ADAPTIVE MOTION ESTIMATION ALGORITHM AND HARDWARE DESIGNS FOR H.264 MULTIVIEW VIDEO CODING

by

Kamil Erdayandı

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of
the requirements for the degree of
Master of Sciences

Sabancı University

August 2014

ADAPTIVE MOTION ESTIMATION ALGORITHM AND HARDWARE DESIGNS FOR H.264 MULTIVIEW VIDEO CODING

APPROVED BY:

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

Assoc. Prof. Dr. Meriç Özcan .....

Assoc. Prof. Dr. Erkay Savaş .....

DATE OF APPROVAL: 06. 08. 2014

To my Family and Friends

ACKNOWLEDGEMENT

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

I would like to thank to all members of System-on-Chip Design and Testing Lab, Yusuf Akşehir, Ercan Kalalı, Yusuf Adıbelli, Zafer Özcan, Erdem Özcan, Serkan Yalıman, and Hasan Azgın who have been greatly supportive during my study.

Special thanks to my friends and family.

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

ADAPTIVE MOTION ESTIMATION ALGORITHM AND HARDWARE DESIGNS FOR H.264 MULTIVIEW VIDEO CODING

Kamil Erdayandı
Electronics, MS Thesis, 2014

Thesis Supervisor: Assoc. Prof. İlker HAMZAOĞLU

Keywords: H.264, Multiview Video Coding, Motion Estimation, Hardware Design

ABSTRACT

Multiview Video Coding (MVC) is the process of efficiently compressing stereo (2 views) or multiview video signals. The improved compression efficiency achieved by H.264 MVC comes with a significant increase in computational complexity. Therefore, in this thesis, we propose novel techniques for significantly reducing the amount of computations performed by full search motion estimation algorithm for H.264 MVC, and therefore significantly reducing the energy consumption of full search motion estimation hardware for H.264 MVC with very small PSNR loss and bitrate increase.

We also propose an adaptive fast motion estimation algorithm for reducing the amount of computations performed by H.264 MVC motion estimation, and therefore reducing the energy consumption of H.264 MVC motion estimation hardware even more with additional very small PSNR loss and bitrate increase. We also propose an adaptive H.264 MVC motion estimation hardware for implementing the proposed adaptive fast motion estimation algorithm. The proposed motion estimation hardware is implemented in Verilog HDL and mapped to a Xilinx Virtex-6 FPGA. The proposed motion estimation hardware has less energy consumption than the full search motion estimation hardware for H.264 MVC and the full search motion estimation hardware for H.264 MVC including the proposed computation reduction techniques.

H.264 ÇOK BAKIŞLI VIDEO KODLAMA İÇİN UYARLANIR HAREKET TAHMİNİ ALGORİTMA VE DONANIM TASARIMLARI

Kamil Erdayandı
Elektronik Müh., Yüksek Lisans Tezi, 2014

Tez Danışmanı: Doç. Dr. İlker HAMZAOĞLU

Anahtar Kelimeler: H.264, Çok Bakışlı Video Kodlama, Hareket Tahmini, Donanım Tasarımı

ÖZET

Çok Bakışlı Video Kodlama (ÇBVK), stereo veya çok bakışlı video sinyallerini etkili biçimde sıkıştırma işlemidir. H.264 ÇBVK ile birlikte, sıkıştırma verimliliği yükselmiştir, fakat bununla birlikte hesaplama karmaşıklığı belirgin biçimde artmıştır. Dolayısı ile bu tezde, H.264 ÇBVK tam arama haraket tahmini algoritmasının işlem miktarını ve dolayısıyla H.264 ÇBVK tam arama hareket tahmini donanımının harcadığı enerji miktarını, bir miktar PSNR kaybı ve bit-hızı artışı ile beraber, önemli oranda azaltan özgün teknikler önerdik.

Ayrıca, H.264 ÇBVK hareket tahmini işlem miktarını ve dolayısıyla H.264 ÇBVK hareket tahmini donanımının harcadığı enerji miktarını, bir miktar daha PSNR kaybı ve bit-hızı artışı ile beraber, daha fazla azaltan uyarlanır bir hızlı hareket tahmini algoritması önerdik. Ayrıca, önerilen uyarlanır hızlı hareket tahmini algoritmasını gerçeklemek için uyarlanır bir H.264 ÇBVK hareket tahmini donanımı önerdik. Önerilen hareket tahmini donanımı, Verilog HDL kullanılarak gerçeklenmiş ve Xilinx Virtex-6 FPGA'ya yerleştirilmiştir. Önerilen hareket tahmini donanımı, H.264 ÇBVK tam arama hareket tahmini donanımından ve önerilen işlem miktarı azaltma tekniklerini içeren H.264 ÇBVK tam arama hareket tahmini donanımından daha az enerji harcamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASR	Adaptive Search Range
BM	Block Matching
BRAM	Block RAM
FPGA	Field Programmable Gate Array
GOP	Group of Pictures
HDL	Hardware Description Language
JMVC	H.264 Multi-view Coding Software Model
LT	Left Temporal
LUT	Look-up Table
MB	Macro Block
ME	Motion Estimation
MV	Motion Vector
MVC	Multiview Coding
PE	Processing Element
PSNR	Peak Signal to Noise Ratio
PBASR	Prediction Based Adaptive Search Range
RT	Right Temporal
RTL	Register Transfer Level
QP	Quantization Parameter
SAD	Sum of Absolute Differences
SR	Search Range

CHAPTER I

INTRODUCTION

Since the recently developed H.264 video compression standard has significantly better video compression efficiency than previous video compression standards, it is already started to be used in many consumer electronic devices [1, 2]. Motion estimation (ME) is used for compressing a video by removing the temporal redundancy between the video frames. Since it constitutes up to 70% of the computations performed by a video encoder, it is the most computationally intensive part of a video encoder hardware. The improved compression efficiency achieved by motion estimation in H.264 standard comes with an increase in computational complexity.

Block matching (BM) is used for ME in H.264 standard. BM partitions the current frame into non-overlapping NxN rectangular blocks and finds a MV for each block by finding the block from the reference frame in a given search range that best matches the current block. Sum of Absolute Differences (SAD) is the most preferred block matching criterion. The SAD value of a search location defined by the motion vector d(dx,dy) is calculated as below where c(x,y) and r(x,y) represent current and reference frames, respectively. The coordinates (i,j) denote the offset locations of current and reference blocks of size NxN.

$$SAD(\vec{d}) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left|c(x+i,y+j) - r(x+i+d_x,y+j+d_y)\right| \qquad (1)$$

Multiview Video Coding (MVC) is the process of efficiently compressing stereo (2 views) or multiview video signals. MVC has many applications in the consumer electronics industry such as 3 dimensional (3D) TV and free viewpoint TV. As shown in Figure 1.1, each view in a multiview video can be independently coded by an H.264 video encoder [4]. However, in order to efficiently compress a multiview video, in addition to removing the temporal redundancy between the frames of a view, the redundancy between the frames of neighboring views should also be removed.

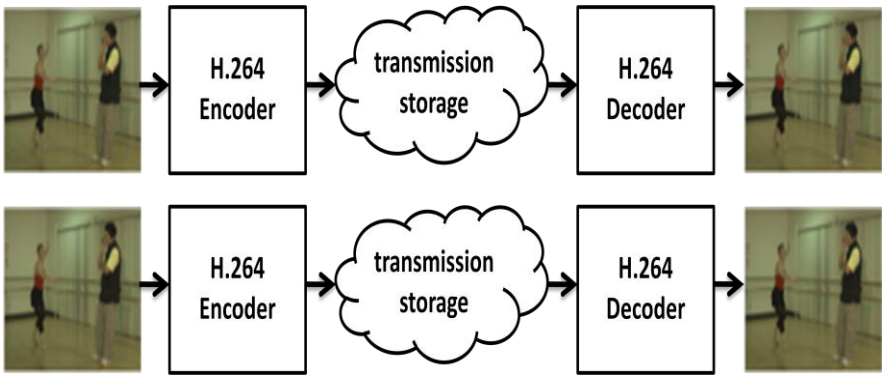

Figure 1.1 H.264 Simulcast Coding for Stereo Video [4]

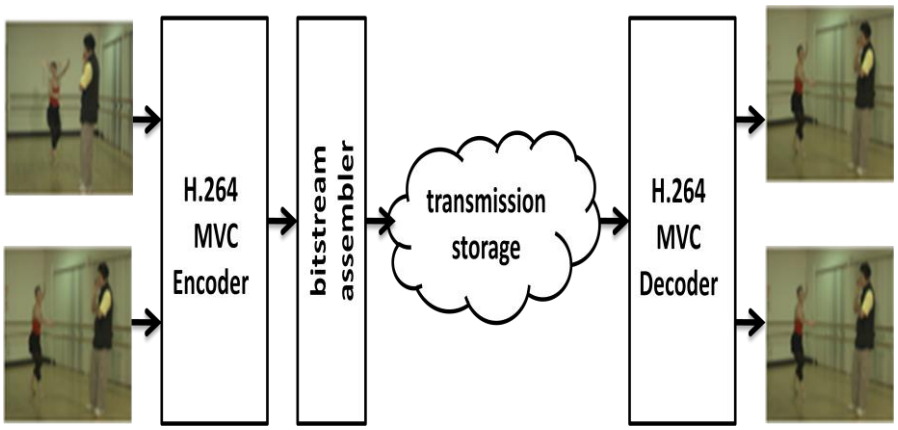

Figure 1.2 H.264 Multiview Coding for Stereo Video [4]

Therefore, H.264 standard is extended with MVC [3, 4, 5]. H.264 MVC codes the frames of the synchronized views by predicting the frames from both the other frames in the same view and the other frames in the neighboring views. In this way, it reduces the bitrate without reducing the quality of the reconstructed video in comparison to coding each view independently. H.264 MVC process for stereo video is shown in Figure 1.2 [4].

An H.264 MVC prediction structure for 5 views captured with 5 linearly arranged cameras is shown in Figure 1.3 [6]. In this prediction structure, eight temporal pictures are considered to form a group of pictures (GOP). The first picture of a GOP (black pictures in Figure 1.3) is called key picture, and the other pictures of a GOP are called nonkey pictures. The key pictures of the first view (I frames) are intra-coded. The blocks in an I frame are predicted from spatially neighboring blocks in the same frame. The key pictures of the other views (P frames) are inter-coded. The blocks in a P frame are predicted from the blocks in the key picture of previous view. Hierarchical B pictures with 3 levels are used for temporal prediction. The nonkey pictures of the first view are inter-predicted only from the previous and future pictures in the same view. The nonkey pictures of the other views are inter-predicted both from the previous and future pictures in the same view and the B pictures in the previous view.

