An Optimized systolic array architecture for Motion Estimation Based FSBMA used in H.264/AVC Standard

A. Yahi
LERICA Laboratory – Badji Mokhtar University
Sidi Amar, B.P# 12, Annaba, Algeria
mira_yahi@hotmail.fr

K. Messaoudi
LERICA Laboratory – Badji Mokhtar University
Electronic Department, Mentouri University
Sidi Amar, B.P# 12, Annaba, Algeria
kamel.messaoud@univ-annaba.org

S. Toumi
LERICA Laboratory – Badji Mokhtar University
Sidi Amar, B.P# 12, Annaba, Algeria
salah.toumi@univ-annaba.org

E. Bourennane
LE2I Laboratory - Burgundy University
BP 47 870, 21078, Dijon Cedex, France
ebourenn@u-bourgogne fr

Abstract—**Motion Estimation (ME) in high-definition H.264/AVC video coding presents a large challenge for memory bandwidth, latency, time consumer and cost because of its large and various search range. This paper presents a novel method based on concatenation of systolic array used in the part of integer motion estimation (IME) which adopts a parallel process. The search range used is 48x48 pixels, thus, this novel approach organize computational resources into groups within the search range. In order to preserve time consuming, this technique leading to reductions up to 75% of clock cycle. The adopted architecture used the full search (FS) method by exhaustively searching all the possible candidates within the search window in a range displacement of one pixel both horizontally and vertically.**

Keywords- video compression, H.264/AVC, Motion Estimation, systolic array

I. INTRODUCTION

Introducing new applications in multimedia technology such as, broadcast services over satellite and terrestrial channels, digital video storage, wireless led to the use of High Definition Digital Video Disc (HD-DVD), Super Hi-Vision Ultra HD-video with a resolution range from 4K to 8K (over than four/eight time of typical high-definition 1080p), obliges video coding methods to include more complex and advanced features. Therefore, standardization of the video compression techniques is essential.

The H.264/AVC also called the MPEG-4 part 10 [1] is the recent standard used in multimedia communication based on digital video compression technique. It was developed by a Joint Video Team (JVT) consisting of experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG). Compared to the previous version of video compression (like MPEG4), H.264/AVC offers more coding efficiency, but this leads to increasing computational complexity by using more powerful tools. Therefore, reducing its implementation complexity becomes a very challenging subject.

In a video encoding, motion estimation is the most consuming part which takes more than 60% of total encoding time. In fact, motion estimation exploit the temporal redundancy existing between one or several images in a video sequence, it proceeds by macroblock according to different size mode: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 and calculate the sum of absolute differences (SAD) between the current MB and each candidate position within a search area, thus, it compute 41 motion vectors for each MB (16x16). To reduce this amount of calculation, one of the methods consists of restricting the number of possible candidate modes and/or computing only partial costs. For that, several algorithms for motion estimation have been proposed [2] [3], these solutions were improved while keeping a good tradeoff between visual quality and compression rates. Recent developments have focused on the topology and regularity considered for the array and the strategy for distributing the data among different processing elements (PEs), PEs can be one-dimensional (1-D) or two-dimensional (2-D) architectures obtaining different results in terms of hardware cost and throughput [4].

This paper presents an efficient organization of 2-Dimensional Systolic Arrays to perform full-search block matching algorithm (FSBMA) in a shorter delay, based on parallel processing of the IME part, this paper introduced firstly an overview of H.264/AVC in section 2. Section 3 explains the used systolic array architecture for FSBMA, and in section 4 the results of processing of ME are reported and a comparison is given. Finally, Section 5 concludes the paper.

