LOW POWER MOTION ESTIMATION HARDWARE DESIGNS

by
ONUR CAN ULUSEL

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

Sabancı University
Spring 2010

LOW POWER MOTION ESTIMATION HARDWARE DESIGNS

APPROVED BY:

Assist. Prof. Dr. İlker Hamzaoğlu (Thesis Supervisor)	…….
Assist. Prof. Dr. Ayhan Bozkurt	…….
Assoc. Prof. Dr. Erkay Savaş	…….
Assist. Prof. Dr. Müjdat Çetin	…….
Dr. Mustafa Parlak	…….

DATE OF APPROVAL: …….

LOW POWER MOTION ESTIMATION HARDWARE DESIGNS

Onur Can Ulusel

EE, Master Thesis, 2010

Thesis Supervisor: Assist. Prof. Dr. İlker Hamzaoğlu

ABSTRACT

Motion Estimation (ME) is the most computationally intensive and most power consuming part of video compression and video enhancement systems. ME is used in video compression standards such as H.264/MPEG-4 and it is used in video enhancement algorithms such as frame rate conversion and de-interlacing. Half pixel (HP) ME increases the video coding efficiency at the expense of increased computational complexity. Therefore, in this thesis, we designed and implemented efficient integer pixel (IP) ME hardware implementing full search ME algorithm, and we proposed techniques for reducing the dynamic power consumptions of IP and HP ME hardware. The proposed ME hardware architectures are implemented in Verilog HDL and mapped to Xilinx FPGAs. The FPGA implementations are verified with post place & route simulations.

We proposed comparison prediction (CP) technique for reducing the power consumption of IP block matching (BM) ME hardware. CP technique reduces the power consumption of absolute difference operations performed by IP BM ME hardware. The proposed technique can easily be used in all IP BM ME hardware. It reduced the power consumption of a fixed block size IP BM ME hardware implementing full search algorithm by 9.3% with 0.04% PSNR loss on a Xilinx XC2VP30-7 FPGA.

We also proposed two techniques for reducing the power consumption of H.264 HP ME hardware. The first technique is vector dependent sum of absolute difference (SAD) reuse which reduces the amount of computations for variable block size H.264 HP ME with no PSNR loss. The second technique is a novel modification of the HP search algorithm which adaptively tries to use the IP motion vector trajectories to reduce HP search to 1-D. This technique causes an average PSNR loss of 0.36 dB. The two techniques reduced the power consumption of a variable block size H.264 HP ME hardware by 6% and 31% on a Xilinx Virtex 6 FPGA respectively.

DÜŞÜK GÜÇ KULLANIMLI HAREKET TAHMİNİ DONANIMLARI

Onur Can Ulusel

EE, Yüksek Lisans Tezi, 2010

Tez Danışmanı: Yard. Doç. Dr. İlker Hamzaoğlu

ÖZET

Hareket Tahmini (HT) video sıkıştırma ve video iyileştirme sistemlerinin en çok işlem yapılan ve en çok güç harcayan kısmıdır. HT, H.264/MPEG-4 gibi video sıkıştırma standartlarında ve çerçeve hızı dönüştürme gibi video iyileştirme uygulamalarında kullanılır. Yarım piksel hassaslığında (YPH) HT video kodlama verimini arttırmakla birlikte yapılan işlem miktarını da arttırır. Bu nedenle bu tezde, tam arama HT algoritmasını kullanan verimli tam sayı hassaslığında (TSH) HT donanımları tasarladık ve gerçekleştirdik. Ayrıca TSH ve YPH HT donanımları için güç azaltma teknikleri önerdik. Önerilen HT donanımları Verilog HDL dili kullanılarak gerçekleştirildiler ve Xilinx FPGA'lerine yerleştirildiler.

Blok eşleştirme (BE) HT donanımlarının güç kullanımını azaltmak için karşılaştırma öngörüsü (KÖ) tekniğini önerdik. KÖ tekniği BE HT donanımlarında yapılan mutlak fark işleminin güç kullanımını azaltır. KÖ tekniği tüm BE HT donanımlarına kolayca uygulanabilir. Bu tezde ise sabit blok boyutlu 256 işlem birimli BE HT donanımına uygulandı.

Ayrıca, H.264 YPH HT için iki güç azaltma tekniği önerdik. Bu tekniklerin bir YPH HT donanımının güç kullanımını olan etkilerini gösterdik. Birinci teknik hareket vektörlerine bağlı olarak mutlak farklar toplamlarının (MFT) yeniden kullanımı tekniğidir. Bu teknik değişken blok boyutlu HT için yapılan işlem miktarını PSNR kaybı olmadan azaltmaktadır. İkinci teknik YPH HT algoritmasında yapılan özgün bir değişikliktir. Bu teknik tam sayı hareket vektörlerinin uzantılarını kullanarak YP arama penceresini uyarlanır bir şekilde tek boyuta indirmektedir. Bu teknik değişken blok boyutlu HT için yapılan işlem miktarını az bir PSNR kaybı ile azaltmaktadır.

ACKNOWLEDGEMENTS

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

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

I would like to thank to all members of System-on-Chip Design and Testing Lab, Aydın Aysu, Abdülkadir Akın, Yusuf Adıbelli, Çağlar Kalaycıoğlu, Zafer Özcan, Mert Çetin, and Murat Can Kıral who have been greatly supportive during my study.

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

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

TABLE OF CONTENTS

ABSTRACT……..……IV

ÖZET……..V

ACKNOWLEDGEMENTS…….……VI

TABLE OF CONTENTS…….....VII

LIST OF FIGURES……..……IX

LIST OF TABLES….....X

ABBREVIATIONS….....XI

1 INTRODUCTION…….……..1

1.1 Thesis Contributions ……. 4

2 FULL SEARCH MOTION ESTIMATION HARDWARE DESIGNS …….6

2.1 Motion Estimation Hardware with 256 Processing Elements ……... 6

2.1 Motion Estimation Hardware with 64 Processing Elements ……... 10

3 POWER REDUCTION TECHNIQUE FOR INTEGER PIXEL MOTION ESTIMATION HARDWARE…….…….……....14

3.1 Absolute Difference Hardware ……..... 14

3.2 Comparison Prediction Technique …….. 16

3.3 Implementation Results ……... 18

4 POWER REDUCTION TECHNIQUES FOR H.264 HALF-PIXEL MOTION ESTIMATION HARDWARE……..23

4.1 Half-Pixel Motion Estimation Algorithm and Hardware……..23

4.2 Vector Dependent SAD Reuse Technique……26

4.3 Integer-Pixel Motion Vector Trajectory Based Adaptive Half-Pixel Motion Estimation Algorithm ……28

4.4 Implementation Results……33

5 CONCLUSIONS AND FUTURE WORK…35

6 REFERENCES……36

LIST OF FIGURES

Figure 1.1 Motion Estimation…….2
Figure 2.1 256 PE VBS ME Hardware Architecture …….6
Figure 2.2 PE Architecture ……...……..7
Figure 2.3 (a) Vertical Search Flow (b) Zigzag Search Flow ……8
Figure 2.4 64 PE VBS ME Hardware Architecture ……...……..11
Figure 3.1 Standard Absolute Difference Hardware ……15
Figure 3.2 (a) Reset based Absolute Difference Prediction Hardware (b) Enable based Absolute Difference Prediction Hardware …….……..16
Figure 3.3 Checkerboard Pattern for Standard (S) and Predicted (P) Absolute Difference Hardware ……..18
Figure 3.4 Rate Distortion Curves for a) Mother & Daughter and b) Mobile Video Sequences ……...20
Figure 4.1 Half Pixel Search Locations …….……..……23
Figure 4.2 Half Pixel Interpolation ….....24
Figure 4.3 Variable Block Size Modes ……....25
Figure 4.4 Half Pixel Interpolation Hardware for 4x4, 4x8 and 8x4 Sub Blocks……26
Figure 4.5 Half Pixel Interpolation for 4x4 Sub Blocks ……..29
Figure 4.6 Half Pixel Search Locations for Each Integer Pixel ….....30
Figure 4.7 Vector Trajectory Estimations for (a) Zero, (b) Twice Bigger Than, and (c) Bigger Than Criteria …….……...……31

LIST OF TABLES

Table 2.1 Dataflow of 256 PE VBS ME Hardware ……..... 8

Table 2.2 The FPGA Resource Usage and the Maximum Clock Frequency of 256 PE VBS ME Hardware…….……...10

Table 2.3 Data Flow of 1st PE Column in 64 PE VBS ME Hardware …..……..12

Table 2.4 The FPGA Resource Usage and the Maximum Clock Frequency of 64 PE VBS ME Hardware ……..13

Table 3.1 Average PSNR and Bit Rate for Several Video Sequences …….19

Table 3.2 Comparison of Motion Estimation Hardware Architectures ……...22

Table 4.1 Average Computation Reduction Obtained from a Foreman CIF (352x288) Video Sequence Using Vector Dependant SAD Reuse Technique ……27

Table 4.2 Comparison of Estimation Methods (Percentage of 1-D Search and the Resulting PSNR) ……...…….. 32

Table 4.3 Area and Power Comparison of Proposed Vector Dependent SAD Reuse (VDSR) and Vector Trajectory Based Search (VTBS) Techniques .…….34

ABBREVIATIONS

ASIC	Application Specific Integrated Circuit
BM	Block Matching
BRAM	Block Random Access Memory
BRF	Best Reference Frame
CAVLC	Context Adaptive Variable Length Coding
CIF	Common Intermediate Format
CP	Comparison Prediction
DVD	Digital Versatile Disc
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FS	Full Search
FSM	Finite State Machine
HDL	Hardware Description Language
HDTV	High Definition Television
HP	Half-Pixel
IP	Integer-Pixel
ISO/IEC	International Standards Organization, International Electrotechnical Commission
ITU-T	International Telecommunications Union, Telecommunications Standardization Sector
JVT	Joint Video Team
LUT	Look-up Table
MB	Macroblock
ME	Motion Estimation
MPEG	Motion Picture Experts Group
MV	Motion Vector
NAL	Network Abstraction Layer

PE	Processing Element
PSNR	Peak Signal Noise Ratio
PVT	Process Voltage Temperature
QP	Quantization Parameter
R-D	Rate-Distortion
RTL	Register Transfer Level
SAD	Sum of Absolute Differences
VBS	Variable Block Size
VCD	Value Change Dump
VGA	Video Graphics Array

CHAPTER I

INTRODUCTION

Motion Estimation (ME) is the most computationally intensive part of video compression and video enhancement systems. ME is used to reduce the bit-rate in video compression systems by exploiting the temporal redundancy between successive frames, and it is used to enhance the quality of displayed images in video enhancement systems by extracting the true motion information. ME is used in video compression standards such as MPEG4 and H.264 [1], and in video enhancement algorithms such as frame rate conversion [2, 3].

