to 87% respectively for chroma components. Since H.264 DBF algorithm is highly adaptive, the number of addition and shift operations in a frame varies from frame to frame. In these tables, the column Total shows the total number of addition and shift operations in a frame, and the column Reduction shows the reductions achieved in addition and shift operations by the proposed PECR and PSCR techniques for a frame.

The proposed techniques, on the other hand, have to check if the results of at least 3 and at most 5 subtraction operations are equal to zero or not for one row/column of a vertical/horizontal edge based on BS, α and β parameters. However, this overhead is quite small considering that checking whether a number is zero or not can be efficiently implemented in hardware.

By using the subtraction operations shown in (4.1) - (4.5), we can only check the equality of three pixels (p2 - q2) on each side of an edge. However, for modes 3, 5, and 7, when BS = 4, DBF algorithm can access up to four pixels on each side of an edge, and therefore p3 or q3 or both can be used in filtering equations.

Table 4.15 Computation Reductions for Luma Components

Frame	QP	PECR						PSCR (1bT)						PSCR (2bT)					
		Addition			Shift			Addition			Shift			Addition			Shift		
		Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%
Foreman	28	803402	82105	10	295290	41353	14	818603	111494	14	302944	56306	19	778095	176418	23	283934	89187	31
	35	913494	143034	16	346610	71066	21	907300	180608	20	343823	90047	26	918952	239294	26	349431	120201	34
	42	991821	198848	20	383863	97580	25	986226	233691	24	381568	115135	30	991359	293135	30	383784	145092	38
Akiyo	28	908417	213855	24	343440	106339	31	919334	267059	29	349299	134398	38	895968	348976	39	337953	176342	52
	35	974458	272054	28	375299	133795	36	972272	316579	33	374437	156580	42	978300	373578	38	377201	186790	50
	42	1034674	315226	30	404414	154579	38	1029293	357081	35	402122	175359	44	1033452	408192	39	403844	201464	50
M & D	28	869923	216830	25	325398	107854	33	883125	261807	30	332110	131702	40	851255	329477	39	317080	166608	53
	35	980533	263040	27	377044	130305	35	976084	309950	32	375197	154037	41	985010	364486	37	379513	183153	48
	42	1052760	310296	29	412626	153361	37	1045649	348563	33	409543	172592	42	1051807	410040	39	412155	203419	49
Ice	28	895329	299755	33	341275	148739	44	901174	362657	40	344190	182279	53	884867	438044	50	336662	221623	66
	35	937683	372397	40	359848	184561	51	934922	422788	45	358693	211434	59	941709	481615	51	361673	242634	67
	42	986790	424246	43	382198	211425	55	981126	468430	48	379826	234123	62	986781	511883	52	382144	257103	67

Table 4.16 Computation Reductions for Chroma (CbCr) Components

Frame	QP	PECR						PSCR (1bT)						PSCR (2bT)					
		Addition			Shift			Addition			Shift			Addition			Shift		
		Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%	Total	Reduc.	%
Foreman	**28**	267688	74982	**28**	72062	35856	**50**	268008	87012	**32**	72180	41766	**58**	267360	105300	**39**	719	51482	**72**
	35	270448	98220	**36**	73064	46734	**64**	270520	106692	**39**	73080	51234	**70**	918952	239294	**26**	349431	120201	**34**
	42	271328	99678	**37**	73356	47982	**65**	271328	108378	**40**	73356	52166	**71**	991359	293135	**30**	383784	145092	**38**
Akiyo	**28**	262544	75744	**29**	70168	35924	**51**	262928	83850	**32**	70332	40098	**57**	261496	99612	**38**	69774	48466	**69**
	35	264224	83592	**32**	70708	39900	**56**	264872	91902	**35**	70986	44106	**62**	265560	103734	**39**	71278	50116	**70**
	42	266000	96756	**36**	71372	45584	**64**	266032	102000	**38**	71386	48376	**68**	265888	109044	**41**	71338	52034	**73**
M & D	**28**	270704	75102	**28**	72982	35212	**48**	270712	85572	**32**	72984	40558	**56**	270616	104262	**39**	72942	50410	**69**
	35	271744	93000	**34**	73472	43106	**59**	271744	102216	**38**	73472	47844	**65**	271744	110736	**41**	73472	52306	**71**
	42	271744	102462	**38**	73472	48514	**66**	271744	109572	**40**	73472	52028	**71**	271744	117420	**43**	73472	55884	**76**
Ice	**28**	268328	102108	**38**	72194	49632	**69**	268328	109398	**41**	72188	53268	**74**	267752	120600	**45**	71978	59242	**82**
	35	269344	104814	**39**	72552	50860	**70**	269480	114336	**42**	72610	55416	**76**	269712	122520	**45**	72712	59716	**82**
	42	270480	125460	**46**	72994	60810	**83**	270496	127638	**47**	72998	61952	**85**	270384	131100	**48**	72966	63780	**87**

Additional comparison operations can be performed to determine the equality of p3 and q3 with the other pixels in the filtering equations of modes 3, 5, and 7. However, the simulation results obtained by H.264 JM reference software version 14.0 for several video sequences showed that a small number of filtering units filtered with mode 3 and mode 5 have all equal or similar pixels. On the other hand, a large number of filtering units filtered with mode 7 have all equal or similar pixels. Therefore, we propose that p3 and q3 are compared only with the pixels in the filtering equations of mode 7.

Table 4.17 shows the number of comparisons of p3 and q3 with the other pixels in the filtering equations, the number of addition reductions achieved by the proposed techniques, and the percentage of the comparisons to the addition reductions. As shown in the table, the overhead of comparing p3 and q3 with the other pixels in the filtering equations is much smaller than the amount of addition reductions achieved by the proposed techniques, and it decreases with higher QP values and truncation amounts.

We also quantified the impact of the proposed PSCR technique on the rate-distortion performance of the H.264 DBF algorithm using H.264 JM reference software 14.0. The rate distortion curves and average PSNR comparison of the original DBF algorithm and the DBF algorithm with the proposed PSCR technique for luma components of several CIF size video frames (one frame from each video) and different pixel truncation amounts are shown in Fig. 4.4 and Table 4.18 respectively. The average PSNR values shown in Table 4.18 are calculated using the technique described in [26]. The proposed PSCR technique does not change the PSNR for some video frames, but it decreases the PSNR slightly for some video frames.

Table 4.17 Comparison Overhead

		PECR			PSCR (1bT)			PSCR (2bT)		
	QP	**# Comp.**	**Addition Reduction**	**%**	**# Comp.**	**Addition Reduction**	**%**	**# Comp.**	**Addition Reduction**	**%**
Foreman	**28**	13570	157087	**8.6**	14166	198506	**7.1**	12914	281718	**4.6**
	35	17330	241254	**7.2**	17224	287300	**6.0**	17600	478588	**3.7**
	42	19878	298526	**6.7**	19802	342069	**5.8**	19890	586270	**3.4**
Akiyo	**28**	16818	289599	**5.8**	17174	350909	**4.9**	16478	448588	**3.7**
	35	18836	355646	**5.3**	18876	408481	**4.6**	19082	477312	**4.0**
	42	21396	41	**5.2**	21386	459081	**4.7**	21382	517236	**4.1**
M & D	**28**	15078	291932	**5.2**	15584	347379	**4.5**	14480	433739	**3.3**
	35	18316	356040	**5.1**	18328	412166	**4.4**	18680	475222	**3.9**
	42	21632	412758	**5.2**	21620	458135	**4.7**	21648	527460	**4.1**
Ice	**28**	17344	401863	**4.3**	17570	472055	**3.7**	17116	558644	**3.1**
	35	18486	477211	**3.9**	18452	537124	**3.4**	18600	604135	**3.1**
	42	19910	549706	**3.6**	19908	596068	**3.3**	19876	642983	**3.1**

Table 4.18 Average PSNR Comparison of PSCR Technique

Frame		**Original (dB)**	**1 bit Trunc. (dB)**	**ΔPSNR (dB)**	**2 bits Trunc. (dB)**	**ΔPSNR (dB)**
Foreman	**Y**	35.28	35.28	**0.00**	35.26	**-0.02**
	Cb	40.40	40.39	**0.00**	40.36	**-0.04**
	Cr	42.49	42.48	**-0.01**	42.42	**-0.07**
Akiyo	**Y**	37.25	37.25	**0.00**	37.23	**-0.02**
	Cb	40.48	40.46	**-0.02**	40.43	**-0.05**
	Cr	42.61	42.60	**-0.01**	42.53	**-0.08**
M & D	**Y**	36.80	36.79	**-0.02**	36.77	**-0.04**
	Cb	42.19	42.19	**0.00**	42.15	**-0.05**
	Cr	43.28	43.28	**0.00**	43.22	**-0.06**
Ice	**Y**	36.74	36.73	**-0.01**	36.71	**-0.02**
	Cb	42.98	42.97	**-0.01**	42.93	**-0.05**
	Cr	43.65	43.64	**-0.01**	43.58	**-0.07**

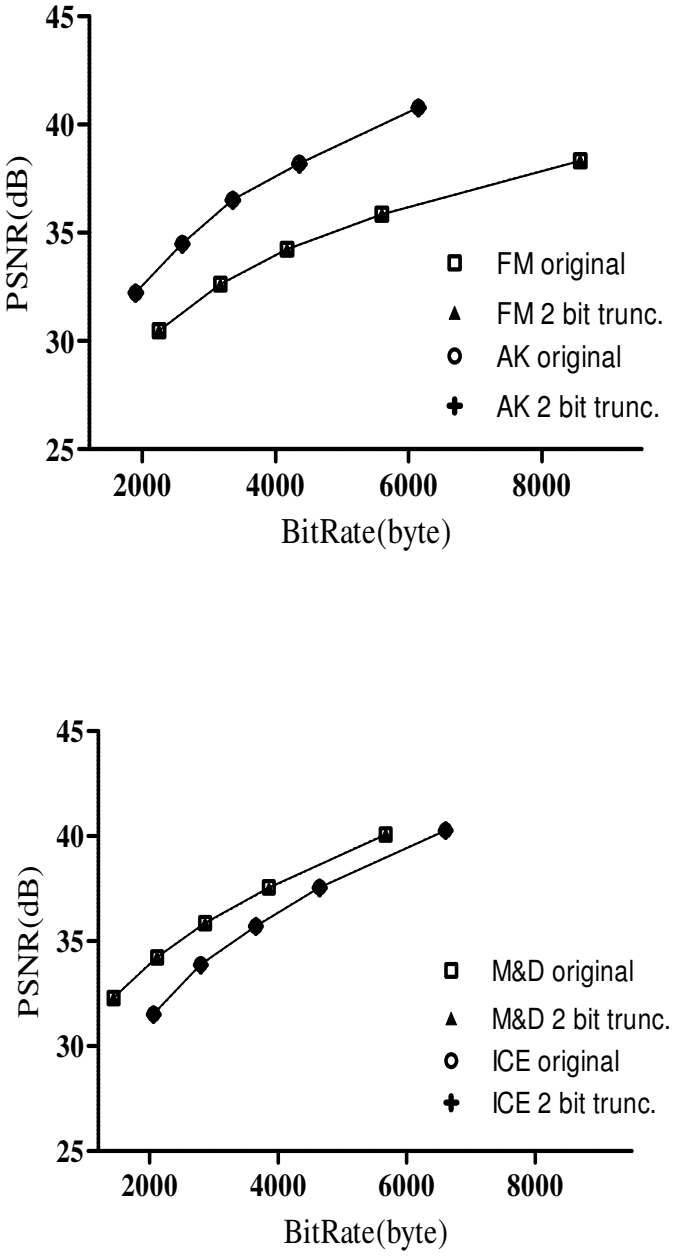


Figure 4.4 Rate Distortion Curves of the Original H.264 DBF Algorithm and H.264 DBF Algorithm with Proposed PSCR Technique

4.3 H.264 DBF Hardware and Its Energy Consumption

The block diagram of H.264 DBF hardware proposed in [37] is shown in Fig. 4.5. This DBF hardware consists of a datapath, a control unit, two 384×8 register files, and two dual-port on-chip SRAMs to store partially filtered pixels.

A 384×8 register file, IBUF, is used to store one 16x16 reconstructed MB (256 luminance pixels and 128 chrominance pixels) that will be filtered by DBF hardware. As shown in Fig. 4.6, there are sixteen 4×4 blocks in a MB and they are processed by IT/IQ in the order given in the H.264 standard [1]. The DBF hardware starts filtering after a new 16x16 reconstructed MB is ready.

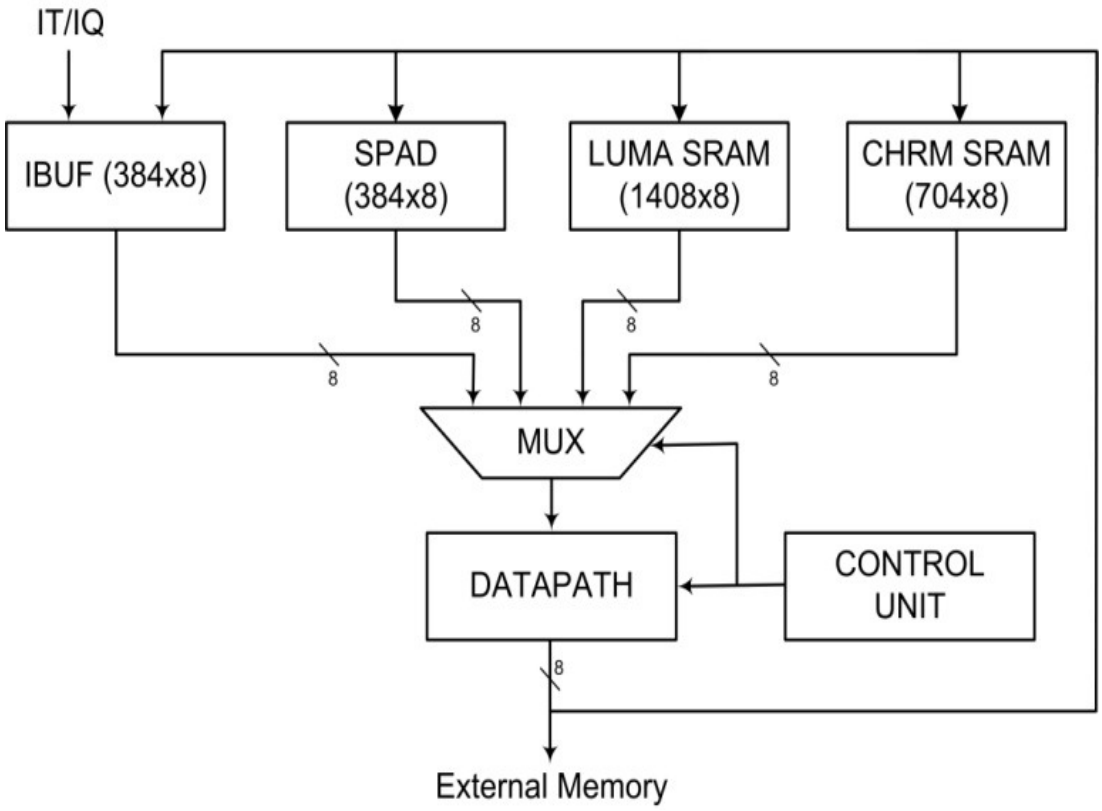


Figure 4.5 H.264 DBF Hardware Architecture

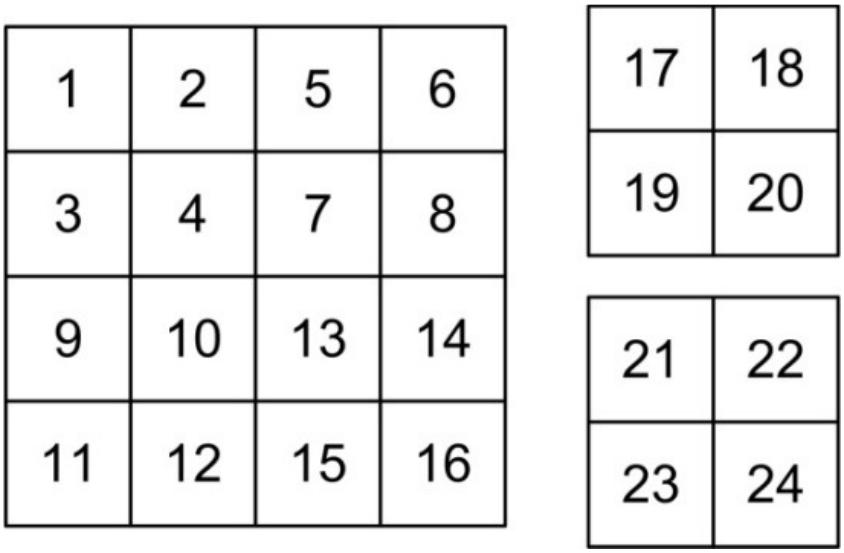


Figure 4.6 Processing Order of 4×4 Blocks

A 384×8 register file, SPAD, is used to store partially filtered pixels in a 16×16 MB until all the edges in this MB are fully filtered. In the M×N frame shown in Fig. 4.7, squares represent 16x16 MBs. In order to filter a MB, its upper and left neighboring 4×4 blocks should be available. Since DBF hardware gets its input MB from IT/IQ hardware and it does not access off-chip frame memory, the upper neighboring 4×4 blocks of all MBs in a row of the frame and the left neighboring 4×4 blocks of the current MB are stored in on-chip local memory. The left neighboring 4×4 blocks of the current MB are stored in SPAD. The upper neighboring 4×4 luminance and chrominance blocks of all MBs in a row of a CIF size frame are stored in the 4×352×8 = 1408×8 LUMA SRAM and 4x88x8+4x88x8 = 704×8 CHRM SRAM respectively.

As shown in Fig. 4.8, the datapath is implemented as a two stage pipeline to improve the clock frequency and throughput. The first pipeline stage includes one 12-bit adder/subtractor and two shifters to perform numerical calculations. The second pipeline stage includes one 12-bit comparator, several two's complementers and multiplexers to determine conditional branch results.