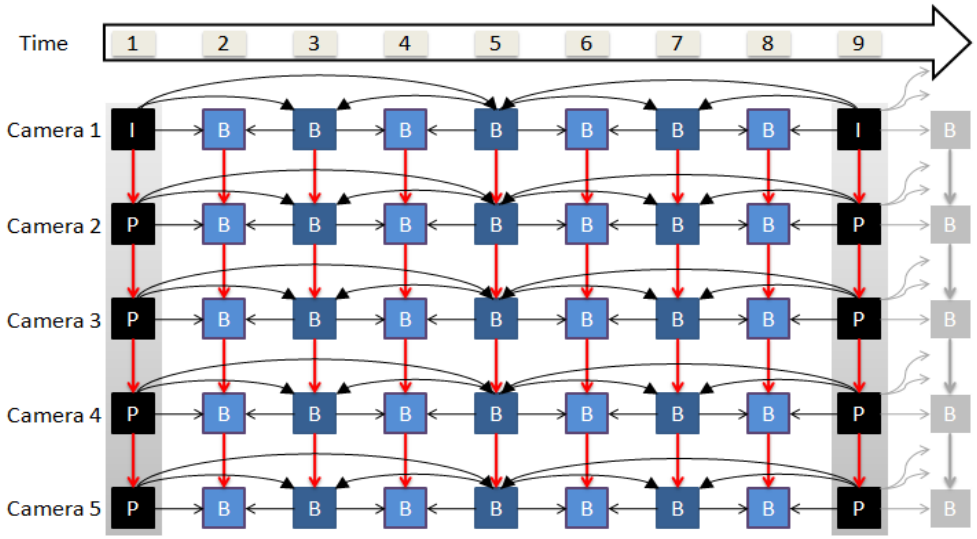

Figure 1.3 An H.264 Multiview Coding Prediction Structure [6]

1.1 Thesis Contributions

The improved compression efficiency achieved by H.264 MVC comes with a significant increase in computational complexity. Temporal prediction (between pictures in the same view) and inter-view prediction (between pictures in the neighboring views) are the most computationally intensive parts of H.264 MVC. Therefore, in this thesis, we propose novel techniques for significantly reducing the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC, and therefore significantly reducing the energy consumption of full search motion estimation hardware for H.264

MVC with very small PSNR loss and bitrate increase [7, 17]. The experimental results obtained by Joint Multiview Video Coding (JMVC) 3.01 H.264 MVC software [8] for VGA (640x480) size Ballroom and Vassar multiview videos with 8 views and 81 frames in each view [9] showed that the proposed techniques reduced the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC up to 66% with very small PSNR loss and bitrate increase.

We also propose an adaptive fast motion estimation algorithm for reducing the amount of computations performed by temporal and inter-view predictions in H.264 MVC motion estimation, and therefore reducing the energy consumption of H.264 MVC motion estimation hardware even more with additional very small PSNR loss and bitrate increase. The experimental results obtained by Joint Multiview Video Coding (JMVC) 3.01 H.264 MVC software [8] for VGA (640x480) size Ballroom and Vassar multiview videos with 8 views and 81 frames in each view [9] showed that the proposed motion estimation algorithm reduced the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC up to 86% with very small PSNR loss and bitrate increase.

We also propose an adaptive H.264 MVC motion estimation hardware for implementing the proposed adaptive fast motion estimation algorithm. The proposed motion estimation hardware is implemented in Verilog HDL and mapped to a Xilinx Virtex-6 FPGA. The proposed motion estimation hardware has up to 95% less energy consumption than the full search motion estimation hardware for H.264 MVC, and up to 81% less energy consumption than the full search motion estimation hardware for H.264 MVC including the proposed computation reduction techniques [17].

1.2 Thesis Organization

The rest of the thesis is organized as follows. Chapter II presents the proposed computation reduction techniques for temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC, and the experimental results. Chapter III presents the proposed adaptive fast motion estimation algorithm, and the experimental results. It also presents the proposed adaptive motion estimation hardware for implementing the proposed adaptive fast motion estimation algorithm, and its implementation results. Chapter IV presents conclusions and future work.

CHAPTER II

COMPUTATION REDUCTION TECHNIQUES FOR FULL SEARCH MOTION ESTIMATION ALGORITHM

The improved compression efficiency achieved by H.264 MVC comes with a significant increase in computational complexity. Temporal prediction (between pictures in the same view) and inter-view prediction (between pictures in the neighboring views) are the most computationally intensive parts of H.264 MVC. Therefore, in this thesis, we propose novel techniques for significantly reducing the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC, and therefore significantly reducing the energy consumption of full search motion estimation hardware for H.264 MVC with very small PSNR loss and bitrate increase [7, 17].

In the H.264 MVC prediction structure shown in Figure 1.3, nonkey pictures in the first GOP are temporally predicted as follows. 5th picture is predicted from 4th previous picture (1st picture) and 4th future picture (9th picture). 3rd picture and 7th picture are predicted from 2nd previous picture and 2nd future picture (1st and 5th, 5th and 9th). The other pictures are predicted from the previous neighboring picture and the future neighboring picture. Therefore, for temporal prediction, we propose using large search window for 5th picture, smaller search window for 3rd and 7th pictures, and even smaller search window for other pictures. In this way, we reduce the amount of computation performed by full search ME algorithm for temporal prediction.

As in the case of the H.264 MVC prediction structure shown in Figure 1.3, when a multiview video is captured with linearly arranged cameras, there is a constant relationship between positions of the cameras. We propose using this relationship between neighboring views for reducing the amount of computation performed by full search ME algorithm for inter-view prediction. In Ballroom and Vassar multiview videos with 8 views, because of the camera positions, we propose searching only the right side of the search window in the neighboring view during inter-view prediction.

Because of the constant relationship between camera positions, during inter-view prediction, disparity vectors found by motion estimation between neighboring views are similar. Therefore, during inter-view prediction, we propose performing full search motion estimation for the current block in a search window of size 16 ([0, +16]) if the previous disparity vector is smaller than 17, in a search window of size 32 ([0, +32]) if previous disparity vector is smaller than 33, otherwise in a search window of size 48 ([0, +48]). In addition, if previous SAD value is larger than a threshold value, the size of the search window is increased by 16. Therefore, search window size can be at most 64 ([0, +64]). The SAD values obtained by motion estimation in JMVC 3.01 H.264 MVC software are analyzed to determine this threshold value. Since most of the SAD values were smaller than 2000, the SAD threshold value is set to 1500.

In order to determine the amount of computation reduction achieved by these techniques and their impact on the rate distortion performance of the H.264 MVC encoder with the prediction structure shown in Figure 1.3, we added the proposed techniques to JMVC 3.01 H.264 MVC software [8] and disabled its following features; adjusting the search window according to the default predicted vector, variable block size search, sub-pixel search, multi-frame search, fast search algorithms, and variable quantization parameter (QP) values. Disabling these features caused 0.55 dB PSNR loss and between 400 and 450 kbit/s bit rate increase. We used VGA (640x480) size Ballroom and Vassar multiview videos which have 8 views and 81 frames in each view with the H.264 MVC prediction structure shown in Figure 1.3 for the experiments [9].

We determined the impact of using 64-64-64 ([-64, +64] - [-64, +64] - [-64, +64]), 32-32-32 ([-32, +32] - [-32, +32] - [-32, +32]), 32-32-16 ([-32, +32] - [-32, +32] - [-16, +16]), 32-16-16 ([-32, +32] - [-16, +16] - [-16, +16]), 16-8-8 ([-16, +16] - [-8, +8] - [-8, +8]), and 8-4-4 ([-8, +8] - [-4, +4] - [-4, +4]) window sizes for 5th picture, 3rd and 7th pictures, and other pictures during temporal prediction in Ballroom multiview video which has 8 views and 81 frames in each view with the H.264 MVC prediction

structure shown in Figure 1.3. The rate distortion curves and the number of SAD calculations performed for Ballroom video with QP 32 are shown in Figure 2.1 and Table 2.1, respectively.

The experimental results for 64-64-64 and 32-32-32 search window sizes are obtained without using the proposed computation reduction techniques. The experimental results for the other search window sizes are obtained using all the proposed computation reduction techniques. The rate distortion curves for 64-64-64, 32-32-16 and 32-16-16 search window sizes are similar. 32-32-32, 16-16-8 and 8-4-4 search window sizes cause higher bit rates at the same quality than the other search window sizes. 32-32-16 and 32-16-16 search window sizes obtain better rate distortion curve by performing less computation than 32-32-32 search window size, and they obtain similar rate distortion curve by performing less computation than 64-64-64 window size. Since 14% less SAD calculations are performed by using 32-16-16 search window size instead of 32-32-16 search window size, we decided using 32-16-16 search window size. As shown in Table 2.2 and Table 2.3, for 32-16-16 search window size, similar numbers of SAD calculations are performed for Ballroom and Vassar videos with different QPs.

Table 2.1 Number of SAD Calculations for H.264 MVC Motion Estimation for Ballroom with QP 32

Search Window Sizes	**Interview SAD Calculations**	**Temporal SAD Calculations**	**Total SAD Calculations**	**Reduction (%)**
32-32-32	2786918400	5505024000	8291942400	0
32-32-16	801155584	3145728000	3946883584	52.40
32-16-16	801860352	1966080000	2767940352	66.61
16-8-8	802980608	491520000	1294500608	84.38
8-4-4	804428032	122880000	927308032	88.81

Table 2.2 Number of SAD Calculations for H.264 MVC Motion Estimation Using 32-16-16 Search Window Size for Ballroom with QP 22, 32, and 42

	Interview SAD Calculations	Temporal SAD Calculations	Total SAD Calculations	Reduction (%)
QP 22	771534848	1966080000	2737614848	66.98
QP 32	801860352	1966080000	2767940352	66.61
QP 42	925411840	1966080000	2891491840	65.12
Average	832935680	1966080000	2799015680	66.24

Table 2.3 Number of SAD Calculations for H.264 MVC Motion Estimation Using 32-16-16 Search Window Size for Vassar with QP 22, 32, and 42

	Interview SAD Calculations	Temporal SAD Calculations	Total SAD Calculations	Reduction (%)
QP 22	624803328	1966080000	2590883328	68.75
QP 32	564774144	1966080000	2530854144	69.47
QP 42	548278272	1966080000	2514358272	69.67
Average	579285248	1966080000	2545365248	69.30

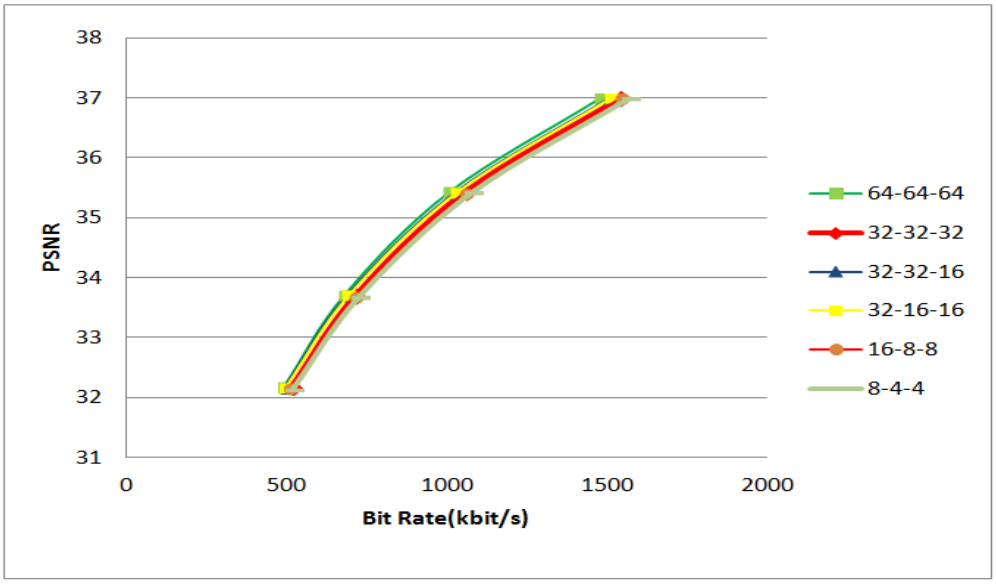

Figure 2.1 Rate Distortion Curves

As shown in Figure 2.1, Table 2.1, Table 2.2 and Table 2.3, the proposed techniques reduce the amount of computations performed by temporal and inter-view predictions in H.264 MVC significantly with very small PSNR loss and bitrate increase. We compared the PSNR and bitrate values obtained by JMVC 3.01 H.264 MVC software (the features mentioned above are disabled) using the proposed motion estimation algorithm (full search motion estimation algorithm including the proposed computation reduction techniques) and using TZ search motion estimation algorithm (fast search motion estimation algorithm used in JMVC 3.01 software). The PSNR and bitrate values for Ballroom and Vassar multiview videos which have 8 views and 81 frames in each view with the H.264 MVC prediction structure shown in Figure 1.3 are shown in Tables 2.4 – 2.15.