Block Matching (BM) is the most preferred method for ME. BM ME partitions current frame into non-overlapping NxN rectangular blocks, and it tries to find the block from the reference frame in a given search window that best matches the current block in the current frame. Sum of Absolute Differences (SAD) is the most preferred block matching criterion. As shown in Figure 1.1, the location of a block in a frame is given using the (x,y) coordinates of top-left corner of this block. The search window in the reference frame is the [-p, p] size region around the location of the current block in the current frame. The SAD value for a current block in the current frame and a candidate block in the reference frame is calculated by adding the absolute differences of corresponding pixels in the two blocks as shown in the formula (1.1). In this formula, B_{mxn} is a block of size mxn, **d**=*(dx, dy)* is the motion vector, *c* and *r* are current and reference frames respectively. Since a motion vector expresses the relative motion of the current block in the reference frame, motion vectors are specified in relative coordinates. If the location of the best matching block in the reference frame is (x+u, y+v), then the motion vector is expressed as (u,v).

$$SAD_{B_{mxn}}(\mathbf{d}) = \sum_{x=1,y=1}^{m,n} \left| c(x,y) - r(x + d_x, y + d_y) \right| \qquad (1.1)$$

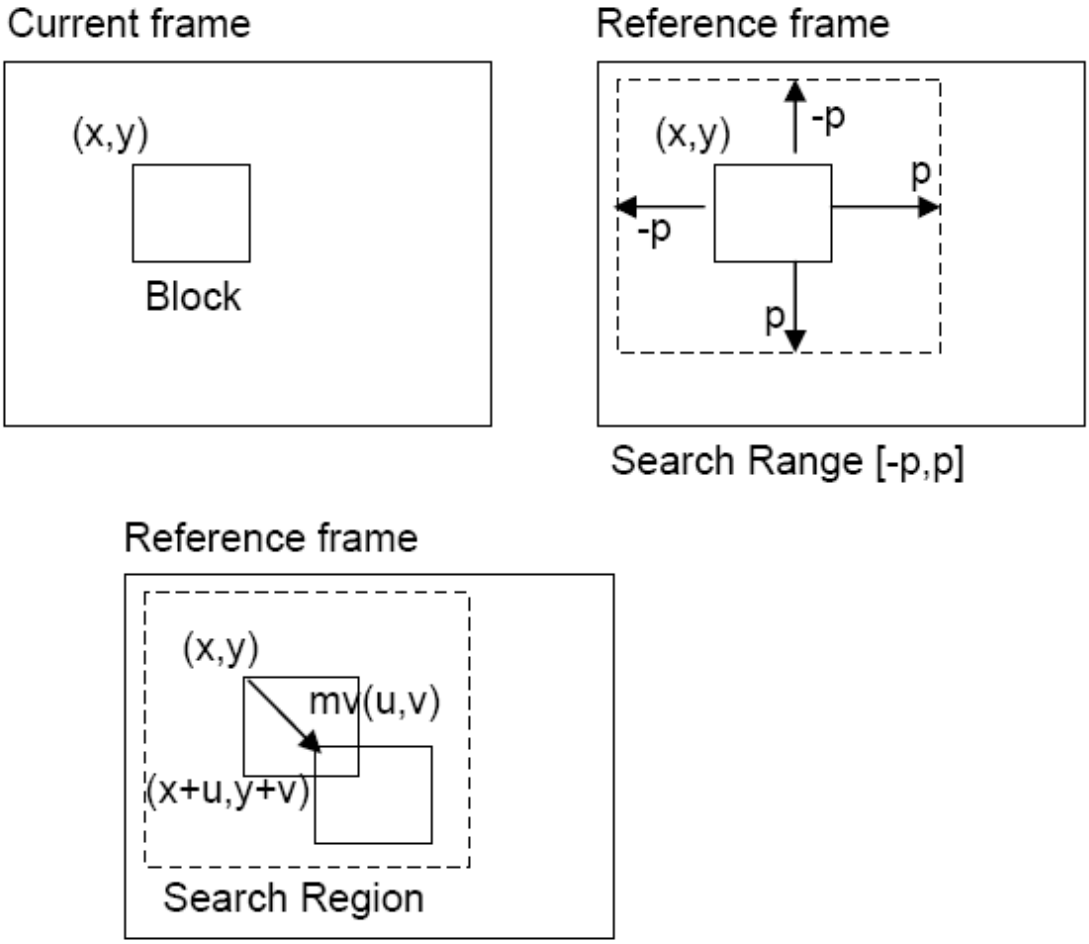


Figure 1.1 Motion Estimation [1]

ME is performed on the luminance (Y) component of a YUV image and the resulting motion vectors are also used for the chrominance (U and V) components. Full Search (FS) ME algorithm finds the reference block that best matches the current block by computing the SAD values for all search locations in a given search range. Therefore, FS algorithm achieves the best performance among BM ME algorithms, but its computational complexity is high.

In fixed block size (FBS) ME, only the motion vectors for 16x16 MBs are computed. In order to improve the ME performance, variable block size (VBS) ME is used in the H.264 standard. But, the computational complexity of FS algorithm for VBS ME is even higher [4, 5]. In VBS ME, each 16x16 MB can be divided into sub-blocks in four different ways; 16x16, 16x8, 8x16 or 8x8. Each 8x8 sub-block can be further divided into 8x8, 8x4, 4x8 or 4x4 sub-blocks. Each 8x8 sub-block can be divided into different size sub-blocks.

Several fast search ME algorithms, such as New Three Step Search [6], Diamond Search [7], Hexagon-Based Search [8], and Adaptive Dual Cross Search [9], are proposed to reduce the computational complexity of FS algorithm. These algorithms try to approach the PSNR of FS algorithm by computing the SAD values for fewer search locations in a given search range. Several hardware architectures for fast search ME algorithms are proposed in the literature [10, 11].

Although many fast search ME algorithms are developed, FS algorithm has remained a popular candidate for hardware implementation because of its regular dataflow and good compression performance [12, 13].

In order to further improve the performance of integer pixel (IP) ME, half-pixel (HP) accurate VBS ME is performed [14, 15]. However, the amount of computation required by HP VBS ME is even more than the amount required by IP VBS ME. Therefore, this coding gain comes with an increase in encoding complexity and hence with an increase in the power consumption of HP VBS ME hardware. Several techniques for reducing the amount of computation by HP ME are proposed in the literature [16 - 21]. Some of these papers present efficient HP ME hardware implementations with efficient memory usage [16, 17]. An interpolation-free HP ME is proposed in [18]. Fast HP ME algorithms using reduced number of HP search locations are proposed in [19 - 21].

Multimedia applications running on portable devices increased recently and this trend is expected to continue in the future. Since portable devices operate on battery, it is important to reduce power consumption so that battery life can be increased. Therefore, power consumption has become a critical design metric for portable applications.

Due to low non-recurring engineering (NRE) costs, Field Programmable Gate Arrays (FPGAs) are ideal solutions for low-to-mid volume designs [22]. However, FPGAs 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, power consumption is an even more important design metric for FPGA implementations, and designers should consider the

impact of their design decisions not only on speed and area, but also on power consumption throughout the entire design process [23].

1.1 Thesis Contributions

In this thesis, we propose two VBS ME hardware architectures which use 256 Processing Elements (PE) and 64 PEs respectively. The proposed VBS ME hardware architectures perform FS ME. 64 PE ME hardware has lower area than 256 PE ME hardware. However, 256 PE ME hardware is faster than the 64 PE ME hardware.

In addition, we propose a comparison prediction (CP) technique for reducing the power consumption of BM ME hardware by reducing the power consumption of absolute difference (AD) operations. CP technique replaces the 8-bit comparator in AD hardware with a D Flip-Flop (DFF) and 1-bit inverter. CP technique can easily be used in all BM ME hardware. In this thesis, it is applied to FBS version of the proposed 256 PE VBS ME hardware. We quantified the impact of the proposed power reduction technique on the power consumption and Peak Signal-to-Noise Ratio (PSNR) obtained by this ME hardware. It reduced the average dynamic power consumption of this ME hardware by 6.1% with 0.01% PSNR loss and by 9.3% with 0.04% PSNR loss on a XC2VP30-7 FPGA.

Finally, we propose two power reduction techniques for H.264 HP ME hardware and show their impact on the power consumption of a VBS HP ME hardware. Both techniques can be easily integrated to any HP ME hardware. The first technique is vector dependent SAD reuse. It reduces the amount of computations for VBS HP ME with no PSNR loss by taking advantage of sub-blocks with equal motion vectors (MV) to calculate the SADs of larger sub-blocks using the SADs of smaller sub-blocks. The second technique is a novel modification of the HP search algorithm which adaptively tries to use the IP MV trajectories to reduce HP search to 1-D. This technique causes an average PSNR loss of 0.36 dB. We integrated both techniques to the H.264 VBS HP ME hardware proposed in [24]. The two techniques reduced the power consumption of this VBS HP ME hardware by 6% and 31% on a Xilinx Virtex 6 FPGA respectively.

The rest of the thesis is organized as follows;

Chapter II presents 256 PE ME Hardware and 64 PE ME Hardware.

Chapter III explains the proposed comparison prediction technique for reducing the power consumption of BM ME hardware, and presents its impact on the power consumption of the proposed 256 PE ME hardware.

Chapter IV explains two power reduction techniques for H.264 HP ME hardware, and presents their impact on the power consumption of a VBS HP ME hardware.

Chapter V presents the conclusions and possible directions for future work.

CHAPTER II

FULL SEARCH MOTION ESTIMATION HARDWARE DESIGNS

2.1 Motion Estimation Hardware with 256 Processing Elements

The block diagram of the proposed ME hardware architecture for implementing VBS FS ME algorithm using 256 PEs is shown in Figure 2.1. FBS version of this ME hardware architecture is also implemented. In the proposed architecture, a 2-D systolic PE array is used and all the PEs are capable of shifting data down, up and left. Each circle in the figure represents a PE. This ME hardware finds the MVs of an MB in a search range of [-16, 15] pixels.

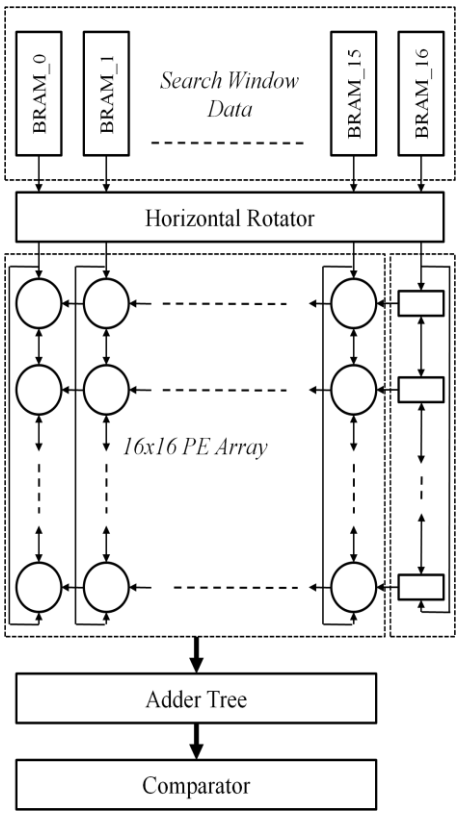


Figure 2.1 256 PE VBS ME Hardware Architecture

The architecture of a PE is shown in Figure 2.2. Each PE calculates the absolute difference between a pixel in the current MB and a pixel in the search window. The SAD of a search location is calculated by adding the absolute differences calculated by PEs using an adder tree. This ME hardware is highly pipelined and its latency is eight clock cycles; one cycle for synchronous read from BRAM, one cycle for horizontal shifting, one cycle for SAD computation in 2-D systolic PE array, two cycles for the adder tree generating 4x4 SADs and three cycles for the adder tree generating 41 MVs for 7 different block sizes.

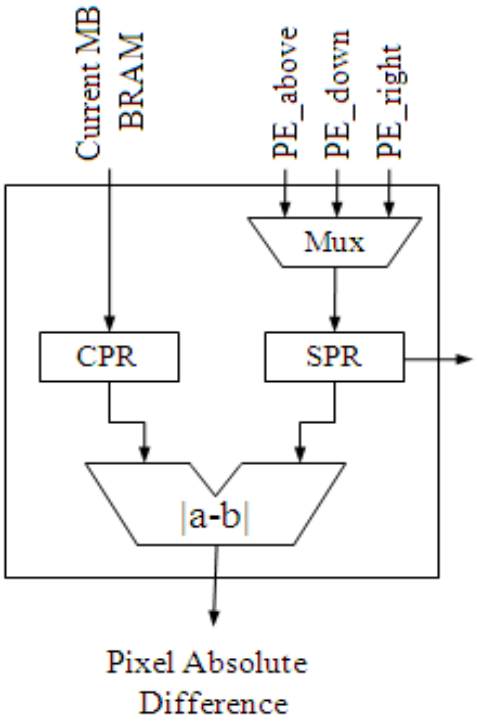


Figure 2.2 PE Architecture

The proposed 256 PE VBS ME hardware searches the search locations in a search window column by column in a zigzag pattern. Most of the proposed ME hardware architectures using 256 PEs use a vertical search flow and when the end of a column is reached the search location at the top of the next column is searched as shown in Figure 2.3 (a). Therefore, it is required to either broadcast multiple pixels into the PEs [25] or delay all the PEs until they are filled. There are 256 PE ME hardware architectures using a 2-D systolic PE array and searching the search window in a zigzag pattern as shown in Figure 2.3 (b) [26]. However, these architectures either use both row and column aligned memories or use data duplication. The proposed ME hardware architecture overcomes this problem by using a pipeline of 16 8-bit temporary registers.