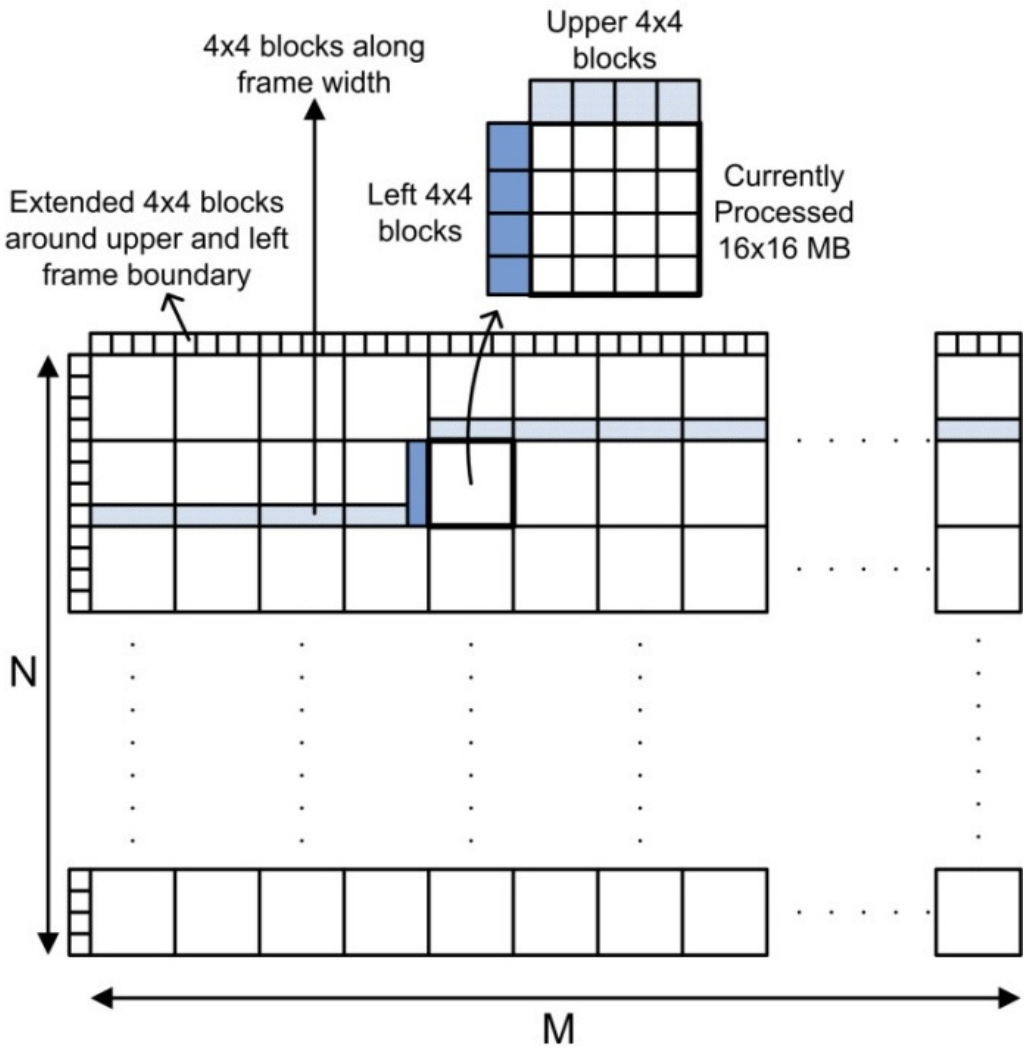


Figure 4.7 4x4 Blocks Stored in LUMA and CHRM SRAMs

The edges 1, 2, 3, 4, 33, 34, 41, 42 of a MB shown in Fig. 4.2 are not filtered if this MB is located in the left frame boundary, and the edges 17, 18, 19, 20, 37, 38, 45, 46 of a MB are not filtered if this MB is located in the top frame boundary. This is not the case for the MBs located inside the frame. In order to avoid this irregularity and therefore simplify the control unit, we have extended the frames at the upper and left frame boundaries for 4 pixels in depth as shown in Fig. 4.7. 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.

This H.264 DBF hardware architecture is implemented in Verilog HDL. We applied the PECR and PSCR techniques separately to this H.264 DBF hardware. The equality and similarity of the pixels are determined by checking whether the output of the 12-bit subtractor in the datapath is zero or not.

The Verilog RTL codes are mapped to Virtex 4 FPGA (an FPGA with 24576 slices implemented in 90nm CMOS technology), and the FPGA implementations are verified with post place&route simulations. The FPGA resource usage and the clock frequency of the DBF hardware implementations are shown in Table 4.19. DBF hardware with PECR technique works at 98 MHz and it takes 5574 clock cycles in the worst-case to process a MB. Therefore, the FPGA implementation can process a CIF (352x288) frame in 22.54 ms (396 MB * 5574 clock cycles per MB * 10.21 ns clock cycle = 22.54 ms), and it can process 1000/22.54 = 44 CIF frames per second.

Fig. 4.9 shows an example unfiltered video frame and the same frame filtered by H.264 DBF algorithm with PECR technique. Akiyo video frame is used as input for timing simulation. As it can be seen from Fig. 4.9, some of the blocking artifacts are reduced and some of them are totally removed.

The power consumptions of the DBF hardware implementations at 25 MHz on the same FPGA are estimated using Xilinx XPower Analyzer tool. In order to estimate the power consumption of a DBF hardware implementation, timing simulation of its placed and routed netlist is done. Foreman, Akiyo, Mother&Daughter and Ice video frames (one frame from each video) are used as inputs for timing simulations and the signal activities are stored in VCD files. These VCD files are used for estimating the power consumptions of the DBF hardware implementations using this gate level power estimation tool.

Table 4.19 FPGA Resource Usage and Clock Frequency After P&R

Resource	**DBF Hardware [37]**	**DBF Hardware with PECR Tech.**	**DBF Hardware with PSCR Tech. (2bT)**
LUTs	1198	1301	1339
DFFs	293	288	307
Block RAMs	7	7	7
Clock Frequency	97 MHz	98 MHz	100 MHz

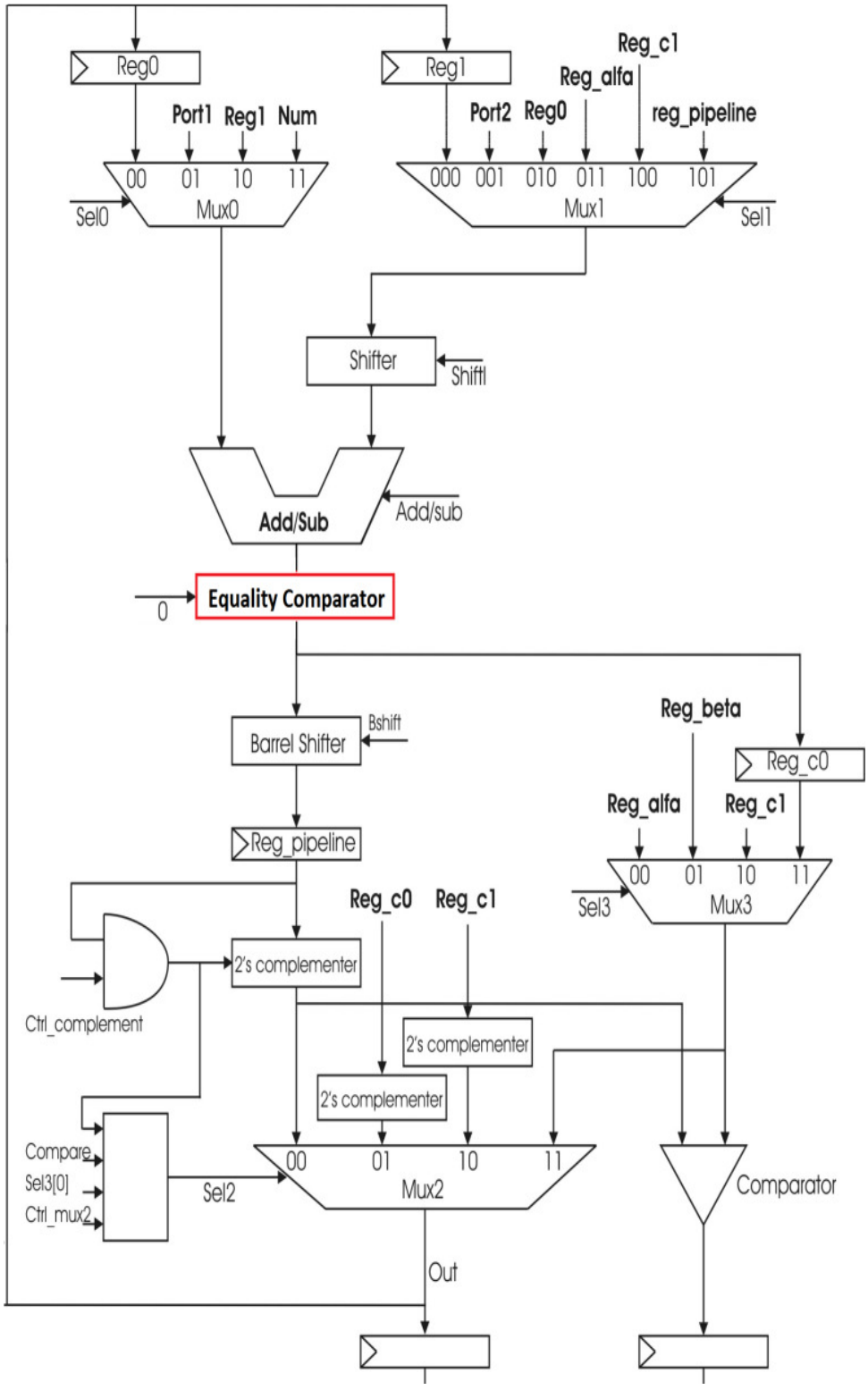


Figure 4.8 H.264 DBF Datapath

Figure 4.9 Unfiltered video frame shown above and the same frame filtered by H.264 Deblocking Filter algorithm shown below

The energy consumptions of the H.264 DBF hardware implementations on the same FPGA for different QP values and video frames are shown in Tables 4.20, 4.21 and 4.22. As shown in these tables, the proposed PECR and PCSR techniques reduced both total computation time and power consumption of the H.264 DBF hardware. The proposed PECR and PSCR techniques reduced the energy consumption of the H.264 DBF hardware up to 35% and 39%, respectively.

Table 4.20 Energy Consumption Reduction By PECR Technique

Frame		Total Computation Time (µs)			Power (mW)		Energy (µJ)		
	QP	Org. [37]	LP	Δ time	Org. [37]	LP	Org. [37]	LP	%
Foreman	*28*	81607	79809	1798	51.44	37.81	4198	3018	**28.1**
	35	84643	81396	3247	51.43	37.76	4353	3074	**29.4**
	42	86689	82096	4593	51.48	37.77	4463	3101	**30.5**
Akiyo	*28*	84317	79454	4863	51.33	37.65	4328	2991	**30.9**
	35	86152	79859	6293	51.35	37.62	4424	3004	**32.1**
	42	87678	80373	7305	51.37	37.59	4504	3021	**32.9**
M & D	*28*	83502	78578	4924	51.32	37.63	4285	2957	**31.0**
	35	86528	80493	6035	51.33	37.62	4441	3028	**31.8**
	42	88295	81106	7189	51.79	37.61	4573	3050	**33.3**
Ice	*28*	83847	77012	6835	51.24	37.45	4296	2884	**32.9**
	35	85049	76461	8588	51.21	37.40	4355	2860	**34.3**
	42	83695	74302	9393	51.03	37.37	4271	2777	**35.0**

Table 4.21 Energy Consumption Reduction By PSCR (1bT) Technique

Frame		Total Computation Time (µs)			Power (mW)		Energy (µJ)		
	QP	Org. [37]	LP	Δ time	Org. [37]	LP	Org. [37]	LP	%
Foreman	*28*	81607	77326	4281	51.44	34.4	4198	2660	**32.6**
	35	84643	78942	5701	51.43	34.37	4353	2713	**33.7**
	42	86689	78975	7714	51.48	34.42	4463	2718	**34.1**
Akiyo	*28*	84317	78432	5885	51.33	36.32	4328	2692	**35.8**
	35	86152	77103	9049	51.35	36.22	4424	2638	**36.4**
	42	87678	78809	8869	51.37	36.47	4504	2717	**36.7**
M & D	*28*	83502	75833	7669	51.32	34.24	4285	2597	**35.4**
	35	86528	78269	8259	51.33	34.25	4441	2681	**36.2**
	42	88295	79953	8342	51.79	34.29	4573	2742	**39.0**
Ice	*28*	83847	74128	9719	51.24	36.21	4296	2536	**37.5**
	35	85049	74512	10537	51.21	36.33	4355	2558	**38.8**
	42	83695	74761	8934	51.03	36.40	4271	2572	**36.3**

Table 4.22 Energy Consumption Reduction By PSCR (2bT) Technique

Frame		Total Computation Time (µs)			Power (mW)		Energy (µJ)		
	QP	**Org. [37]**	**LP**	**Δ time**	**Org. [37]**	**LP**	**Org. [37]**	**LP**	**%**
Foreman	***28***	81607	78398	3209	51.44	36.03	4198	2825	**32.7**
	35	84643	79778	4865	51.43	36.01	4353	2873	**34.0**
	42	86689	80381	6308	51.48	36.01	4463	2895	**35.1**
Akiyo	***28***	84317	77368	6949	51.33	35.83	4328	2772	**35.9**
	35	86152	78063	8089	51.35	35.83	4424	2797	**36.8**
	42	87678	78676	9002	51.37	35.82	4504	2818	**37.4**
M & D	***28***	83502	76917	6585	51.32	35.83	4285	2756	**35.7**
	35	86528	78717	7811	51.33	35.82	4441	2820	**36.5**
	42	88295	79418	8877	51.79	35.84	4573	2846	**37.8**
Ice	***28***	83847	74846	9001	51.24	35.63	4296	2667	**37.9**
	35	85049	74767	10282	51.21	35.59	4355	2661	**38.9**
	42	83695	75332	8363	51.03	35.56	4271	2679	**37.3**

CHAPTER V

A NOVEL ENERGY REDUCTION TECHNIQUE FOR H.264 INTRA MODE DECISION

H.264 intra prediction algorithm uses 9 4x4 luma, 4 16x16 luma, and 4 8x8 chroma modes. The luma component of each MB in a frame has 16 4x4 blocks and each 4x4 block can be coded with one of 9 different 4x4 prediction modes. The same MB can also be coded with one of 4 different 16x16 prediction modes. Therefore, in order to choose the best mode for the luma component of a MB, intra predictions for 148 different prediction modes are calculated.

H.264 Joint Model (JM) reference software encoder implements two different intra mode decision algorithms; Lagrangian Rate Distortion Optimization (RDO) based mode decision and Sum of Absolute Transformed Difference (SATD) based mode decision [17]. Lagrangian RDO based mode decision algorithm selects the prediction mode that minimizes the Lagrangian cost function shown in (5.1). Distortion (D) and rate (R) for each prediction mode are determined by encoding the current block using this prediction mode and calculating the distortion and rate. λ is calculated based on Quantization Parameter (QP). This technique has extremely high computational complexity.

$$J = D + \lambda R \tag{5.1}$$

SATD based intra mode decision algorithm also selects the prediction mode that minimizes the Lagrangian cost function shown in (5.1). However, it estimates distortion as SATD and rate as the number of bits used for encoding the prediction mode. This SATD based cost function is defined as

$$J_{SATD} = SATD + 4\lambda R \tag{5.2}$$

SATD based intra 16x16 mode decision algorithm used in JM software calculates the cost of each intra 16x16 mode and selects the mode with minimum cost. For each intra 16x16 mode, the SATD value for a MB is calculated as follows: For each 4x4 block in a MB, denoted as (0,…,15) in Figure 5.1, find residue block by subtracting predicted block from current block, and apply Hadamard Transform (HT) to each 4x4 residue block as illustrated in Figure 5.2. Form a 4x4 DC block, as shown in Figure 5.1, by extracting DC coefficients (upper leftmost coefficient, shown as gray in Figure 5.1) of each transformed 4x4 block and dividing them by 2, and apply HT to this 4x4 DC block. Add the absolute values of all AC coefficients and Hadamard transformed and scaled DC coefficients.

Intra 4x4 mode decision algorithm calculates the cost of each 4x4 mode for each 4x4 block denoted as 0,…,15 in Figure 5.1 and chooses the mode with the minimum cost for each 4x4 block. After the best modes are selected for all 4x4 blocks, the costs of the best modes for all 4x4 blocks are added to determine the total cost of the current MB. This cost is compared with the cost of the best 16x16 mode to decide the intra mode for the luma component of this MB. The Intra 8x8 mode decision algorithm is very similar to intra 16x16 mode decision algorithm except that a 4x4 DC block is not formed.

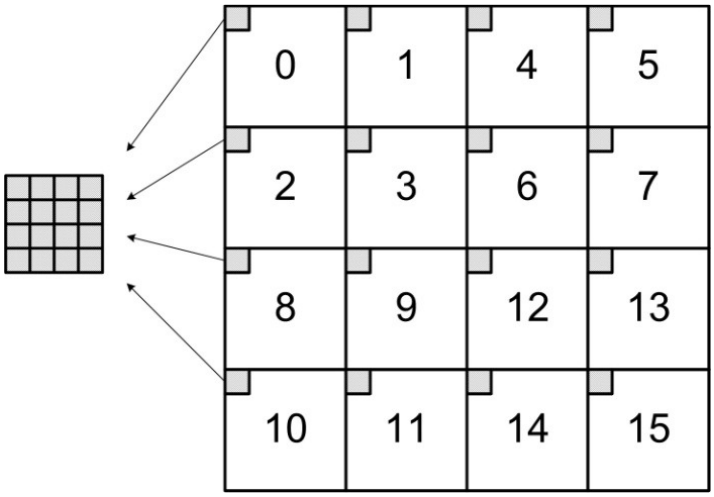


Figure 5.1 Formation of DC Block for Intra 16x16 Prediction Modes

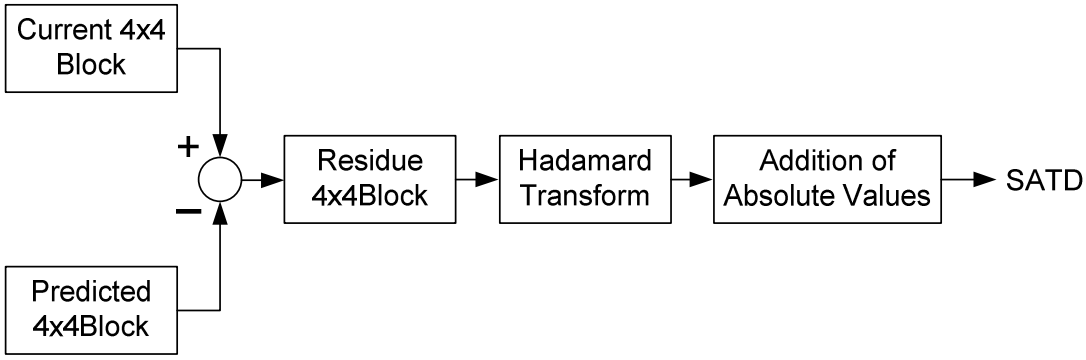


Figure 5.2 SATD Calculation for Each 4x4 Block

The computational complexity of the SATD based mode decision algorithm is also high. As it is shown in Figure 5.3, only 11% of all the addition operations performed for intra search are performed for intra prediction and 89% are performed for intra mode decision. Intra prediction shown in Figure 5.3 is implemented using only addition and shift operations as explained in [22, 24, 25]. Intra mode decision shown in Figure 5.3 includes residue operations (subtractions are counted as additions), HT operations, and addition of absolute values.

In this thesis, we propose a novel energy reduction technique for H.264 intra mode decision. The proposed technique reduces the number of additions and shifts performed by 16x16 and 8x8 intra prediction algorithms by 80% and it reduces the number of additions performed by SATD based 4x4, 16x16 and 8x8 intra mode decision algorithms used in

H.264 JM reference software encoder by 46%, 64% and 62% respectively for a CIF size frame with very small PSNR loss by using fixed predicted block patterns of intra modes and distribution property of HT and by slightly modifying intra 16x16 and 8x8 plane mode prediction equations used for cost calculation by SATD based 16x16 and 8x8 intra mode decision algorithms.

We also implemented an efficient H.264 16x16 intra mode decision hardware including the proposed technique using Verilog HDL. The proposed technique reduced the energy consumption of this H.264 16x16 intra mode decision hardware up to 59.6%.

Several techniques are proposed in the literature to reduce the computational complexity of H.264 intra mode decision. In [29], a new cost function and rate predictor, and a technique similar to the technique proposed in this thesis are proposed only for intra 4x4 mode decision. Selective intra mode decision techniques proposed in [22, 30, 31, 32, 33, 34] calculate only the cost of the intra modes likely to be selected by the mode decision and select one of these intra modes at the expense of PSNR loss. The proposed technique can be used together with these selective mode decision techniques for further reducing computational complexity of H.264 intra mode decision.