PSNR and bit rate comparison of the proposed motion estimation algorithm with TZ search algorithm is given in Table 2.16. Since the features mentioned above are disabled for the proposed motion estimation algorithm, in this comparison, the same features are disabled for TZ search algorithm as well. PSNR and bit rate comparison of the motion estimation algorithm proposed in [13] with TZ search algorithm is also given in Table 2.16. Since these features are not disabled for the motion estimation algorithm proposed in [13], in this comparison, they are not disabled for TZ search algorithm either. The average number of SAD calculations per macroblock for the proposed motion estimation algorithm is 2800. In [13], the average numbers of SAD calculations per macroblock for TZ search algorithm and for the motion estimation algorithm proposed in [13] are given as 920 and 330, respectively.

These results show that the proposed motion estimation algorithm achieves better rate distortion performance than TZ search motion estimation algorithm by performing more computations. TZ search motion estimation algorithm achieves better rate distortion performance than the H.264 MVC motion estimation algorithm proposed in [13] by performing more computations. Therefore, the amount of computations performed by temporal and inter-view predictions in H.264 MVC can be reduced more than the proposed techniques by using fast search motion estimation algorithms such as TZ search and the algorithm proposed in [13] at the expense of more PSNR loss and bitrate increase.

Table 2.4 PSNR and Bit-rate for Ballroom with QP 22 Using Proposed Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	40.59	43.07	43.14	4236.66
1	40.26	42.84	42.87	4219.35
2	40.39	43.20	43.22	3810.11
3	40.28	43.02	42.99	3895.18
4	40.24	42.94	42.82	4066.67
5	40.54	43.42	43.33	3625.36
6	40.02	42.68	42.78	4599.65
7	40.16	42.78	42.68	4232.48
Average	40.31	42.99	42.98	4085.68

Table 2.5 PSNR and Bit-rate for Ballroom with QP 32 Using Proposed Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	34.90	39.03	38.96	1097.16
1	34.85	39.24	38.96	862.99
2	35.01	39.61	39.47	787.93
3	34.61	39.08	39.01	836.32
4	34.59	39.05	38.77	867.85
5	35.00	39.42	39.31	833.18
6	34.57	39.24	39.06	833.11
7	34.33	38.74	38.38	900.41
Average	34.73	39.17	38.99	877.37

Table 2.6 PSNR and Bit-rate for Ballroom with QP 42 Using Proposed Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	29.58	36.44	36.38	349.15
1	29.57	36.75	36.44	301.10
2	29.96	37.25	37.01	288.83
3	29.36	36.69	36.67	295.68
4	29.31	36.78	36.43	300.47
5	29.76	36.64	36.72	302.02
6	29.69	36.57	36.56	300.21
7	28.87	36.26	36.10	314.31
Average	29.51	36.67	36.54	306.47

Table 2.7 PSNR and Bit-rate for Ballroom with QP 22 Using TZ Search Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	40.59	43.08	43.14	4210.64
1	40.27	42.85	42.87	4251.02
2	40.40	43.20	43.21	3838.80
3	40.30	43.01	42.99	3921.46
4	40.25	42.92	42.82	4118.59
5	40.56	43.41	43.34	3668.96
6	40.03	42.67	42.78	4651.26
7	40.18	42.78	42.67	4263.42
Average	40.32	42.99	42.98	4115.52

Table 2.8 PSNR and Bit-rate for Ballroom with QP 32 Using TZ Search Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	34.90	39.04	38.97	1082.58
1	34.86	39.23	38.95	883.84
2	35.02	39.59	39.45	809.70
3	34.61	39.07	38.99	856.67
4	34.59	39.02	38.75	894.06
5	35.01	39.40	39.26	861.90
6	34.58	39.19	39.04	861.17
7	34.33	38.71	38.35	923.68
Average	34.74	39.16	38.97	896.70

Table 2.9 PSNR and Bit-rate for Ballroom with QP 42 Using TZ Search Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	29.55	36.46	36.40	344.61
1	29.53	36.75	36.43	307.59
2	29.91	37.17	36.98	297.59
3	29.31	36.62	36.61	303.04
4	29.26	36.71	36.36	309.10
5	29.71	36.61	36.65	313.56
6	29.64	36.51	36.48	311.72
7	28.82	36.16	36.03	324.77
Average	29.47	36.62	36.49	314.00

Table 2.10 PSNR and Bit-rate for Vassar with QP 22 Using Proposed Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	40.09	42.88	42.57	3663.85
1	40.03	42.86	42.42	3859.09
2	40.26	43.29	42.79	3409.82
3	40.12	43.14	42.95	3377.05
4	40.06	42.98	42.73	3363.54
5	40.60	43.99	43.51	2614.19
6	39.98	42.68	42.22	4484.85
7	40.13	42.90	42.57	3453.61
Average	40.16	43.09	42.72	3528.25

Table 2.11 PSNR and Bit-rate for Vassar with QP 32 Using Proposed Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	34.92	40.18	39.46	421.69
1	34.85	40.36	39.33	351.94
2	35.15	40.98	39.81	339.82
3	34.92	40.66	40.25	348.75
4	34.74	40.33	39.65	344.71
5	35.49	41.70	40.83	319.20
6	34.60	40.40	39.18	417.46
7	34.69	40.30	39.63	399.10
Average	34.92	40.61	39.77	367.84

Table 2.12 PSNR and Bit-rate for Vassar with QP 42 Using Proposed Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	30.80	38.42	37.53	131.76
1	30.77	38.88	37.55	138.36
2	31.12	39.44	38.05	134.02
3	30.78	39.27	38.75	133.03
4	30.67	39.02	37.92	138.30
5	31.17	40.11	39.28	144.00
6	30.41	39.05	37.74	142.75
7	30.17	38.85	38.25	141.80
Average	30.74	39.13	38.13	138.00

Table 2.13 PSNR and Bit-rate for Vassar with QP 22 Using TZ Search Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	40.09	42.88	42.57	3659.53
1	40.04	42.86	42.42	3856.76
2	40.26	43.29	42.80	3405.40
3	40.12	43.14	42.94	3374.12
4	40.06	42.98	42.73	3357.58
5	40.60	43.99	43.51	2610.46
6	39.98	42.69	42.21	4482.18
7	40.13	42.90	42.57	3447.02
Average	40.16	43.09	42.72	3524.13

Table 2.14 PSNR and Bit-rate for Vassar with QP 32 Using TZ Search Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	34.92	40.18	39.45	417.47
1	34.84	40.36	39.32	355.27
2	35.15	40.97	39.80	339.49
3	34.93	40.64	40.24	349.67
4	34.73	40.33	39.65	340.49
5	35.48	41.67	40.81	321.94
6	34.60	40.38	39.19	419.63
7	34.70	40.26	39.62	398.75
Average	34.92	40.60	39.76	367.84

Table 2.15 PSNR and Bit-rate for Vassar with QP 42 Using TZ Search Motion Estimation Algorithm

View	Y	U	V	Bit Rate
0	30.79	38.41	37.52	129.38
1	30.73	38.85	37.52	136.10
2	31.07	39.41	38.04	131.54
3	30.73	39.23	38.71	130.83
4	30.62	39.01	37.90	136.50
5	31.11	40.05	39.25	143.42
6	30.40	39.02	37.70	143.77
7	30.14	38.79	38.22	143.63
Average	30.70	39.10	38.11	136.90

Table 2.16 PSNR and bit rate comparison

	QP	[13]		Prop. Algorithm	
		Δ PSNR (dB)	**Bit Rate (%)**	**Δ PSNR (dB)**	**Bit Rate (%)**
Ballroom	22	-0.011	8.4	-0.002	-0.7
	32	-0.06	11.7	0.012	-2.2
	42	-0.19	19.8	0.048	-2.5
Vassar	22	-0.01	1.1	-0.001	0.1
	32	-0.013	6.7	0.009	0
	42	-0.043	6.7	0.033	0.8
Average		-0.055	9.1	0.017	-0.8

CHAPTER III
AN ADAPTIVE FAST MOTION ESTIMATION ALGORITHM AND ITS HARDWARE IMPLEMENTATION

3.1 Proposed Motion Estimation Algorithm

The improved compression efficiency achieved by H.264 MVC comes with a significant increase in computational complexity. Temporal prediction (between pictures in the same view) and inter-view prediction (between pictures in the neighboring views) are the most computationally intensive parts of H.264 MVC. Therefore, in this thesis, we propose prediction based adaptive search range (PBASR) fast motion estimation algorithm for reducing the amount of computations performed by temporal and inter-view predictions in H.264 MVC motion estimation, and therefore reducing the energy consumption of H.264 MVC motion estimation hardware even more with additional very small PSNR loss and bitrate increase.

The proposed fast motion estimation algorithm determines the position and size of the search range, that will be used for inter-view and temporal predictions of the current macroblock (MB), adaptively by using the motion vectors of previously coded neighboring MBs in current frame, motion vectors of previously coded MBs in previous

and future reference frames in the current view, and motion vectors of previously coded MBs in inter-view reference frame in the previous view.

The position of the search range for inter-view prediction of current MB in a B frame is determined by using inter-view vectors of previously coded neighboring MBs of current MB in current frame (blue MBs 1, 2, 3, 4 in Figure 3.1), and inter-view vectors of previously coded neighboring MBs of current MB in its previous and future reference frames in the current view (brown MBs 1, 2, 3, 4, 5 in Figure 3.1) as candidate vectors. The brown MBs 1 in the previous and future reference frames are located in the same position as the current MB in current frame. The position of the search range for inter-view prediction of current MB in a P frame (key picture) is determined by using inter-view vectors of previously coded neighboring MBs of current MB in current frame, and inter-view vectors of previously coded MBs in the 8th previous reference frame in the current view as candidate vectors.