II. OVERVIEW OF H.264/AVC

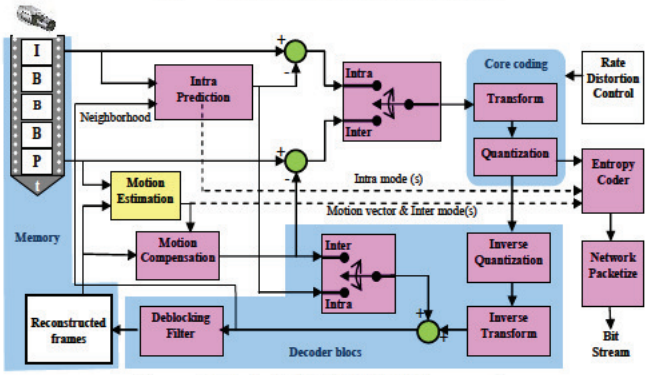


Figure. 1 Typical H.264/AVC video encoder

The standard H.264/AVC is a video coder in closed loop, it use already decoded information for coding the current part. The diagram of coding of H.264/AVC standard is represented in fig 1. In the encoding process, the video is processed from one picture (frame) to another; the dependence between images is related to the length of GOP (Group of Pictures) which consists of a succession of several types of images of type (I,B,P). For the process, H.264/AVC divides each frame into one or multiple slices, and then every slice is divided into macroblocks of size 16x16 luma pixels, the macrobloc being the unit of coding in the standard. Each macrobloc is coded either by an Intra-prediction, or an Inter-prediction to eliminate respectively spatial and temporal redundancy existing between frames. In fact, the images coded in Intra are independent of the other images, it only exploit the redundancy between pixels and their neighbors in the same image, the resulting frame is referred to as an I-picture [5]. The inter prediction uses the temporal redundancy between successive images, it performs motion estimation to create Motion vector (MV). The estimated motion vectors are transmitted and used by the decoder to assemble a prediction of the current frame. When a selected reference frame(s) for motion estimation is a previously encoded frame(s), the frame to be encoded is referred to as a P-picture. When both a previously encoded frame and a future frame are chosen as reference frames, then the frame to be encoded is referred to as a B-picture [5]. Thus, H.264/AVC supports multiple reference frames (MRF) in inter-picture coding (can reach a maximum of 15 reference frames in the buffer) to provide more efficient encoding, it also offers better motion-compensation than the previous coding standards. In some previously works, the effectiveness of the MRF is improved, mainly relies on the nature of video sequences, it is very effective for sequences which have fruitful high frequency signal with movement in different direction [6][7].

The result of these codings generates residuals which are compared with a module of decision. The decision is done by Rate-Distortion criterion (RD) to select the best residual. Inside the encoder H.264/AVC, these residual selected are transformed with the discrete cosine transform (DCT) then quantified and then entropy coded to produce the bit-stream (flux), more than, the standard H.264/AVC includes a loop of decoding inside the encoder. Consequently, the inverse quantization and the inverse transformation are respectively applied inside the encoder to reconstruct this residual. After that, the inverse prediction is applied; this operation consists of adding the predictor selected in the decision module (the best Intra or Inter predictor) to reconstruct pictures. Finally, a deblocking filter is applied to the reconstruct picture to eliminate some degradations (losses of information) produced by the quantification module [8].

A. Motion estimation (ME)

The motion estimation (ME) can be performed in two directions forward or backward (or both ways in case of bi-directional prediction) [9]. It compares the current block with candidate blocks within a search window (SW) in a reference frame and finds the region that gives the 'best' match (fig 2).

There are several algorithms for searching a 'best match' which tend to reduce of the computational complexity at the price of a slight loss of performance. Based on previously works, Line Diamond Parallel Search algorithm (LDPS), Diamond Search (DS), the HEXagon-Based Search (HEXBS), and the Nearest Neighbors Search (NNS) are proposed [1] [3]. The FSBMA is the most used because of its regularity, and its suitable for real time application, but