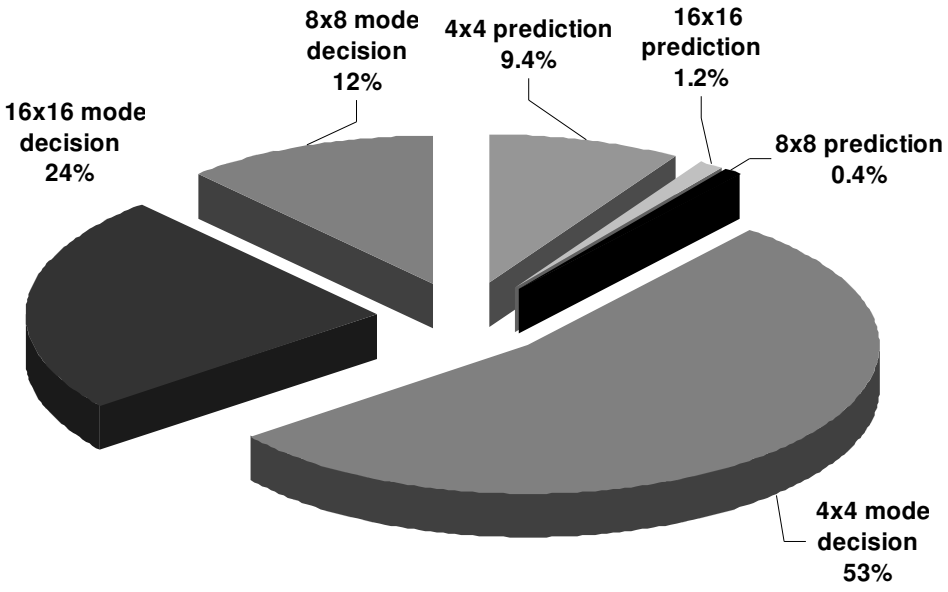


Figure 5.3 Addition Operations Performed by Intra Prediction and Mode Decision

5.1 Hadamard Transform

HT is a linear transform and HT of a 4x4 block Z is defined as:

$$T = H * Z * H \tag{5.3}$$

where

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{5.4}$$

If we write block Z explicitly then, Equation (4.3) becomes:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} z_0 & z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 & z_7 \\ z_8 & z_9 & z_{10} & z_{11} \\ z_{12} & z_{13} & z_{14} & z_{15} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{5.5}$$

Only binary shift and integer addition/subtraction operations are used in HT. HT defined in (5.5) can be implemented with 64 additions using the fast HT algorithm shown in Figure 5.4 [35].

As part of H.264 intra mode decision hardware, we designed a high speed HT hardware based on this fast HT algorithm. The designed hardware is two-stage pipelined to improve clock frequency and has 16 adders/subtractors. It finishes HT operations of a 4x4 block in 4 clock cycles.

5.2 Proposed Computational Complexity Reduction Technique

HT is a linear operation and it can be applied before subtraction operation as shown in (5.6). H, C, P are the Hadamard matrix, current 4x4 block, predicted 4x4 block respectively and the Hadamard matrix is shown in (5.4). In this way, two HTs are performed instead of one. However, this decreases the computational complexity of SATD

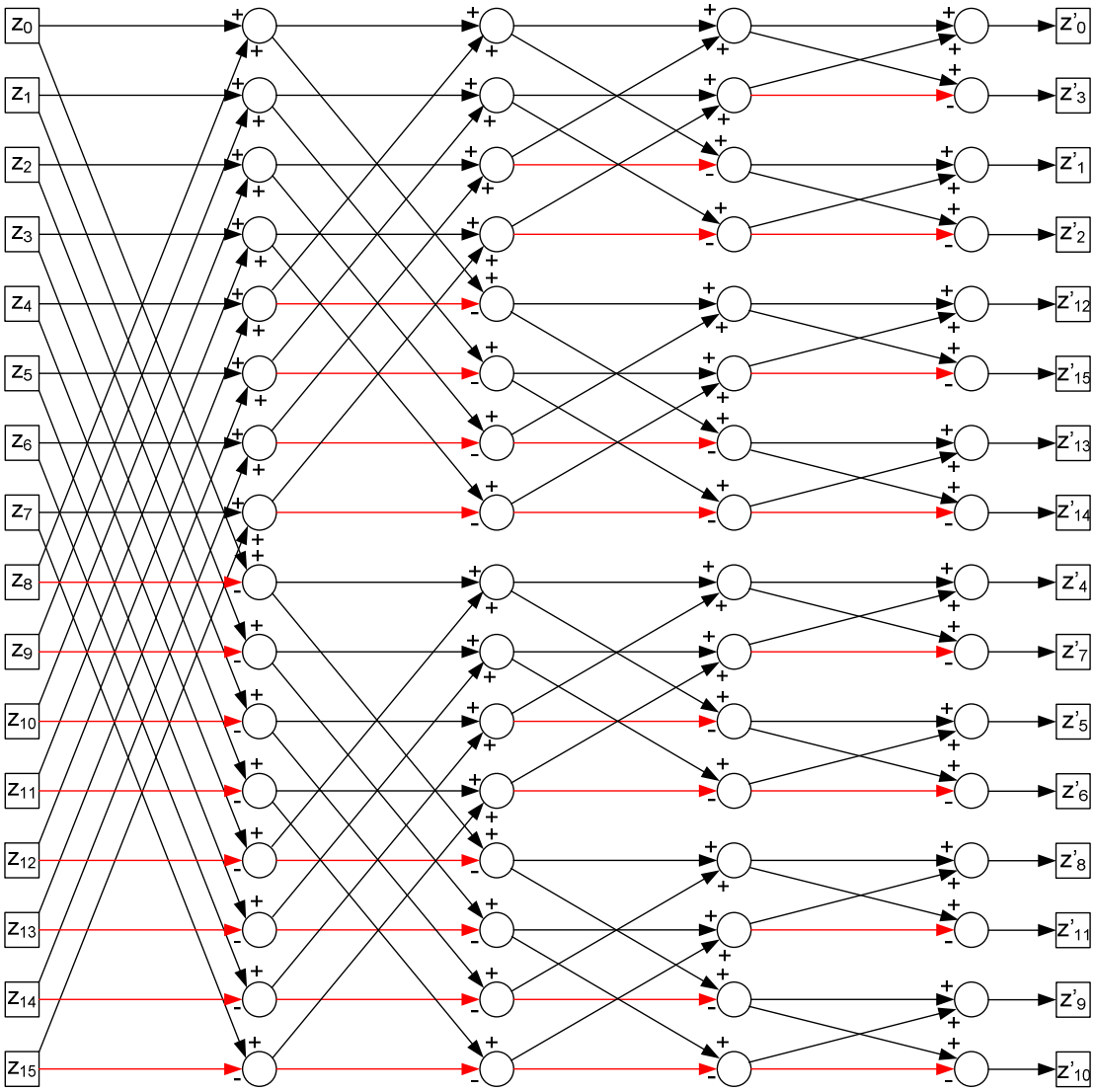


Figure 5.4 Fast HT Algorithm for a 4x4 Block

based H.264 intra mode decision. Since the 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 modes, once HT of the current block is found it can be used for all intra prediction modes.

$$T = H*(C-P)*H = (H*C*H)-(H*P*H) \tag{5.6}$$

A similar technique is proposed only for intra 4x4 mode decision in [29, 35]. In this thesis, we generalized this technique for the mode decision of all intra prediction modes, we showed that this technique reduces the number of residue calculations required for intra mode decision as well and we applied this technique to 16x16 and 8x8 plane modes by

proposing a small modification in the prediction equations used for calculating the cost of the 16x16 and 8x8 plane modes for intra mode decision.

5.2.1 HT of Predicted Blocks by Intra 4x4 Modes

The predicted block patterns of horizontal, vertical and DC prediction modes and the result of performing HT for these predicted block patterns are shown in Figure 5.5. HT of a 4x4 block can be calculated with 64 addition operations [29]. On the other hand, as shown in Figure 5.5, HT of a 4x4 block predicted by vertical or horizontal modes can be calculated with 8 addition and 4 shift operations and HT of a 4x4 block predicted by DC mode can be calculated with only 1 shift operation.

Vertical

$$\begin{bmatrix} k & m & n & p \\ k & m & n & p \\ k & m & n & p \\ k & m & n & p \end{bmatrix} \xrightarrow{HT} \begin{bmatrix} 4(k+m+n+p) & 4(k+m-n-p) & 4(k-m-n+p) & 4(k-m+n-p) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Horizontal

$$\begin{bmatrix} k & k & k & k \\ m & m & m & m \\ n & n & n & n \\ p & p & p & p \end{bmatrix} \xrightarrow{HT} \begin{bmatrix} 4(k+m+n+p) & 0 & 0 & 0 \\ 4(k+m-n-p) & 0 & 0 & 0 \\ 4(k-m-n+p) & 0 & 0 & 0 \\ 4(k-m+n-p) & 0 & 0 & 0 \end{bmatrix}$$

DC

$$\begin{bmatrix} p & p & p & p \\ p & p & p & p \\ p & p & p & p \\ p & p & p & p \end{bmatrix} \xrightarrow{HT} \begin{bmatrix} 16p & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 5.5 Hadamard Transform of Vertical, Horizontal and DC Modes

The predicted block pattern of DDL mode is shown in (5.7) where k-s are defined in [1]. HT of this predicted block, shown as TDDL in (5.8), can be efficiently calculated if equations in Table 5.1 are pre-calculated. TDDL can be calculated by using pre-calculated values as shown in Table 5.2. The predicted block pattern of DDR mode is shown in (5.9)

where k-s are defined in [1]. HT of this predicted block, shown as TDDR in (5.10), can be efficiently calculated if equations in Table 5.3 are pre-calculated. TDDR be calculated by using pre-calculated values as shown in Table 5.4.The predicted block pattern of VR mode is shown in (5.11) where k-s are defined in [1]. HT of this predicted block, shown as TVR in (5.12), can be efficiently calculated if equations in Table 5.5 are pre-calculated. TVR can be calculated by using pre-calculated values as shown in Table 5.6.

The predicted block pattern of HD mode is shown in (5.13). HT of this predicted block is given in (5.14). Since this mode is similar to VR, we do not give explicit equations. The predicted block pattern of VL mode is shown in (5.15). HT of this predicted block is given in (5.16). Since this mode is similar to VR, we do not give explicit equations. The predicted block pattern of HUP mode is shown in (5.17) where k-s are defined in [1]. HT of this predicted block, shown as THUP in (5.18), can be efficiently calculated if equations in Table 5.7 are pre-calculated. THUP can be calculated by using pre-calculated values as shown in Table 5.8.

5.2.2 HT of Predicted Blocks by Intra 16x16 and 8x8 Horizontal, Vertical and DC Modes

In addition to the computation reduction achieved for HT of a 4x4 block, since a MB is partitioned into 4x4 blocks for HT as shown in Figure 5.6, the proposed technique significantly reduces amount of computations required for intra 16x16 and 8x8 mode decisions by data reuse. For intra 16x16 vertical mode, the predicted pixels in the 4x4 predicted blocks 0, 2, 8, and 10 are the same as shown in (5.19) and HT of this block is shown in (5.20). HT of the predicted block 0 can be reused for predicted blocks 2, 8, and 10 as well. The same is true for the other vertical predicted 4x4 blocks in the same column. For intra 16x16 horizontal mode, the predicted pixels in the 4x4 predicted blocks 0, 1, 4, and 5 are the same as shown in (5.21) and HT of this block is shown in (5.22). Therefore, HT of the predicted block 0 can be reused for predicted blocks 1, 4, and 5 as well. The same is true for the other horizontal predicted 4x4 blocks in the same row. For DC mode, the predicted pixels in all the 4x4 predicted blocks are the same as shown in (5.23) and HT of this block is shown in (5.24). Therefore, HT of the predicted block 0, shown in (5.24), can be reused for all the other 4x4 DC predicted blocks.

$$DDL = \begin{bmatrix} k & m & n & p \\ m & n & p & q \\ n & p & q & r \\ p & q & r & s \end{bmatrix} \tag{5.7}$$

$$TDDL = \begin{bmatrix} k+2m+3n+4p+3q+2r+s & k+2m+n-q-2r-s & k-n-q+s & k+n-q-s \\ k+2m+n-q-2r-s & k+2m-n-4p-q+2r+s & k-3n+3q-s & k-n-q+s \\ k-n-q+s & k-3n+3q-s & k-2m-n+4p-q-2r+s & k-2m+n-q+2r-s \\ k+n-q-s & k-n-q+s & k-2m+n-q+2r-s & k-2m+3n-4p+3q-2r+s \end{bmatrix} \tag{5.8}$$

$$DDR = \begin{bmatrix} k & m & n & p \\ q & k & m & n \\ r & q & k & m \\ s & r & q & k \end{bmatrix} \tag{5.9}$$

$$TDDR = \begin{bmatrix} 4k+3q+2r+s+3m+2n+p & q+2r+s-m-2n-p & -q+s-m+p & q+s-m-p \\ -q-2r-s+m+2n+p & 4k+q-2r-s+m-2n-p & 3q-s-3m+p & q-s+m-p \\ -q+s-m+p & -3q+s+3m-p & 4k-q-2r+s-m-2n+p & q-2r+s-m+2n-p \\ -q-s+m+p & q-s+m-p & -q+2r-s+m-2n+p & 4k-3q+2r-s-3m+2n-p \end{bmatrix} \tag{5.10}$$

$$VR = \begin{bmatrix} k & m & n & p \\ q & r & s & t \\ x & k & m & n \\ y & q & r & s \end{bmatrix} \tag{5.11}$$

$$TVR = \begin{bmatrix} 2k + 2q + x + y + 2m + 2r + 2n + 2s + p + t & 2k + 2q + x + y - 2n - 2s - p - t & x + y - 2m - 2r + p + t & x + y - p - t \\ - x - y + p + t & - x - y + 2m + 2r - p - t & 2k + 2q - x - y - 2n - 2s + p + t & 2k + 2q - x - y - 2m - 2r + 2n + 2s - p - t \\ - x + y + p - t & - x + y + 2m - 2r - p + t & 2k - 2q - x + y - 2n + 2s + p - t & 2k - 2q - x + y - 2m + 2r + 2n - 2s - p + t \\ 2k - 2q + x - y + 2m - 2r + 2n - 2s + p - t & 2k - 2q + x - y - 2n + 2s - p + t & x - y - 2m + 2r + p - t & x - y - p + t \end{bmatrix} \tag{5.12}$$

$$HD = \begin{bmatrix} k & m & n & p \\ q & r & k & m \\ s & t & q & r \\ x & y & s & t \end{bmatrix} \tag{5.13}$$

$$THD = \begin{bmatrix} 2k + 2q + 2s + x + 2m + 2r + 2t + y + n + p & x + y - n - p & x - y - n + p & 2k + 2q + 2s + x - 2m - 2r - 2t - y + n - p \\ 2k - 2s - x + 2m - 2t - y + n + p & 2q - x + 2r - y - n - p & 2q - x - 2r + y - n + p & 2k - 2s - x - 2m + 2t + y + n - p \\ - 2q + x - 2r + y + n + p & 2k - 2s + x + 2m - 2t + y - n - p & 2k - 2s + x - 2m + 2t - y - n + p & - 2q + x + 2r - y + n - p \\ - x - y + n + p & 2k - 2q + 2s - x + 2m - 2r + 2t - y - n - p & 2k - 2q + 2s - x - 2m + 2r - 2t + y - n + p & - x + y + n - p \end{bmatrix} \tag{5.14}$$

$$VL = \begin{bmatrix} k & m & n & p \\ q & r & s & t \\ m & n & p & x \\ r & s & t & y \end{bmatrix} \tag{5.15}$$

$$TVL = \begin{bmatrix} 2m + k + q + 2r + 2n + 2s + 2p + 2t + x + y & 2m + k + q + 2r - 2p - 2t - x - y & k + q - 2n - 2s + x + y & k + q - x - y \\ k + q - x - y & k + q - 2n - 2s + x + y & - 2m + k + q - 2r + 2p + 2t - x - y & k - 2m + q - 2r + 2n + 2s - 2p - 2t + x + y \\ k - q - x + y & k - q - 2n + 2s + x - y & - 2m + k - q + 2r + 2p - 2t - x + y & k - 2m - q + 2r + 2n - 2s - 2p + 2t + x - y \\ 2m + k - q - 2r + 2n - 2s + 2p - 2t + x - y & 2m + k - q - 2r - 2p + 2t - x + y & k - q - 2n + 2s + x - y & k - q - x + y \end{bmatrix} \tag{5.16}$$

$$HUP = \begin{bmatrix} k & m & n & p \\ n & p & q & r \\ q & r & s & s \\ s & s & s & s \end{bmatrix} \tag{5.17}$$

$$THUP = \begin{bmatrix} 2n + k + 2q + 6s + m + 2p + 2r & k - 2s + m & k - m & 2n + k + 2q - m - 2p - 2r \\ k + 2n - 6s + m + 2p & k - 2q + 2s + m - 2r & k - 2q - m + 2r & k + 2n - m - 2p \\ k - 2q + 2s + m - 2r & k - 2n + 2s + m - 2p & k - 2n - m + 2p & k - 2q - m + 2r \\ k - 2s + m & k - 2n + 2q - 2s + m - 2p + 2r & k - 2n + 2q - m + 2p - 2r & k - m \end{bmatrix} \tag{5.18}$$

Table 5.1 Pre-calculated Values for DDL Prediction Mode

Equations	Number of Addition/Subtractions	Number of Shift
$a(1) = 4p$	0	1
$a(2) = q + n$	1	0
$a(3) = q - n$	1	0
$a(4) = 2(q+n) + (q+n) = 3(q+n)$	1	1
$a(5) = 2(q-n) + (q-n) = 3(q-n)$	1	1
$a(6) = 2(m+r)$	1	1
$a(7) = 2(m-r)$	1	1
$a(8) = k + s$	1	0
$a(9) = k - s$	1	0
Total	8	5

Table 5.2 DDL Mode Prediction Calculations Using Pre-calculated Values

Equations	Number of Addition/Subtractions	Number of Shift
$TDDL[0,0] = a(1) + a(4) + a(6) + a(8)$	3	0
$TDDL[0,1] = a(7) + a(9) - a(3)$	2	0
$TDDL[0,2] = a(8) - a(2)$	1	0
$TDDL[0,3] = a(9) - a(3)$	1	0
$TDDL[1,0] = TDDL[0,1]$	0	0
$TDDL[1,1] = a(6) + a(8) - a(1) - a(2)$	3	0
$TDDL[1,2] = a(5) + a(9)$	1	0
$TDDL[1,3] = a(8) - a(2)$	1	0
$TDDL[2,0] = TDDL[0,2]$	0	0
$TDDL[2,1] = TDDL[1,2]$	0	0
$TDDL[2,2] = a(1) - a(2) - a(6) + a(8)$	3	0
$TDDL[2,3] = a(9) - a(3) - a(7)$	2	0
$TDDL[3,0] = TDDL[0,3]$	0	0
$TDDL[3,1] = TDDL[1,3]$	0	0
$TDDL[3,2] = TDDL[2,3]$	0	0
$TDDL[3,3] = a(8) - a(6) + a(4) - a(1)$	3	0
Total	20	0

Table 5.3 Pre-calculated Values for DDR Prediction Mode

Equations	Number of Additions/Subtractions	Number of Shifts
$a(1) = 4k$	0	1
$a(2) = q + m$	1	0
$a(3) = q - m$	1	0
$a(4) = 2(q+m) + (q+m) = 3(q+m)$	1	1
$a(5) = 2(q-m) + (q-m) = 3(q-m)$	1	1
$a(6) = 2(n+r)$	1	1
$a(7) = 2(n-r)$	1	1
$a(8) = p + s$	1	0
$a(9) = p - s$	1	0
Total	8	5