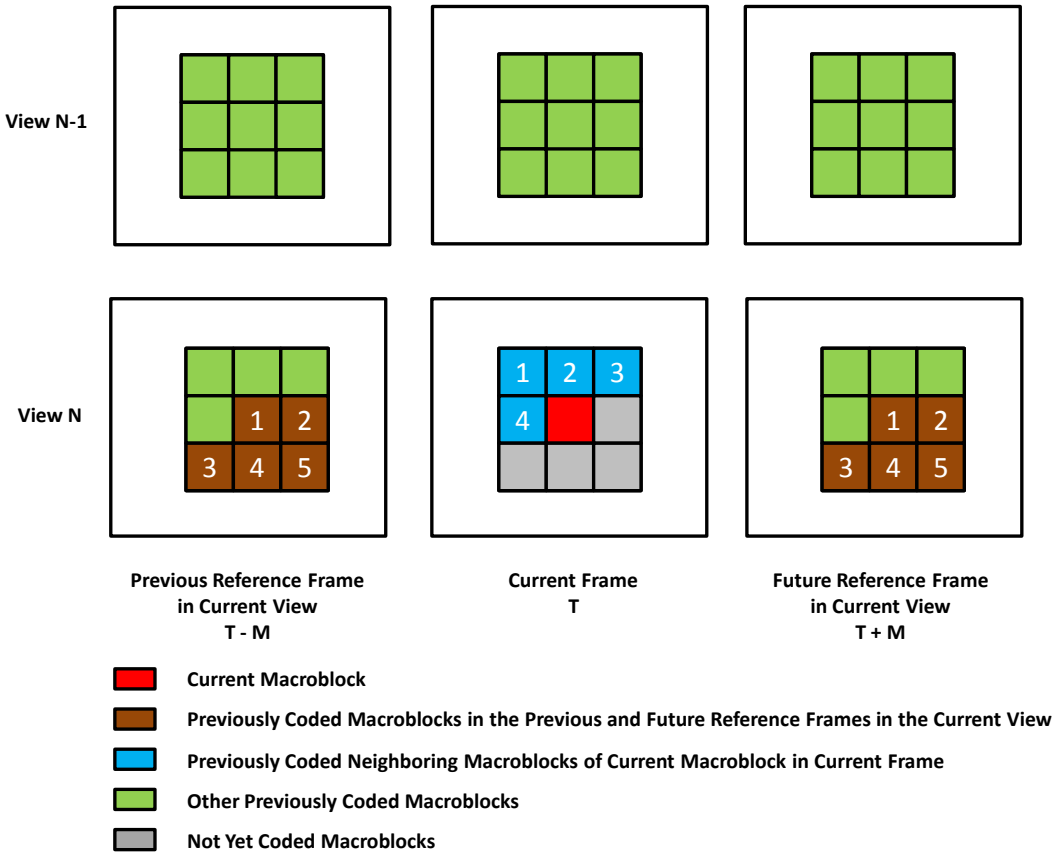

Figure 3.1 Candidate Vectors for Inter-view Prediction

For each candidate vector, SAD value is calculated between current MB and reference MB pointed by that candidate vector. The candidate vector with the smallest SAD value is selected as the predicted vector. The predicted vector points to the center of the search range for inter-view prediction of current MB.

If a candidate vector is pointing outside a frame, it is not used and the SAD value for this candidate vector is not calculated. If a MB, whose inter-view vector would be used as a candidate vector, is outside the frame, this candidate vector is not used and the SAD value for this candidate vector is not calculated. For example, as shown in Figure 3.2, when current MB is at the top border of current frame, neighboring MBs 1, 2, 3 of current MB are outside the current frame and therefore the corresponding candidate vectors are not used. Similarly, as shown in Figure 3.3, when current MB is at down-right corner of current frame, neighboring MB 3 of current MB and neighboring MBs 2, 3, 4, 5 in its previous and future reference frames are outside the frame and therefore the corresponding candidate vectors are not used.

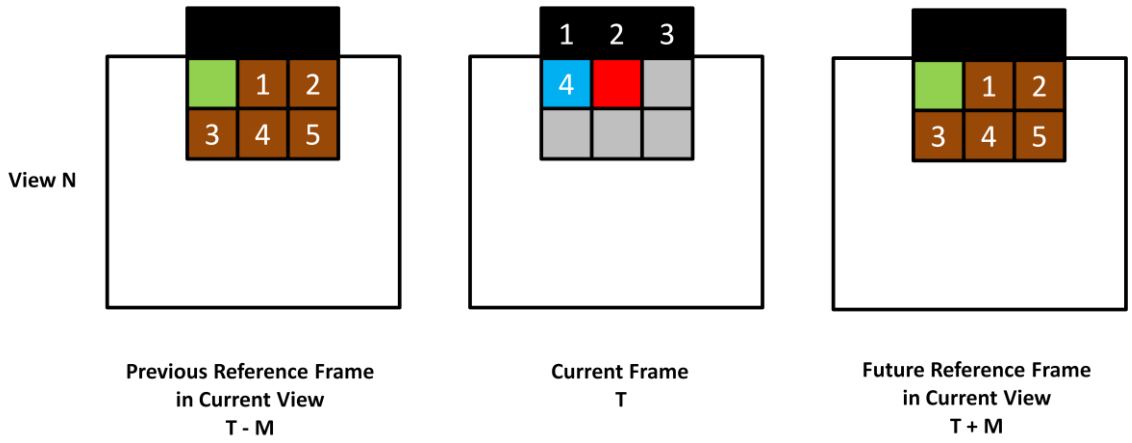

Figure 3.2 Candidate Vectors When Current MB is at Top Border

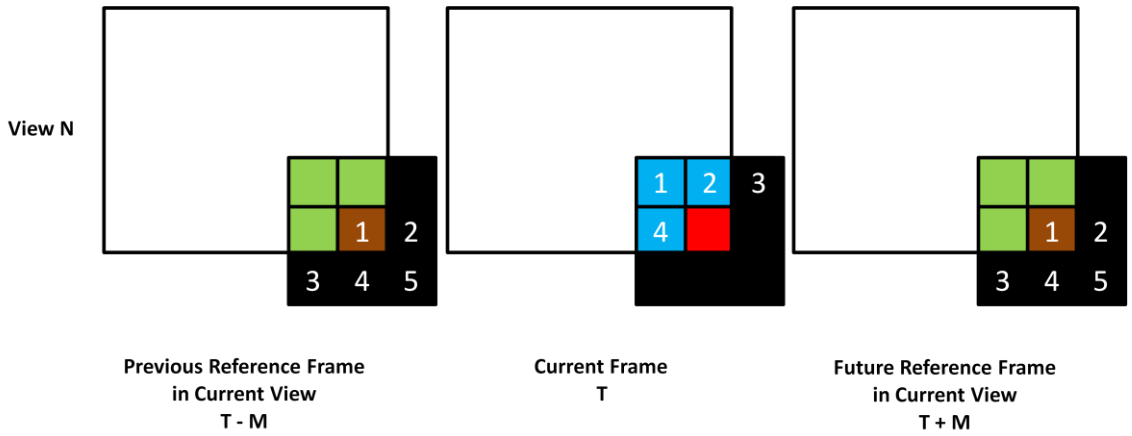

Figure 3.3 Candidate Vectors When Current MB is at Down-Right Corner

The size of the search range for inter-view prediction of current MB is determined based on the difference between the predicted vector and inter-view vectors of the corresponding MB in the previous reference frame in current view and its neighboring MBs. Since predicted vector and these vectors are good estimates for the inter-view vector of current MB, their difference is a good estimate for the size of the search range for the inter-view vector of current MB.

The corresponding MB in the previous reference frame in current view is pointed by temporal vector (TV_N) of current MB. Since TV_N is not calculated yet while current MB is being coded, as shown in Figure 3.4 instead of TV_N, temporal vector (TV_{N-1}) of the MB in the neighboring frame in previous view pointed by the predicted vector of current MB is used to determine the corresponding MB in the previous reference frame in current view. Therefore, the MB pointed by TV_{N-1} (MB 9) is taken as the corresponding MB. If the MB pointed by TV_{N-1} is outside the previous reference frame in current view or if the current frame is a P frame (key picture), the MB in the previous reference frame in current view which is in the same position as the current MB in current frame is taken as the corresponding MB.

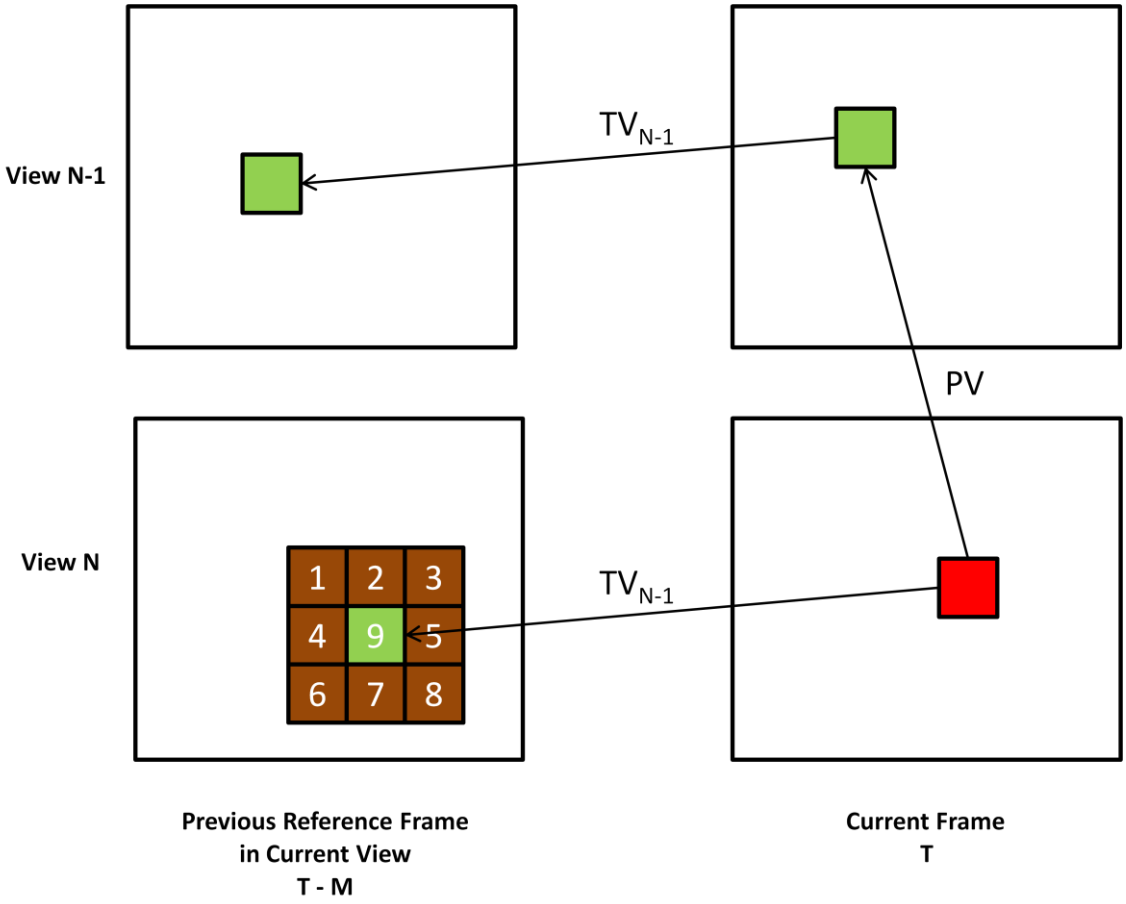

Figure 3.4 Corresponding MB in Previous Reference Frame in Current View

The size of the search range for inter-view prediction of current MB in x direction and y direction are calculated as shown in (2) and (3)

$$\text{SearchRange}_x = \frac{\frac{\sum_{i=1}^{i=8}(\text{IVcm}_{i_x} - \text{PVx})}{8} + \text{IVcm}_{9_x} - \text{PVx}}{2} \quad (2)$$

$$\text{SearchRange}_y = \frac{\frac{\sum_{i=1}^{i=8}(\text{IVcm}_{i_y} - \text{PVy})}{8} + \text{IVcm}_{9_y} - \text{PVy}}{2} \quad (3)$$

where IVcm_i denote inter-view vectors of corresponding MB and its neighboring MBs, i indicates the MB number as shown in Figure 3.4, and PV denotes the predicted vector. IVcm_{9_x} and IVcm_{9_y} are given higher weights, since they are expected to be more similar to the inter-view vector of current MB.