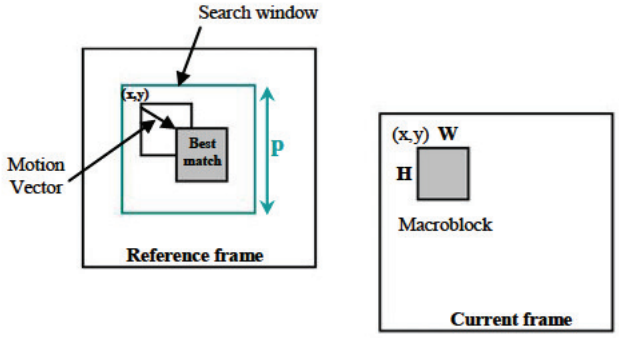


Figure 2. Block matching motion estimation process

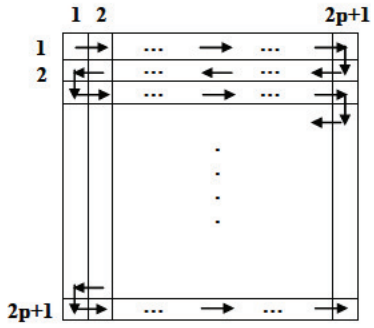


Figure 3. Movement of search locations in the search window in FSBMA

the FSBMA is an exhaustively search; it demand a lot of computation to calculate the distortion for all the possible positions of the candidate MBs within the search window and takes into account about $(2p+1)^2$ positions (fig 3). In the rest of the paper, we use an architecture based on FSBMA search.

To compute Motion Estimation (ME), the most used matching criterion is defined by:

$$SAD(i,j) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} |R(x+i, y+j) - C(x,y)| \quad (1)$$

$$\begin{cases} i \in [0, RW-16] \\ j \in [0, RH-16] \end{cases}$$

The coefficients W and H represent the width and height of the macroblock. RW and RH represent the width and height of the search window (SW). C(x, y) represents the value of pixel (x, y) in the current macroblock to process, Finally, R(x+i, y+j) represents the value of pixel (x+i, y+j) in the search window. Note that this paper uses a horizontal search range of [-16, +32] pixels and a vertical search range of [-16, +32] pixels for each macroblock. These search range values translate to an RW value of 47 and RH value of 48. Generally the minimal value of the SAD is given for each encoder, exceeding this value the block is not regarded as best candidate; this value is called "threshold". Thus, the candidate region that minimizes the residual energy is chosen as the best match, once the best match is selected, motion vectors are generated by block motion estimation to describe at which position a similar block can be found in already decoded frames.

To get the SAD values, the H.264/AVC standard subdivides a macroblock into 40 sub-blocks of size 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4. Consequently, for a macroblock, 41 SAD values are needed per reference block (fig 4). The sub-partitioning type to be used for a MB is decided by the encoder. Typically, the encoder performs motion estimation for several or all available predictive MB types. The final MB type and motion vectors are often selected by rate-distortion optimization [10] using a Lagrangian cost criterion.

Besides the new technologies added to the video compression standard, H.264/AVC adopts another

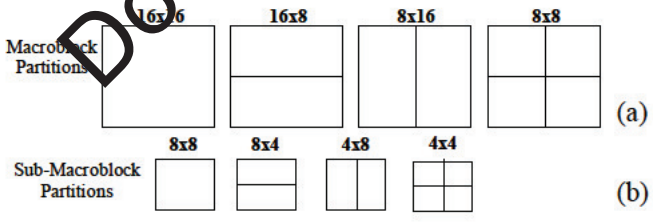


Figure 4. (a) Four block sizes for a 16×16 macroblock. (b) Four subblock sizes for 8×8 subblocks.

specificity in order to increase coding efficiency. In fact, for motion estimation process, we can distinguish two types of motion estimation: The integer motion estimation (IME) and Fractional Motion Estimation (FME), the first checks all modes to determine the best partition mode beginning by the largest which is 16×16, until 8×8 (fig 4). If the 8×8 mode is chosen as the best mode (according to RD), the modes with smaller block size will be checked, in that way 41 Motion vectors are generated [11]. The second one, which is FME, involves searching sub-sample interpolated positions, choosing the position that gives the best match which minimizes the residual energy; it uses the integer- or sub-sample values at this position for generating motion compensated prediction. Thus, the subpixel accuracy increases significantly the efficiency of the ME because the most similar block can be found in a fractional position, indicating a movement smaller than one pixel [11]. In this paper, we consider only the part for IME calculations.