Table 5.4 DDR Mode Prediction Calculations Using Pre-calculated Values

Equations	Number of Additions/Subtractions	Number of Shifts
$TDDR[0,0] = a(1) + a(4) + a(6) + a(8)$	3	0
$TDDR[0,1] = a(3) - a(7) - a(9)$	2	0
$TDDR[0,2] = a(8) - a(2)$	1	0
$TDDR[0,3] = a(3) - a(9)$	1	0
$TDDR[1,0] = -TDDR[0,1]$	1	0
$TDDR[1,1] = a(1) + a(2) - a(6) - a(8)$	3	0
$TDDR[1,2] = a(5) + a(9)$	1	0
$TDDR[1,3] = a(2) - a(8)$	1	0
$TDDR[2,0] = TDDR[0,2]$	0	0
$TDRR[2,1] = -TDDR[1,2]$	1	0
$TDDR[2,2] = a(1) - a(2) - a(6) + a(8)$	3	0
$TDDR[2,3] = a(3) + a(7) - a(9)$	2	0
$TDDR[3,0] = -TDDR[0,3]$	1	0
$TDDR[3,1] = TDDR[1,3]$	0	0
$TDDR[3,2] = -TDDR[2,3]$	1	0
$TDDR[3,3] = a(1) - a(4) + a(6) - a(8)$	3	0
Total	24	0

Table 5.5 Pre-calculated Values for VR Prediction Mode

Equations	Number of Additions/Subtractions	Number of Shifts
$a(1) = y + x$	1	0
$a(2) = y - x$	1	0
$a(3) = p + t$	1	0
$a(4) = p - t$	1	0
$a(5) = a(1) + a(3)$	1	0
$a(6) = a(2) + a(4)$	1	0
$a(7) = a(1) - a(3)$	1	0
$a(8) = a(4) - a(2)$	1	0
$a(9) = 2(k + n)$	1	1
$a(10) = 2(k - n)$	1	1
$a(11) = 2(q + s)$	1	1
$a(12) = 2(q - s)$	1	1
$a(13) = a(9) - a(11)$	1	0
$a(14) = a(9) + a(11)$	1	0
$a(15) = a(10) + a(12)$	1	0
$a(16) = a(10) - a(12)$	1	0
$a(17) = 2(m + r)$	1	1
$a(18) = 2(m - r)$	1	1
Total	18	6

Table 5.6 VR Mode Prediction Calculations Using Pre-calculated Values

Equations	Number of Additions/Subtractions	Number of Shifts
$TVR[0,0] = a(14) + a(17) + a(5)$	2	0
$TVR[0,1] = a(15) + a(7)$	1	0
$TVR[0,2] = a(5) - a(17)$	1	0
$TVR[0,3] = a(7)$	0	0
$TVR[1,0] = -a(7)$	1	0
$TVR[1,1] = a(17) - a(5)$	1	0
$TVR[1,2] = a(15) - a(7)$	1	0
$TVR[1,3] = a(14) - a(17) - a(5)$	2	0
$TVR[2,0] = a(6)$	0	0
$TVR[2,1] = a(18) - a(8)$	1	0
$TVR[2,2] = a(6) + a(16)$	1	0
$TVR[2,3] = a(13) - a(18) - a(8)$	2	0
$TVR[3,0] = a(8) + a(13) + a(18)$	2	0
$TVR[3,1] = a(16) - a(6)$	1	0
$TVR[3,2] = a(8) - a(18)$	1	0
$TVR[3,3] = -a(6)$	1	0
Total	18	0

Table 5.7 Pre-calculated Values for HUP Prediction Mode

Equations	Number of Additions/Subtractions	Number of Shifts
$a(1) = k - m$	1	0
$a(2) = k + m$	1	0
$a(3) = 2(n + p)$	1	1
$a(4) = 2(n - p)$	1	1
$a(5) = 2(q + r)$	1	1
$a(6) = 2(q - r)$	1	1
$a(7) = 2s$	0	1
$a(8) = a(7) + 4s = 6s$	1	1
$a(9) = a(2) + a(7)$	1	0
$a(10) = a(2) + a(5)$	1	0
$a(11) = a(1) + a(6)$	1	0
Total	10	6

Table 5.8 HUP Mode Prediction Calculations Using Pre-calculated Values

Equations	Number of Additions/Subtractions	Number of Shifts
$THUP[0,0] = a(10) + a(3) + a(8)$	2	0
$THUP[0,1] = a(2) - a(7)$	1	0
$THUP[0,2] = a(1)$	0	0
$THUP[0,3] = a(11) + a(4)$	1	0
$THUP[1,0] = a(2) + a(3) - a(8)$	2	0
$THUP[1,1] = a(9) - a(5)$	1	0
$THUP[1,2] = a(1) - a(6)$	1	0
$THUP[1,3] = a(1) + a(4)$	1	0
$THUP[2,0] = THUP[1,1]$	0	0
$THUP[2,1] = a(9) - a(3)$	1	0
$THUP[2,2] = a(1) - a(4)$	1	0
$THUP[2,3] = THUP[1,2]$	0	0
$THUP[3,0] = a(2) - a(7)$	1	0
$THUP[3,1] = a(10) - a(3) - a(7)$	2	0
$THUP[3,2] = a(11) - a(4)$	1	0
$THUP[3,3] = a(1)$	0	0
Total	15	0

$$B0 = B2 = B8 = B10 = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \\ h_0 & h_1 & h_2 & h_3 \\ h_0 & h_1 & h_2 & h_3 \\ h_0 & h_1 & h_2 & h_3 \end{bmatrix} \tag{5.19}$$

$$\mathrm{HT(B0)} = \mathrm{HT(B2)} = \mathrm{HT(B8)} = \mathrm{HT(B10)} = \begin{bmatrix} 4(h_0 + h_1 + h_2 + h_3) & 4(h_0 + h_1 - h_2 - h_3) & 4(h_0 - h_1 - h_2 + h_3) & 4(h_0 - h_1 + h_2 - h_3) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.20}$$

$$B0 = B2 = B8 = B10 = \begin{bmatrix} v_0 & v_0 & v_0 & v_0 \\ v_1 & v_1 & v_1 & v_1 \\ v_2 & v_2 & v_2 & v_2 \\ v_3 & v_3 & v_3 & v_3 \end{bmatrix} \tag{5.21}$$

$$HT(B0) = HT(B1) = HT(B4) = HT(B5) = \begin{bmatrix} 4(v_0 + v_1 + v_2 + v_3) & 0 & 0 & 0 \\ 4(v_0 + v_1 - v_2 - v_3) & 0 & 0 & 0 \\ 4(v_0 - v_1 - v_2 + v_3) & 0 & 0 & 0 \\ 4(v_0 - v_1 + v_2 - v_3) & 0 & 0 & 0 \end{bmatrix} \tag{5.22}$$

	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
v_0	0				1				4				5			
v_1																
v_2																
v_3																
v_4	2				3				6				7			
v_5																
v_6																
v_7																
v_8	8				9				12				13			
v_9																
v_{10}																
v_{11}																
v_{12}	10				11				14				15			
v_{13}																
v_{14}																
v_{15}																

Figure 5.6 16x16 MB and its Neighboring Pixels

$$B0 = ... = B15 = \begin{bmatrix} p & p & p & p \\ p & p & p & p \\ p & p & p & p \\ p & p & p & p \end{bmatrix}$$

$$where\ p = \left(\sum_{i=0}^{15} (h_i + v_i) + 16 \right) >> 5 \tag{5.23}$$

$$HT(B0) = ... = HT(B15) \begin{bmatrix} 16p & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.24}$$

Intra 16x16 mode decision algorithm also includes applying HT to 4x4 DC blocks formed by DC coefficients of HT of each 4x4 block shown in Figure 5.6. We propose to apply the same technique to 4x4 DC blocks as well. After HT is applied to current block (C) and predicted block (P), a 4x4 DC block is formed by DC coefficients of HT of C and a 4x4 DC block is formed by DC coefficients of HT of P as shown in (5.6). Then, HT is applied to these 4x4 DC blocks and the results are subtracted. 4x4 DC blocks formed by DC coefficients of HT of predicted blocks by intra modes have the same block patterns as the HT of predicted blocks themselves. For example, 4x4 DC block formed by DC coefficients of HT of predicted block by vertical mode is shown in (5.25) and its HT is shown in (5.26). It has 4 nonzero elements same as the HT of predicted block itself. Horizontal mode is similar to vertical mode. 4x4 DC block formed by DC coefficients of HT of predicted block by horizontal mode is shown in (5.27) and its HT is shown in (5.28). 4x4 DC block formed by DC coefficients of HT of predicted block by DC mode is shown in (5.29) and its HT is shown in (5.30). It has 1 nonzero element same as the HT of predicted block itself. Therefore, HT of 4x4 DC blocks for each intra 16x16 prediction mode can be calculated with small amount of computation by using the proposed technique.

Intra 8x8 Vertical, Horizontal and DC modes are very similar to corresponding intra 16x16 modes except that no 4x4 DC block is formed. Therefore, similar computation reductions are achieved for intra 8x8 Vertical, Horizontal and DC modes.

$$4\times\begin{bmatrix} h_0+h_1+h_2+h_3 & h_4+h_5+h_6+h_7 & h_8+h_9+h_{10}+h_{11} & h_{12}+h_{13}+h_{14}+h_{15} \\ h_0+h_1+h_2+h_3 & h_4+h_5+h_6+h_7 & h_8+h_9+h_{10}+h_{11} & h_{12}+h_{13}+h_{14}+h_{15} \\ h_0+h_1+h_2+h_3 & h_4+h_5+h_6+h_7 & h_8+h_9+h_{10}+h_{11} & h_{12}+h_{13}+h_{14}+h_{15} \\ h_0+h_1+h_2+h_3 & h_4+h_5+h_6+h_7 & h_8+h_9+h_{10}+h_{11} & h_{12}+h_{13}+h_{14}+h_{15} \end{bmatrix} \quad (5.25)$$

$$HT(DC)=\begin{bmatrix} X & Y & Z & T \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where

$$X = 16(h_0+h_1+h_2+h_3+h_4+h_5+h_6+h_7+h_8+h_9+h_{10}+h_{11}+h_{12}+h_{13}+h_{14}+h_{15})$$
$$Y = 16(h_0+h_1+h_2+h_3+h_4+h_5+h_6+h_7-h_8-h_9-h_{10}-h_{11}-h_{12}-h_{13}-h_{14}-h_{15})$$
$$Z = 16(h_0+h_1+h_2+h_3-h_4-h_5-h_6-h_7-h_8-h_9-h_{10}-h_{11}+h_{12}+h_{13}+h_{14}+h_{15})$$
$$T = 16(h_0+h_1+h_2+h_3-h_4-h_5-h_6-h_7+h_8+h_9+h_{10}+h_{11}-h_{12}-h_{13}-h_{14}-h_{15})$$

(5.26)

$$4\times\begin{bmatrix} v_0+v_1+v_2+v_3 & v_0+v_1+v_2+v_3 & v_0+v_1+v_2+v_3 & v_0+v_1+v_2+v_3 \\ v_4+v_5+v_6+v_7 & v_0+v_1+v_2+v_3 & v_0+v_1+v_2+v_3 & v_0+v_1+v_2+v_3 \\ v_8+v_9+v_{10}+v_{11} & v_8+v_9+v_{10}+v_{11} & v_8+v_9+v_{10}+v_{11} & v_8+v_9+v_{10}+v_{11} \\ v_{12}+v_{13}+v_{14}+v_{15} & v_{12}+v_{13}+v_{14}+v_{15} & v_{12}+v_{13}+v_{14}+v_{15} & v_{12}+v_{13}+v_{14}+v_{15} \end{bmatrix}$$

(5.27)

$$HT(DC)=$$
$$16\times\begin{bmatrix} v_0+v_1+v_2+v_3+v_4+v_5+v_6+v_7+v_8+v_9+v_{10}+v_{11}+v_{12}+v_{13}+v_{14}+v_{15} & 0 & 0 & 0 \\ v_0+v_1+v_2+v_3+v_4+v_5+v_6+v_7-v_8-v_9-v_{10}-v_{11}-v_{12}-v_{13}-v_{14}-v_{15} & 0 & 0 & 0 \\ v_0+v_1+v_2+v_3-v_4-v_5-v_6-v_7-v_8-v_9-v_{10}-v_{11}+v_{12}+v_{13}+v_{14}+v_{15} & 0 & 0 & 0 \\ v_0+v_1+v_2+v_3-v_4-v_5-v_6-v_7+v_8+v_9+v_{10}+v_{11}-v_{12}-v_{13}-v_{14}-v_{15} & 0 & 0 & 0 \end{bmatrix}$$

(5.28)

$$DC = \begin{bmatrix} 16p & 16p & 16p & 16p \\ 16p & 16p & 16p & 16p \\ 16p & 16p & 16p & 16p \\ 16p & 16p & 16p & 16p \end{bmatrix} \tag{5.29}$$

$$HT(DC) = \begin{bmatrix} 256p & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.30}$$

5.2.3 HT of Predicted Blocks by Intra 16x16 and 8x8 Plane Mode

Plane mode is the most complex prediction mode and it constitutes almost 90% of addition and 100% of shift operations performed by 16x16 and 8x8 intra predictions. Plane mode first calculates a, b, c parameters from the neighboring pixels of the current MB. It then calculates the predicted pixels using a, b, c as shown in (2.7.d). If the following two small modifications are made in plane mode equations, HT of a block predicted by plane mode can be calculated with a very small amount of computation; Clip1 in (5.31) is removed (Clip1 is a function which clips the predicted pixel value between 0 and 255) and right shift by 5 in (5.31) is changed to divide by 32. The new plane mode equation shown in (5.32) is only used for calculating the cost of 16x16 and 8x8 plane modes for intra mode decision. If plane mode is selected by mode decision, the actual predicted pixels will be calculated using a, b, and c.

$$pred[x, y] = Clip1((a + b*(x-7) + c*(y-7) + 16) >> 5) \tag{5.31}$$

$$pred[x, y] = (a + b*(x-7) + c*(y-7) + 16)/32 \tag{5.32}$$

Modified plane mode equation shown in (5.32) simplifies HT of 16x16 plane mode significantly. Equation (5.33) shows HT of 16x16 plane mode.

Using the modified equation given in (5.32), we can calculate the cost of the plane mode by only using a, b, c parameters without calculating actual predicted pixels. Therefore, the number of additions and shifts performed by 16x16 and 8x8 intra

prediction algorithms for intra mode decision is reduced by approximately 80%. As shown in (5.33) for modified plane mode, HT of predicted blocks 1,…,15 are exactly the same as HT of predicted block 0 except DC coefficient. Therefore, HT of predicted block 0 can be reused for all other predicted 4x4 blocks.

In addition, proposed technique can be applied to 4x4 DC block formed by DC coefficients of HT of predicted block by plane mode as well. 4x4 DC block formed by DC coefficients of HT of predicted block by plane mode is shown in (5.34) and its HT is shown in (5.35). As shown in (5.34) and (5.35), HT of plane mode as well as HT of its DC block can be found easily once a, b, c parameters are calculated.

5.3 Computation Reduction for Residue Calculations

Residue calculations require more subtraction operations than the addition operations required by intra prediction. The proposed technique also significantly reduces the number of residue calculations in intra mode decision algorithm. The residue calculation is only needed for the nonzero elements in the HT of a predicted block. As shown in Figure 5.5, since HT of a 4x4 block predicted by DC prediction mode has only 1 nonzero element, only 1 residue calculation is needed and 15 subtraction operations are avoided for this 4x4 block. Similarly, since HT of a 4x4 block predicted by vertical and horizontal prediction modes have only 4 nonzero elements, 12 subtraction operations are avoided for each 4x4 block for vertical and horizontal prediction modes. In addition, as shown in (5.33), since HT of a 4x4 block predicted by modified plane mode has 5 nonzero elements, 11 subtraction operations are avoided during residue calculations for each 4x4 block.

$$\frac{1}{32}\times\begin{bmatrix}
16a-88b-88c+256 & -16b & 0 & -8b & 16a-24b-88c+256 & -16b & 0 & -8b & 16a+40b-88c+256 & -16b & 0 & -8b & 16a+104b-88c+256 & -16b & 0 & -8b \\
-16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 \\
16a-88b-24c+256 & -16b & 0 & -8b & 16a-24b-24c+256 & -16b & 0 & -8b & 16a+40b-24c+256 & -16b & 0 & -8b & 16a+104b-24c+256 & -16b & 0 & -8b \\
-16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 \\
16a-88b+40c+256 & -16b & 0 & -8b & 16a-24b+40c+256 & -16b & 0 & -8b & 16a+40b+40c+256 & -16b & 0 & -8b & 16a+104b+40c+256 & -16b & 0 & -8b \\
-16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 \\
16a-88b+104c+256 & -16b & 0 & -8b & 16a-24b+104c+256 & -16b & 0 & -8b & 16a+40b+104c+256 & -16b & 0 & -8b & 16a+104b+104c+256 & -16b & 0 & -8b \\
-16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 & -16c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0 & -8c & 0 & 0 & 0
\end{bmatrix} \tag{5.33}$$

$$\frac{1}{32}\times\begin{bmatrix}
16a-88b-88c+256 & 16a-24b-88c+256 & 16a+40b-88c+256 & 16a+104b-88c+256 \\
16a-88b-24c+256 & 16a-24b-24c+256 & 16a+40b-24c+256 & 16a+104b-24c+256 \\
16a-88b+40c+256 & 16a-24b+40c+256 & 16a+40b+40c+256 & 16a+104b+40c+256 \\
16a-88b+104c+256 & 16a-24b+104c+256 & 16a+40b+104c+256 & 16a+104b+104c+256
\end{bmatrix} \tag{5.34}$$

$$\begin{bmatrix}
8a+4b+4c+128 & -32b & 0 & -16b \\
-32c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-16c & 0 & 0 & 0
\end{bmatrix} \tag{5.35}$$

5.4 Computation Reduction Results

We quantified the computation reductions achieved by the proposed technique for the SATD based intra mode decision algorithm used in H.264 JM software encoder version 14.0 [17]. For 4x4 modes the computation amounts for a 4x4 block and for 16x16 and 8x8 modes the computation amounts for a 16x16 MB are shown in Table 5.9. The columns labeled I show the amount of computation performed by the original SATD mode decision and the columns labeled II show the amount of computation performed by the SATD mode decision using the proposed technique. Since current block HT is common for both intra 16x16 and 4x4 mode decision, the results of the current block HT for intra 16x16 mode decision are reused for intra 4x4 mode decision. The results show that the proposed technique significantly reduces the computational complexity of SATD based intra 4x4, 16x16 and 8x8 mode decision algorithms.

We also quantified the impact of the proposed modifications for the 16x16 and 8x8 plane mode equations on the rate-distortion performance of the SATD based intra mode decision algorithm used in H.264 JM reference software encoder version 14.0. Rate distortion curves and average PSNR comparison of the original SATD mode decision and the SATD mode decision using modified plane mode equations for several CIF size benchmark video frames are shown in Figure 5.7 and Table 5.10. The proposed plane mode equation modifications don't affect the PSNR for Football, they increase the PSNR slightly for Foreman and Mother&Daughter, and they decrease the PSNR slightly for other video frames shown in Table 5.10. The average PSNR values shown in Table 5.10 are calculated using the technique described in [26].

