Implementation of A Optimized Systolic Array Architecture for FSBMA using FPGA for Real-time Applications

Mohammad Mahdi Azadfar

Multimedia System Group , IT Faculty Iran Telecom Research Center (ITRC) , Tehran , IRAN

Summary
In this paper, a optimized systolic array architecture for Full Search Block Matching Algorithm is presented. This Array Architecture is implemented by RTL-level VHDL for using as a motion estimation unit in low bit rate and real-time applications such as video telephony. This implementation is synthesized for two FPGA families, Xilinx Spartan and Virtex, The results for area occupation and maximum operating frequency are presented. This results show that this array architecture is suitable for real-time video encoding systems with minimum hardware utilization and high performance.

Key words:
Full Search Block Matching Algorithm (FSBMA), parametrizable Implementation, Systolic Array Architecture, Real-Time , FPGA.

1. Introduction

Digital video has a big amount of raw data and needs a big storage area to be recorded or a high bandwidth media to be transmitted. For example, transmitting a video with CIF format (4:1:1 or 4:2:0) and 10Frames/Sec, needs more than 11Mbps bandwidth. This high amount of data is reduced by using video coding methods which exploit spatial and temporal redundancies [1].Motion Estimation (ME) is used in many standard video Codecs like ITU-T H.264 and ISO MPEG-4 to exploit temporal redundancy within frames. In this method, instead of transmitting data correspond to the current frame, Motion Vectors of objects regarding to previous frames are calculated and are transmitted with differential information to decoder. Block Matching is the most popular method for Motion Estimation. ME is computationally more complex and takes %50 to %70 of processing time of video encoding [2]. Software based methods are not suitable for real-time implementing of ME with optimum result [1]. Accordingly, hardware based methods are needed to implement a real-time ME. By using pipelining and parallel processing techniques, Motion Estimation can be done in a short period of time. Accordingly, Systolic Array Processors are a good choicefor hardware implementation of Motion Estimation [3].Since pipelining and parallel processing techniques increase the hardware size, it is important to do a trade-off between hardware size and speed. FPGA is a good candidate to be used to implement a realtime ME, because of its flexibility, reconfigurability, reliability, and capability for rapid prototyping.

In this paper, it is shown that the 1-Dimensional Systolic Arrays which have less hardware size than 2-Dimensional arrays, are more suitable for Common Intermediate Format (CIF) (or smaller) frame size applications. Also, Xilinx FPGAs (Spartan and Virtex) is employed to implement a systolic array architecture for Full Search Block Matching Algorithm (FSBMA).In section 2, the FSBMA is described. Section 3 explains the used systolic array architecture for FSBMA, and in section 4 the results of implementation of ME are reported. Finally, section 5 concludes the paper.

2. Full Search Block Matching Algorithm

Block Matching is the most popular method for implementing ME [1]. In this method, the current frame is divided into non-overlapped N ×N blocks. For each block, a Search Area is assumed in previous frame with respect to maximum displacement value (P). The current block is matched to candidate blocks within Search Area by a matching criterion (Fig 1) and then Motion Vector (MV) is calculated from the best match state. Popular matching criteria are the Mean of Absolute Differences (MAD) and the Mean Squared Error (MSE)[1]. Experiments show that the results of these criteria are very similar [3], [4]. However, because of its simplicity and low computational complexity, with respect to MSE, the MAD criterion is used in many implementations of the ME's. The MAD is given by:

$$MAD(m,n) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1}|f(i,k) - g(i+m,k+n)| \quad (1)$$

$$-p \le m,n \le p$$

Where f(i, k) is the current block data and g(i+m, k +n) is the candidate block data. Various algorithms are dedicated for Block Matching method. Full Search Block Matching Algorithm is one of them which candidates all possible blocks within the search area. Consequently, FSBMA always calculates the best match state and determines optimum MV. Other proposed algorithms search among less candidate blocks. Consequently, they have less computation and are faster than FSBMA [1]. However, they suffer from irregular data flow that is an important characteristic in the hardware implementation. They also don't guarantee the optimum result. As a result, the FSBMA is usually used in the hardware implementation of ME, because of its simplicity, regularity and optimum result.