III. SYSTOLIC ARRAY

A systolic array is an arrangement of identical processing elements (PE) in an array where data flows synchronously across the array between neighbors which allows a flow of data between PE and its adjacent PE (fig 5). In fact, signals entering or leaving a processing element are mainly addressed to its closer neighbors. As a result, the existing interconnections are very regular [12]. Systolic array are used in application-specific implementations of such a process, they represent a high level of parallelism, pipelinability, regularity, modularity, spatial and temporal locality [12], thus, systolic algorithms are now being implemented as a practical hardware. Some 1-Dimensional and 2-Dimensional Array Architectures are proposed for FSBMA implementation in [13] [14] proving that 2-D architectures are faster than 1-D architectures, but their hardware are more complex; and they also need a high memory bandwidth and a high pin-count. In fact, Systolic Arrays are computational networks, which take a high local data storage (especially the 2-D architecture), that use the parallel processing and pipelining techniques to achieve a high throughput.

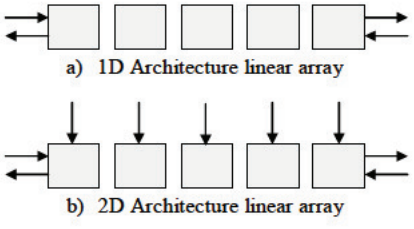


Figure 5. 1D-2D Systolic array architecture

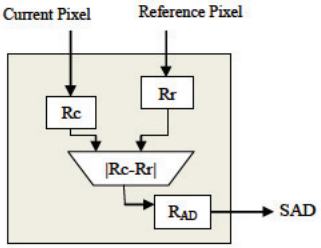


Figure 6. Processing element (PE)

In this paper, we have used the 2D architecture because of their rapidity compared to 1D architecture, but their hardware are more complex and need a high pin-count because of the large amount of data require.

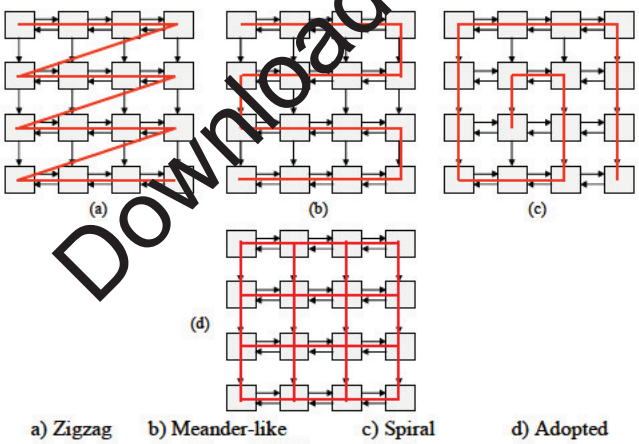
A. Functionality and Number of Units in PEs

Each cell (PE) is an independent processor and has some registers and Arithmetic and Logic Units (ALUs), it compute data and store it independently of each other and share the information with their neighbors. Their function differs for each application, it can do: multiplication, addition, subtraction and other basic operations. In [15], examples are demonstrated and therefore several ways of PE organization (inter-connection) are possible.

In this paper, we rely on a PE that consists of a current pixel register (Rc), a reference pixel register (Rr), these registers store the current and the candidate pixel of macroblock. The operator (Rc-Rr) computes the absolute difference between the Rc and the Rr at every cycle. The output final is stored in another register RAD (fig 6).