Table 5.9 Computation Reductions for Intra Prediction Modes

	Prediction Modes	Hadamard Transform				Residue	
		Addition		Shift		Subtraction	
		I	II	I	II	I	II
Intra 4x4	Vertical	64	8	0	4	16	4
	Horizontal	64	8	0	4	16	4
	DC	64	0	0	1	16	1
	Diagonal down left	64	28	0	5	16	16
	Diagonal down right	64	32	0	5	16	16
	Vertical right	64	36	0	6	16	16
	Horizontal down	64	36	0	6	16	16
	Vertical left	64	36	0	6	16	16
	Horizontal up	64	25	0	6	16	16
	Total	576	209	0	43	144	105
Intra 16x16	Vertical	1088	40	16	36	256	52
	Horizontal	1088	40	16	36	256	52
	DC	1088	0	16	17	256	1
	Plane	1088	3	16	26	256	69
	Current block HT	0	1088	0	0	0	0
	Total	4352	1171	64	115	1024	174
Intra 8x8	Vertical	256	16	0	8	64	16
	Horizontal	256	16	0	8	64	16
	DC	256	0	0	4	64	4
	Plane	256	7	0	9	64	20
	Current block HT	0	256	0	0	0	0
	Total	1024	295	0	29	256	56

Table 5.10 Average PSNR (dB) Comparison of Original SATD Mode Decision and SATD Mode Decision with Proposed Technique

VIDEO FRAME	Original	Proposed	Difference
MOBILE	30.638	30.634	-0.004
MOTHER&DAUGHTER	36.804	36.815	0.011
FOREMAN	35.279	35.303	0.024
FOOTBALL	31.980	31.980	0
SOCCER	32.461	32.448	-0.013
AKIYO	37.253	37.226	-0.027

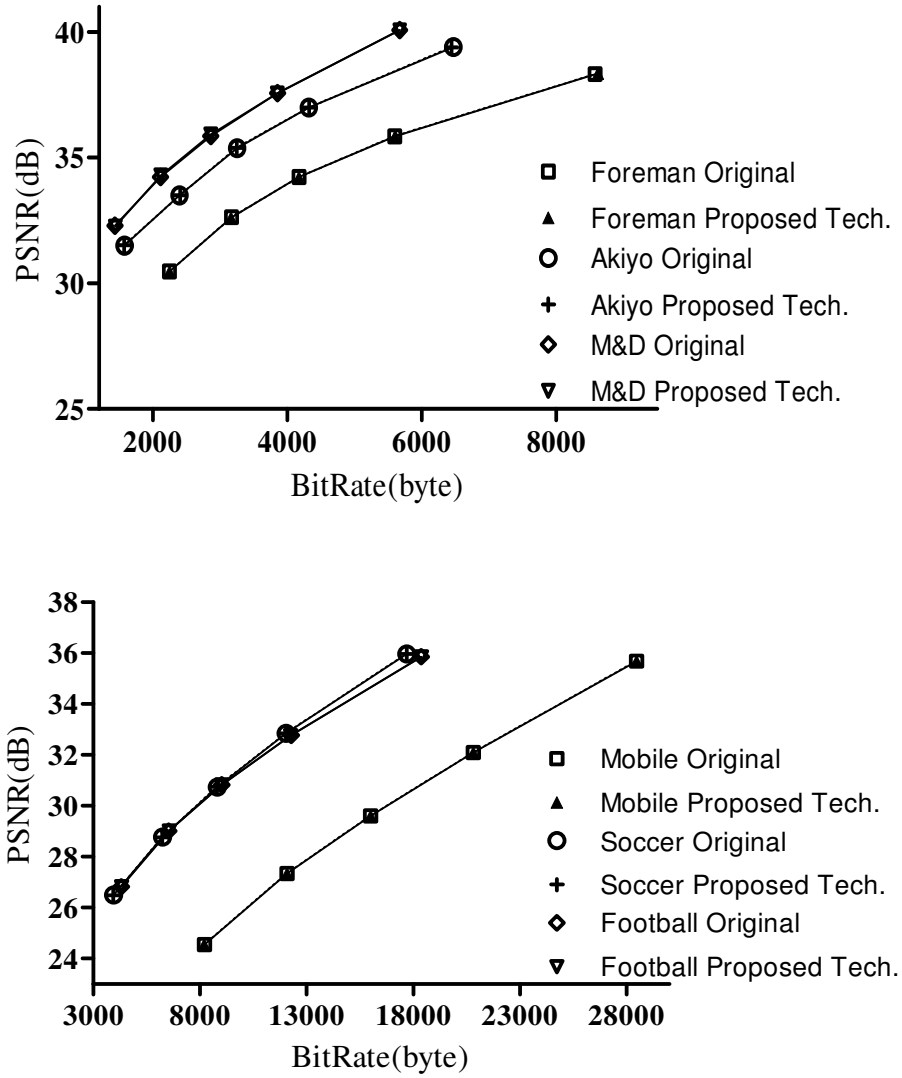


Figure 5.7 Rate Distortion Curves of Original SATD Mode Decision and SATD Mode Decision with Proposed Technique

5.5 Proposed 16x16 Intra Mode Decision Hardware Architectures

We designed two different hardware architectures for H.264 16x16 intra mode decision. The first hardware architecture, shown in Figure 5.8, implements the original SATD intra mode decision algorithm used in H.264 JM software encoder. The second

hardware architecture, shown in Figure 5.9, includes the proposed computational complexity and power reduction technique.

H.264 16x16 intra mode decision hardware consists of two parts; the first part generates predicted blocks by each prediction mode in parallel and the second part calculates SATD cost for each prediction mode using the predicted blocks. The main differences between two architectures are the residue operation and simplification of HT because of fixed prediction block pattern of each intra mode. The first hardware architecture first performs the residue operation and then performs HT. The second hardware architecture, on the other hand, first performs HT and then performs residue operation.

As shown in Figure 5.8, three local buffers are used to store the inputs to intra mode decision hardware; 352x8 top neighboring buffer, 16x8 left neighboring buffer and 256x8 current block buffer. Horizontal predicted block (16x8), vertical predicted block (16x8), DC predicted block (1x8), and plane predicted block (256x8) are used to store the predicted blocks by the corresponding intra prediction modes. Residue block (256x8) is used to store the difference between the current MB and the predicted MB. Top neighboring buffer, plane predicted block, current block, and residue block are implemented as Block SelectRAMs, and other buffers are implemented as Distributed SelectRAMs.

When a new MB arrives, the intra prediction module starts to calculate prediction values for each mode in parallel. After intra prediction is finished, the mode decision hardware starts to process each mode by subtracting predicted block from current block. HT is applied to residue block and absolute values of resulting AC coefficients are added. Then, HT is applied to the 4x4 DC block formed by DC coefficients and the resulting coefficients are added. HT module in Figure 5.8 implements the fast HT algorithm described in section 5.1.

As shown in Figure 5.9, the proposed hardware for SATD intra mode decision with proposed energy reduction technique has the same intra prediction search hardware except that the size of the buffer used for storing the predicted block by plane mode is reduced from 256x8 to 3x8. Since this hardware calculates the SATD cost using only a, b, c parameters, it stores only a, b, c parameters.

After intra prediction, HT is applied to predicted blocks by each prediction mode. Since HT of horizontal, vertical, DC and plane prediction modes simplify significantly using the proposed technique, HT module in Figure 5.9 is much simpler than HT module in Figure 5.8. After HT, AC coefficients of predicted blocks by each prediction mode are subtracted from corresponding AC coefficients of transformed current block. HT is applied to DC coefficients of both current block and predicted blocks by each prediction mode again. Then, DC coefficients of predicted blocks by each prediction mode are subtracted from corresponding DC coefficients of current block. Finally, absolute values of AC and DC coefficient differences are added to find SATD cost.

5.6 Energy Consumption Analysis

The energy consumption of 16x16 intra mode decision hardware including proposed technique on a Xilinx Virtex II FPGA is estimated using Xilinx XPower tool. In order to estimate its energy consumption, timing simulation of the placed and routed netlist of 16x16 intra mode decision hardware is done using Mentor Graphics ModelSim SE. Foreman, Akiyo and Mother&Daughter frames are used as inputs for timing simulations and the signal activities are stored in VCD files. These VCD files are used for estimating the energy consumption of 16x16 intra mode decision hardware using Xilinx XPower tool.

The energy consumption of 16x16 intra mode decision hardware implementation on a Xilinx Virtex II FPGA at 25 MHz is shown in Table 5.11 for different video frames. As shown in the table, the proposed energy reduction technique reduces the energy consumption of 16x16 intra mode decision hardware up to 59.6%.

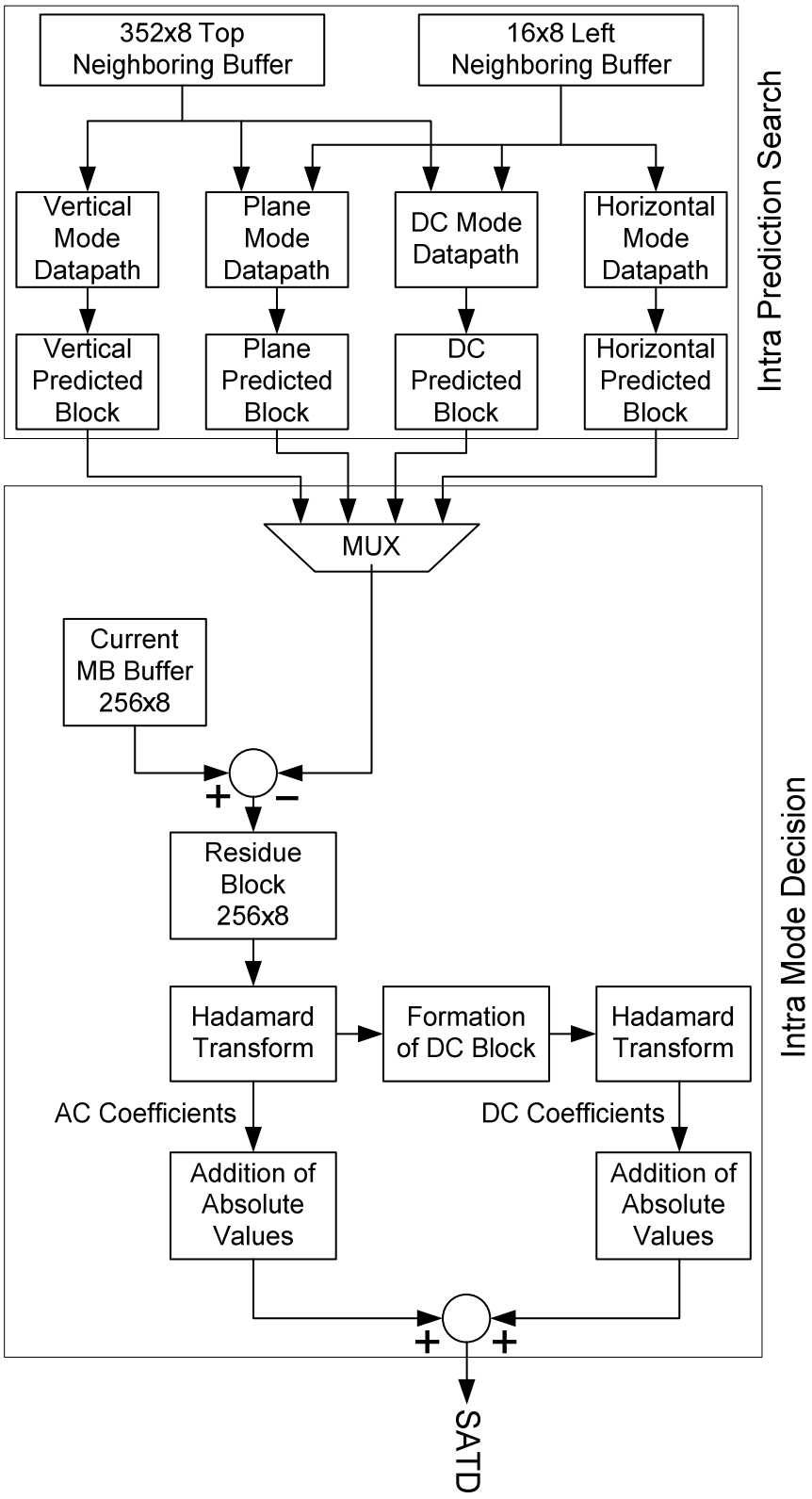


Figure 5.8 Proposed Hardware for Original Intra 16x16 Mode Decision

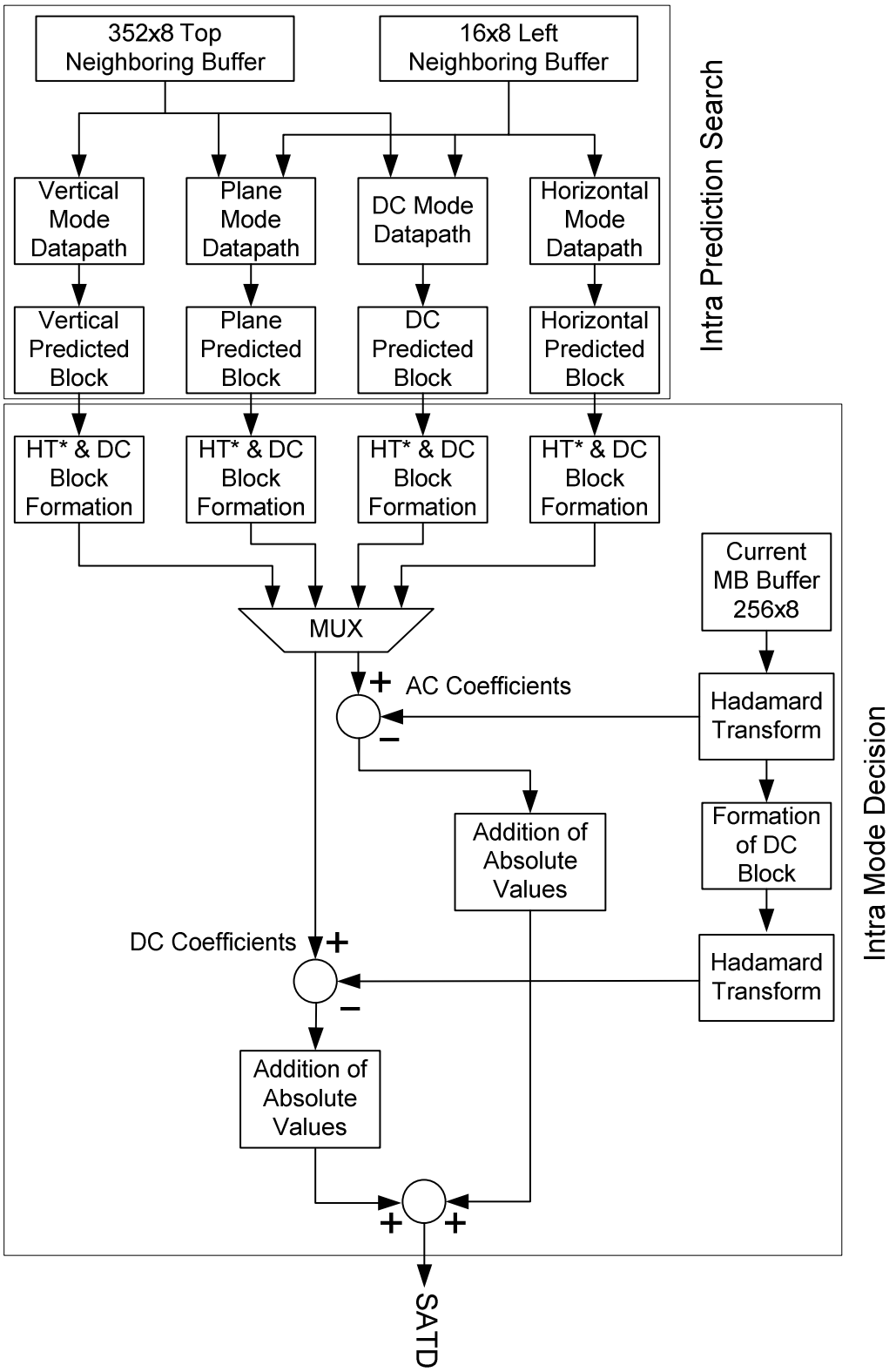


Figure 5.9 Proposed Hardware for Intra 16x16 Mode Decision with Proposed Technique

Table 5.11 Energy Consumption Reduction

		Computation Time(μs)			Power (mW)		Energy (μJ)		
	Q	***Org.***	***Prop.***	***Δt***	***Org.***	***Prop***	***Org.***	***Prop.***	***%***
FM	***28***	15351	6332	9019	114.2	111.8	1753	708	**59.6**
	35	15351	6332	9019	105.5	104.1	1620	659	**59.3**
	42	15351	6332	9019	99.4	98.7	1526	625	**59.0**
AK	***28***	15351	6332	9019	101.7	102	1561	646	**58.6**
	35	15351	6332	9019	96.4	98.1	1480	621	**58.0**
	42	15351	6332	9019	92.8	94.9	1425	601	**57.8**
M&D	***28***	15351	6332	9019	105	104.7	1612	663	**58.9**
	35	15351	6332	9019	96.6	97.7	1483	619	**58.3**
	42	15351	6332	9019	90.9	93.5	1395	592	**57.6**

CHAPTER VI

A NOVEL ENERGY REDUCTION TECHNIQUE FOR INTRA PREDICTION WITH TEMPLATE MATCHING

H.264 intra prediction algorithm predicts the pixels in a MB using the pixels in the available neighboring blocks. A 4x4 luma block consisting of the pixels a to p is shown in Fig. 6.1. The pixels A to M belong to the neighboring blocks and are assumed to be already encoded and reconstructed, and are therefore available in the encoder and decoder to generate a prediction for the current MB.

For the luminance (luma) component of a MB, a 16x16 predicted luma block is formed by performing intra predictions for each 4x4 luma block in the MB and by performing intra prediction for the 16x16 MB. There are 9 prediction modes for each 4x4 luma block and 4 prediction modes for a 16x16 luma block. 4x4 prediction modes are generally selected for highly textured regions while 16x16 prediction modes are selected for flat regions.

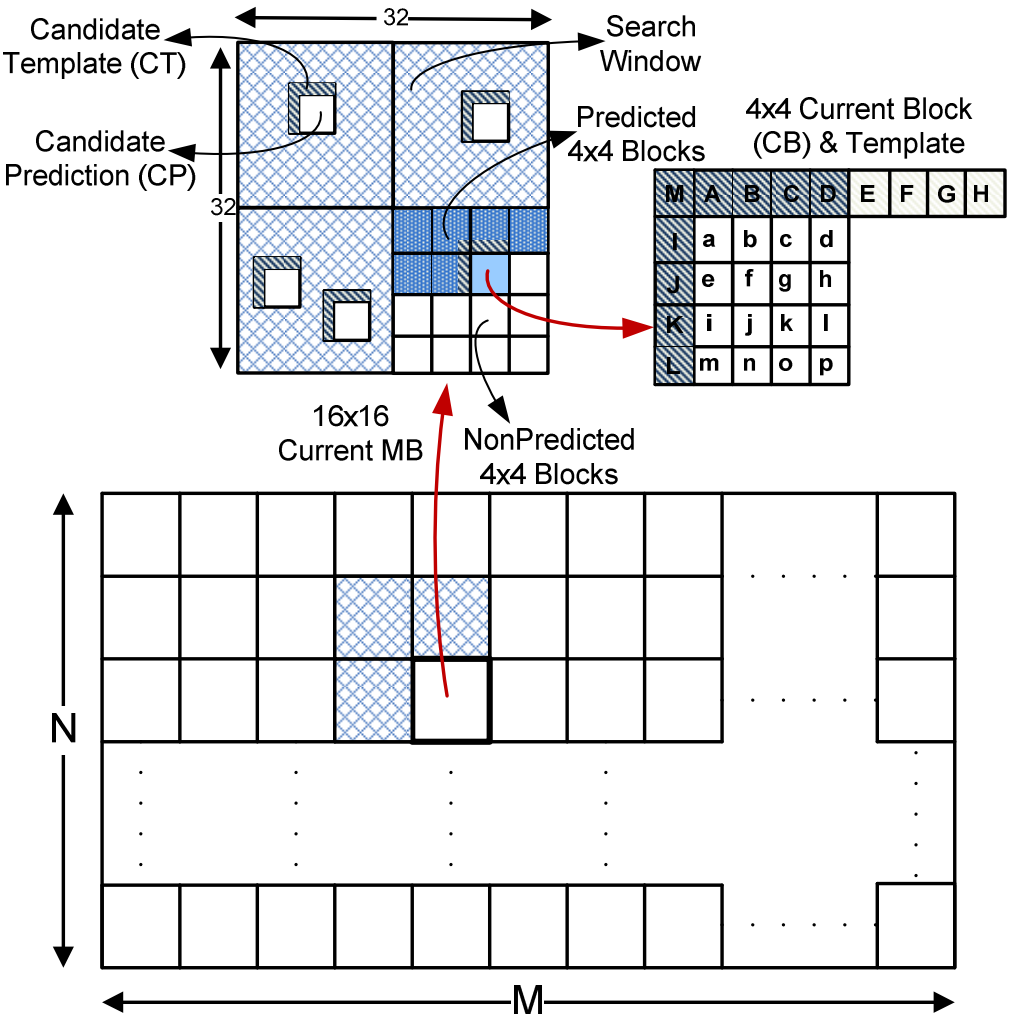


Figure 6.1 Intra Prediction with Template Matching

H.264 intra prediction algorithm has better compression efficiency than the intra prediction algorithms used in previous video compression standards. However, it is not well suited for processing complex textures at low bit rates. Therefore, several new intra prediction algorithms such as bi-directional intra prediction [44] and intra prediction with Template Matching (TM) [45, 46, 47] are proposed to improve H.264 intra prediction. Template matching is used for performing image-based texture synthesis [48], where the current pixel to be synthesized is generated by looking at a neighbourhood of pixels that are already synthesized. It is later proposed for performing intra prediction.