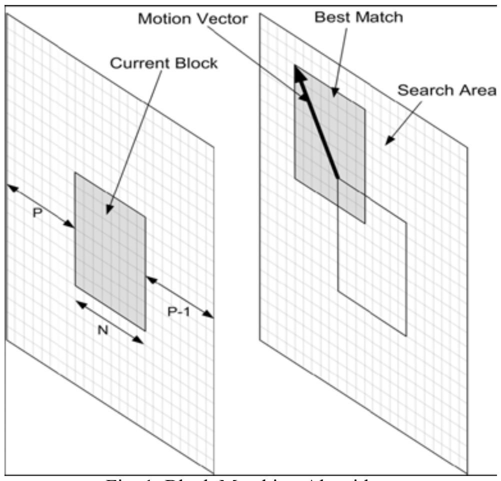

Fig. 1 Block Matching Algorithm.

3. Systolic Array Architecture for FSBMA

FSBMA needs a total $3\times N^2 + (2P)^2$ operations (subtraction, addition, magnitude, and comparative operation) to calculate motion vector of each block. In order to execute such a great amount of operations, in a real-time application, a high throughput system is needed.Systolic Arrays are computational networks, with local data storage, that use the parallel processing and pipelining techniques to achieve a high throughput. They contain array processors with synchronous clock and synchronous control signals. Their major features are modularity and regularity. For a given algorithm, a systolic array can be derived by methodology given by Kung [5]. Various 1-Dimensional and 2-Dimensional Array Architectures have been proposed for FSBMA's implementation [6], [7], [8], [9], [10].Komarek and Pirsch derived four fundamental architectures of FSBMA [6].The number of Processing Elements (PEs) of this array processor is 2P and it is independent of N. Each PE needs 1 clock cycle to process its input data and to generate output. This systolic array requires only a sequential data input.
It needs a total of $C_{AS1}=2P\times(2P+N-1)\times N$ time instances to calculate the Motion Vector of each block [3]. Table 1 shows the number of processing elements and computation time required in these architectures.
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 [6].They are proper for high frame size and high frame rate applications such as High Definition Television (HDTV).

Table 1: Systolic Array Architectures

Architecture	PE count	Computation time
Single PE	1	$N^2(2P+1)^2$
AB1	N	$N(2P+1)(2P+N)$
AS1	$2P+1$	$N(2P+1)(2P+N)$
AB2	N^2	$(2P+1)(2P+N)$
AS2	$N(2P+1)$	$N(2P+N)$

For low bit rate applications 1-D architectures are better considering their less hardware requirement. Two major 1- D architectures, as presented in [6] are AB1 and AS1. AS1 architecture is proper for a FPGA implementation because of its low memory bandwidth and low local memory requirement with respect to AB1 architecture. AS1 architecture is shown in Fig 2. This systolic array requires only a sequential data input. For achieving regular data flow without any data permutation, in each time interval, only N out of 2P + 1 PEs are used. This architecture has 2P + 1 comparators which compare the MAD values of different MVs. Each comparator only is used in 2P out of C_{AS1} time instance.

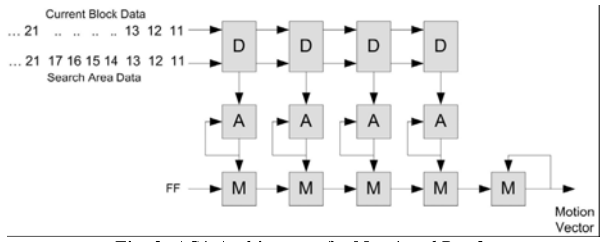

Fig. 2 AS1 Architecture for N = 4 and P = 2.

The AS1 architecture reduces the hardware efficiency.Since FSBMA has totally $(2P)^2$ comparison operation for calculating a MV, it is possible to use only one comparator in AS1 architecture. For this aim, AS1 can be modified as shown in Fig 3 in order to use the hardware efficiently. Implementation results show this modification decreases hardware of AS1 architecture without any effect on its performance. AS1 architecture is proper for an FPGA implementation because of its low memory bandwidth and low local memory requirement.
In the next section, implementation of AS1 architecture is presented.By adding some hardware, AS1 architecture can be modified in order to achieve better efficiency of PE's. This modification increases the total hardware of the system. In the most of the literature, the maximum displacement in the both positive and negative directions is assumed P. This assumption results that the MVs need $\log_2 P + 2$ bits, when N is a power of 2. The number of bits can be reduced to $\log_2 P + 1$ if the maximum displacement in the positive direction is assumed to be $P-1$ (Fig 1).

This assumption reduces the hardware considerably. With this assumption, AS1 architecture has 2P PEs and needs 2P×N×(2P+N−1) time instances to compute the MV of each block.