B. Interconnection between PE and delay communication

Designers are enable to specify the communication delay on connection between PEs (fig 7), this can be controlled by adding some register inside PE to get more delay connection compared to other PE and this increases the use of PE in several fields of application according to the request.



a) Zigzag b) Meander-like c) Spiral d) Adopted

Figure 7. PE interconnections

IV. FSBMA-IME SYSTEM ARCHITECTURE

Figure 8 below shows the proposed architecture which contains a number of PE 16x16 array for SAD computation, two local memories for current block and search area respectively. The data of search areas and current blocks can be inputted into the PE array for each clock cycle through the port from the two memories. Adder trees are used in order to calculate the SAD of blocks. Finally, the comparator will compare values to obtain the chosen one which is represented by a minimal result and then generate motion vector (MV).

Since a PE proceeds by pixel (fig 6), the 2-D 16x16 processing elements (PE) in 16 PE 4x4 unit can compute synchronously 256 absolute values of difference between the candidate pixels and the current pixels of 16x16 block (fig 9). The 16x16 block's SAD can be obtained by accumulating the 256 absolute values of difference in the SAD adder and deduced the other values SAD for the remainder partitions of the block (i.e. : 16x8,8x16,8x8, 8x4,4x8 and 4x4).

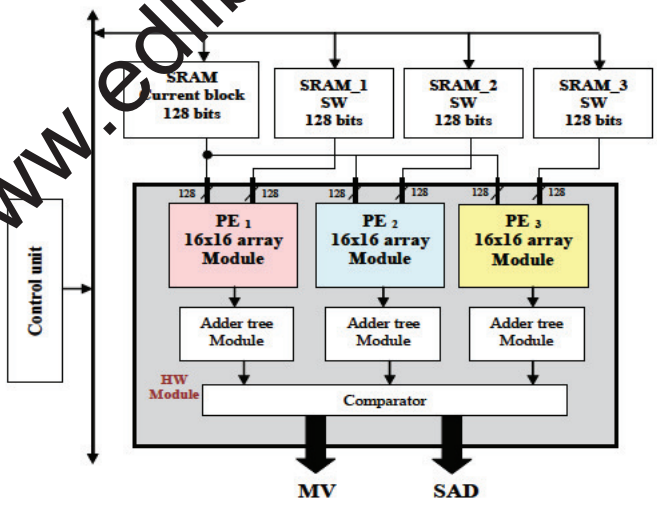


Figure 8. System architecture

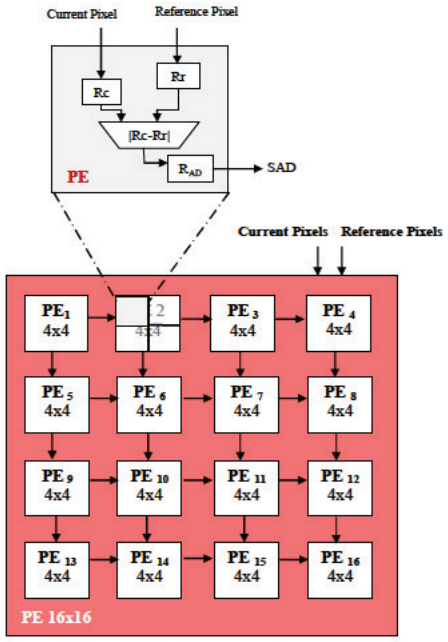


Figure 9. Architecture of a module PE

At the beginning, the 16 pixels data of the current macroblock containing in SRAM are inputted in the 16x16 PE array during the 16 cycle, and 16 pixels data of the candidate block containing in SRAM are inputted in the 16x16 PE array at every cycle, each PE receives two pixels, therefore the first horizontal row of PE will be charged. In the second clock cycle, the data of the first row are transferred to the 2nd row of the PE, while the PE of the first empty row will receive the new data sent by the two memories. This mechanism is repeated in this way until the whole PU (PE16x16) is loaded, and in 16 clock cycles, the PU will be fully charged and the sixteen 4x4 SADs of the candidate block can be computed. In total, this takes 1041 clock cycles (according to [16]).