Fig. 6.1 illustrates intra prediction with TM. A template is formed by a group of neighboring pixels on the top and to the left of the current 4x4 block (A, B, C, D, I, J, K, L, M), and best matching candidate template (CT) is searched in the reconstructed search window (SW) in the current frame based on minimum SAD criterion. The 4x4 candidate prediction (CP) block adjacent to the best matching CT is assigned as the TM prediction for the current block.

Intra prediction with TM has high computational complexity. Therefore, in this thesis, we propose a novel technique for reducing the amount of computations performed by intra prediction with TM, and therefore reducing the energy consumption of intra prediction with TM hardware. This technique reduces the amount of computations by reducing the amount of template search operations. For a 4x4 current block, the proposed technique first calculates the predictions for 9 H.264 4x4 intra prediction modes and their Sum of Absolute Difference (SAD) values. It, then, determines the minimum SAD value among these 9 SAD values. If the minimum SAD value is less than a pre-defined SAD threshold (Th_{SAD}), it selects the prediction with minimum SAD value as the intra prediction of the 4x4 current block.

If the minimum SAD value is larger than Th_{SAD}, then it performs TM search for the current block. In order to increase the compression efficiency of intra prediction with TM, N best matching CTs are saved while TM search is performed in the already coded and reconstructed search window. After TM search is performed, SAD values of the CPs for these N best matching CTs are calculated. If the SAD value of a CP is less than the minimum SAD value of 9 H.264 4x4 intra prediction modes, it is selected as the intra prediction of the 4x4 current block.

The simulation results for several video sequences reconstructed by H.264 reference software, JM 14.0 [17], showed that the proposed technique reduces the amount of template search operations up to 41% with a small comparison overhead. For each 4x4 block, the proposed technique requires one comparison with minimum SAD value of 9 H.264 4x4 intra prediction modes. The simulation results also showed that the proposed technique does not change the PSNR for some video frames, but it decreases the PSNR slightly for some video frames.

We also designed a high performance 4x4 intra prediction with TM hardware including the proposed technique. The proposed hardware is implemented in Verilog HDL. Verilog RTL code is mapped to a Xilinx Virtex 6 FPGA. The FPGA implementation is verified to work at 150 MHz with post place & route simulations. It is capable of processing 53 HD (1280x720) frames per second. The proposed technique reduced the energy consumption of this hardware on this FPGA up to 50%.

Several hardware architectures for H.264 4x4 intra prediction algorithm are proposed in the literature [8, 9, 24, 49]. However, a hardware architecture for 4x4 intra prediction with TM is not reported in the literature.

6.1 Proposed Computation and Energy Reduction Technique

For a 4x4 current block, the proposed technique first calculates the predictions for 9 H.264 4x4 intra prediction modes. For each 4x4 intra prediction mode, it generates 16 predicted pixels using some or all of the neighboring pixels. Then, it calculates the SAD values for each 4x4 intra prediction mode by substracting their predicted pixels from the 4x4 current block pixels. It compares these SAD values and determines the intra prediction mode that has the minimum SAD value. If the minimum SAD value is less than the pre-defined Th_{SAD}, it does not perform TM search for this 4x4 current block, and it selects the prediction with minimum SAD value as the intra prediction of the 4x4 current block.

If the minimum SAD value is larger than Th_{SAD}, it performs TM search for the current block. In order to increase the compression efficiency of intra prediction with TM, N best matching CTs are saved while TM search is performed in the already coded and reconstructed search window. After TM search is performed, SAD values of the CPs for these N best matching CTs are calculated. If the SAD value of a CP is less than the minimum SAD value of 9 H.264 4x4 intra prediction modes, it is selected as the intra prediction of the 4x4 current block.

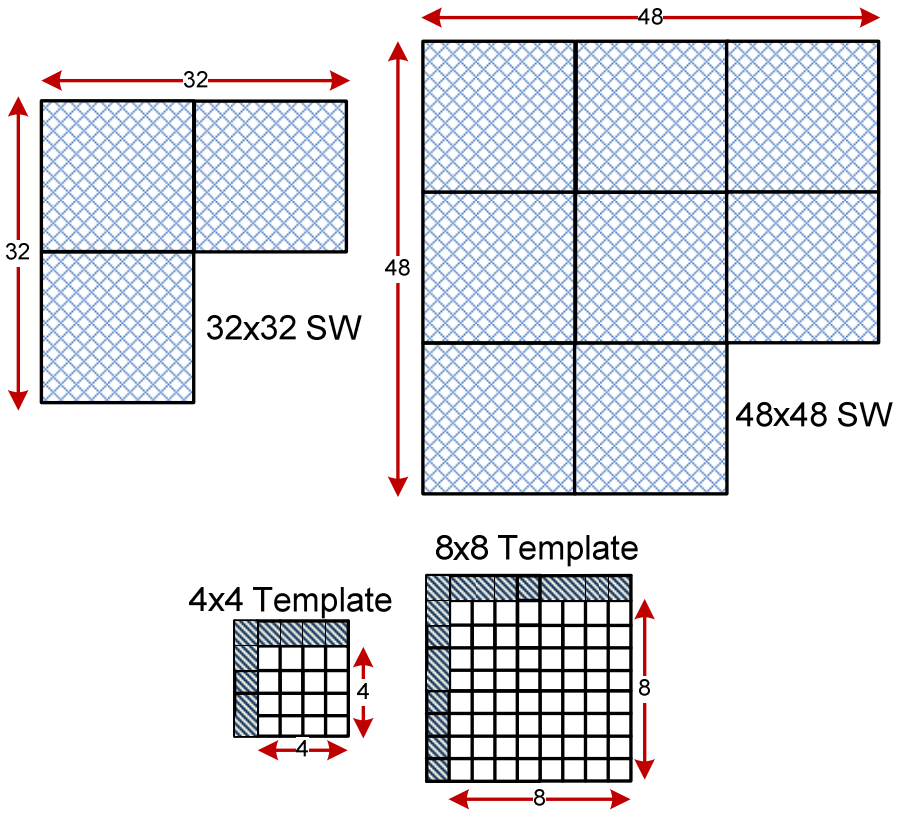


Figure 6.2 Different Size Templates and Search Windows

Table 6.1 PSNR Results (dB) of Different Size SWs and Templates

Frame	Size	32x32		48x48		Diff.	Diff.
		4x4	8x8	4x4	8x8	4x4	8x8
ParkRun	**1280x720**	21.28	**20.06**	21.44	**20.21**	0.16	**0.15**
NMobCal	**1280x720**	24.45	**23.28**	24.69	**23.50**	0.24	**0.22**
T.Tennis	**704x480**	27.21	**26.09**	26.94	**25.94**	-0.27	**-0.15**
Susie	**704x480**	29.32	**26.86**	29.38	**26.12**	0.06	**-0.74**
Flower	**704x480**	24.07	**22.56**	24.09	**22.40**	0.02	**-0.16**
Akiyo	**352x288**	31.34	**28.96**	31.37	**28.71**	0.03	**-0.25**
MD	**352x288**	33.79	**31.67**	34.13	**31.11**	0.34	**-0.56**
Container	**352x288**	23.13	**21.92**	22.85	**21.26**	-0.28	**-0.66**

We determined the PSNR results obtained by using the different size templates and SWs shown in Fig. 6.2 during TM search for several HD (1280x720), VGA (704x480) and CIF (352x288) size video frames (one frame from each video sequence) reconstructed by H.264 reference software JM 14.0. 32x32 SW has 528 4x4 CTs and 320 8x8 CTs, and 48x48 SW has 1680 4x4 CTs and 1344 8x8 CTs. As shown in Table 6.1, the simulation results show that using 4x4 template gives better PSNR results than using 8x8 template in both 32x32 and 48x48 SWs. The simulation results also show that, although 48x48 SW has more CTs than 32x32 SW, using 4x4 and 8x8 templates in a 48x48 SW gives similar PSNR results with using 4x4 and 8x8 templates in a 32x32 SW. Therefore, we decided using 4x4 template in a 32x32 SW.

As shown in Table 6.2, the simulation results for several video frames show that better PSNR results are obtained with larger N values. However, computational complexity significantly increases with larger N values.

In order to achieve more computation reduction with less PSNR loss, Th_{SAD} should be determined based on the following criteria.

a) Th_{SAD} should separate the SAD values of H.264 4x4 intra predictions and TM predictions (CPs) as much as possible.

b) The number of TM predictions (CPs) selected by intra mode decision when Th_{SAD} is not used and when Th_{SAD} is used should be the same as much as possible.

We analyzed 9 HD (1280x720), VGA (704x480) and CIF (352x288) size video frames (one frame from each video sequence) to determine Th_{SAD}. The SAD values of H.264 4x4 intra prediction modes selected by intra mode decision are distributed in the [0, 120] interval with the mean, $\mu = 65$, and standard deviation, $\sigma = 39$. The SAD values of TM predictions selected by intra mode decision are distributed in the [40, 170] interval with the mean, $\mu = 85$, and standard deviation, $\sigma = 30$.

Table 6.3 shows the total number of 4x4 blocks in a frame, the number of TM predictions selected by intra mode decision when Th_{SAD} is not used, and the number of TM predictions selected by intra mode decision when Th_{SAD} is used. The average PSNR comparison of the H.264 4x4 intra prediction algorithm and the 4x4 intra prediction algorithm with TM including the proposed technique for several HD, VGA and CIF size video frames (one frame from each video sequence) for 4 CTs are shown in Table 6.4 and 6.5. The proposed technique does not change the PSNR for some video frames, but it decreases the PSNR slightly for some video frames. The average PSNR values shown in Tables 6.1, 6.2, 6.4 and 6.5 are calculated using the technique described in [26].

As shown in Tables 6.4 and 6.5, using a large Th_{SAD} value achieves more computation reduction, but it causes more PSNR loss. Using a small Th_{SAD} value causes less PSNR loss, but it achieves less computation reduction.

We calculated the computation reduction achieved by the proposed technique for intra prediction algorithm with TM using for several HD, VGA and CIF size video frames (one frame from each video sequence) for 4 CTs are shown in Table 6.6. As shown in the Table 6.6, the amount of reductions achieved in addition and comparison operations ranges from 7% to 56%. In this table, the column Total Block shows the total number of blocks TM search is done.

Table 6.2 PSNR Results (dB) of Intra Prediction with TM

	Frame	Intra	Intra w TM (1 CT)		Intra w TM (2 CTs)		Intra w TM (4 CTs)	
		PSNR	PSNR	Diff.	PSNR	Diff.	PSNR	Diff.
HD	**ParkRun**	21.28	21.43	**0.15**	21.56	**0.28**	21.64	**0.36**
	NMobCal	24.45	24.77	**0.32**	24.94	**0.49**	25.01	**0.56**
	Ducks	27.57	27.87	**0.30**	28.02	**0.44**	28.11	**0.54**
VGA	**T.Tennis**	27.21	27.66	**0.46**	27.98	**0.77**	27.95	**0.75**
	MobCal	35.00	35.53	**0.54**	35.66	**0.66**	35.76	**0.77**
	Flag	34.04	34.49	**0.45**	34.60	**0.56**	34.66	**0.62**
CIF	**Akiyo**	31.34	31.71	**0.38**	31.92	**0.59**	31.87	**0.53**
	MD	33.79	33.96	**0.16**	34.01	**0.22**	34.08	**0.29**
	Container	23.13	23.26	**0.12**	23.39	**0.26**	23.38	**0.26**

Table 6.3 Number of TM Predictions Selected when Th_{SAD} Used

Frame	Total block	TM pred.		Th_{SAD} = 40	Th_{SAD} = 50	Th_{SAD} = 60
Ducks	**58832**	**6517**	#	6441	6221	5929
			%	**98.83**	**95.46**	**90.98**
ParkRun	**58832**	**4573**	#	4248	4236	4214
			%	**92.89**	**92.63**	**92.15**
TableTennis	**21932**	**2779**	#	2743	2672	2547
			%	**98.7**	**96.15**	**91.65**
Container	**6320**	**662**	#	538	525	499
			%	**81.27**	**79.31**	**75.38**

Table 6.4 Average PSNR (dB) Comparison of the Proposed Technique

Frame	Intra	Intra w TM (4 CTs)		Intra w TM (4 CTs) Th_{SAD} = 40		Intra w TM (4 CTs) Th_{SAD} = 50		Intra w TM (4 CTs) Th_{SAD} = 60	
	PSNR	**PSNR**	**Diff.**	**PSNR**	**Diff.**	**PSNR**	**Diff.**	**PSNR**	**Diff.**
ParkRun	21.28	21.64	**0.36**	21.64	**0.36**	21.64	**0.36**	21.64	**0.36**
NewMobCal	24.45	25.01	**0.56**	25.01	**0.56**	25.01	**0.56**	25.01	**0.56**
Ducks	27.57	28.11	**0.54**	28.11	**0.54**	28.11	**0.54**	28.11	**0.54**
T.Tennis	27.21	27.95	**0.75**	27.95	**0.75**	27.95	**0.75**	27.95	**0.75**
MobCal	35.00	35.76	**0.77**	35.76	**0.76**	35.76	**0.76**	35.75	**0.76**
Flag	34.04	34.66	**0.62**	34.66	**0.62**	34.65	**0.61**	34.64	**0.60**
Akiyo	31.34	31.87	**0.53**	31.86	**0.52**	31.85	**0.52**	31.85	**0.52**
MD	33.79	34.08	**0.29**	34.07	**0.28**	34.06	**0.27**	34.05	**0.26**
Container	23.13	23.38	**0.25**	23.38	**0.25**	23.38	**0.25**	23.38	**0.25**

Table 6.5 Average PSNR (dB) Comparison of the Proposed Technique for Higher Th_{SAD}

Frame	Intra	Intra w TM (4 CTs)		Intra w TM (4 CTs) Th_{SAD} = 70		Intra w TM (4 CTs) Th_{SAD} = 90		Intra w TM (4 CTs) Th_{SAD} = 110	
	PSNR	**PSNR**	**Diff.**	**PSNR**	**Diff.**	**PSNR**	**Diff.**	**PSNR**	**Diff.**
ParkRun	21.28	21.64	**0.36**	21.56	**0.28**	21.40	**0.12**	21.34	**0.06**
NewMobCal	24.45	25.01	**0.56**	24.86	**0.41**	24.69	**0.24**	24.56	**0.11**
Ducks	27.57	28.11	**0.54**	28.01	**0.44**	27.79	**0.22**	27.67	**0.10**
T.Tennis	27.21	27.95	**0.75**	27.93	**0.72**	27.79	**0.58**	27.35	**0.14**
MobCal	35.00	35.76	**0.77**	35.65	**0.66**	35.47	**0.47**	35.16	**0.16**
Flag	34.04	34.66	**0.62**	34.54	**0.50**	34.31	**0.27**	34.09	**0.05**
Akiyo	31.34	31.87	**0.53**	31.75	**0.41**	31.63	**0.29**	31.42	**0.08**
MD	33.79	34.08	**0.29**	34.01	**0.22**	33.89	**0.10**	33.81	**0.02**
Container	23.13	23.38	**0.25**	23.30	**0.17**	23.21	**0.08**	23.13	**0.00**

Table 6.6 Computation Reduction in Intra Prediction with TM Algorithm for Different Th_{SAD} values

	Original			Th_{SAD} = 40			Th_{SAD} = 50			Th_{SAD} = 60		
Frame	**Total Block**	**# Total Operation**		**Total Block**	**# Operation**	**%**	**Total Block**	**# Operation**	**%**	**Total Block**	**# Operation**	**%**
Ducks	**58832**	**# Add/Sub**	528076032	**54770**	36460512	**7%**	**50848**	71664384	**14%**	**46670**	109166112	**21%**
		# Comp.	31063296		2144736	**7%**		4215552	**14%**		6421536	**21%**
ParkRun	**58832**	**# Add/Sub**	528076032	**56682**	19298400	**4%**	**56119**	24351888	**50%**	**54620**	37806912	**70%**
		# Comp.	31063296		1135200	**4%**		1432464	**50%**		2223936	**70%**
Table Tennis	**21932**	**# Add/Sub**	196861632	**19846**	18723936	**10%**	**17520**	39602112	**20%**	**15133**	61027824	**31%**
		# Comp.	11580096		1101408	**10%**		2329536	**20%**		3589872	**31%**
Container	**6320**	**# Add/Sub**	56728320	**4320**	17952000	**32%**	**3424**	25994496	**46%**	**2772**	31846848	**56%**
		# Comp.	3336960		1056000	**32%**		1529088	**46%**		1873344	**56%**

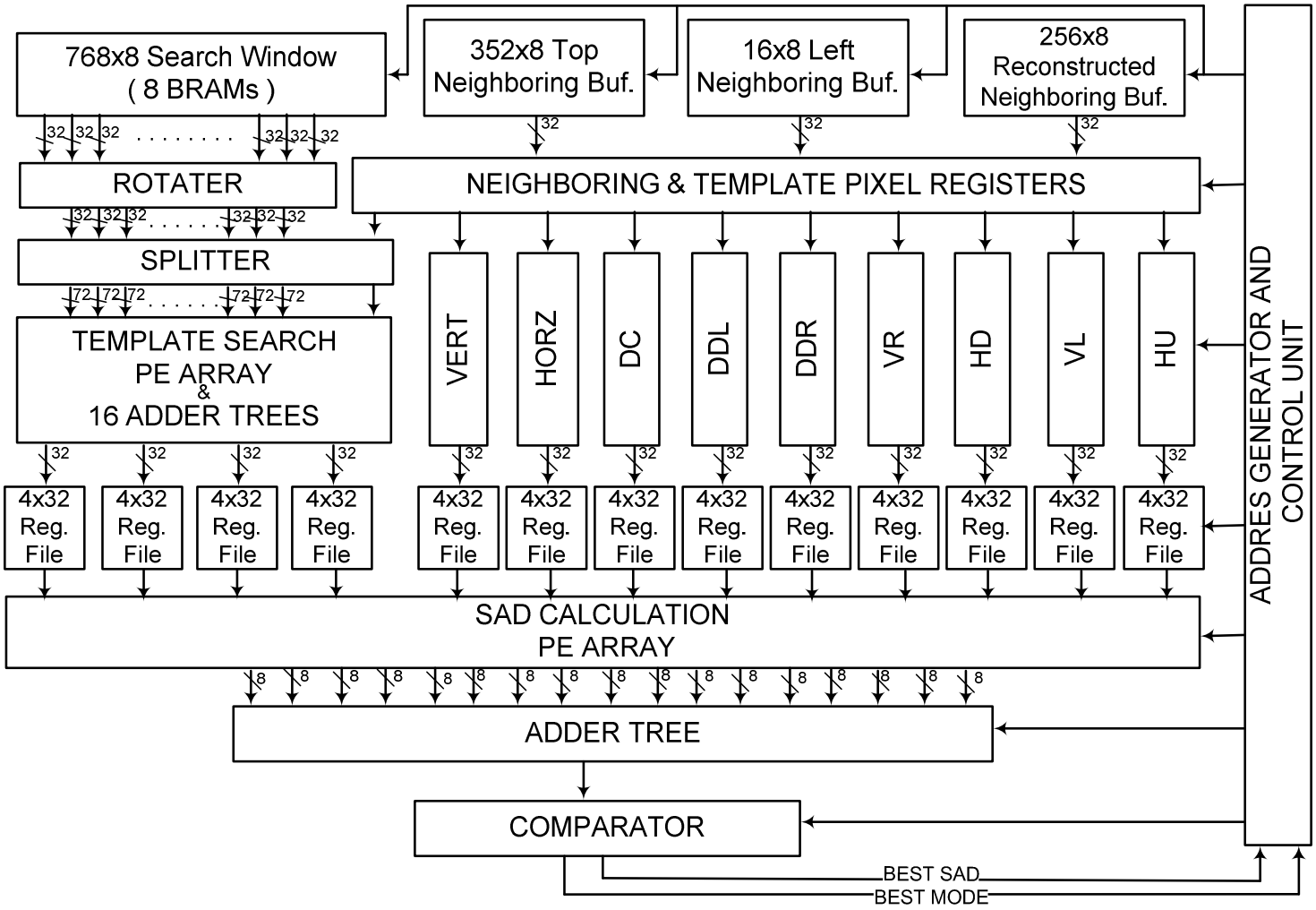


Figure 6.3 Top Level Block Diagram of Proposed 4x4 Intra Prediction with Template Matching Hardware