The temporal vectors of current MB are determined after its inter-view vector is determined. The position of the search range for temporal prediction of current MB is determined by using temporal vectors of previously coded neighboring MBs of current MB in current frame (blue MBs 1, 2, 3, 4 in Figure 3.5), temporal vector of previously coded MB in the inter-view reference frame of current MB in the previous view pointed by the inter-view vector of current MB (brown MB 5 in Figure 3.5) and temporal vectors of its neighboring MBs (brown MBs 1, 2, 3, 4, 6, 7, 8, 9 in Figure 3.5) as candidate vectors.

For each candidate vector, SAD value is calculated between current MB and reference MB pointed by that candidate vector. The candidate vector with the smallest SAD value is selected as the predicted vector. The predicted vector points to the center of the search range for temporal prediction of current MB.

If a candidate vector is pointing outside a frame, it is not used and the SAD value for this candidate vector is not calculated. If a MB, whose temporal vector would be used as a candidate vector, is outside the frame, this candidate vector is not used and the SAD value for this candidate vector is not calculated.

The size of the search range for temporal prediction of current MB is determined based on the difference between the predicted vector, and temporal vector of previously coded MB in the inter-view reference frame of current MB in the previous view pointed by the inter-view vector of current MB (green MB 9 in Figure 3.6) and temporal vectors of its neighboring MBs (brown MBs 1, 2, 3, 4, 5, 6, 7, 8 in Figure 3.6). Since predicted vector and these vectors are good estimates for the temporal vector of current MB, their difference is a good estimate for the size of the search range for the temporal vector of current MB.

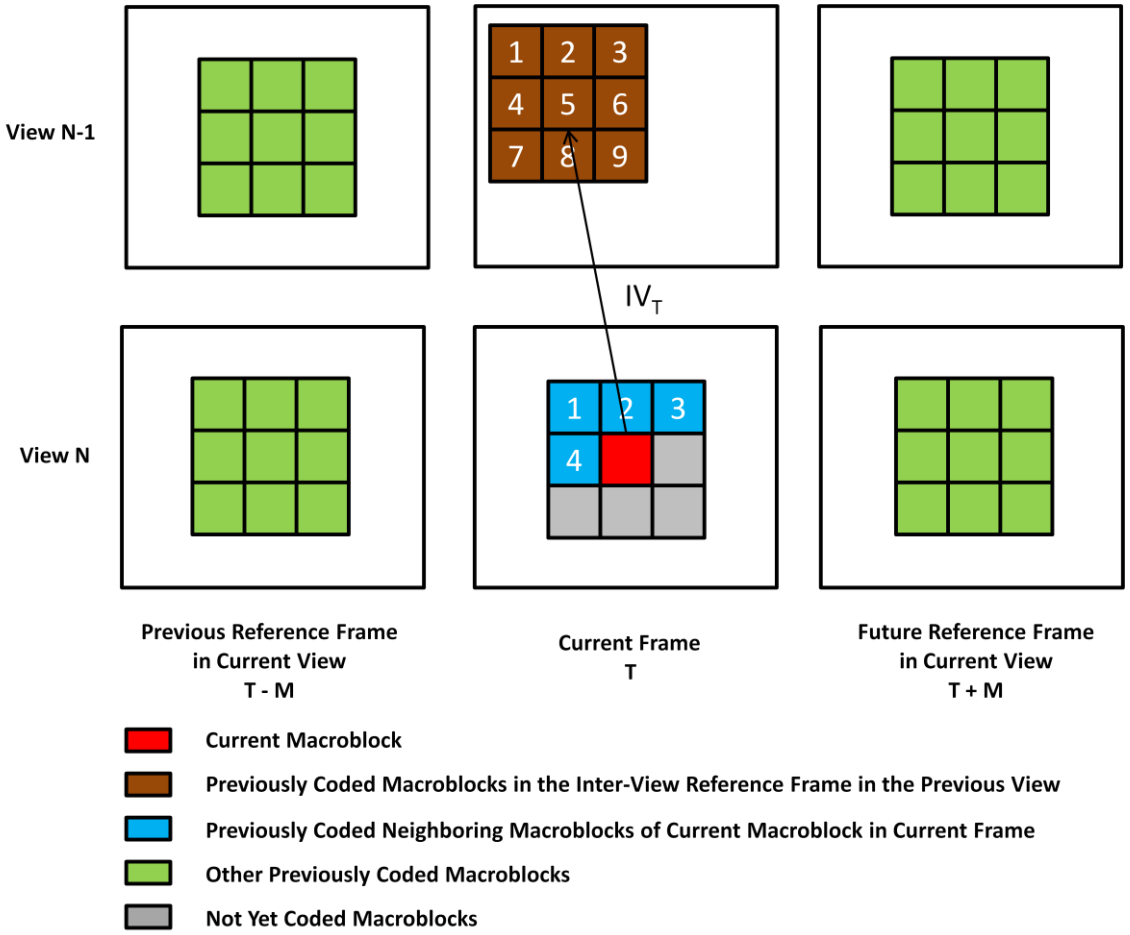

Figure 3.5 Candidate Vectors for Temporal Prediction

The size of the search range for temporal prediction of current MB in x direction and y direction are calculated as shown in (4) and (5)

$$\text{SearchRange}_x = \frac{\frac{\sum_{i=1}^{i=8}(\text{TVcm}_{i_x} - \text{PVx})}{8} + \text{TVcm}_{9_x} - \text{PVx}}{2} \quad (4)$$

$$\text{SearchRange}_y = \frac{\frac{\sum_{i=1}^{i=8}(\text{TVcm}_{i_y} - \text{PVy})}{8} + \text{TVcm}_{9_y} - \text{PVy}}{2} \quad (5)$$

where TVcm_i denote temporal vectors of corresponding MB and its neighboring MBs, i indicates the MB number as shown in Figure 3.6, and PV denotes the predicted vector.

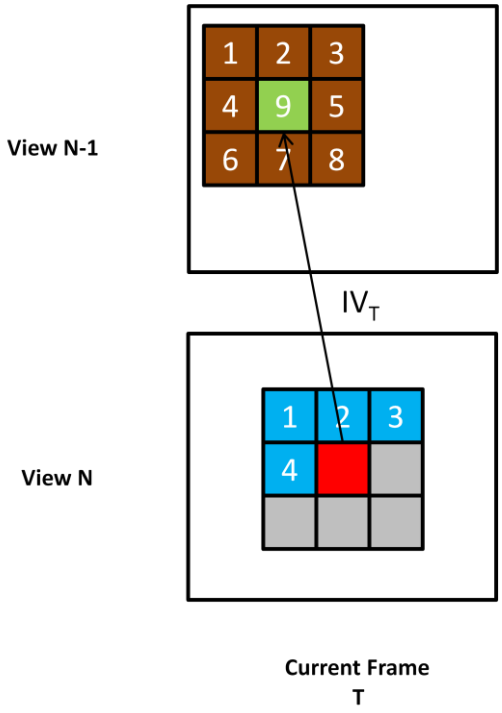

Figure 3.6 Corresponding MB in Inter-View Reference Frame in Previous View

The proposed fast motion estimation algorithm does not adaptively determine the position and size of the search range for the MBs in the first view. It performs temporal predictions of current MB in the [-32, +32] size search range pointed by zero motion vector. It adaptively determines the position and size of the search range for the MBs in the other views. It performs inter-view prediction of current MB in the search range whose size is determined by the equations (2) and (3), and pointed by the predicted vector of current MB. It performs temporal prediction of current MB in the search range whose size is determined by the equations (4) and (5), and pointed by the predicted vector of current MB.

We added the proposed fast motion estimation algorithm to JMVC 3.01 H.264 MVC software [8] and disabled its following features; adjusting the search window according to the default predicted vector, variable block size search, sub-pixel search, multi-frame search, fast search algorithms, and variable quantization parameter (QP) values. We used VGA (640x480) size Ballroom and Vassar multiview videos which have 8 views and 81 frames in each view with the H.264 MVC prediction structure shown in Figure 1.3 for the experiments [9].

Table 3.1 and Table 3.2 show the average number of SAD calculations for Ballroom and Vassar multiview videos, respectively. Table 3.3 shows the percentage of SAD calculation reductions compared to full search motion estimation for Ballroom and Vassar multiview videos. These results show that the proposed fast motion estimation algorithm reduced the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC up to 86%.

Table 3.1 Number of SAD Calculations for Ballroom

View	# Interview SAD	# Temporal SAD
0	0	688128000
1	27139558	49790290
2	22921424	46854459
3	20928354	45463997
4	20526138	44550753
5	21088811	44017779
6	22626781	43453368
7	21171134	41537532
Total	156402200	1003796178

Table 3.2 Number of SAD Calculations for Vassar

View	# Interview SAD	# Temporal SAD
0	0	688128000
1	16628679	30910900
2	17605759	30872422
3	18248413	30868952
4	14557858	30856970
5	16218042	30859682
6	21617815	30885457
7	39045593	30845146
Total	143922159	904227529

Total number of SAD calculations for Ballroom = 156402200 + 1003796178 (6)
= 1160198378

Total number of SAD calculations for Vassar = 143922159 + 904227529 (7)
= 1048149688

Table 3.3 Percentage of SAD Calculation Reductions Compared to Full Search Motion Estimation for Ballroom and Vassar

	Ballroom	**Vassar**
Interview Reduction	% 94.38	% 94.83
Temporal Reduction	% 81.76	% 83.57
Total Reduction	% 86.00	% 87.35

Table 3.4 - Table 3.9 show the PSNR and bitrate values for proposed motion estimation algorithm for Ballroom and Vassar videos for QP = 22, 32 and 42 respectively. Table 3.10 - Table 3.15 show the PSNR and bitrate values for full search motion estimation algorithm for Ballroom and Vassar videos for QP = 22, 32 and 42 respectively. These results show that the proposed motion estimation algorithm reduced the amount of computations performed by temporal and inter-view predictions in H.264 MVC motion estimation significantly with very small PSNR loss and bitrate increase.