Our optimization takes advantage of the parallelism by operating multiple PE 16x16 at the same time; the number of PE that can be used is defined by:

$$Nbr\ of\ PE_{array} = \sqrt{Nbr\ of\ non\ overlapped\ (16x16)\ in\ SW}$$

So for a search window of 48x48 pixels, 3 PE (16x16) array are used in parallel (fig10) and we need a memory of 128 bits for the current block, and 3 memories (128 bits each) for reference block.

TABLE I. DATA-FLOW SCHEDULE FOR PE_ARRAY1,2,3 FUNCTIONING IN PARALLEL

Clk	1st column	2nd column	..	16th column
	PE0 PE1 ... PE15	PE16 PE17 ...PE31	...	PE240 PE241 ...PE255
15	C(0,0) C(0,1) C(0,15) R(-16,-16) R(-16,-15) R(-16,-1)	C(1,0) C(1,1) C(1,15) R(-15,-16) R(-15,-15) R(-15,-1)		C(15,0) C(15,1) C(15,15) R(-1,-16) R(-1,-15) R(-1,-1)
62	C(0,0) C(0,1) C(0,15) R(31,-16) R(31,-15) R(31,-1)	C(1,0) C(1,1) C(1,15) R(32,-16) R(32,-15) R(32,-1)		C(15,0) C(15,1) C(15,15) R(46,-16) R(46,-15) R(46,-1)
110	C(0,0) C(0,1) C(0,15) R(-16,-15) R(-16,-14) R(-16,0)	C(1,0) C(1,1) C(1,15) R(-15,-15) R(-15,-14) R(-15,0)		C(15,0) C(15,1) C(15,15) R(-1,-15) R(-1,-14) R(-1,0)
782	C(0,0) C(0,1) C(0,15) R(-16,-1) R(-16,0) R(-16,14)	C(1,0) C(1,1) C(1,15) R(-15,-1) R(-15,0) R(-15,14)		C(15,0) C(15,1) C(15,15) R(-1,-1) R(-1,0) R(-1,14)
15	C(0,0) C(0,1) C(0,15) R(-16,0) R(-16,1) R(-16,15)	C(1,0) C(1,1) C(1,15) R(-15,0) R(-15,1) R(-15,15)		C(15,0) C(15,1) C(15,15) R(-1,0) R(-1,1) R(-1,15)
62	C(0,0) C(0,1) C(0,15) R(31,0) R(31,1) R(31,15)	C(1,0) C(1,1) C(1,15) R(32,0) R(32,1) R(32,15)		C(15,0) C(15,1) C(15,15) R(46,0) R(46,1) R(46,15)
110	C(0,0) C(0,1) C(0,15) R(-16,1) R(-16,2) R(-16,16)	C(1,0) C(1,1) C(1,15) R(-15,1) R(-15,2) R(-15,16)		C(15,0) C(15,1) C(15,15) R(-1,1) R(-1,2) R(-1,16)
782	C(0,0) C(0,1) C(0,15) R(-16,15) R(-16,0) R(-16,30)	C(1,0) C(1,1) C(1,15) R(-15,15) R(-15,0) R(-15,30)		C(15,0) C(15,1) C(15,15) R(-1,15) R(-1,0) R(-1,30)
15	C(0,0) C(0,1) C(0,15) R(-16,16) R(-16,17) R(-16,31)	C(1,0) C(1,1) C(1,15) R(-15,16) R(-15,17) R(-15,31)		C(15,0) C(15,1) C(15,15) R(-1,16) R(-1,17) R(-1,31)
62	C(0,0) C(0,1) C(0,15) R(31,16) R(31,17) R(31,31)	C(1,0) C(1,1) C(1,15) R(32,16) R(32,17) R(32,31)		C(15,0) C(15,1) C(15,15) R(46,16) R(46,17) R(46,31)
110	C(0,0) C(0,1) C(0,15) R(-16,17) R(-16,18) R(-16,32)	C(1,0) C(1,1) C(1,15) R(-15,17) R(-15,18) R(-15,32)		C(15,0) C(15,1) C(15,15) R(-1,17) R(-1,18) R(-1,32)
782	C(0,0) C(0,1) C(0,15) R(-16,31) R(-16,32) R(-16,46)	C(1,0) C(1,1) C(1,15) R(-15,31) R(-15,32) R(-15,46)		C(15,0) C(15,1) C(15,15) R(-1,31) R(-1,32) R(-1,46)