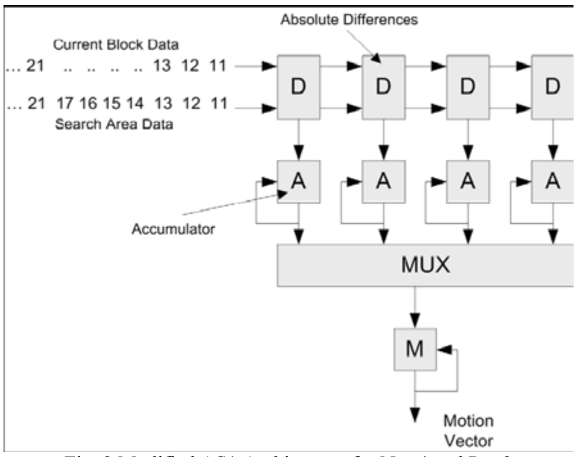

Fig. 3 Modified AS1 Architecture for N = 4 and P = 2

4. Implementation

Fig 4 shows the AS1-based array processor's Block Diagram and Fig 5 shows the architecture of the AS1-based array processor.The implementation is performed by RTL-Level VHDL sing Xilinx ISE 6.1 software and XST synthesis tool [11]. Two parameters are considered in order to perform the adaptable and parameterizable implementation. These parameters are the current block Size (N) and the maximum displacement value (P). The VHDL implementation is optimized to achieve minimum area occupation and maximum operating frequency.

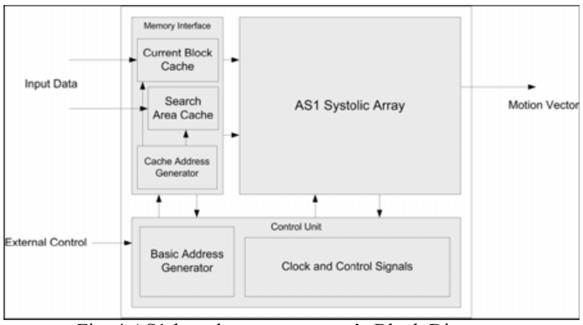

Fig. 4 AS1-based array processor's Block Diagram.

Absolute differences operation ("D" modules in Fig. 2) is optimized and implemented as shown in Fig 6.

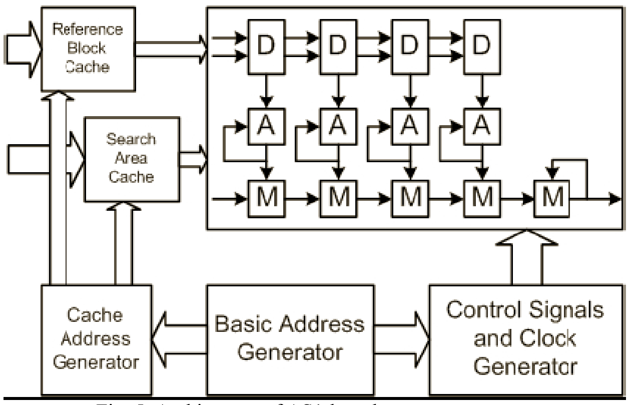

Fig. 5 Architecture of AS1-based array processor.

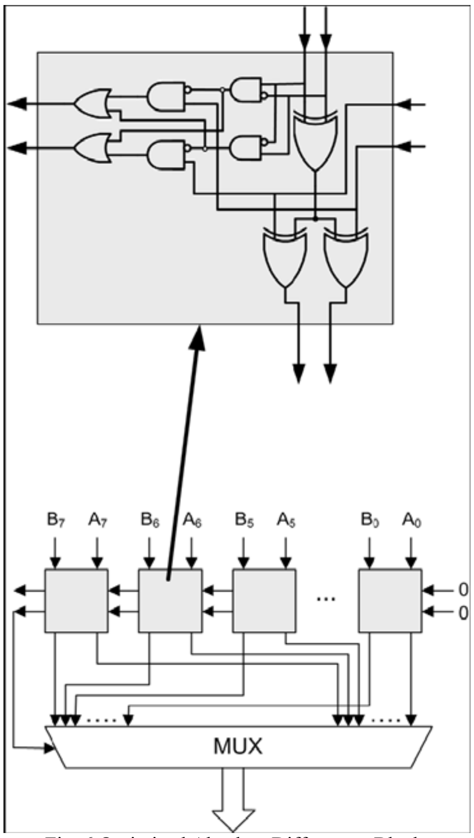

Fig. 6 Optimized Absolute Differences Block.