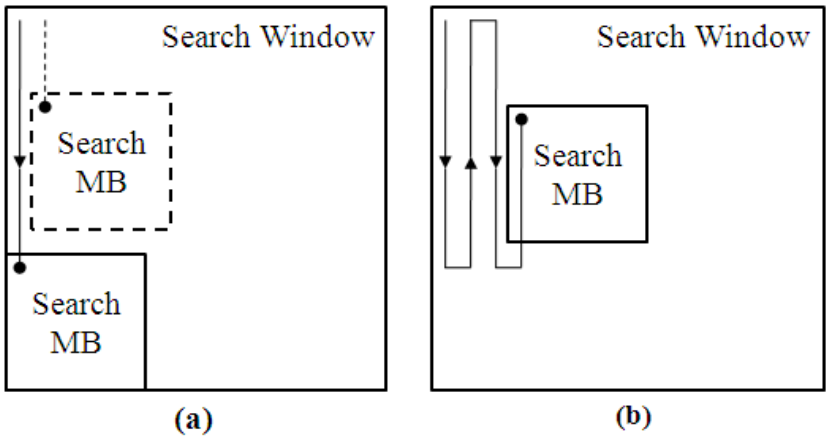


Figure 2.3 (a) Vertical Search Flow (b) Zigzag Search Flow

In the proposed 256 PE VBS ME hardware, the search starts at top left search location of the search window and proceeds down until the last search location of this column is searched. Then, the search continuous with the last search location of the next column and proceeds up until the first search location of this column is searched. Only 16 new search window pixels are required by the PE array in each cycle to calculate the SAD of the next search location regardless of its position in the search window.

Table 2.1 Dataflow of 256 PE VBS ME Hardware

Clock	1st Column				…	16th Column				Temp Column			
	PE(0,15)	PE(0,14)	…	PE(0,0)		PE(15,15)	PE(15,14)	…	PE(15,0)	Reg15	Reg14	…	Reg0
0	S(0,0)	*nop*		*nop*		S(15,0)	*nop*		*nop*	S(16,0)	*nop*		*nop*
1	S(0,1)	S(0,0)	…	*nop*	…	S(15,1)	S(15,0)	…	*nop*	S(16,1)	S(16,0)	…	*nop*
…	…	…		…		…	…		…	…	…		…
15	S(0,15)	S(0,14)		S(0,0)		S(15,15)	S(15,14)		S(15,0)	S(16,15)	S(16,14)		S(16,0)
16	S(0,16)	S(0,15)		S(0,1)		S(15,16)	S(15,15)		S(15,1)	S(16,16)	S(16,15)		S(16,1)
17	S(0,17)	S(0,16)	…	S(0,2)	…	S(15,17)	S(15,16)	…	S(15,2)	S(16,17)	S(16,16)	…	S(16,2)
…	…	…		…		…	…		…	…	…		…
46	S(0,46)	S(0,45)		S(0,31)		S(15,46)	S(15,45)		S(15,31)	S(16,46)	S(16,45)		S(16,31)
47	S(1,46)	S(1,45)		S(1,31)		S(16,46)	S(16,45)		S(16,31)	*nop*	*nop*		*nop*
48	S(1,45)	S(1,44)	…	S(1,30)	…	S(16,45)	S(16,44)	…	S(16,30)	*nop*	*nop*	…	S(17,30)
…	…	…		…		…	…		…	…	…		…
78	S(1,15)	S(1,14)		S(1,0)		S(16,15)	S(16,14)		S(16,0)	S(17,15)	S(17,14)		S(17,0)
…						…							
1007	S(31,46)	S(31,45)		S(31,31)		S(46,46)	S(46,45)		S(46,31)	*nop*	*nop*		*nop*
1008	S(31,45)	S(31,44)	…	S(31,30)	…	S(46,45)	S(46,44)	…	S(46,30)	*nop*	*nop*	…	*nop*
…	…	…		…		…	…		…	…	…		…
1038	S(31,15)	S(31,14)		S(31,0)		S(46,15)	S(46,14)		S(46,0)	*nop*	*nop*		*nop*

The data flow of the PEs is shown in Table 2.1 where S(x, y) is a search window pixel. Current MB pixels are not shown in the table, because each PE stores the same current MB pixel (e.g. PE(0,0) stores C(0,0)) while searching all the search locations in a search window. The PE array is filled in the first 15 cycles. While searching the search locations in the first column of the search window, in each cycle, vertical up shift is performed in the PE array and all the PEs except the ones in the last row are provided search window pixels from their neighboring PEs. PEs in the last row of PE array, in each cycle, read 16 new search window pixels from 16 BRAMs.

The 17th BRAM is used to be able to perform a left shift in the PE array after all the search locations in a column are searched. The 17th BRAM is connected to the temporary registers and by the time there is a need for left shift, the pixels needed for the right most PEs in the PE array become ready in these temporary registers. After the search locations in the first column are searched, a left shift is performed in the PE array while the PEs in the 16th column of the PE array receive search window pixels from the temporary registers.

While searching the search locations in the second column of the search window, in each cycle, vertical down shift is performed in the PE array and all the PEs except the ones in the first row are provided search window pixels from their neighboring PEs. PEs in the first row of PE array, in each cycle, read 16 new search window pixels from 16 BRAMs.

Each BRAM stores the pixels in every 17th column of the search window, e.g. the first BRAM stores the pixels in the 1st, 18th and 35th columns. The order of the search window pixels read from the BRAMs is static. However, the order of the search window pixels required by the PE array and the temporary registers varies depending on the column being processed. This problem is solved by reordering the 16+1 pixels in a search MB row by the horizontal rotator hardware.

The proposed 256 PE VBS ME hardware is implemented in Verilog HDL. The Verilog RTL code is synthesized to a XC2VP30 Xilinx Virtex II Pro FPGA with speed grade 7 using Mentor Graphics Precision RTL tool. The resulting netlist is placed and routed to the same FPGA using Xilinx ISE tool. The FPGA resource usage and the maximum clock frequency of the placed & routed design are given in Table 2.2. The ME hardware takes 1091 clock cycles to process a MB. Therefore, it can process a VGA (640x480) frame in 11.23 ms

(1200 MBs x 1091 clock cycles per MB x 8.578 ns clock cycle = 11.23 ms). Therefore, it can process 1000/11.23 = 89 VGA frames per second.

Table 2.2 The FPGA Resource Usage and the Maximum Clock Frequency of 256 PE VBS ME Hardware

Function Generators	17252	62.98%
CLB Slices	8626	62.98%
DFFs	8736	29.79%
BRAMs	17	12.5%
Frequency (MHz)	113.23	

2.2 Motion Estimation Hardware with 64 Processing Elements

The proposed 64 PE VBS ME hardware architecture is shown in Figure 2.4. This ME hardware finds the MVs of an MB in a search range of [-16, 15] pixels. This ME hardware has a systolic array of 64 (16x4) PEs. Each PE calculates the SAD values of 4 search locations for each MB. Each PE has 4 search pixel registers and 4 current pixel registers. The current pixels are loaded once for each current MB. The search pixels are reloaded for each search MB. However, by using data reuse technique, each PE column reads only one search pixel from the Block RAMs. Other search pixels are shifted among PEs. This reduces the required memory bandwidth compared to broadcasting search pixels to PEs every clock cycle as proposed in [27].

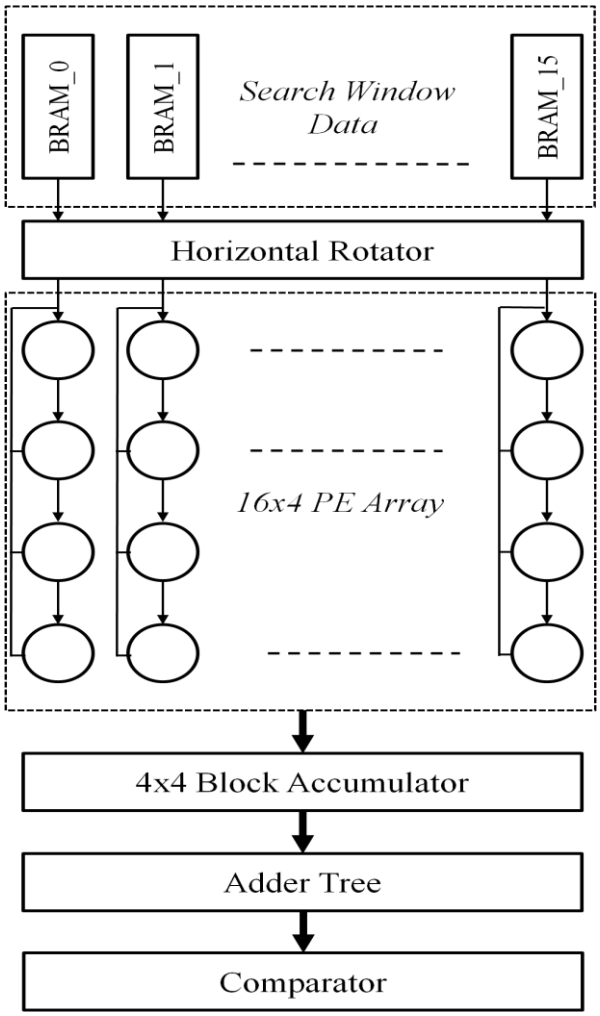


Figure 2.4 64 PE VBS ME Hardware Architecture

The data flow of the 1st PE column is shown in Table 2.2 where S(x, y) is a search window pixel. Current MB pixels are not shown in the table, because each PE stores the same four pixels for each MB. Each PE stores four search pixels and calculates their absolute differences with corresponding current pixels in four cycles. It takes 12 cycles for each PE to receive at least one search pixel and start execution at the beginning of each MB column in the search window. The first search pixels read from BRAMs are written directly to their registers in the PEs. Once the registers in all PEs are filled with the pixels in the first search MB of a MB column, each PE column reads a new search pixel in every four cycles and the search pixels in the PEs are shifted down in each PE column.

Table 2.2 Data Flow of 1st PE Column in 64 PE VBS ME Hardware

Clock	1st Column												
	PE(0,0)				PE(0,1)					PE(0,3)			
	Reg3	Reg2	Reg1	Reg0	Reg3	Reg2	Reg1	Reg0		Reg3	Reg2	Reg1	Reg0
0				S(0,0)									
1			S(0,1)	S(0,0)									
…	…	…	…	…	…	…	…	…		…	…	…	…
5	S(0,3)	S(0,2)	S(0,1)	S(0,0)				S(0,4)					
6	S(0,3)	S(0,2)	S(0,1)	S(0,0)			S(0,5)	S(0,4)					
…	…	…	…	…	…	…	…	…	…	…	…	…	…
12	S(0,3)	S(0,2)	S(0,1)	S(0,0)	S(0,7)	S(0,6)	S(0,5)	S(0,4)					S(0,12)
13	S(0,3)	S(0,2)	S(0,1)	S(0,0)	S(0,7)	S(0,6)	S(0,5)	S(0,4)				S(0,13)	S(0,12)
14	S(0,3)	S(0,2)	S(0,1)	S(0,0)	S(0,7)	S(0,6)	S(0,5)	S(0,4)			S(0,14)	S(0,13)	S(0,12)
15	S(0,3)	S(0,2)	S(0,1)	S(0,0)	S(0,7)	S(0,6)	S(0,5)	S(0,4)		S(0,15)	S(0,14)	S(0,13)	S(0,12)
16	S(0,4)	S(0,3)	S(0,2)	S(0,1)	S(0,8)	S(0,7)	S(0,6)	S(0,5)		S(0,16)	S(0,15)	S(0,14)	S(0,13)
…	…				…				…	…			
139	S(0,34)	S(0,33)	S(0,32)	S(0,31)	S(0,38)	S(0,37)	S(0,36)	S(0,35)		S(0,46)	S(0,45)	S(0,44)	S(0,43)
140				S(1,0)									
141			S(1,1)	S(1,0)									
…	…	…	…	…	…	…	…	…	…	…	…	…	…
155	S(1,3)	S(1,2)	S(1,1)	S(1,0)	S(1,7)	S(1,6)	S(1,5)	S(1,4)		S(1,15)	S(1,14)	S(1,13)	S(1,12)
…	…				…					…			
4476													
4477	S(31,34)	S(31,33)	S(31,32)	S(31,31)	S(31,38)	S(31,37)	S(31,36)	S(31,35)	…	S(31,16)	S(31,15)	S(31,14)	S(31,13)
4478													
4479													

The proposed ME hardware calculates SAD values for all 4x4 sub-blocks in a search MB in 4 cycles. The partial SAD values for 4x4 sub-blocks are stored in an accumulator. After the SAD values for all 4x4 sub-blocks are calculated, the SAD values for the other sub-blocks are calculated by an Adder Tree. After all the search locations in the search window are searched, the sub-blocks with smallest SAD values are stored in the Comparator with their respective motion vectors.