Table 1 show the processing of a current macroblock 16x16 with all possible position in the search area (48x48pixels), this takes 782 clock cycles

During the 16 cycles, the 16x16 pixels of the current macroblock and 16x16 pixels of the reference block 1 are inputted in the PE_array 1 (fig. 13), in parallel during the first 16 clock cycles, the same 16x16 pixels of the current macroblock and 16x16 pixels of the reference block 4 are inputted in the PE_array 2 (fig. 13), in parallel after 16 cycles, the same 16x16 pixels of the current macroblock and 16x16 pixels of the reference block 7 are inputted in the PE_array 3 (fig. 13). Thus, after 782 cycles, all the SW is swept, the three PE_array compute the difference between R and C for all the possible position of a current block (16x16) within the SW by following FSBMA search.

ISBN : 978-81-925233-8-5

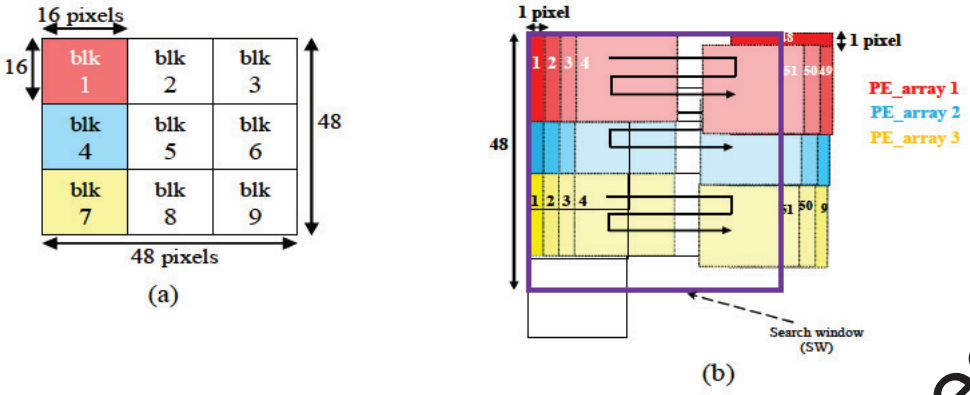


Figure 10. a) Different non-overlapped block (16x16) in SW
b) The sweeping of the three PE_array in the SW

A. Comparison :

TABLE II. THE COMPARISON BETWEEN THE PROPOSED ARCHITECTURE AND OTHER FULL-SEARCH FSBMA ARCHITECTURES

Design	Search window	Nbr of PE	Internal Memory	Cycles (CLK)	percentage of use of CLK
Pyen, Min, Chong [16]	32x32 48x48	16x16 16x16	(128x2) bits	1041 2321	100%
Our	48x48	3x(16x16)	(128x4) bits	784	33,78%

V. CONCLUSION

In this paper, an optimized architecture for H.264/AVC was proposed, based on FSBMA-IME, a modified structure using several parallel PE simultaneously operating is discussed. It includes all the necessary components to support the standard requirements like a memory of 512 bits, three systolic 16×16 PEs array, the adder tree to calculate the 41 blocks SADs and the comparator to generate motion vectors (MV). The adopted architecture has optimal performance characteristics: low latency, high processing speed leading to reductions up to 75% of clock cycle required. However, this architecture is not suitable for reel time implementation because of its large memory bandwidth and the high pin count resulting from the use of several 2-D PE systolic array. While waiting to find solutions, this architecture permits a good organization of several modules that can be used in the most beneficial way.