Table 3.4 PSNR and bit rate for ballroom with QP 22 using proposed motion estimation algorithm

View	**Y**	**U**	**V**	**Bit Rate**
0	40.59	43.08	43.14	4214.10
1	40.27	42.85	42.88	4218.05
2	40.40	43.19	43.22	3806.53
3	40.29	43.01	42.99	3873.48
4	40.24	42.93	42.83	4048.39
5	40.55	43.42	43.34	3600.93
6	40.03	42.67	42.78	4594.72
7	40.17	42.79	42.67	4224.62
Average	40.32	42.99	42.98	4072.60

Table 3.5 PSNR and bit rate for ballroom with QP 32 using proposed motion estimation algorithm

View	Y	U	V	Bit Rate
0	34.90	39.05	38.97	1085.54
1	34.87	39.23	38.95	870.20
2	35.02	39.58	39.45	796.75
3	34.62	39.07	39.01	835.04
4	34.60	39.04	38.75	869.68
5	35.02	39.39	39.28	835.60
6	34.59	39.21	39.05	836.60
7	34.35	38.72	38.38	913.14
Average	34.75	39.16	38.98	880.32

Table 3.6 PSNR and bit rate for ballroom with QP 42 using proposed motion estimation algorithm

View	Y	U	V	Bit Rate
0	29.60	36.48	36.42	351.53
1	29.52	36.72	36.40	304.75
2	29.90	37.16	36.97	293.81
3	29.30	36.63	36.63	298.19
4	29.25	36.73	36.38	302.38
5	29.72	36.61	36.67	305.46
6	29.66	36.49	36.52	302.60
7	28.84	36.18	36.04	320.52
Average	29.48	36.62	36.50	309.91

Table 3.7 PSNR and bit rate for vassar with QP 22 using proposed motion estimation algorithm

View	Y	U	V	Bit Rate
0	40.09	42.88	42.57	3660.64
1	40.04	42.86	42.42	3857.79
2	40.26	43.29	42.80	3405.00
3	40.13	43.14	42.94	3378.84
4	40.07	42.98	42.73	3366.95
5	40.61	44.00	43.51	2615.09
6	39.99	42.69	42.22	4487.45
7	40.14	42.90	42.57	3462.23
Average	40.17	43.09	42.72	3529.25

Table 3.8 PSNR and bit rate for vassar with QP 32 using proposed motion estimation algorithm

View	Y	U	V	Bit Rate
0	34.92	40.18	39.46	418.72
1	34.84	40.35	39.31	350.48
2	35.15	40.96	39.80	338.00
3	34.92	40.63	40.23	348.34
4	34.73	40.31	39.64	343.59
5	35.49	41.65	40.81	321.27
6	34.60	40.38	39.17	418.71
7	34.69	40.26	39.63	406.94
Average	34.92	40.59	39.75	368.26

Table 3.9 PSNR and bit rate for vassar with QP 42 using proposed motion estimation algorithm

View	Y	U	V	Bit Rate
0	30.80	38.42	37.53	129.53
1	30.69	38.83	37.52	133.82
2	31.04	39.38	38.03	131.48
3	30.69	39.22	38.70	126.91
4	30.57	38.98	37.89	137.09
5	31.02	40.06	39.22	140.08
6	30.39	38.98	37.68	139.09
7	30.08	38.80	38.20	138.72
Average	30.66	39.08	38.10	134.59

Table 3.10 PSNR and bit rate for ballroom with QP 22 using full search motion estimation algorithm

View	Y	U	V	Bit Rate
0	40.59	43.08	43.14	4214.10
1	40.27	42.85	42.87	4248.29
2	40.40	43.19	43.22	3835.65
3	40.29	43.01	42.99	3921.42
4	40.25	42.93	42.82	4112.91
5	40.56	43.42	43.33	3667.44
6	40.03	42.67	42.78	4650.34
7	40.17	42.77	42.67	4262.53
Average	40.32	42.99	42.98	4114.08

Table 3.11 PSNR and bit rate for ballroom with QP 32 using full search motion estimation algorithm

View	Y	U	V	Bit Rate
0	34.90	39.05	38.97	1085.54
1	34.86	39.24	38.97	885.12
2	35.02	39.60	39.46	811.31
3	34.62	39.10	39.00	859.73
4	34.60	39.05	38.76	895.46
5	35.02	39.41	39.30	863.45
6	34.58	39.21	39.04	863.86
7	34.34	38.74	38.37	925.99
Average	34.74	39.17	38.98	898.81

Table 3.12 PSNR and bit rate for ballroom with QP 42 using full search motion estimation algorithm

View	Y	U	V	Bit Rate
0	29.60	36.48	36.42	351.53
1	29.58	36.76	36.45	313.99
2	29.97	37.19	37.01	301.60
3	29.38	36.65	36.63	309.37
4	29.32	36.73	36.40	314.33
5	29.78	36.63	36.66	318.50
6	29.71	36.53	36.56	314.82
7	28.88	36.19	36.05	329.24
Average	29.53	36.65	36.52	319.17

Table 3.13 PSNR and bit rate for vassar with QP 22 using full search motion estimation algorithm

View	Y	U	V	Bit Rate
0	40.09	42.88	42.57	3661.45
1	40.03	42.86	42.42	3857.79
2	40.26	43.29	42.80	3407.65
3	40.12	43.14	42.95	3374.79
4	40.06	42.98	42.72	3359.53
5	40.60	43.99	43.52	2613.54
6	39.98	42.69	42.22	4485.22
7	40.13	42.90	42.56	3449.43
Average	40.16	43.09	42.72	3526.17

Table 3.14 PSNR and bit rate for vassar with QP 32 using full search motion estimation algorithm

View	Y	U	V	Bit Rate
0	34.92	40.18	39.46	420.33
1	34.85	40.37	39.33	361.97
2	35.16	40.98	39.82	346.41
3	34.93	40.66	40.24	355.31
4	34.74	40.33	39.65	348.85
5	35.49	41.70	40.83	325.59
6	34.60	40.40	39.20	424.27
7	34.69	40.28	39.64	402.82
Average	34.92	40.61	39.77	373.19

Table 3.15 PSNR and bit rate for vassar with QP 42 using full search motion estimation algorithm

View	Y	U	V	Bit Rate
0	30.80	38.42	37.53	132.28
1	30.76	38.87	37.55	141.43
2	31.13	39.44	38.06	137.58
3	30.79	39.25	38.73	139.41
4	30.68	39.02	37.91	141.74
5	31.19	40.08	39.29	149.39
6	30.42	39.06	37.72	148.22
7	30.19	38.83	38.25	149.24
Average	30.74	39.12	38.13	142.41

3.2 Proposed Motion Estimation Hardware

We also propose an adaptive H.264 MVC motion estimation hardware for implementing the proposed prediction based adaptive search range (PBASR) fast motion estimation algorithm. The proposed hardware has three hardware modules working in parallel; inter-view hardware, left temporal (LT) hardware and right temporal (RT) hardware. Inter-view, LT and RT hardware have the same hardware architecture which is shown in Figure 3.7. The register files, processing element array and adder tree hardware architectures are shown in Figure 3.8.

Inter-view, LT and RT hardware determine disparity (inter-view) vector, LT motion vector and RT motion vector, respectively. LT and RT hardware determine the motion vectors of current MB after inter-view hardware determines the disparity vector of current MB. Therefore, while inter-view hardware is determining the disparity vector of nth MB, LT and RT hardware determine motion vectors of (n-1)th MB.

First, the current MB data is read from off-chip video frame memory and stored into current MB register file. Duplicate candidate vector elimination is done while current MB register file is loaded. Then, for each candidate vector in the candidate vector list, the reference MB data pointed by that candidate vector is read from off-chip video frame memory and stored into reference MB register file. SAD value is calculated between current MB and reference MB pointed by that candidate vector. After the SAD values of all candidate vectors are calculated, the candidate vector with the smallest SAD value is selected as the predicted vector. Then, search range size is calculated, and the pixels in the search range are read from off-chip video frame memory and stored into Block RAMs (BRAM). For each search location in the search range, reference MB data is read from BRAMs and stored into reference MB register file. Then, in each clock cycle, SAD value is calculated between the current MB and a reference MB. After the SAD values of all reference MBs are calculated, the vector pointing to the reference MB with the smallest SAD value is selected as the best motion vector.

Only LT and RT hardware are used for the pictures in first view. They perform temporal predictions of current MB in the [-32, +32] size search range pointed by zero motion vector. Therefore, predicted vector and search range size calculation are not done for the pictures in first view.

Only inter-view hardware is used for the key pictures. The candidate vectors in the future reference frames are not used for predicted vector calculation. The candidate

vectors in the 8th previous reference frames are used for predicted vector calculation. The corresponding MB is not determined for search range calculation. Instead, the MB in the 8th previous frame located in the same position as the current MB in current frame is used as the corresponding MB.

3.2.1 Current MB Register File Loading

The current MB data (16 x 16 pixels = 16 x 16 x 8 bits) is read from off-chip video frame memory and stored into current MB register file in 16 clock cycles. First, bottom 15 rows (16 x 15 pixels) of current MB are read from off-chip video frame memory and stored into temporary current MB register file in 15 clock cycles. In each clock cycle, one row (16 pixels = 16 x 8 bits) of current MB is read and stored into temporary current MB register file. In 16th clock cycle, top row of current MB is read from off-chip video frame memory and stored directly into top row of current MB register file. In the same clock cycle, all 15 rows of temporary current MB register file are stored into bottom 15 rows of current MB register file. In this way, no switching activity occurs in the processing element array while current MB register file is loaded. This reduces the dynamic power consumption of the proposed motion estimation hardware.