The proposed 64 PE VBS ME hardware is implemented in Verilog HDL. The Verilog RTL code is synthesized to a XC2VP30 Xilinx Virtex II Pro FPGA with speed grade 7 using Mentor Graphics Precision RTL tool. The resulting netlist is placed and routed to the same FPGA using Xilinx ISE tool. The FPGA resource usage and the maximum clock frequency of the placed & routed design are given in Table 2.3. The ME hardware takes 4160 clock cycles to process a MB. Therefore, it can process a VGA (640x480) frame in 43.70 ms (1200 MBs x 4160 clock cycles per MB x 8.755 ns clock cycle = 43.70 ms). Therefore, it can process 1000/43.70 = 22 VGA frames per second.

Table 2.3 The FPGA Resource Usage and the Maximum Clock Frequency of 64 PE VBS ME Hardware

Function Generators	11608	42.38%
CLB Slices	5804	42.38%
DFFs	6241	21.28%
BRAMs	17	12.5%
Frequency (MHz)	114.22	

CHAPTER III

POWER REDUCTION TECHNIQUE FOR INTEGER PIXEL MOTION ESTIMATION HARDWARE

BM ME hardware architectures perform absolute difference (AD) operations for calculating SAD values [10, 11, 28]. The number of AD operations performed by BM ME algorithms is very high. For example, FS algorithm performs 103,809,024 AD operations for finding motion vectors of a CIF (352x288) size frame in a [-16, 15] search range. Using larger frame sizes, larger search ranges or multiple reference frames significantly increases the number of AD operations performed.

Therefore, we propose comparison prediction (CP) technique for reducing the power consumption of BM ME hardware by reducing the power consumption of absolute difference operations. CP technique replaces the 8-bit comparator in AD hardware with a D Flip-Flop (DFF) and 1-bit inverter. CP technique can easily be used in all BM ME hardware. In this thesis, it is applied to the FBS version of the proposed 256 PE VBS ME hardware. It reduced the average dynamic power consumption of this ME hardware by 6.1% with 0.01% PSNR loss and by 9.3% with 0.04% PSNR loss on a XC2VP30-7 FPGA.

3.1 Absolute Difference Hardware

The 256 PE FBS ME hardware implements full search algorithm with a zigzag search flow in a [-16, 15] search range. It finds the search location in the search window (SW) that best matches the current 16x16 MB based on minimum SAD criterion. While the SW is searched for the current MB, each PE stores a current MB pixel and calculates the AD with corresponding pixels in the SW. ADs calculated by the 256 PEs for a search location are added by a pipelined adder tree in order to calculate the SAD value of this search location.

After the SAD values for all search locations in the SW are calculated, the search process for the current MB finishes.

The AD hardware used in this ME hardware is shown in Figure 3.1. It includes an 8-bit comparator, two 8-bit 2to1 multiplexers, an 8-bit subtractor, and an 8-bit register. AD hardware compares the 8-bit current MB pixel with the 8-bit SW pixel and subtracts the smaller one from the larger one. The result of the comparison is used to select the proper pixels for subtraction so that the result of the subtraction is always positive.

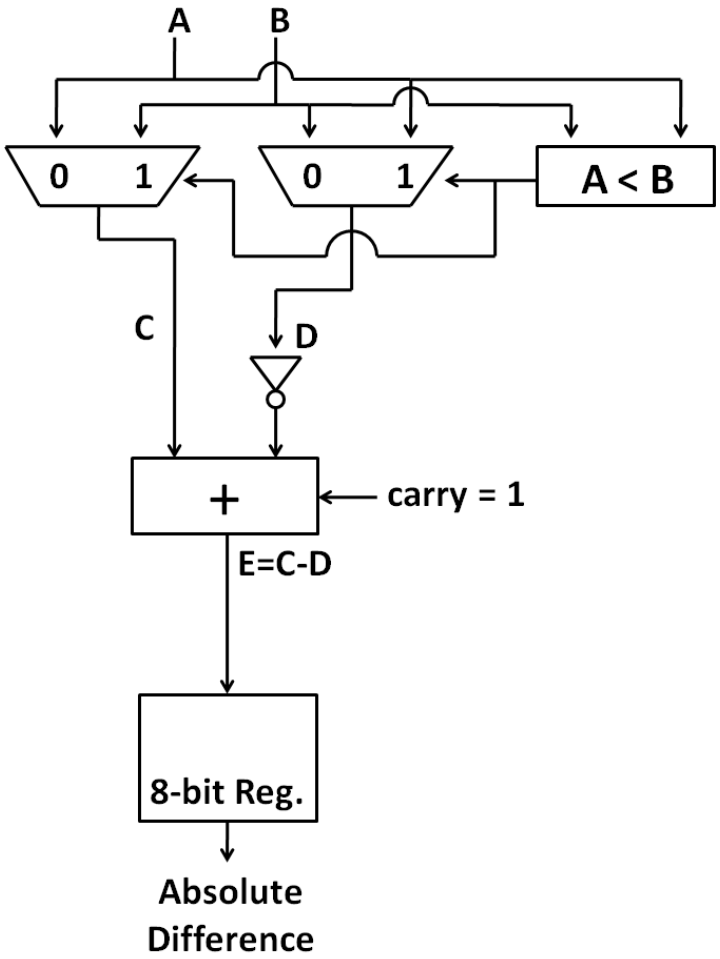


Figure 3.1 Standard Absolute Difference Hardware

3.2 Comparison Prediction Technique

The proposed CP technique avoids the comparison in the AD hardware by predicting the comparison result using the previous subtraction result. As shown in Figure 3.2 (a) and (b), the proposed technique stores an initial prediction in a DFF, and updates it after each incorrect prediction. The initial prediction predicts that current MB pixel will be subtracted from the SW pixel. If the sign bit of the subtraction result is 0, the prediction is correct and the DFF is not updated. If the sign bit of the subtraction result is 1, the prediction is incorrect and the prediction in the DFF is reversed. This new prediction is used for predicting the comparison results for the following pixels.

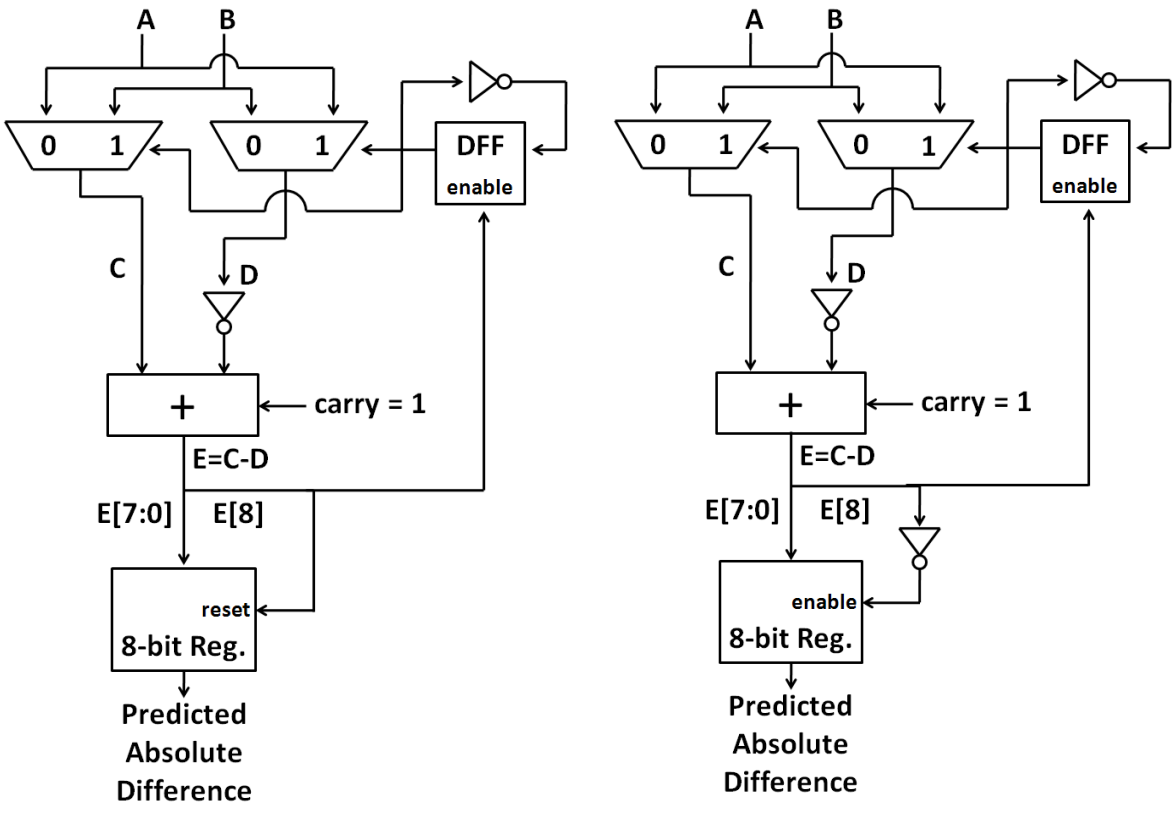


Figure 3.2 (a) Reset based Absolute Difference Prediction Hardware (b) Enable based Absolute Difference Prediction Hardware

When the comparison prediction is incorrect, the result of the subtraction operation is different from the absolute difference of the two input pixels. This causes PSNR loss. Since the pixels in the SW usually have high spatial correlation, CP technique has very high prediction accuracy. Therefore, it causes very small PSNR loss. We determined the accuracy of the comparison prediction on 5 video sequences each with 80 frames. The results show that the proposed CP technique correctly predicts the results of 90.1% of the comparisons performed by the PEs.

When there is an incorrect prediction, the larger pixel value is subtracted from the smaller one and the subtraction result is negative. Using this negative value for SAD calculation will result in an incorrect SAD. In order to reduce the impact of this negative value on the SAD and therefore reduce the impact of using incorrect predictions on the PSNR obtained by ME, we propose four different methods; reset based AD prediction (R-ADP), enable based AD prediction (E-ADP), R-ADP used with a checkerboard pattern (CR-ADP) and E-ADP used with a checkerboard pattern (CE-ADP).

As shown in Figure 3.2 (a), R-ADP method uses the sign bit of the subtraction result as a reset signal for the 8-bit register used for storing the absolute difference result. When an incorrect prediction is done, this 8-bit register is set to 0. Therefore, instead of a negative value, 0 is used for SAD calculation. Since the SW pixels have high spatial correlation, in case of consecutive incorrect comparison predictions, it is likely that the current MB pixel value is close to the SW pixel values. Therefore, predicting absolute difference as 0 will have a small impact on SAD.

As shown in Figure 3.2 (b), E-ADP method uses the inverse of the sign bit of the subtraction result as an enable signal for the 8-bit register used for storing the absolute difference result. When an incorrect prediction is done, this 8-bit register is disabled. Therefore, instead of a negative value, the previous absolute difference is used for SAD calculation. In case of consecutive incorrect comparison predictions, predicting the absolute differences as 0 may cause the SAD to be smaller than it should be and this SAD value may incorrectly be selected as the minimum SAD. E-ADP method avoids this by using the previous absolute difference in case of incorrect comparison prediction. In addition, since E-ADP method keeps the previous AD value in the 8-bit register, it does not consume dynamic power for setting the 8-bit register to 0.

CR-ADP method applies the R-ADP method to 128 of the 256 PEs in the PE array. CE-ADP method applies the E-ADP method to 128 of the 256 PEs in the PE array. In both CR-ADP and CE-ADP methods, the 128 PEs are determined by a checkerboard pattern as shown in Figure 3.3.