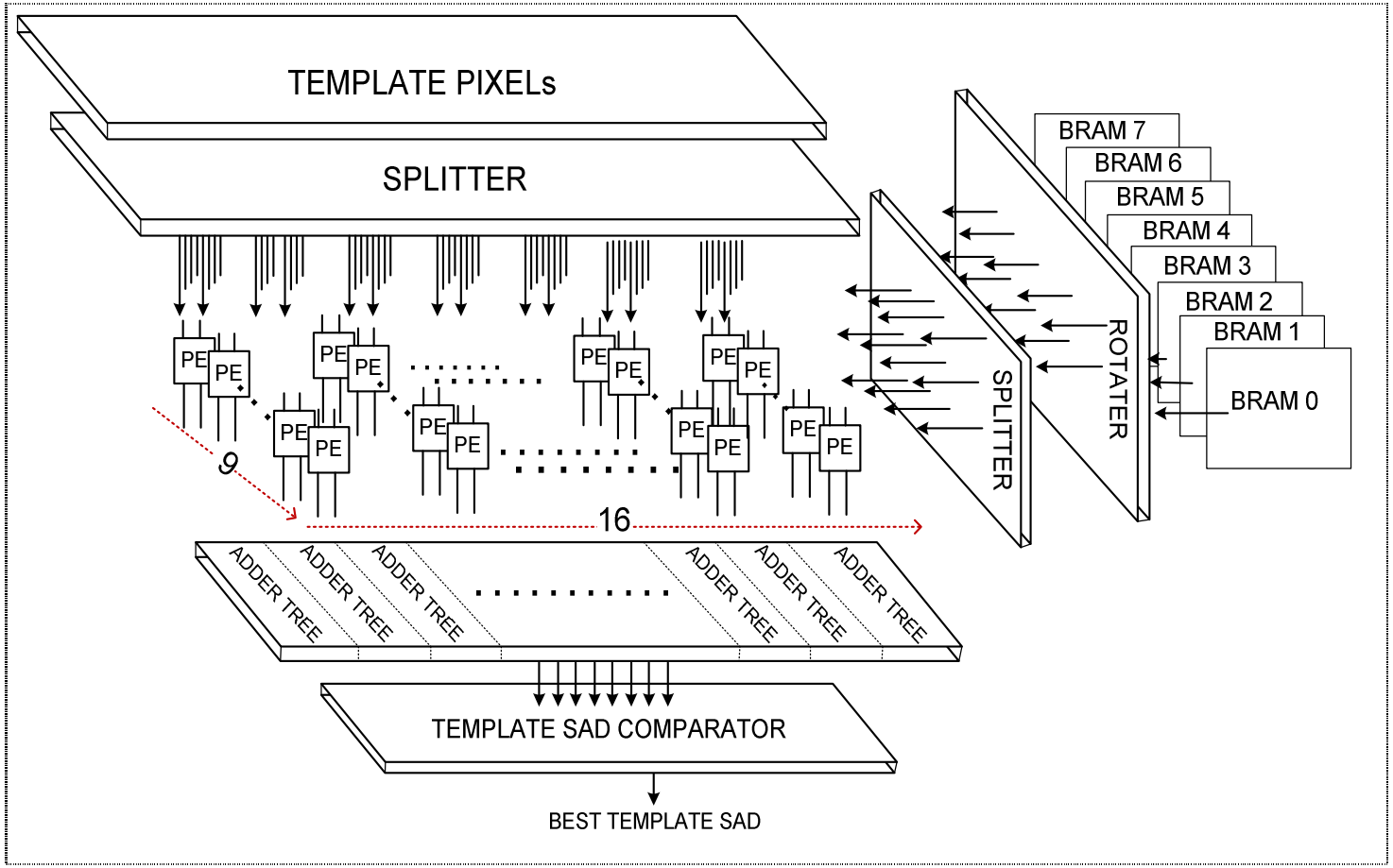


Figure 6.4 Template Search PE Array and 16 Adder Tree

6.2 Proposed Intra Prediction with Template Matching Hardware

Top level block diagram of the proposed 4x4 intra prediction with TM hardware is shown in Fig. 6.3. Top, left and reconstructed local neighboring buffers are used to store the neighboring pixels in the previously coded and reconstructed neighboring blocks. Search Window memory is used to store the pixels in the previously coded and reconstructed search window in the current frame. These on-chip memories reduce the required off-chip memory bandwidth.

For a current 4x4 block, the proposed hardware calculates the predictions for 9 H.264 4x4 intra prediction modes using the neighboring pixels, and their SAD values. Depending on the minimum SAD value of H.264 intra prediction modes, it performs TM search to find the best matching 4 CTs in a [-32, 32] pixel search window. If TM search is not performed, the H.264 4x4 intra prediction mode which has the minimum SAD value is selected as the best prediction mode. If TM search is performed, then, it compares the SAD values of 4 CPs with the minimum SAD value of H.264 intra prediction modes. The prediction which has the minimum SAD value is selected as the best prediction mode.

6.2.1 PE Array Architectures

6.2.1.1 Template Search PE Array (TSPEA)

The architecture of Template Search PE Array (TSPEA) is shown in Fig. 6.4. TSPEA latency is 8 clock cycles; 1 cycle for Addres Generator and Control Unit, 1 cycle for synchronous read from memory and Rotator, 2 cycles for Adder Tree and 4 cycles for Comparator. Control Unit generates the required address and control signals for TSPEA to calculate the SAD values of CTs in 32x32 SW.

There are 9x16 = 144 PEs in the TSPEA. The architecture of a PE is shown in Fig. 6.5. Each PE is composed of a comparator, two 2x1 multiplexers and an 8-bit subtractor. The comparator and the multiplexers are used to send the larger pixel to the first input of the subtractor and the smaller pixel to the second input of the subtractor. This ensures that the result of the subtractor is always positive.

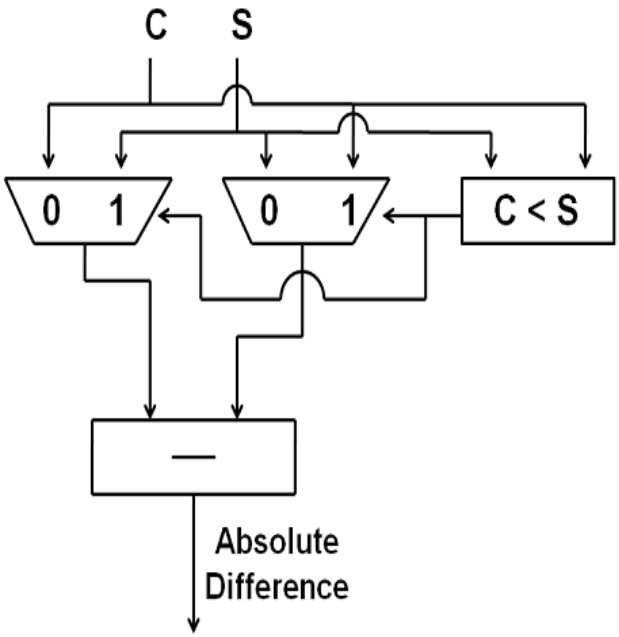


Figure 6.5 PE Architecture

144 PEs calculate the absolute differences in one clock cycle for 16 different CTs. The outputs of 9 parallel PEs are connected to an adder tree which calculates sum of absolute differences. The adder tree has 2 pipeline stages for faster operation. Even though the SAD calculation for a block takes 2 clock cycles, after the first SAD calculation, the throughput is 1 SAD calculation per clock cycle. The 32x32 SW has 528 CTs, and TSPEA calculates the SAD values of 16 different CTs in 1 cycle. Therefore, calculation of SAD values of 528 CTs takes 34 (528/16 + 1 = 34) clock cycles.

After PE array calculates the SAD values of 16 CTs, Comparator compares these 16 SAD values and determines the CT that has the minimum SAD value and the corresponding CP. Then, PE array calculates the SAD values of the next 16 CTs in the same 32x32 SW. The pixels of 16 CTs needed for calculating the SAD values of these first 16 CTs in SW are loaded from BRAMs into PE arrays. Data alignment for the PEs is achieved by using a Rotator and 2 Splitters. Pixels in 16 different CTs and pixels in the current template are connected to the 144 PEs by 2 Splitters.

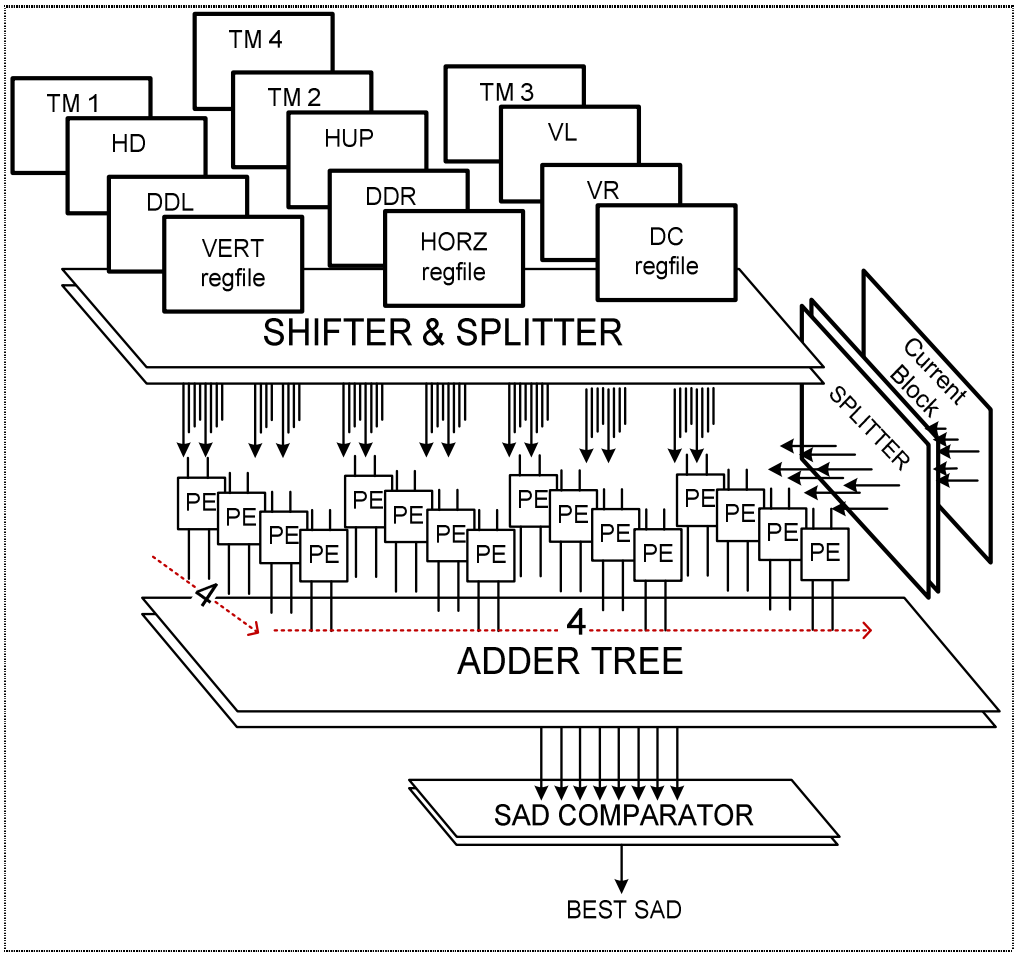


Figure 6.6 SAD Calculation PE Array and Adder Tree

6.2.1.2 SAD Calculation PE Array (SCPEA)

The architecture of SAD Calculation PE Array (SCPEA) is shown in Fig. 6.6. SCPEA latency is 7 clock cycles; 1 cycle for Addres Generator and Control Unit, 1 cycle for synchronous read from memory, 4 cycles for Adder Tree and 1 cycles for Comparator. Control Unit also generates the required address and control signals to calculate the SAD values of 9 H.264 4x4 intra predictions and 4 CPs. After the SAD values are calculated, Comparator compares them and determines the prediction mode that has the minimum SAD value. There are 4x4 = 16 PEs in the SCPEA. The architecture of a PE is shown in Fig. 6.5.

16 PEs calculate the absolute differences between the 4x4 current block and a 4x4 block predicted by a prediction mode in one clock cycle. The outputs of the 16 PEs are connected to an adder tree. The adder tree has 4 pipeline stages for faster operation. Even though the SAD calculation for a prediction mode takes 4 clock cycles, after the first SAD calculation, the throughput is 1 SAD calculation per clock cycle. Therefore, calculating the SAD values of 9 H.264 4x4 intra prediction modes takes 14 cycles.

After PE array calculates the SAD value of an intra prediction mode, it calculates the SAD value of the next intra prediction mode. 16 pixels needed for calculating the SAD value of a 4x4 block predicted by a prediction mode are loaded from 4x32 register files into PE arrays. Data alignment for the PEs is achieved by using a Shifter and 2 Splitters.

6.2.2 Memory Organization and Data Alignment

The memory organization of 32x32 Search Window (SW) is shown in Fig. 6.7. Horizontally and vertically adjacent pixels of 32x32 SW can be read with one clock cycle latency using the proposed ladder-shaped SW data organization. Each address of a BRAM contains four pixels. 8 dual-port BRAMs in the FPGA are used to store the 32x32 SW. In Fig. 6.7, the numbers show the BRAM addresses containing the corresponding pixels in SW. The control overhead of address signals used for reading from BRAMs and the control overhead of the Rotator, the Splitter and the multiplexers in the PEs are reduced by symmetric arrangement 32x32 SW pixels in the BRAMs.

6.2.3 Implementation Results

The proposed intra prediction with TM hardware architecture is implemented in Verilog HDL. The Verilog RTL code is synthesized and mapped to a Xilinx XC6VLX550T-2FF1760 FPGA using Xilinx ISE 11.5. The hardware implementation is verified with post place & route simulations using Mentor Graphics ModelSim SE. The FPGA implementation consumes 10,678 LUTs and 5,291 DFFs. In addition, 6144 bits on-chip memory is used for storing 32x32 SW. These 6144 bits are stored in 8 dual port BRAMs. 6656 bits on-chip memory is used for storing 352x8 top neighboring buffer, 16x8 left neighboring buffer, 256x8 reconstructed neighboring buffer and 13 Register Files. These 6656 bits are stored in 2 RAMB36E1 and 16 RAMB18E1.

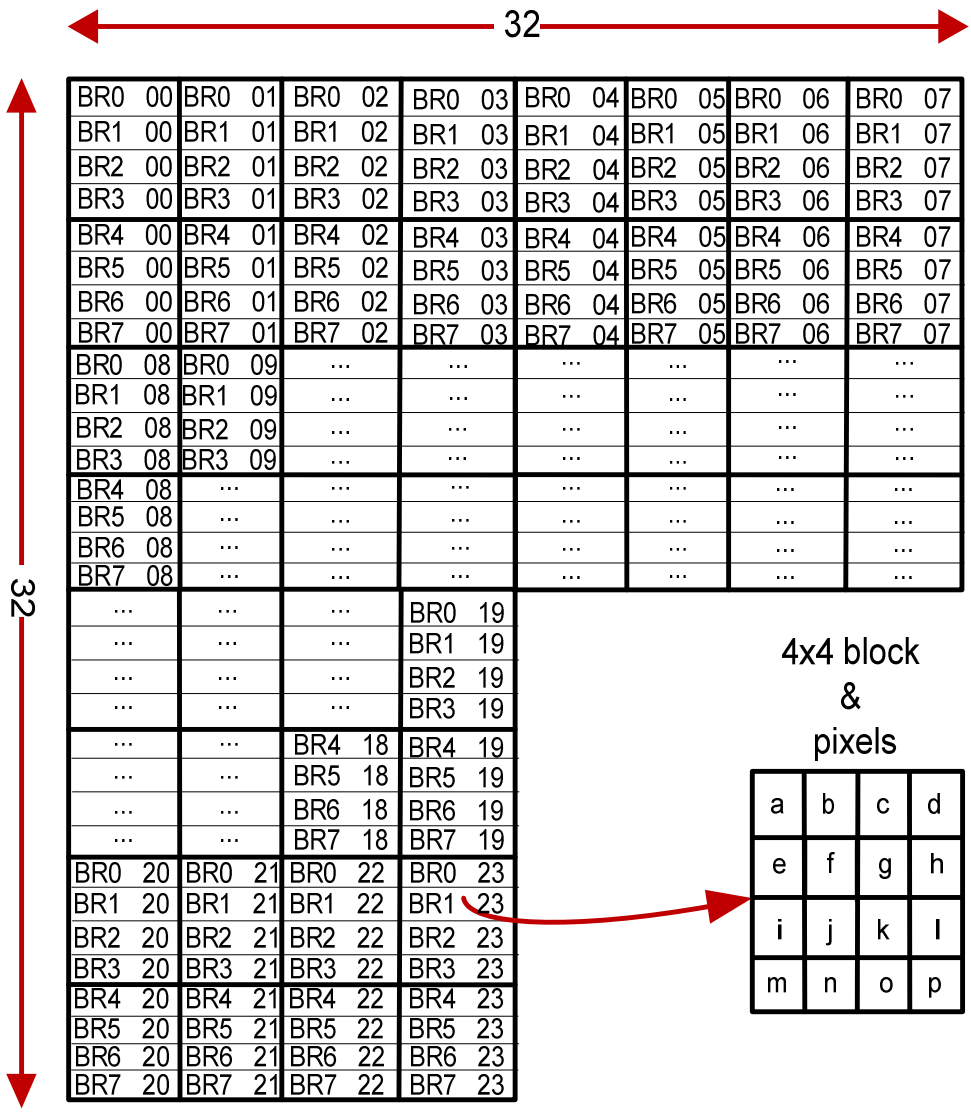


Figure 6.7 Memory Organization of 32x32 SW

The proposed hardware has 4 clock cycles initial latency for starting the intra prediction due to neighboring pixels and current block loading. Calculating the predictions for 9 H.264 4x4 intra prediction modes and their SAD values takes 17 clock cycles. TM search and comparison of SADs take 34+7=41 clock cycles. Calculating the SADs for 4 CPs takes 9 clock cycles. The proposed hardware calculates the predictions for 9 H.264 intra prediction modes for the current 4x4 block and searches the best matching CTs for the previous 4x4 block in parallel. Therefore, in worst case, it requires 34 + 7 + 9 = 50 clock

cycles. The proposed hardware can work at 150 MHz on the same FPGA after place & route. Therefore, it is capable of processing 53 HD (1280x720) frames per second.