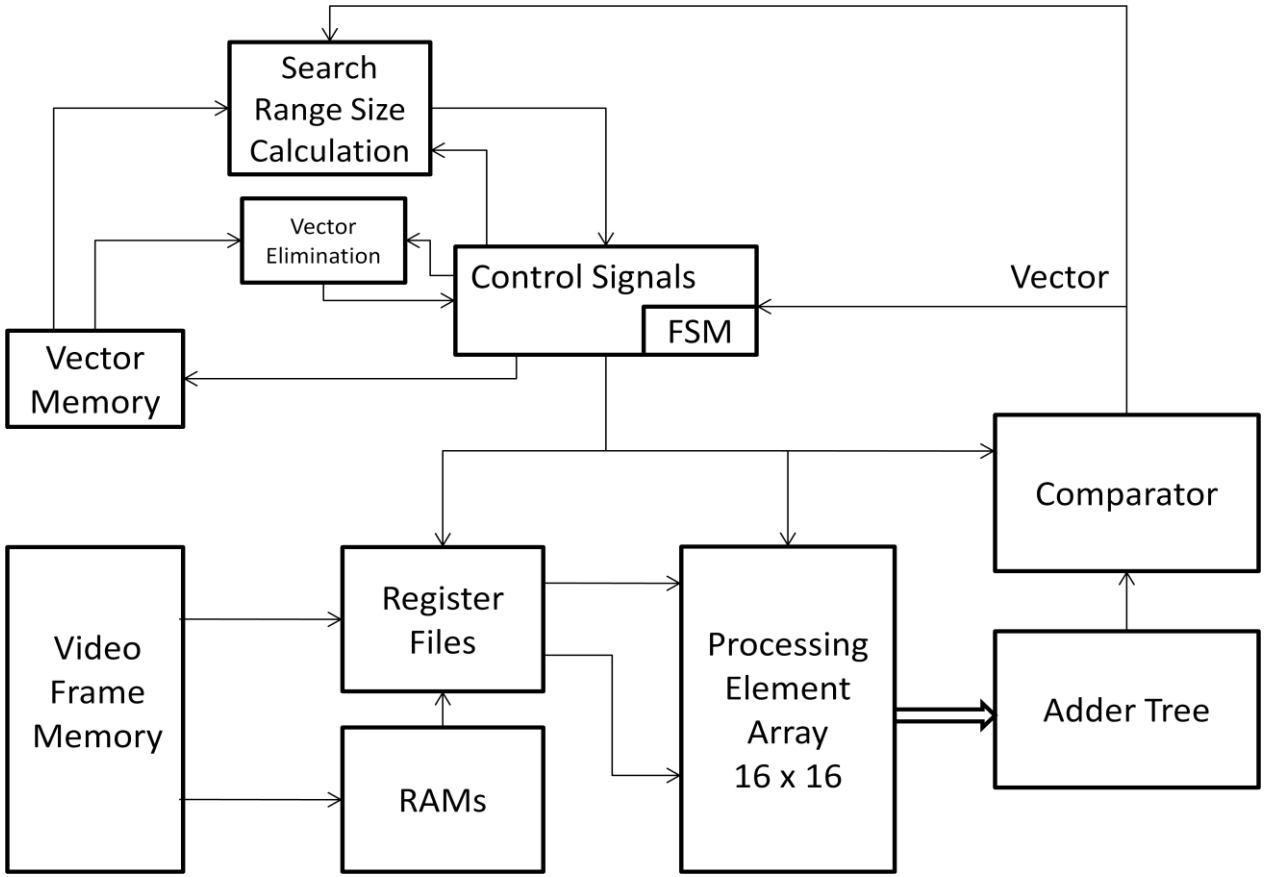

Figure 3.7 Proposed Motion Estimation Hardware

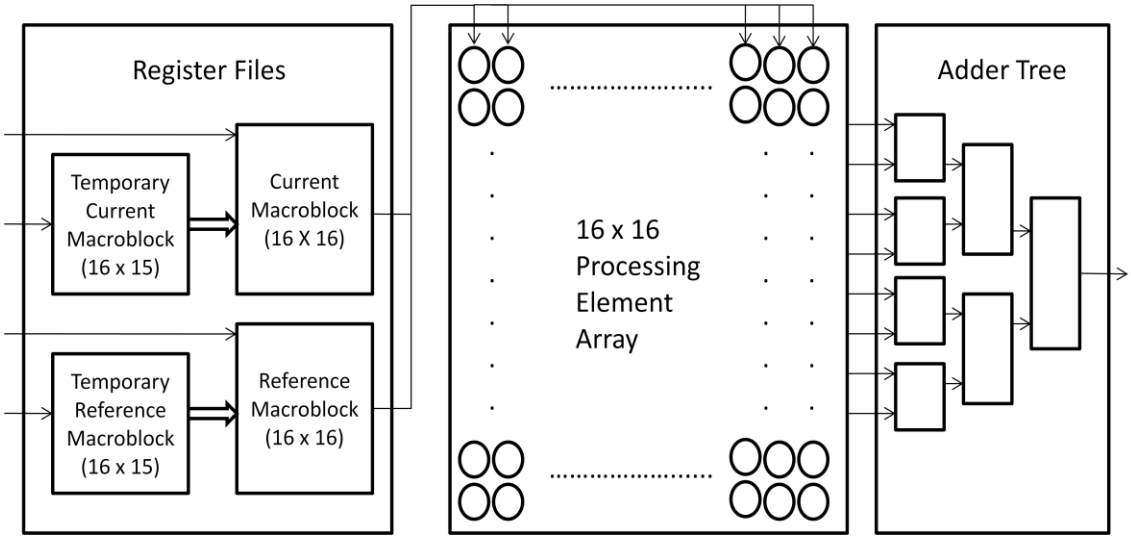

Figure 3.8 Processing Element Array and Adder Tree

3.2.2 Duplicate Candidate Vector Elimination

Duplicate candidate vector elimination is done while current MB register file is loaded. The address of each candidate vector is calculated in one clock cycle. If a MB, whose vector would be used as a candidate vector, is inside the frame, the candidate vector is read from off-chip vector memory. If this candidate vector is pointing inside the frame and it is different from the previous candidate vectors, it is stored into candidate vector list. A 64x64x1 bit table is used to determine whether a candidate vector is already stored into candidate vector list or not. When a candidate vector is stored into candidate vector list, the corresponding entry in the table is set to 1. This table can store this information for the candidate vectors in the range [-32, +32]. In order to determine whether the current candidate vector is different from the previous candidate vectors, the corresponding entry in the table is checked. If it is 1, the current candidate vector is not stored into candidate vector list. In this way, calculating the SAD values of the identical candidate vectors is avoided.

3.2.3 Predicted Vector Calculation

For each candidate vector in the candidate vector list, the reference MB data (16 x 16 pixels = 16 x 16 x 8 bits) pointed by that candidate vector is read from off-chip video frame memory and stored into reference MB register file in 16 clock cycles. First, bottom 15 rows (16 x 15 pixels) of reference MB are read from off-chip video frame memory and stored into temporary reference MB register file in 15 clock cycles. In each clock cycle, one row (16 pixels = 16 x 8 bits) of reference MB is read and stored into temporary reference MB register file. In 16th clock cycle, top row of reference MB is read from off-chip video frame memory and stored directly into top row of reference MB register file. In the same clock cycle, all 15 rows of temporary reference MB register file are stored into bottom 15 rows of reference MB register file. In this way, no switching activity occurs in the processing element array while reference MB register file is loaded. This reduces the dynamic power consumption of the proposed motion estimation hardware.

After reference MB register file is loaded, SAD value is calculated between current MB and reference MB pointed by that candidate vector. Absolute differences between each current MB pixel and reference MB pixel are calculated in the 16x16 processing element array in one clock cycle, and these 256 absolute differences are added in the adder tree in four clock cycles. While the absolute differences are added in the adder tree, reference MB data pointed by the next candidate vector is loaded into reference MB register file. After the SAD values of all candidate vectors are calculated, the candidate vector with the smallest SAD value is selected as the predicted vector.

3.2.4 Search Range Size Calculation

In inter-view, LT and RT hardware, 9 vectors used for search range size calculation are read from off-chip vector memory. The differences between the predicted vector and these vectors are calculated in parallel. 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 6^{th}, 7^{th}, 8^{th}, 9^{th} vector differences are added, and the result is shifted right by 3 bits. This value and 5^{th} vector difference are added, and the result is shifted right by 1 bit. The search range size is set to this value, if it is less than 32. Otherwise, the search range size is set to 32.

3.2.5 Loading Search Range into BRAMs

After the search range size is determined, the pixels in the search range are read from off-chip video frame memory and stored into BRAMs. There are 20 BRAMs in Inter-view, LT and RT hardware. Therefore, there are 60 BRAMs in the proposed motion estimation hardware. The organization of the pixels in the BRAMs when the search range size is [-32, +32] in both x and y directions is shown in Figure 3.9. In the figure, (x, y) show the position of the pixel in the search range. In this case, there are 2 x 32 + 16 = 80 rows and 2 x 32 + 16 = 80 columns in the search range. Each word in a BRAM is 32 bits, and it can store 4 pixels. In each clock cycle, 20 pixels in the search range are read from off-chip video frame memory and stored into one word of 5 BRAMs. If the search range size is [-SRx, +SRx] in x direction and [-SRy, +SRy] in y direction, there are 2 x SRy + 16 rows and 2 x SRx + 16 columns in the search range. Therefore, in this case, (2 x SRy + 16) x (2 x SRx + 16) pixels will be read from off-chip video frame memory and stored into BRAMs.

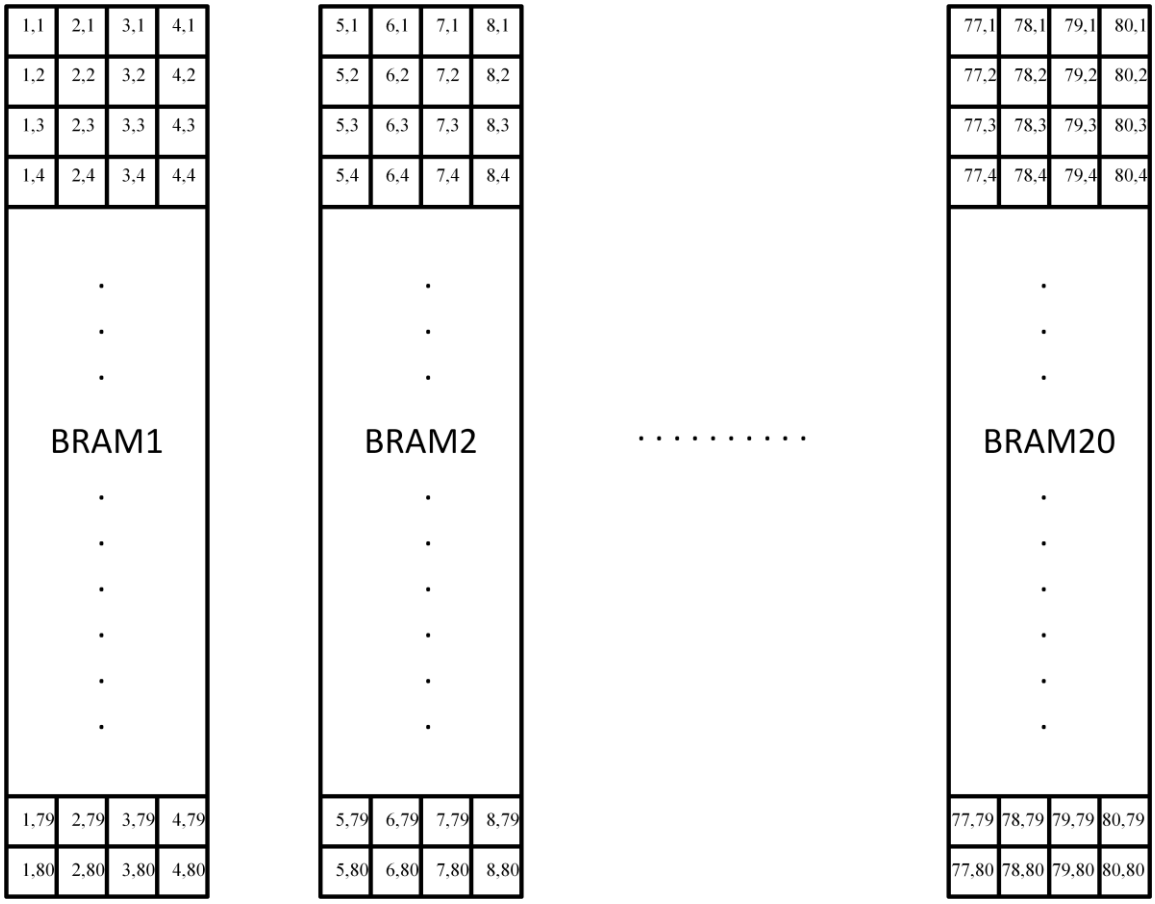

Figure 3.9 BRAM Organization

3.2.6 Reference MB Register File Loading and SAD Calculation

For each search location in the search range, reference MB data is read from BRAMs and stored into reference MB register file. In the first 16 clock cycles, the reference MB data is read from BRAMs and stored into reference MB register file as explained in Predicted Vector Calculation section except that pixels are read from BRAMs instead of off-chip video frame memory. In the following clock cycles, the pixels in the reference MB register file are shifted up, down or right, and only a new row or column of pixels are stored into reference MB register file. In case of down shift, bottom 15 rows in reference MB register file are shifted one row up, a new row of pixels are read from BRAMs and stored into bottom row of reference MB register file. In case of up shift, top 15 rows in reference MB register file are shifted one row down, a new row of pixels are read from BRAMs and stored into top row of reference MB register file. In case of right shift, right 15 columns in reference MB register file are