REFERENCES

[1] A.B. Abdallah, H. Loukil, N. Masmoudi, "FPGA DESIGN FOR H.264/AVC ENCODER", *International Journal of Computer Science, Engineering and Applications (IJCSEA)* Vol.1, No.5, October 2011.

[2] Z. Chen, J. Xu, Y. He, J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC", *ScienceDirect Elsevier*, pp. 264-290. Tsinghua University, Beijing, China, October 2005.

[3] O. Ndili and T. Ogunfunmi, "Algorithm and Architecture Co-Design of Hardware-Oriented, Modified Diamond Search for Fast Motion Estimation in H.264/AVC", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, VOL. 21, NO. 9, pp. 1214-1227, September 2011.

[4] S. López, G.M. Callicó, J. Lobajas, V.de Armas, J.F. López, and R. Sarmiento, "Grouped Approach for the Design of H.264/AVC Motion Estimation Architectures", *ETRI Journal*, Volume 30, Number 6, pp. 862-864. December 2008.

[5] "MPEG-4 Part 10 AVC (H.264) Video Encoding", *Scientific-Atlanta*, Part Number 7007887 Rev B. June 2005.

[6] K. Hosseini, D. Faura, O. Romain, P. Garda, "Accelerating the multiple reference frames compensation in the H.264 video coder", *Springer, J Real-Time Image Proc*, October 2008.

[7] S. Hsia and Y.C. Hung, "Fast multi-frame motion estimation for H264/AVC system", *Springer, SIViP*, 2009.

[8] K. Messaoudi, E. Bourennane, S. Toumi, G. Ochoa-ruiz, M. Yahi, "A Highly Parallel Hardware Implementation of the Deblocking Filter Used in H.264/AVC CODECs", 2010.

[9] M.E. RIZKALLA, P. SALAMA, M.EL-SHARKAWY AND M. SUSHMITHA, "Hardware Implementation of Block-based Motion Estimation for Real Time Applications", *Springer, Journal of VLSI Signal Processing 49*, pp. 139-159. 2006.

[10] S. Kamp, M. Evertz, and M. Wien, "Decoder Side Motion Vector Derivation For Inter Frame Video Coding", *RWTH Aachen University*, Germany, 2007.

[11] Y.K. Lin, C.C. Lin, T.Y. Kuo, T.S. Chang, "A Hardware-Efficient H.264/AVC Motion-Estimation Design for High-Definition Video", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I REGULAR PAPERS*, VOL. 55, NO. 6, pp. 1526-1535, JULY 2008.

[12] « Systolic Processors », *Advanced Logic Synthesis ECE 572*.

[13] C-Y. Chen, S-Y. Chien, Y-W. Huang, T-C. Chen, T-C. Wang, and L-G. Chen, "Analysis and Architecture Design of Variable Block-Size Motion Estimation for H 264/AVC", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, VOL. 53, NO. 2, pp. 578-593, February 2006.

[14] M M. Azadfar, "Implementation of A Optimized Systolic Array Architecture for FSBMA using FPGA for Real-time Applications", *IJCSNS International Journal of Computer Science and Network Security*, VOL.8 No.3,pp. 46-51, March 2008.

[15] S. Prakash, R.S. Kumar, V. Murthy, "Design and FPGA Implementation of Systolic Array Architecture for Full Search Block Matching Algorithm", *SASTECH*, pp. 103-108, Volume 8, Issue 2, September 2009.

[16] S.M. Pyen, K.Y. Min, J.W. Chong, "An Efficient Hardware Architecture for Full-Search Variable Block Size Motion Estimation in H.264/AVC", *IEEE, University, Seoul, Korea, Engineering UC Berkeley, CA. US.*