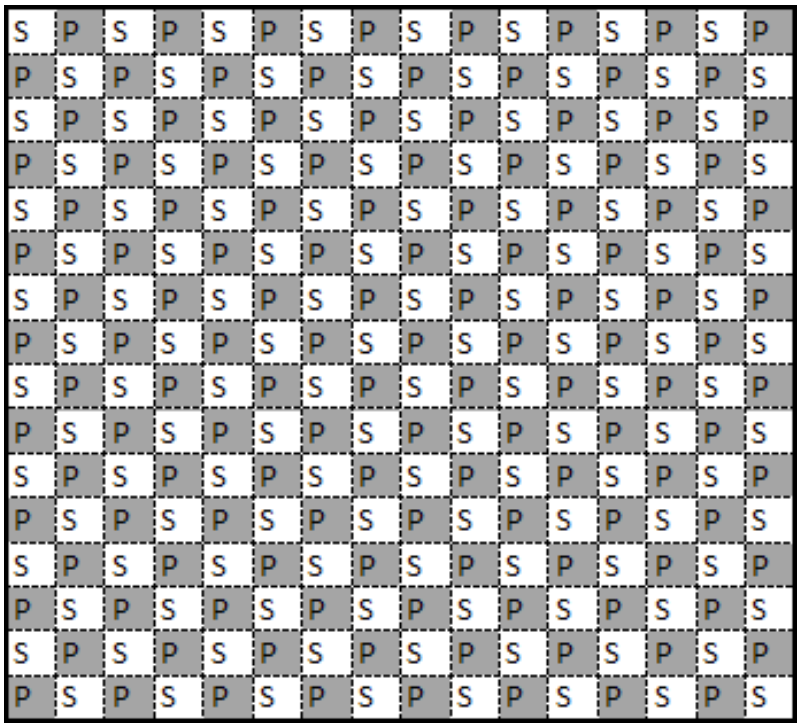


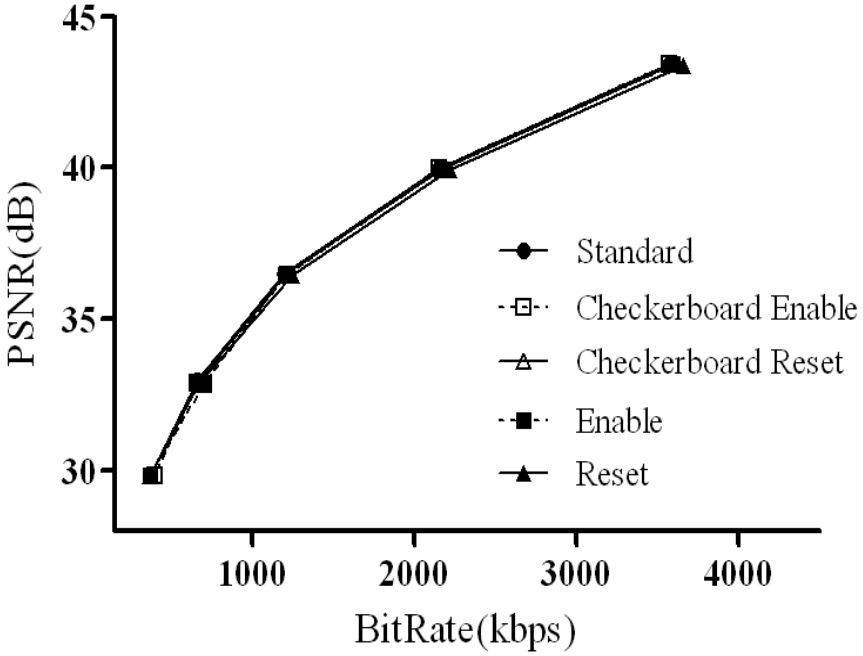
Figure 3.3 Checkerboard Pattern for Standard (S) and Predicted (P) Absolute Difference Hardware

3.3 Implementation Results

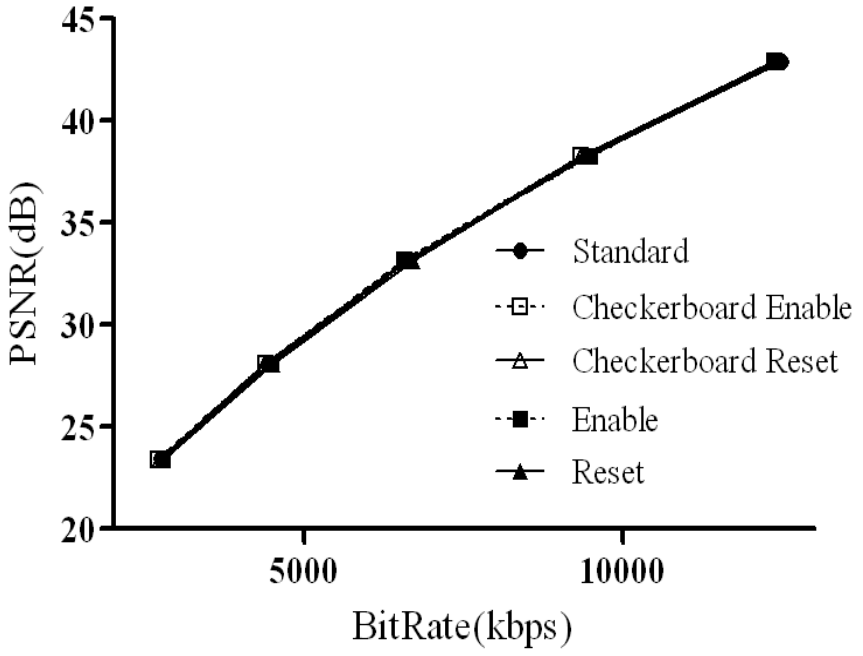
The proposed R-ADP, E-ADP, CR-ADP and CE-ADP methods are integrated to H.264 JM reference encoder software version 14.2. The average PSNR (dB) and bit rate (Kbps) obtained by these methods for several video sequences for 16x16 FBS FS ME in a [-16, 15] search range with zigzag search flow are given in Table 3.1. The rate distortion curves for the video sequences are shown in Figure 3.4. The data for rate distortion curves are obtained for quantization parameters 20, 25, 30, 35 and 40.

Table 3.1 Average PSNR and Bit Rate for Several Video Sequences

		Comparison Prediction Accuracy	Quantization Parameter	Absolute Difference		Proposed Methods							
						R-ADP		CR-ADP		E-ADP		CE-ADP	
				Bit Rate	PSNR	Bit Rate	PSNR	Bit Rate	PSNR	Bit Rate	PSNR	Bit Rate	PSNR
Video Sequences	Foreman (288x352)	90.9%	25	1753	38.916	2088	38.893	1800	38.909	1814	38.902	2070	38.910
			30	804	35.588	1016	35.466	837	35.556	1023	35.474	833	35.563
			35	398	32.252	534	31.920	417	32.147	525	31.968	411	32.197
	Mobile (288x352)	86.5%	25	9440	38.226	9382	38.216	9331	38.221	9487	38.216	9338	38.225
			30	6647	33.146	6692	33.130	6602	33.140	6557	33.139	6566	33.141
			35	4444	28.074	4496	28.036	4376	28.069	4481	28.048	4383	28.071
	Mother & Daughter (288x352)	91.3%	25	2149	39.972	2217	39.942	2170	39.966	2163	39.948	2150	39.967
			30	1198	36.496	1248	36.437	1200	36.480	1203	36.459	1214	36.462
			35	658	32.966	663	32.829	660	32.861	699	32.834	662	32.901
	Akiyo (288x352)	93.3%	25	2219	41.193	2258	41.166	2211	41.183	2329	41.180	2257	41.186
			30	1447	37.336	1495	37.254	1415	37.299	1444	37.279	1422	37.300
			35	955	35.183	903	35.006	955	35.128	955	35.100	898	35.174
	Paris (288x352)	89.6%	25	6102	38.703	6288	38.678	6243	38.690	6170	38.689	6121	38.695
			30	4126	34.155	4268	34.129	4136	34.145	4221	34.139	4178	34.145
			35	2716	29.395	2817	29.357	2719	29.388	2775	29.363	2718	29.390
Average PSNR Loss			25	—	0%	—	0.06%	—	0.02%	—	0.04%	—	0.01%
			30		0%		0.17%		0.06%		0.13%		0.06%
			35		0%		0.44%		0.17%		0.34%		0.08%



(a)



(b)

Figure 3.4 Rate Distortion Curves for
a) Mother & Daughter and b) Mobile Video Sequences

The results show that E-ADP method performs better than R-ADP method, and CE-ADP and CR-ADP methods perform very close to using standard absolute difference operation.

The proposed R-ADP, E-ADP, CR-ADP and CE-ADP methods are implemented in Verilog HDL and these hardware implementations are integrated to 256 PE ME hardware. The resulting Verilog RTL codes are synthesized to a XC2VP30-7 FPGA using Precision RTL 2005b and mapped to the same FPGA using ISE 8.2i. The ME hardware implementations are verified with post place & route simulations using Modelsim 6.1c.

The power consumptions of the ME hardware are estimated using Xilinx XPower tool. In order to estimate the dynamic power consumption of a ME hardware, timing simulation of the placed & routed netlist of that ME hardware is done at 50 MHz for a full frame of the Foreman video sequence using Mentor Graphics ModelSim 6.1c 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 ME hardware.

The area and power consumptions of ME hardware with standard AD and ME hardware with proposed methods are given in Table 3.2. The ME hardware with R-ADP, E-ADP, CR-ADP and CE-ADP methods use 8%, 9%, 4% and 3% less slices than the ME hardware with standard AD. The ME hardware with R-ADP, E-ADP, CR-ADP and CE-ADP methods have 8.1%, 9.3%, 6.0% and 6.1% less dynamic power consumption than the ME hardware with standard AD.

R-ADP and E-ADP methods reduce the dynamic power consumption and area of the ME hardware more than the CR-ADP and CE-ADP methods. However, R-ADP and E-ADP methods have a PSNR loss of 0.06% and 0.04% respectively, whereas CR-ADP and CE-ADP methods have a PSNR loss of 0.02% and 0.01% respectively. Therefore, one of the four methods can be used for ME depending on performance and power consumption requirement of the video compression or video enhancement application.

Table 3.2 Comparison of Motion Estimation Hardware Architectures

		AD	Proposed Techniques							
			R-ADP		CR-ADP		E-ADP		CE-ADP	
		Value	Value	%	Value	%	Value	%	Value	%
Area	Slice	9353	8628	8	8934	4	8542	9	9039	3
	LUT	16145	14353	11	15210	6	14217	12	15261	5
	DFF	7377	7633	-3	7505	-2	7633	-3	7505	-2
Average Dynamic Power (mW)		775.90	713.39	8.1	729.57	6.0	704.09	9.3	728.77	6.1

CHAPTER IV

POWER REDUCTION TECHNIQUES FOR H.264 HALF-PIXEL MOTION ESTIMATION HARDWARE

4.1 Half-Pixel Motion Estimation Algorithm and Hardware

The search locations for half-pixel accurate motion estimation are shown in Figure 4.1. First, integer-pixel motion estimation is performed at the integer-pixel search locations and best integer-pixel motion vector is determined based on minimum SAD criterion. Then, half-pixel motion estimation is performed at the eight half-pixel search locations around the best integer-pixel motion vector with a search range of [-1, 1], and the integer-pixel motion vector is refined by the best half-pixel motion vector.

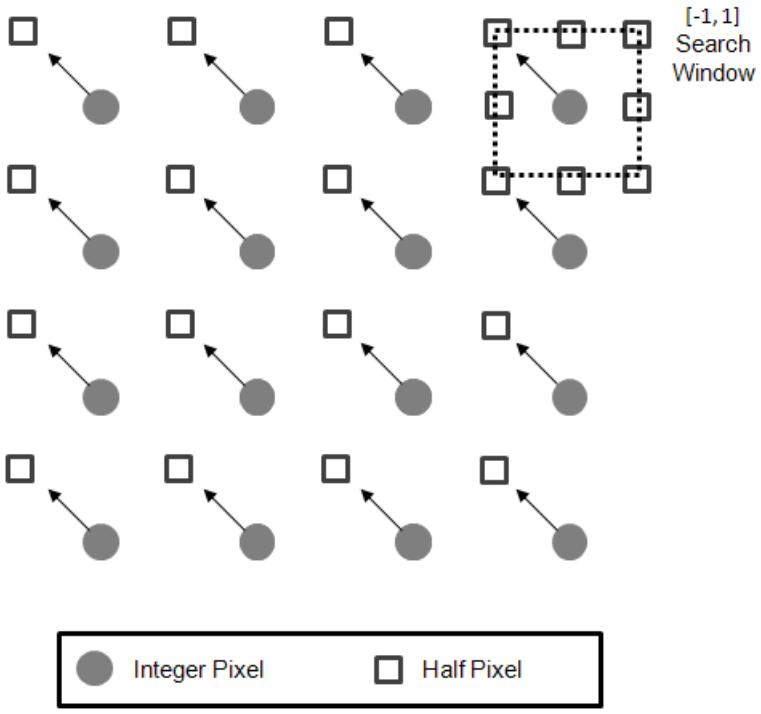


Figure 4.1 Half Pixel Search Locations

Before searching for the best half-pixel motion vector, half pixels in the half-pixel search window are interpolated from the neighboring pixels using the following 6-tap finite impulse response (FIR) filter:

$$Half\ Pixel_c = (A - 5B + 20C + 20D - 5E + F)/32 \qquad (4.1)$$

First, the half pixels that are adjacent to two integer pixels are interpolated from 6 integer pixels. Then, the remaining half pixels are interpolated from 6 horizontal or 6 vertical half pixels. A half-pixel interpolation example is shown in Figure 4.2. First, the half pixels a, b, c, d, e, f are interpolated from 6 corresponding horizontal integer pixels. For example, half pixel c is interpolated from the 6 horizontal integer pixels A, B, C, D, E, and F. Then, the half pixels g, h, i, j, k, m are interpolated from 6 corresponding vertical integer pixels. For example, half pixel i is interpolated from the 6 vertical integer pixels M, N, C, I, O, P. Finally, half-pixel n can be interpolated from either horizontal half pixels g, h, i, j, k, m or vertical half pixels a, b, c, d, e, f.