Fig. 6.8 shows an example predicted by H.264 9 intra 4x4 modes video frame and the same frame predicted by H.264 9 intra 4x4 modes with TM including the proposed technique for 4 CTs for Th_{SAD} = 40 for VGA size MobileCalander video frame used as input for timing simulation.

6.3 Energy Consumption Analysis

The power consumption of the proposed 4x4 intra prediction with TM hardware on a Xilinx Virtex 6 FPGA is estimated using Xilinx XPower Analyzer tool. In order to estimate its power consumption, timing simulation of the placed and routed netlist of the proposed hardware is done using Mentor Graphics ModelSim SE. VGA size Flag, MobileCalendar and CIF size Foreman, Akiyo and Mother&Daughter video frames (one frame from each video sequence) are used as input for timing simulations and the signal activities are stored in VCD files. These VCD files are used for estimating the power consumption of the proposed 4x4 intra prediction with TM hardware using Xilinx XPower Analyzer tool.

The energy consumptions of the proposed hardware implementation on a Xilinx Virtex 6 FPGA at 50 MHz are shown in Tables 6.7, 6.8 and 6.9 for different Th_{SAD} values. As shown in the tables, the proposed technique reduced both total computation time and power consumption of this hardware. The proposed technique reduced the energy consumption of this hardware up to 50%.

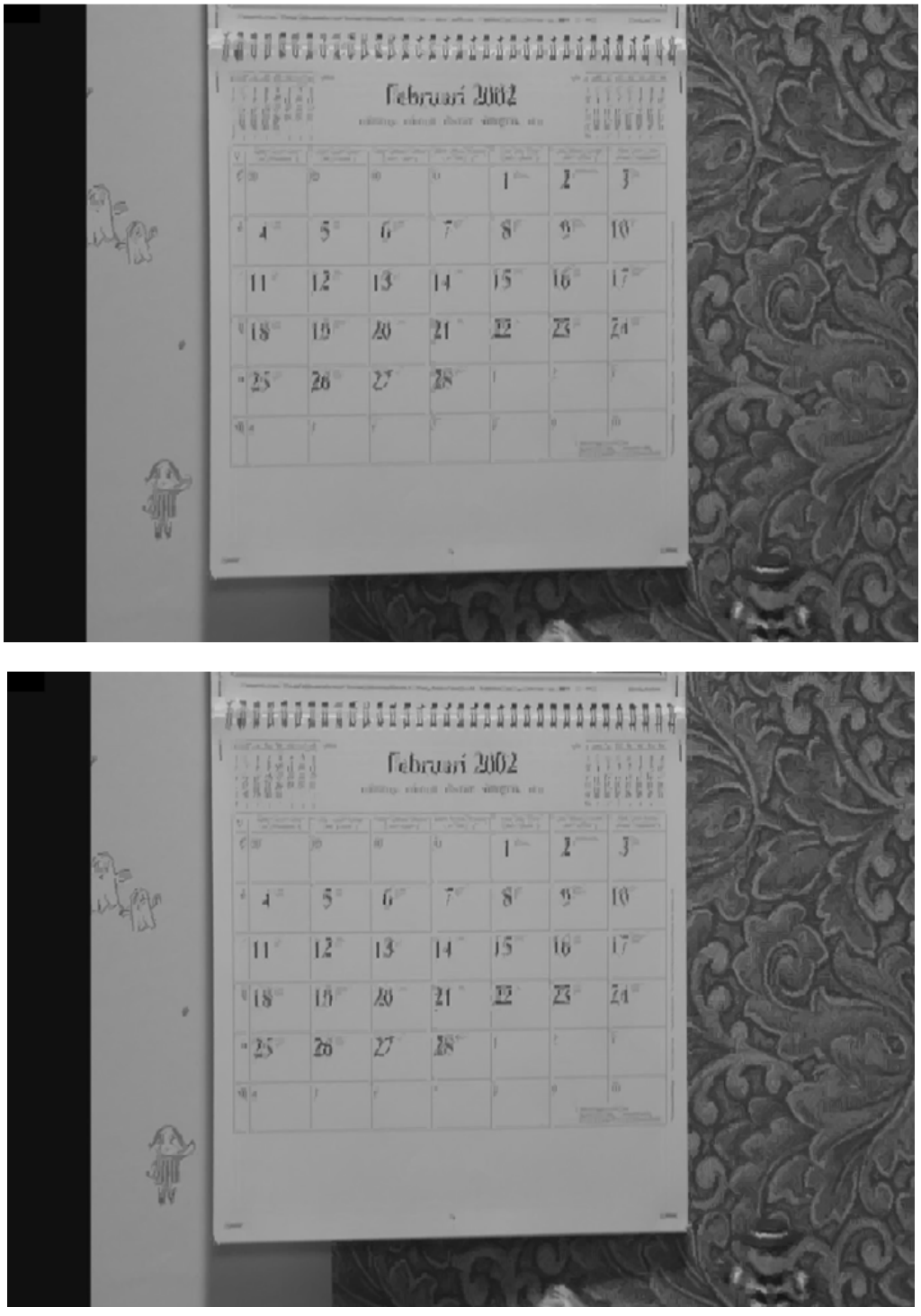


Figure 6.8 Predicted by H.264 9 intra 4x4 modes video frame shown on the above and the same frame predicted by H.264 9 intra 4x4 modes with TM including proposed technique shown on the below

Table 6.7 Energy Consumption Reduction when Th_{SAD} = 40

Frame	Size	Total Computation Time (μs)		Power (mW)		Energy (mJ)		
		Org.	LE	Org.	LE	Org.	LE	%
Flag	**720x480**	24921	17095	116	113	2895	1931	**33.3**
MobCal	**720x480**	24921	15588	104	106	2596	1658	**36.1**
Form.	**352x288**	7257	4563	115	123	831	560	**32.7**
Akiyo	**352x288**	7257	3977	109	107	788	427	**45.8**
M&D	**352x288**	7257	4398	108	111	783	489	**37.6**

Table 6.8 Energy Consumption Reduction when Th_{SAD} = 50

Frame	Size	Total Computation Time (μs)		Power (mW)		Energy (mJ)		
		Org.	LE	Org.	LE	Org.	LE	%
Flag	**720x480**	24921	15655	116	109	2895	1702	**41.2**
MobCal	**720x480**	24921	15324	104	104	2596	1598	**38.4**
Form.	**352x288**	7257	4340	115	118	831	511	**38.6**
Akiyo	**352x288**	7257	3879	109	105	788	408	**48.2**
M&D	**352x288**	7257	4156	108	111	783	463	**40.9**

Table 6.9 Energy Consumption Reduction when Th_{SAD} = 60

Frame	Size	Total Computation Time (μs)		Power (mW)		Energy (mJ)		
		Org.	LE	Org.	LE	Org.	LE	%
Flag	**720x480**	24921	14597	116	105	2895	1532	**47.1**
MobCal	**720x480**	24921	15055	104	103	2596	1554	**40.1**
Form.	**352x288**	7257	4123	115	113	831	466	**44.0**
Akiyo	**352x288**	7257	3793	109	103	788	391	**50.4**
M&D	**352x288**	7257	3965	108	107	783	424	**45.8**

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

In this thesis, we proposed novel computational complexity and power reduction techniques for intra prediction, DBF, and intra mode decision modules of an H.264 video encoder hardware, and intra prediction with TM hardware. We quantified the computation reductions achieved by these techniques using H.264 Joint Model reference software encoder. We designed efficient hardware architectures for these video compression algorithms and implemented them in Verilog HDL. We integrated the proposed techniques to these hardware implementations and quantified their impact on the power consumptions of these hardware implementations on Xilinx Virtex FPGAs. The proposed techniques significantly reduced the power consumptions of these FPGA implementations in some cases with no PSNR loss and in some cases with very small PSNR loss.

We proposed a pixel equality and pixel similarity based computation and power reduction techniques for H.264 intra prediction algorithm. The proposed PECR technique reduced the amount of computations performed by intra 4x4, 16x16 and 8x8 prediction modes up to to 60%, 28%, and 68% respectively with a small comparison overhead. The proposed PSCR technique reduced the amount of computations performed by intra 4x4,

16x16 and 8x8 prediction modes up to 68%, 39%, and 65% respectively with a small comparison overhead. The proposed PECR and PSCR techniques reduced the power consumption of a 4x4 intra prediction hardware up to 46% and 57%, respectively.

In this thesis, we also proposed pixel equality and similarity based computation and power reduction techniques for H.264 intra prediction algorithm. The proposed pixel equality and similarity based techniques reduced the amount of computations performed by 4x4 intra prediction modes up to 78% and 89%, respectively, with a small comparison overhead. They reduced the power consumption of a 4x4 intra prediction hardware up to 13.7% and 17.2%, respectively.

We, then, proposed PECR and PSCR techniques for reducing the amount of computations performed by H.264 DBF algorithm, and therefore reducing the energy consumption of H.264 DBF hardware. The proposed techniques reduced the amount of addition and shift operations performed by H.264 DBF algorithm up to 52% and 67% respectively with a small comparison overhead. We also implemented an efficient H.264 DBF hardware including the proposed techniques using Verilog HDL. The proposed techniques reduced the energy consumption of this DBF hardware up to 35% and 39%, respectively.

In addition, we proposed a novel energy reduction technique for H.264 intra mode decision. The proposed technique reduces the number of additions performed by Sum of Absolute Transformed Difference based 4x4, 16x16 and 8x8 intra mode decision algorithms used in H.264 joint model reference software encoder by 46%, 43% and 42% respectively for a CIF size frame without any PSNR loss. The proposed energy reduction technique reduced the energy consumption of 16x16 intra mode decision hardware up to 59.6%.

Also, we proposed a novel technique for reducing the amount of computations performed by intra prediction with TM. The proposed technique does not change the PSNR for some video frames, but it decreases the PSNR slightly for some video frames.The proposed FPGA implementation is capable of processing 53 HD (1280x720) frames per second, and the proposed technique reduced its energy consumption up to 50%.

As future work, the proposed computational complexity and power reduction techniques can be applied to other modules of an H.264 video encoder such as motion estimation and to the modules of a High Efficiency Video Coding (HEVC), the emerging international standard for video compression, video encoder. The impact of the proposed computational complexity and power reduction techniques on the power consumptions of ASIC implementations of the proposed hardware designs can be quantified.

BIBLIOGRAPHY

[1] Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification”, *ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC*, May 2003.

[2] Thomas Wiegand and Heiko Schwarz Ralf Schäfer, "The emerging H.264/AVC standard," *Heinrich Hertz Institute, Berlin, EBU Technical Review* 2003.

[3] I. Richardson, “The H.264 Advanced Video Compression Standard”, Wiley, 2010.

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

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

[6] L. Benini, G. De Micheli and E. Macii, "Designing Low-power Circuits: Practical Recipes," *IEEE Circuits and Systems Magazine*, vol. 1, no. 1, pp. 6 – 25, 2001.

[7] E. Ross, "Beat the Heat", *IEEE Spectrum*, vol. 41, no. 5, p. 38–43, May 2004.

[8] Mustafa Parlak, Yusuf Adibelli and Ilker Hamzaoglu, "A Novel Computational Complexity and Power Reduction Technique for H.264 Intra Prediction," *IEEE Transactions on Consumer Electronics,* vol. 54, no. 4, pp. 2006-2014, November 2008.

[9] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "Pixel Similarity Based Computation and Power Reduction Technique for H.264 Intra Prediction", *IEEE Transactions on Consumer Electronics,* vol. 56, no. 2, pp. 1079-1087, May 2010.

[10] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "Pixel Similarity Based Computation and Power Reduction Technique for H.264 Intra Prediction", *International Conference on Field Programmable Logic and Applications*, pp. 171 - 174, Aug. 2010.

[11] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "Computation and Power Reduction Techniques for H.264 Intra Prediction", *Microprocessors and Microsystems: Embedded Hardware Design,* Volume 36, Issue 3, Pages 205–214, May 2012.

[12] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "A Computation and Power Reduction Technique for H.264 Intra Prediction", *Euromicro Conference on Digital System Design*, Pages: 753 – 760, September 2010.

[13] Yusuf Adibelli and Ilker Hamzaoglu, "A High Performance and Low Energy Hardware for Intra Prediction with Template Matching", (Submitted to) *IEEE Symposium on Embedded Systems for Real-Time Multimedia*, October 2012.

[14] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "Energy Reduction Techniques for H.264 Deblocking Filter Hardware", *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, August 2011.

[15] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "A Computation and Energy Reduction Technique for H.264 Deblocking Filter Hardware", *IEEE Signal Processing and Communications Applications Conference,* Pages: 210 – 213, April 2011.

[16] Yusuf Adibelli, Mustafa Parlak and Ilker Hamzaoglu, "A Novel Energy Reduction Technique for H.264 Intra Mode Decision", *IEEE International Conference on Image Processing*, September 2011.

[17] Joint Video Team of ITU-T VCEG and ISO/IEC MPEG. Joint Model Reference Software, Version 14.0. [Online]. http://iphome.hhi.de/suehring/tml

[18] Feng Pan, Xiao Lin, S. Rahardja, K.P. Lim, Z.G. Li, Dajun Wu, Si Wu, "Fast Mode Decision Algorithm for Intraprediction in H.264/AVC Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 7, p. 813 – 822, July 2005.

[19] I. Choi, J. Lee and B. Jeon, "Fast Coding Mode Selection With Rate-Distortion Optimization for MPEG-4 Part-10 AVC/H.264," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 12, p. 1557 – 1561, December 2006.

[20] An-Chao Tsai, A. Paul, Jai-Ching Wang and Jhing-Fa Wang, , "Efficient Intra Prediction in H.264 Based on Intensity Gradient Approach," *IEEE International Symposium on Circuits and Systems*, pp. 3952 – 3955, May 2007.

[21] Ling-Jiao Pan and Yo-Sung Ho, "A Fast Mode Decision Algorithm for H.264/AVC Intra Prediction*," IEEE Workshop on Signal Processing Systems*, pp. 704 – 709, October 2007.

[22] Yu-Wen 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, March 2005.

[23] Genhua Jin, Jin-Su Jung and Hyuk-Jae Lee, "An Efficient Pipelined Architecture for H.264/AVC Intra Frame Processing," *IEEE International Symposium on Circuits and Systems*, pp. 1605 – 1608, May 2007.

[24] Esra Sahin and Ilker Hamzaoglu, "An Efficient Hardware Architecture for H.264 Intra Prediction Algorithm," *Design, Automation and Test in Europe*, April 2007.

[25] Ilker Hamzaoglu, Ozgur Tasdizen and Esra Sahin, "An Efficient H.264 Intra Frame Coder System," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1903 - 1911, November 2008.

[26] G. Bjontegaard, “Calculation of average PSNR differences between RD-curves,” *13th Video Coding Experts Group Meeting*, 2001.

[27] Y. Lai, T. Liu, Y. Li, C. Lee, Design of an intra predictor with data reuse for high-profile H.264 applications, *IEEE International Symposium on Circuits and Systems*, May 2009.

[28] Esra Sahin and Ilker Hamzaoglu, "An Efficient Intra Prediction Hardware Architecture for H.264 Video Decoding", *Euromicro Conference on Digital System Design*, August 2007.

[29] C. H. Tseng, H. M. Wang and J. F. Yang, "Enhanced Intra-4x4 Mode Decision for H.264/AVC Coders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 8, pp. 1027-1032, August 2006.

[30] F.Pan; X. Lin; S.Rahardja,; K.P.Lim,; Li, Z.G.; D. Wu; Si Wu, “Fast mode decision algorithm for intra prediction in H.264/AVC video coding”, *IEEE Trans. on CAS for Video Technology*, July 2005.

[31] Choi, I.; Lee, J.; Jeon, B. “Fast Coding Mode Selection with Rate-Distortion Optimization for MPEG-4 Part-10 AVC/H.264”, *IEEE Trans. on CAS for Video Technology*, vol. 16, no. 12, Dec. 2006.

[32] An-Chao Tsai, Paul, A., Jai-Ching Wang, Jhing-Fa Wang, “Efficient Intra Prediction in H.264 Based on Intensity Gradient Approach”, *IEEE ISCAS*, May 2007.

[33] Changsung Kim and C. C. J. Kuo, "Feature-Based Intra-/InterCoding Mode Selection for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 4, pp. 441 - 453, April 2007.

[34] Jia-Ching Wang, Jhing-Fa Wang, Jar-Ferr Yang and Jang-Ting Chen, "A Fast Mode Decision Algorithm and Its VLSI Design for H.264/AVC Intra-Prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 10, pp. 1414 - 1422 , October 2007.

[35] H.-M. Wang, C.-H. Tseng, and J.-F. Yang, "Computation Reduction for Intra 4x4 Mode Decision with SATD Criterion in H.264/AVC," *IET Signal Processing*, vol. 1, no. 2, pp. 121 - 127, September 2007.

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

[37] Mustafa Parlak and Ilker Hamzaoglu, "Low Power H.264 Deblocking Filter Hardware Implementations", *IEEE Trans. on Consumer Electronics*, vol. 54, no. 2, May 2008.

[38] Y. Huang, T. Chen, B. Hsieh, T. Wang, T. Chang, L. Chen, “Architecture design for deblocking filter in H.264/JVT/AVC”, *IEEE International Conference on Multimedia and Expo*, July 2003.

[39] H. Lin, J. Yang, B. Liu, J. Yang, “Efficient deblocking filter architecture for H.264 video coders”, *IEEE International Symposium on Circuits and Systems*, May 2006.

[40] Y-K. Lai, L-F. Chen, W-C. Chiou, "A Memory Interleaving and Interlacing Architecture for Deblocking Filter in H.264/AVC", *IEEE Trans. on Consumer Electronics*, vol. 56, no. 4, November 2010.

[41] G. Khurana, A.A. Kassim, T.P. Chua, M.B. Mi, "A pipelined hardware implementation of in-loop deblocking filter in H.264/AVC", *IEEE Trans. on Consumer Electronics*, May2006.

[42] K. Xu, C.-S.Choy, "A Five-Stage Pipeline, 204 Cycles/MB, Single-Port SRAM-Based Deblocking Filter for H.264/AVC", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 18, no. 3, March 2008.

[43] J. Lou, A. Jagmohan, D. He, L. Lu, M.-T. Sun, "H.264 Deblocking Speedup", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 19, no. 8, August 2009.

[44] Yan Ye and M. Karczewicz, "Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning," *IEEE International Conference on Image Processing*, 2008.

[45] T. K. Tan, C. S. Boon, and Y. Suzuki, "Intra prediction by template matching," *IEEE International Conference on Image Processing*, 2006.

[46] C. Lan, J. Xu, F. Wu, G Shi "Intra frame coding with template matching prediction and adaptive transform", *IEEE International Conference on Image Processing*, 2010.

[47] T.K. Tan, C.S. Boon, and Y. Suzuki, "Intra prediction by averaged template matching predictors," *IEEE Consumer Communications and Networking Conference*, Jan. 2007.

[48] A.A. Efros and T.K. Leung, "Texture synthesis by non-parametric sampling," *International Conference on Computer Vision*, 1999.

[49] L.V. Agostini, S. Bampi, "FPGA Based Architectures for H. 264/AVC Video Compression Standard", *International Conference on Field Programmable Logic and Applications*, Aug. 2006.