shifted one column left, a new column of pixels are read from off-chip video frame memory, and stored into right column of reference MB register file.

After reference MB register file is loaded, in each clock cycle, SAD value is calculated between the current MB and a reference MB. Absolute differences between each current MB pixel and reference MB pixel are calculated in the 16x16 processing element array in one clock cycle, and these 256 absolute differences are added in the adder tree in four clock cycles. Since these operations are pipelined, while the absolute differences are added in the adder tree, the next reference MB is loaded to reference MB register file and its SAD value is calculated. After the SAD values of all reference MBs are calculated, the vector pointing to the reference MB with the smallest SAD value is selected as the best motion vector.

3.2.7 Implementation Results

The proposed motion estimation hardware is implemented in Verilog HDL. The Verilog RTL code is verified with RTL simulations using Mentor Graphics Modelsim. The RTL simulation results matched the results of a MATLAB implementation of the proposed motion estimation algorithm. The Verilog RTL code is synthesized and mapped to a Xilinx XC6VLX760 FF760 FPGA with speed grade -2 using Xilinx ISE 13.4. The FPGA implementation is verified with post place & route simulations using Mentor Graphics Modelsim. It consumes 24,678 slices, 65,110 LUTs, 49,021 DFFs and 60 BRAMs, and it works at 60 MHz. The performance of the FPGA implementation for the frames in the third GOP of Ballroom (640 x 480) video sequence is shown in Table 3.16. The FPGA implementation processes the 63 frames in the third GOP in Ballroom (640 x 480) video sequence in 998.6 ms. Therefore, it can process 63 VGA frames per second.

Table 3.16 Performance of the FPGA Implementation

View \ Frame	17	18	19	20	21	22	23	24	Total (ms)
0	Intra	83.2	83.2	83.2	83.2	83.2	83.2	83.2	582.4
1	7.7	8.8	10.1	8.9	12.7	9.4	10.3	10.2	78.1
2	5.5	6.5	7.8	6.6	11.4	7	8.5	7.3	60.6
3	5	6.1	6.9	6.4	12.8	6.4	8.1	6.4	58.1
4	5.3	5.7	7.1	5.8	11.6	6.1	8.6	6.4	56.6
5	5.4	5.9	6.6	6.1	9.2	6.5	7.8	6.8	54.3
6	7.4	6.6	7.3	6.6	9.7	6.9	7.8	6.6	58.9
7	5.1	5.2	6.8	5.4	9.5	5.1	6.9	5.6	49.6

We estimated the power consumption of the FPGA implementation using Xilinx XPower tool. In order to estimate the power consumption of the proposed motion estimation hardware, timing simulation of its post place & route netlist is done using Mentor Graphics ModelSim. The signal activities of these timing simulations are stored in VCD files, and these VCD files are used for estimating the power consumption of the proposed motion estimation hardware using Xilinx XPower tool. Since motion estimation hardware is used as part of an H.264 multiview video encoder, only internal power consumption is considered, and input and output power consumptions are ignored. Therefore, the power consumption of the proposed motion estimation hardware can be divided into four main categories; clock power, logic power, signal power and BRAM power.

The power consumption of the FPGA implementation for one fourth of all MBs in fifth frame in second view of first GOP in Ballroom (640 x 480) video sequence is shown in Table 3.17. The FPGA implementation processes fifth frame in second view of first GOP in 7.8 ms. Energy consumption comparison of motion estimation hardware is shown in Table 3.18. The proposed motion estimation hardware has up to 95% less energy consumption than the full search motion estimation hardware for H.264 MVC with search range [-32, +32], and it has up to 81% less energy consumption than the full search motion estimation hardware for H.264 MVC including the computation reduction techniques proposed in Chapter II [17].

Table 3.17 Power Consumption

Clock	250 mW
Logic	36 mW
Signals	60 mW
BRAMs	93 mW
Total	439 mW

Table 3.18 Energy Consumption Comparison of Motion Estimation Hardware

	Average Power (mW)	**Time (µs)**	**Energy (mj)**	**Energy Reduction (%)**
Full Search Motion Estimation Hardware	1489.62	10079	15.41	0
[17]	1529.82	2901	4.32	71.97
Proposed Motion Estimation Hardware	439	1865	0.82	95.32

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

In this thesis, we proposed novel techniques for significantly reducing the amount of computations performed by full search motion estimation algorithm for H.264 MVC, and therefore significantly reducing the energy consumption of full search motion estimation hardware for H.264 MVC with very small PSNR loss and bitrate increase. The experimental results obtained by Joint Multiview Video Coding (JMVC) 3.01 H.264 MVC software showed that the proposed techniques reduced the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC up to 66% with very small PSNR loss and bitrate increase.

We also proposed an adaptive fast motion estimation algorithm for reducing the amount of computations performed by temporal and inter-view predictions in H.264 MVC motion estimation, and therefore reducing the energy consumption of H.264 MVC motion estimation hardware even more with additional very small PSNR loss and bitrate increase. The experimental results obtained by Joint Multiview Video Coding (JMVC) 3.01 H.264 MVC software showed that the proposed motion estimation algorithm reduced the amount of computations performed by temporal and inter-view predictions in full search motion estimation algorithm for H.264 MVC up to 86% with very small PSNR loss and bitrate increase.

We also proposed an adaptive H.264 MVC motion estimation hardware for implementing the proposed adaptive fast motion estimation algorithm. The proposed

motion estimation hardware is implemented in Verilog HDL and mapped to a Xilinx Virtex-6 FPGA. The proposed motion estimation hardware has up to 95% less energy consumption than the full search motion estimation hardware for H.264 MVC, and up to 81% less energy consumption than the full search motion estimation hardware for H.264 MVC including the proposed computation reduction techniques [17].

As future work, the proposed adaptive fast motion estimation algorithm can be improved to further reduce its computational complexity with additional small PSNR loss and bitrate increase. The energy consumption of the proposed adaptive H.264 MVC motion estimation hardware can be further reduced by using energy reduction techniques such as clock gating.

BIBLIOGRAPHY

[1] Christos Grecos, "Editorial of Special Issue on Real-Time Aspects of the H.264 Family of Standards", *Journal of Real-Time Image Processing*, 4(1), 1-2 (2009)

[2] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services", *ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC)* (2010)

[3] ISO/IEC JTC1/SC29/WG11, Text of ISO/IEC 14496-10:200X/FDAM 1 "Multiview Video Coding", *Doc N9978*, Hannover, Germany (2008)

[4] P. Merkle, K. Muller, T. Wiegand, 3D Video: "Acquisition, Coding, and Display", *IEEE Trans. on Consumer Electronics*, 56(2), 946-950 (2010)

[5] A. Vetro, T. Wiegand, G. J. Sullivan, "Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard*", Proceedings of the IEEE*, 99(4), 626-642 (2011)

[6] P. Merkle, A. Smolic, K. Muller, T. Wiegand, "Efficient Prediction Structures for Multiview Video Coding", *IEEE Trans. on CAS for Video Tech*, 17(11), 1461-1473 (2007)

[7] Y. Aksehir, K. Erdayandi, T. Z. Ozcan, I. Hamzaoglu, "A Low Energy Adaptive Motion Estimation Hardware for H.264 Multiview Video Coding", *Proceedings of the Conference on Design and Architectures for Signal and Image Processing*, pp. 1-6 (2012)

[8] http://wftp3.itu.int/av-arch/jvt-site/2009_01_Geneva

[9] http://www.merl.com/pub/avetro/mvc-testseq/orig-yuv

[10] O. Tasdizen, H. Kukner, A. Akin, I. Hamzaoglu," Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation", *IEEE Trans. on Consumer Electronics*, 55(3), 1645-1653 (2009)

[11] A. Aysu, G. Sayilar, I. Hamzaoglu, “A Low Energy Adaptive Hardware for H.264 Multiple Reference Frame Motion Estimation”, *IEEE Trans. on Consumer Electronics*, 57(3), 1377-1383 (2011)

[12] G. Sanchez, F. Sampaio, M. Porto, S. Bampi, L. Agostini, “DMPDS: A Fast Motion Estimation Algorithm Targeting High Resolution Videos and Its FPGA Implementation”, *Int. Journal of Reconfigurable Computing* (2012)

[13] B. Zatt, M. Shafique, S. Bampi, J. Henkel, “Multi-Level Pipelined Parallel Hardware Architecture for High Throughput Motion and Disparity Estimation in Multiview Video Coding*”, Proceedings of the DATE Conference*, pp. 1-6 (2011)

[14] B. Zatt, M. Shafique, S. Bampi, J. Henkel, “A Low-Power Memory Architecture with Application-Aware Power Management for Motion & Disparity Estimation in Multiview Video Coding”, *Proceedings of the IEEE/ACM ICCAD*, pp. 40-47 (2011)

[15] F. Sampaio, B. Zatt, S. Bampi, L. Agostini, “Memory Efficient FPGA Implementation of Motion and Disparity Estimation for the Multiview Video Coding”, *Proceedings of the VIII Southern Conference on Programmable Logic*, pp. 1-6 (2012)

[16] C. Kalaycioglu, O. C. Ulusel, I. Hamzaoglu, “Low Power Techniques for Motion Estimation Hardware”, *Proceedings of the Int. Conference on Field Programmable Logic*, pp. 180-185 (2009)

[17] Y. Aksehir, K. Erdayandi, T. Z. Ozcan, I. Hamzaoglu, “Low Energy Adaptive Motion Estimation Hardware for H.264 Multiview Video Coding”, *Journal of Real Time Image Processing* (2013)