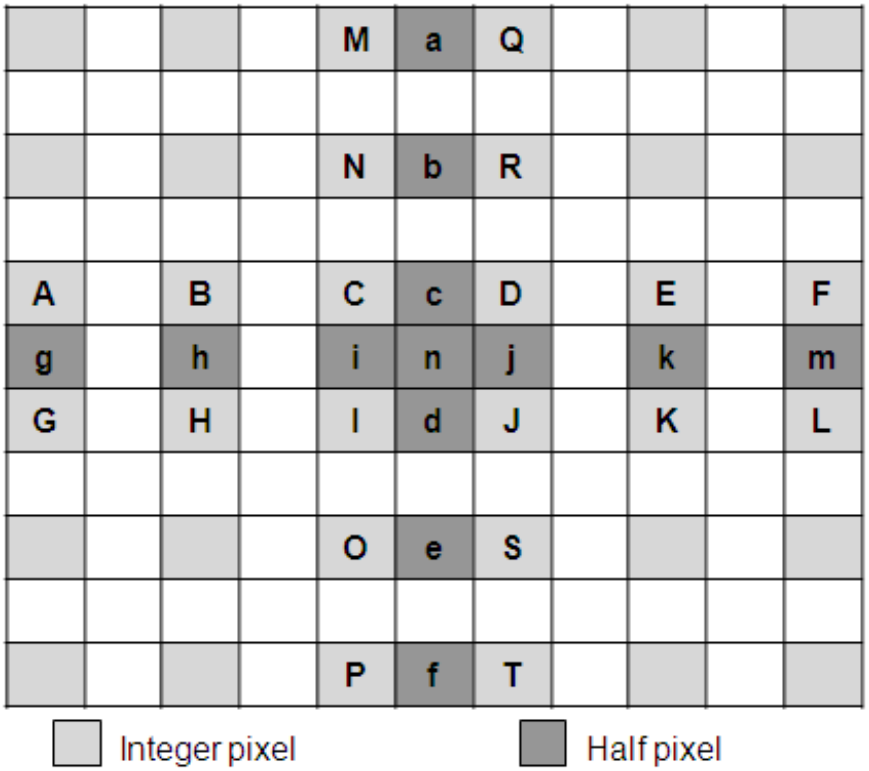


Figure 4.2 Half Pixel Interpolation

According to H.264 standard, there are 41 sub-blocks in a 16x16 MB as shown in Figure 4.3. When performing VBS HP ME, each sub-block is searched in the search window defined by the integer motion vector of that sub-block. Therefore, for each sub-block, HP ME may be performed in a different search window.

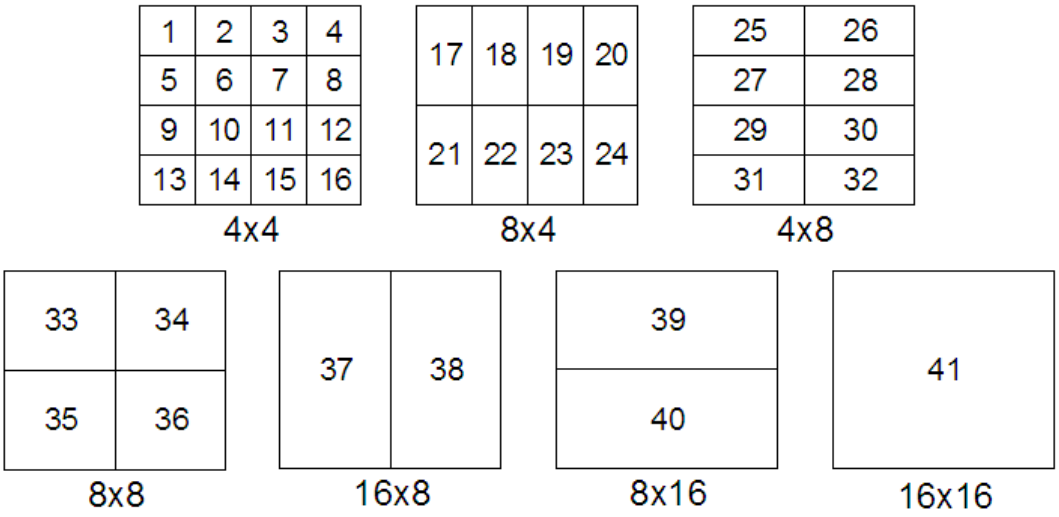


Figure 4.3 Variable Block Size Modes

The VBS HP ME hardware proposed in [24] performs HP interpolation and HP search for each sub-block. For each sub-block, first, interpolation hardware calculates the half pixels in the half-pixel search window of that sub-block. Then, search hardware searches the half-pixel search locations and determines the best half-pixel motion vector for that sub-block. In the VBS HP ME hardware proposed in [24], interpolation datapaths are shared by different block sizes. 4x4, 4x8 and 8x4 blocks share 10 interpolation datapaths. 8x8, 8x16, 16x8 and 16x16 blocks share 18 interpolation datapaths. The block diagram of the half-pixel interpolation hardware for 4x4, 4x8 and 8x4 sub-blocks is given in Figure 4.4. The interpolation and search for different size sub-blocks run in parallel, while the interpolation and search for the same size sub-blocks run sequentially.

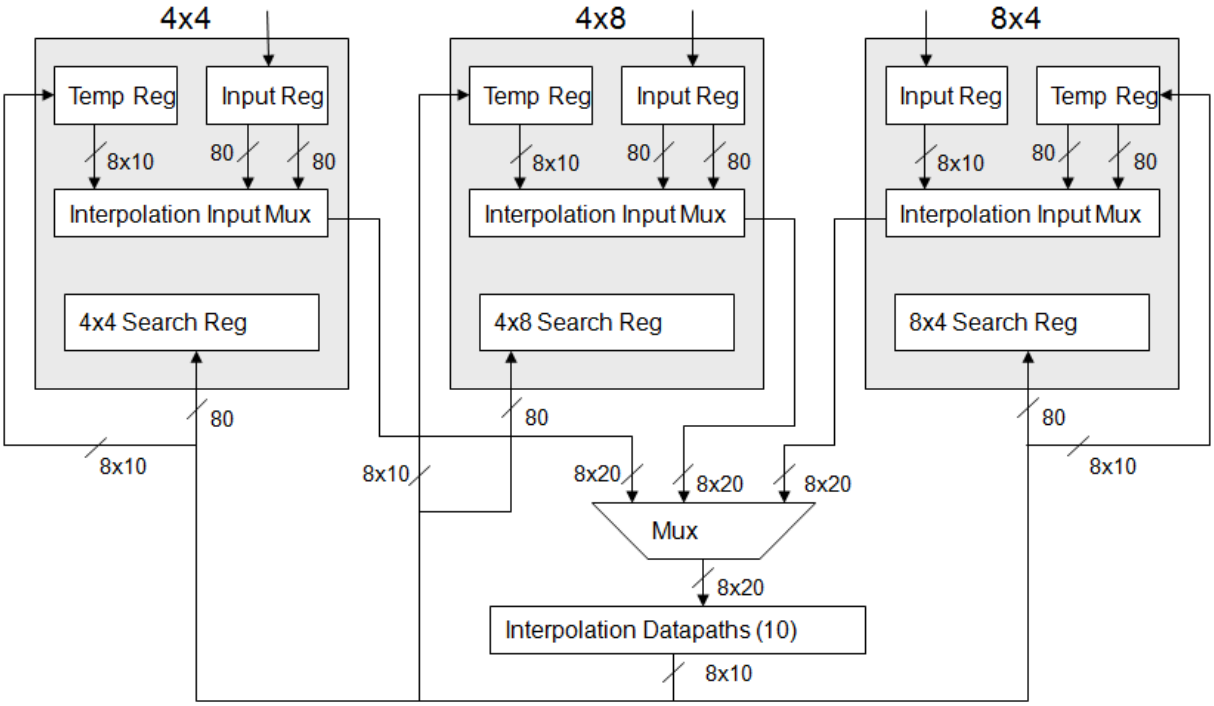


Figure 4.4 Half Pixel Interpolation Hardware for 4x4, 4x8 and 8x4 Sub Blocks [24]

4.2 Vector Dependent SAD Reuse Technique

In VBS IP ME, SAD value of each sub-block should be calculated for each search location. SAD reuse is a commonly used technique that allows calculating the SADs of larger sub-blocks by adding the SADs of smaller sub-blocks. As shown in Figure 4.3, SADs of all 41 sub-blocks for a search location can be calculated by using the SADs of 16 4x4 sub-blocks. However, SAD reuse technique is not directly applicable to VBS HP ME. HP MVs of each sub-block may be searched in different search windows. Since larger sub-blocks may have a different search window than the search windows of smaller sub-blocks, in VBS HP ME, the SADs of each sub-block may have to be calculated separately.

In this thesis, we propose vector dependent SAD reuse technique for VBS HP ME in order to reduce the power consumption of VBS HP ME hardware by eliminating redundant computations. When IP MVs of neighboring sub-blocks are equal, their cumulative search window is the same as the search window of the larger sub-block composed of these sub-blocks. In this case, performing HP ME for the larger block is redundant, because its SADs for the half-pixel search locations can be calculated by using the SADs of the smaller sub-

blocks for the same half-pixel search locations. Vector dependent SAD reuse technique requires comparing the IP MVs of the sub-blocks. When the IP MVs of the proper sub-blocks are equal, the HP ME hardware can be disabled for the larger blocks, and the SADs of the larger blocks for the half-pixel search locations can be computed using the SADs of these sub-blocks for the same half-pixel search locations.

For each sub-block size, the percentage of the smaller sub-blocks with equal IP MVs for a CIF size Foreman video sequence is shown in Table 4.1. It can be seen that IP MVs of the same size sub-blocks are highly correlated and they are likely to have the same IP MV with the one larger size sub-block. The probability of having equal IP MVs decreases as the number of same size sub-blocks increase, e.g. it is unlikely for 16 4x4 sub-blocks of a 16x16 MB to have the same IP MV.

In our implementation of the vector dependent SAD reuse technique, HP ME for 8x4, 4x8 and 8x8 sub-blocks is not enabled if the IP MVs of the corresponding 4x4 sub-blocks are equal. Similarly, HP ME for 8x16, 16x8 and 16x16 sub-blocks is not enabled if the IP MVs of the corresponding 8x8 sub-blocks are equal. For example, HP ME for sub-block 25 in Figure 4.3 is not enabled if the MVs of the sub-blocks 1 and 2 are equal. Similarly, HP ME for sub-block 34 is not enabled if the MVs of the sub-blocks 3, 4, 7 and 8 are equal.

Table 4.1 Average Computation Reduction Obtained from a Foreman CIF (352x288) Video Sequence Using Vector Dependant SAD Reuse Technique

Average Computation Reduction		Computed Block Size					
		8x4	4x8	8x8	16x8	8x16	16x16
Compared Block Size	4x4	15.06 %	15.34 %	4.42%	0.88%	1.39%	0.00 %
	8x4			29.92 %	9.22 %	10.35 %	3.79 %
	4x8			28.09 %	8.46 %	9.47%	1.77 %
	8x8				33.84 %	35.86 %	15.66 %
	16x8						45.20 %
	8x16						41.67 %

If the IP MVs of the corresponding 4x4 sub-blocks are equal, the SADs of 8x4, 4x8, and 8x8 sub-blocks for the half-pixel search locations are computed using the SADs of these 4x4 sub-blocks for the same half-pixel search locations. Similarly, if the MVs of the corresponding 8x8 sub-blocks are equal, the SADs of 8x16, 16x8, and 16x16 sub-blocks for the half-pixel search locations are computed using the SADs of these 8x8 sub-blocks for the same half-pixel search locations.

The proposed technique requires comparing IP MVs of two 4x4 sub-blocks for each 4x8 and 8x4 sub-block. Since the comparison results for 8x4 and 4x8 sub-blocks can be reused for the 8x8 sub-blocks, 16 comparisons are enough to determine the equality of the corresponding 4x4 IP MVs. The proposed technique also requires 4 comparisons for determining the equality of the corresponding 8x8 IP MVs. Therefore, 20 comparisons per MB are required.

In addition, the proposed technique requires storing the SADs of half-pixel search locations for the 4x4 sub-blocks with equal IP MVs in a register file. Similarly, the SADs of half-pixel search locations for the 8x8 sub-blocks with equal IP MVs are stored in a register file. After the SADs of all corresponding 4x4 sub-blocks are calculated, an adder calculates the sum of 2 SADs for 4x8 and 8x4 blocks, and 4 SADs for 8x8 blocks. The same adder is used for calculating the sum of 2 8x8 sub-block SADs for 8x16 and 16x8 blocks, and 4 8x8 sub-block SADs for 16x16 MB.

4.3 Integer-Pixel Motion Vector Trajectory Based Adaptive Half-Pixel Motion Estimation Algorithm

In order to decrease the power consumption of HP ME hardware, we propose an adaptive algorithm that reduces the 2-D search window of HP ME to a 1-D search window. This reduction is done if the IP MVs meet a certain criterion so that the trajectory of the MV is estimated to be either on the x-axis or y-axis. In this case, 1-D search is performed on the estimated trajectory of the IP MV resulting in a significant amount of computation reduction for HP ME. If this cannot be estimated, the original 2-D HP ME algorithm is used.

HP ME requires both interpolation and search. Since only the half pixels required in HP search need to be interpolated, reducing the search window to 1-D reduces the amount of computation for both interpolation and search.

The half-pixel interpolation flow for 4x4 sub-blocks is shown in Figure 4.5. Set A half-pixels are interpolated from a column of integer pixels, set B half-pixels are interpolated from a row of integer pixels, and set C half pixels are interpolated from a row of set A half-pixels. Set A half-pixels required for interpolating set C half-pixels are also computed. In 2-D HP ME, 130 half-pixels should be interpolated, whereas 1-D HP ME requires interpolating only the 35 neighboring set A or set B half-pixels depending on the axis chosen. Since set C half-pixels are not used in 1-D HP ME, set A half-pixels required for interpolating the set C half-pixels are not computed either. Similarly, for a 16x16 MB, 2-D HP ME requires interpolating 374 set A half-pixels, 272 set B half-pixels and 289 set C half-pixels, whereas 1-D HP ME requires interpolating 272 set A or set B half-pixels.

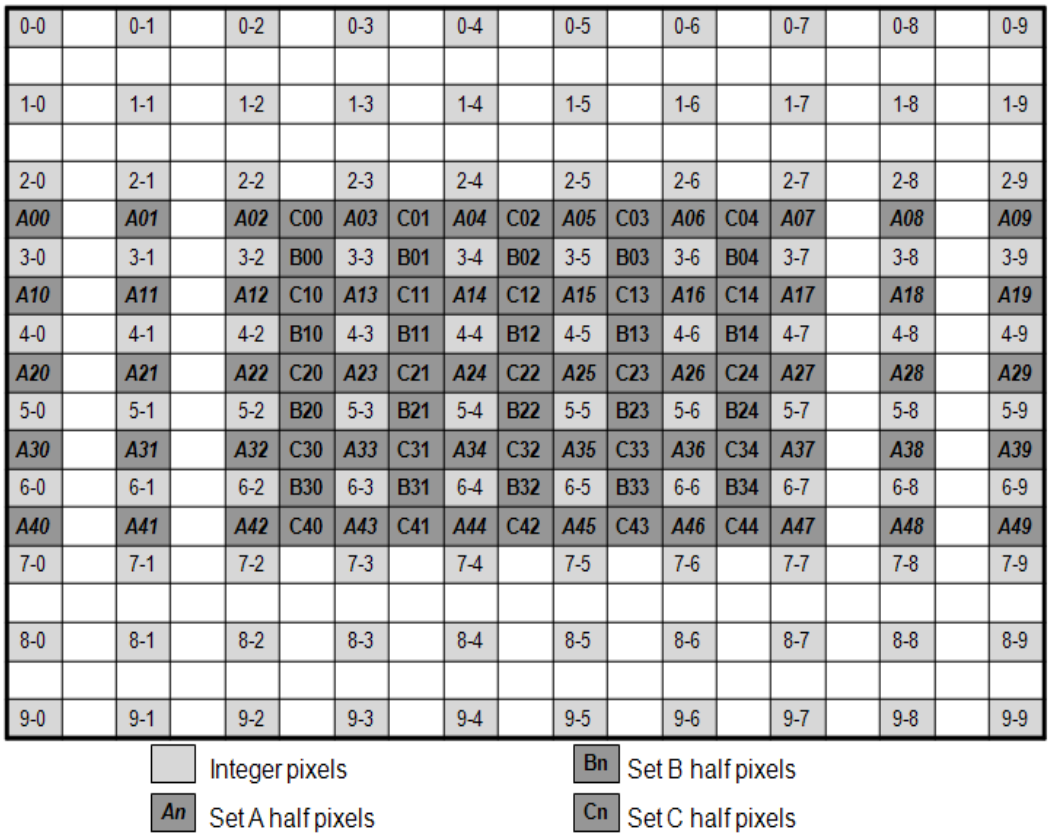


Figure 4.5 Half Pixel Interpolation for 4x4 Sub Blocks [24]

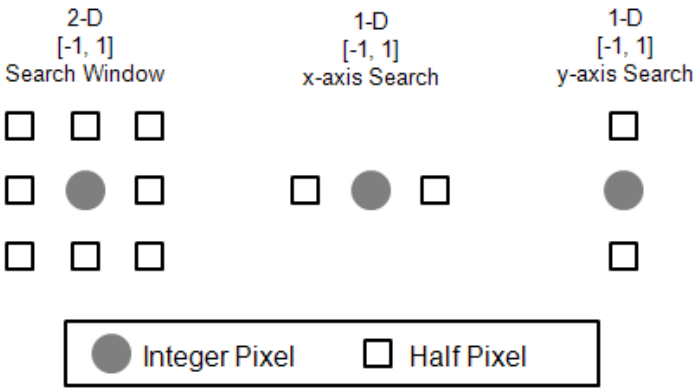


Figure 4.6 Half Pixel Search Locations for Each Integer Pixel

The number of search locations required for 2-D and 1-D HP ME are shown in Figure 4.6. 8 SAD values should be calculated in a search window of [-1, 1] for 2-D HP ME, whereas only 2 SAD values should be calculated for 1-D HP ME.

We propose three different methods, *Zero* (Z), *Twice Bigger Than* (TBT) and *Bigger Than* (BT), for estimating the trajectory of IP MVs. Z method estimates the trajectory of an IP MV only if x or y component of the MV is equal to zero. In case of an IP MV with zero x component and non-zero y component, 1-D HP ME on the y axis will be performed. Similarly, in case of an IP MV with zero y component and non-zero x component, 1-D HP ME on the x axis will be performed. TBT method performs 1-D HP ME on an axis if the MV component of that axis has an absolute value at least twice larger than the absolute value of the MV component of the other axis. In all other cases, i.e. when x and y components are equal or close to each other, 2-D HP ME is performed. BT method estimates the trajectory of the IP MV if the x or y component has a larger absolute value than the other component. Since this method performs 1-D HP ME unless the x and y components of the IP MV are equal, it favors 1-D HP ME the most.

The cases where each method performs 1-D HP ME instead of 2-D HP ME depending on IP MV values are illustrated in Figure 4.7. Z method performs 1-D HP search only when the MV is on the x or y axis. BT method performs 1-D HP search for almost all MVs unless the x and y components of the MV are equal. TBT method performs 1-D HP search for about half of all MVs.

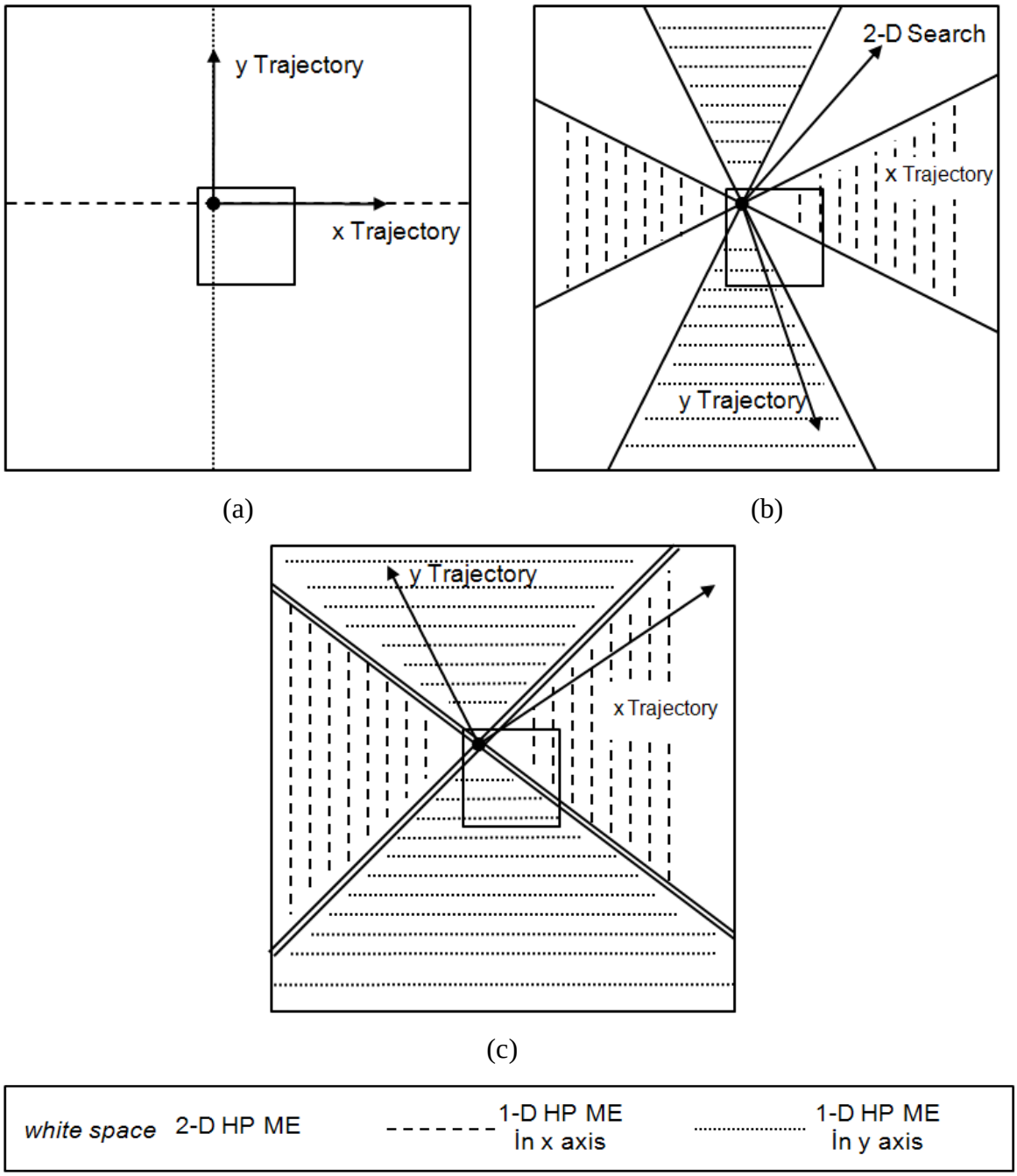


Figure 4.7 Vector Trajectory Estimations for (a) Zero, (b) Twice Bigger Than, and (c) Bigger Than Criteria

Table 4.2 Comparison of Estimation Methods (Percentage of 1-D Search and the Resulting PSNR)

Algorithm		Percentage of 1-D Search		Original PSNR (dB)		Proposed Techniques PSNR (dB)	
		FBS HP ME	VBS HP ME	FBS HP ME	VBS HP ME	FBS HP ME	VBS HP ME
Zero	Foreman	40.84%	33.36%	34.87	38.97	34.70	38.76
	Mobile	69.69%	58.18%	27.21	28.13	26.03	27.57
	Akiyo	3.06%	5.51%	44.75	46.16	44.67	46.08
Twice Bigger Than	Foreman	58.64%	57.74%	34.87	38.97	34.63	38.67
	Mobile	74.34%	71.22%	27.21	28.13	26.01	27.50
	Akiyo	3.12%	7.94%	44.75	46.16	44.67	46.06
Bigger Than	Foreman	62.22%	68.06%	34.87	38.97	34.62	38.64
	Mobile	74.70%	74.82%	27.21	28.13	26.01	27.48
	Akiyo	3.12%	8.96%	44.75	46.16	44.67	46.05

The percentage of performing 1-D search and the PSNR results obtained by each method are shown in Table 4.2 for both FBS and VBS HP ME. The percentage of 1-D search shows the percentage of making trajectory estimation for the IP MV and therefore performing 1-D HP ME instead of 2-D HP ME.

The results show that all three methods mostly perform 2-D HP ME for the low motion video sequence '*Akiyo*'. Therefore, they have minor PSNR loss. For the other video sequences, both TBT and BT methods have similar results both in computation reduction and PSNR loss. Z method performs 1-D HP ME considerably less than the other two methods, therefore it achieves less computation reduction with better PSNR performance. Because of the better computation reduction performance of BT technique, we used it in the hardware implementation of IP MV trajectory based HP ME.

The proposed IP MV trajectory based HP ME technique is also integrated to VBS HP ME hardware proposed in [24]. The integration required adding an absolute difference hardware and two comparators, and modifying the control unit. The BT method which

determines whether to perform 1-D HP search or 2-D HP search is implemented as a pre-computation step before interpolation of each sub-block. The BT method is implemented by checking whether x and y components of the MV are equal or one of them is larger than the other one. If they are equal, than 2-D HP ME is performed. If one of the components is larger than the other one, then a trajectory estimation towards the larger component is done and the appropriate control signal is sent to the HP ME module.

Depending on the result of the trajectory estimation, the interpolation module interpolates either all the half pixels required for performing 2-D HP ME or only the half-pixels required for performing 1-D HP ME. Similarly, the search module calculates either the SADs of all 8 half-pixel search locations for performing 2-D HP ME or the SADs of 2 half-pixel search locations on the selected axis for performing 1-D HP ME. The data flow of the hardware for 1-D HP ME is similar to 2-D HP ME. But, the input registers of both interpolation and search modules, and the accumulator in the processing elements are disabled accordingly to reduce power consumption.

4.4 Implementation Results

The proposed VBS HP ME hardware architectures are implemented in Verilog HDL and mapped to a Xilinx Virtex 6 FPGA using Synopsys Synplify synthesis tool and Xilinx ISE 11.4 place and route tool.

In order to estimate the power consumptions of the VBS HP ME hardware implementations, timing simulations of their placed and routed netlists are done using Mentor Graphics ModelSim SE 6.1c. A frame from CIF (352x288) size Foreman video sequence is used as input for timing simulations and the signal activities are stored in VCD files. These VCD files are used for estimating the power consumptions using Xilinx XPower Analyzer 11.4 tool.

Table 4.3 Area and Power Comparison of Proposed Vector Dependent SAD Reuse (VDSR) and Vector Trajectory Based Search (VTBS) Techniques

Area			
	Original	VDSR	VTBS
LUTs	15595	15576	15640
D Flip-Flops	13921	14021	13921
BRAMs	37	39	24
Power Consumption (mW)			
	Original	VDSR	VTBS
Clock	28.39	28.36	26.75
Logic	5.50	4.09	2.83
Signal	15.68	11.28	10.51
BRAM	44.70	44.60	24.67
Total	94.27	88.33	64.76

The power consumptions at 50 MHz and areas of the proposed hardware implementations on a Virtex 6 FPGA are shown in Table 4.3. As shown in the table, the proposed vector dependent SAD reuse and vector trajectory based HP search techniques reduced the power consumption of the VBS HP ME hardware by 6% and 31% respectively.

Since HP ME 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 the HP ME hardware can be divided into four main categories; signal power, logic power, clock power and BRAM 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. BRAM power is the power dissipated in BRAMs.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

In this thesis, first, we proposed two FS ME hardware architectures which use 256 PEs and 64 PEs respectively. The 256 PE ME hardware takes 1091 clock cycles to process a MB. Therefore, it can process 89 VGA frames per second. The 64 PE ME hardware takes 4160 clock cycles to process a MB. Therefore, it can process 22 VGA frames per second.

Then, we proposed CP technique for reducing power consumption of BM ME hardware. CP technique replaces the 8-bit comparator in AD hardware with a DFF and 1-bit inverter. CP technique can easily be used in all BM ME hardware. In this thesis, it is applied to a fixed-block size 256 PE ME hardware implementing full search algorithm. It reduced the average dynamic power consumption of this ME hardware by 6.1% with 0.01% PSNR loss and by 9.3% with 0.04% PSNR loss on a XC2VP30-7 FPGA.

Finally, we proposed two power reduction techniques for HP ME hardware. Vector dependent SAD reuse technique reduced the power consumption of a VBS HP ME hardware by 6% with no PSNR loss and minor area overhead. The vector trajectory based adaptive HP ME technique reduced the power consumption of the same VBS HP ME hardware by 31% with minor PSNR loss and area overhead.

As future work, the proposed CP technique can be improved to have no PSNR loss by updating the results after incorrect predictions. The proposed low power techniques for HP ME hardware can be applied to quarter pixel accurate ME hardware. The impact of the proposed low power techniques on the ASIC implementations of ME hardware architectures can be presented.

REFERENCES

[1] I. Richardson, *H.264 and MPEG-4 Video Compression,* Wiley, 2003.

[2] B.-D. Choi, J.-W. Han, C.-S. Kim, S.-J. Ko, "Motion-compensated Frame Interpolation Using Bilateral Motion Estimation and Adaptive Overlapped Block Motion Compensation," *IEEE Trans. on CAS for Video Technology*, vol. 17, no.4, pp. 407–416, Apr. 2007.

[3] Y. Ling, J. Wang, Y. Liu, and W. Zhang, "A Novel Spatial and Temporal Correlation Integrated Based Motion-compensated Interpolation for Frame Rate Up-conversion," *IEEE Trans. on Consumer Electronics*, vol. 54, no.2, pp. 863-869, May 2008.

[4] C. Wei, H. Hui, T. Jiarong, and M. Hao, "A High-performance Reconfigurable VLSI Architecture for VBSME in H.264," *IEEE Trans. on Consumer Electron*ics, vol. 54, no. 3, pp. 1338-1345, Aug. 2008.

[5] T. Moorthy, and A. Ye, "A Scalable Computing and Memory Architecture for Variable Block Size Motion Estimation on Field-Programmable Gate Arrays," *International Conference on Field Programmable Logic*, pp. 83-88, Sept. 2008.

[6] R. Li, B. Zeng, and M.L. Liou, "A New Three-step Search Algorithm for Block Motion Estimation," *IEEE Trans. on CAS for Video Technology*, vol. 4, pp. 438–442, 1994.

[7] S. Zhu and K.-K. Ma, "A New Diamond Search Algorithm for Fast Block Matching Motion Estimation," *IEEE Trans. on Image Processing*, vol. 9, pp. 287–290, 2000.

[8] C. Zhu, X. Lin, and L. P. [30] Chau, "Hexagon-based Search Pattern for Fast Block Motion Estimation," *IEEE Trans. on CAS for Video Technology*, vol. 12, pp. 349–355, 2002.

[9] X.-Q. Banh and Y.-P. Tan, "Adaptive Dual-cross Search Algorithm for Block-matching Motion Estimation", *IEEE Trans. on Consumer Electronics*, vol. 50, no. 2, pp. 766-775, May 2004.

[10] W. M. Chao, C. W. Hsu, Y. C. Chang, and L. G. Chen, "A Novel Motion Estimator Supporting Diamond Search and Fast Full Search," *IEEE ISCAS*, May 2002.

[11] O. Tasdizen, A. Akin, H. Kukner, and I. Hamzaoglu, "Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation," *IEEE Trans. on Consumer Electronics*, vol. 55, no. 3, Aug 2009.

[12] G. Stewart, D. Renshaw, and M. Riley, “A Novel Motion Estimation Power Reduction Technique,” *International Conference on Field Programmable Logic*, pp. 546–549, August 2007.

[13] S. Yalcin, H. F. Ates and I. Hamzaoglu, “A High Performance Hardware Architecture for an SAD Reuse based Hierarchical Motion Estimation Algorithm for H.264 Video Coding”, *International Conference on Field Programmable Logic*, August 2005.

[14] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard”, *IEEE Trans. on CAS for Video Technology*, July 2003.

[15] K. Minoo, T. Q. Nquyen, “Reverse, Sub-Pixel Block Matching: Applications within H.264 and Analysis of Limitations”, *IEEE Int. Conf. on Image Processing*, 2006.

[16] T. Dias, N. Roma, L. Sousa, “Efficient Motion Vector Refinement Architecture for Sub-Pixel Motion Estimation Systems”, *IEEE SIPS*, 2005.

[17] T. C. Chen, Y. H. Chen, C. Y. Tsai, L. G. Chen, “Low Power and Power Aware Fractional Motion Estimation of H.264/AVC for Mobile Applications”, *IEEE ISCAS*, 2006.

[18] J. W. Suh, J. Jechang, “Fast Sub-pixel Motion Estimation Techniques Having Lower Computational Complexity”, *IEEE Trans. on Consumer Electronics*, August 2004.

[19] Y. J. Wang, C. C. Cheng, T. S. Chang, “A Fast Algorithm and Its VLSI Architecture for Fractional Motion Estimation for H.264/MPEG-4 AVC Video Coding”, *IEEE Trans. on CAS for Video Technology*, May 2007.

[20] Y. Song, Y. Ma, Z. Liu, T. Ikenaga, S. Goto, “Hardware-Oriented Direction-Based Fast Fractional Motion Estimation Algorithm in H.264/AVC”, *IEEE ICME*, 2008.

[21] H. Nisar, T. S. Choi, “Fast and Efficient Fractional Pixel Motion Estimation for H.264/AVC Video Coding”, *IEEE ICIP*, 2009.

[22] Degalahal V. and Tuan T., "Methodology for High Level Estimation of FPGA Power Consumption", *Asia and South Pacific Design Automation Conference*, pp. 657-660, Jan 2005.

[23] Abdelli N., Fouilliart A.-M., Mien N., Senn E., "High-Level Power Estimation of FPGA", *IEEE International Symposium on Industrial Electronics*, pp. 925-930, June 2007.

[24] S. Yalcin, I. Hamzaoglu, “A High Performance Hardware Architecture for Half-Pixel Accurate H.264 Motion Estimation”, *IFIP Int. Conf. on VLSI-SoC*, October 2006.

[25] Y. W. Huang, T. C. Wang, B. Y. Hsieh, and L. G. Chen, “Hardware architecture design for variable block-size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264,” *IEEE Int. Symp. on Circuits Syst.*, pp. 796–799, 2003.

[26] M. Kim, I. Hwang, and S.-I. Chae, “A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264,” *ASP DAC*, pp. 631–634, Jan 2005.
[27] S. Khawam, et.al., “Efficient Implementations of Mobile Video Computations on Domain Specific Reconfigurable Arrays,” *DATE Conference*, February 2004.
[28] T. C. Chen, Y. H. Chen, S. F. Tsai, S. Y. Chien, and L. G. Chen, “Fast Algorithm and Architecture Design of Low-Power Integer Motion Estimation for H.264/AVC,” *IEEE Trans. on CAS for Video Technology*, vol. 17, pp. 568-577, 2007.