"A" modules in AS1 architecture are accumulators. In each time instance, they add input value to their value of internal registers.The output ports of Accumulators have 2 × log2 N + 8 bits.Output port of these modules is N LSB bits of internal register which is integer part of MAD when the N is a power of 2.Therefore, for an accurate comparison of MAD values, comparators also should have 2×log2 N+8 bits ports. But this needs more hardware for comparators, specially when the N is big. To reduce

the hardware of comparators, comparison can be done by using the 8 most-significant bits of accumulators output. This equals integer part of MAD when N is a power of 2. This assumption does not have major effect in finding the best match and MV calculation. Modified AS1 architecture calculates the MAD values of all Candidate Blocks of a row within the Search Area, simultaneously. In each 2P+N−1 time instances, the MAD value of one row of data is calculated and is stored in the respective accumulator.
Since the Current Block has N rows, the total MAD value of a block is calculated after N×(2P+N−1) time instances. Calculated MAD values are compared with previously calculated best match which was stored in a local storage within the comparator. The best match data will be updated if a better match is found. Overall best match is calculated after 2P×N×(2P+N−1) time instances when All respective MAD values of Candidate Blocks was calculated and compared with best match. Motion vector of Current Block is extracted from the overall best match data.
Implementation of Modified AS1-based array processor is performed by Xilinx Spartan and Virtex FPGA families [12], [13]. These results show Modified AS1 architecture can be implemented on a single moderate-capacity FPGA. Also, it is possible to implement the whole of the video encoder system on a single high-capacity FPGA. The results show Modification of AS1 decreases the hardware about %20.Fig. 7 shows the hardware size of Modified AS1 vs. AS1.

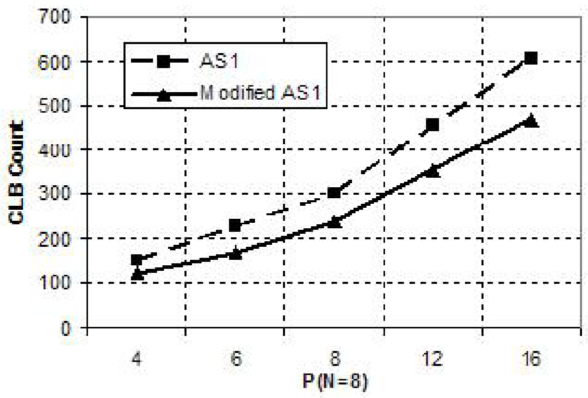

Fig. 7 Hardware size of Modified AS1 vs. AS1 on Virtex FPGA.

Table 2 show the occupied area (CLB count) for N=8 and different P values on Spartan and Virtex FPGAs, respectively in AS1.

Table 2: CLB count for N=8 and Different P values

$P(N=8)$	Spartan2	Virtex2
4	301	150
6	459	227
8	605	301
12	912	454
16	1216	607

Table 3 show the occupied area (CLB count) for P=8 and different N values on Spartan and Virtex FPGAs, respectively in AS1.

Table 3: CLB count for P=8 and Different N values

$N(P=8)$	Spartan2	Virtex2
4	597	297
8	605	301
16	609	303

Tables 4 and 5 show the occupied area (CLB count) for N=8 and different P values on Spartan and Virtex FPGAs, respectively, Im this tables AS1 with modified AS1 is compared.

Table 4:CLB count for N=8 and Different P values using Spartan FPGA

$P(N=8)$	AS1	Modified AS1
4	301	244
6	459	371
8	605	479
12	912	715
16	1216	939

Table 5: CLB count for N=8 and Different P values using Virtex FPGA

$P(N=8)$	AS1	Modified AS1
4	150	122
6	227	168
8	301	238
12	454	355
16	607	469

Table 6 shows the occupied area for P = 8 and different N values. As expected, the occupied area is almost independent from N. The maximum operating frequency varies by occupied area size and it is constant in the later case.

Table 6: CLB count for P=8 and Different N values using Spartan FPGA

$N(P=8)$	AS1	Modified AS1
4	597	471
8	605	479
16	609	483

Tables 7 and 8 show the Maximum operating frequency for N=8 and different P values. They show the maximum operating frequency is not changed by modification.

Table 7: Maximum Operating Frequencies (MHz) For N=8 and Different P Values using Spartan FPGA

$P(N=8)$	AS1	Modified AS1
4	109.878	109.878
6	109.878	109.878
8	109.878	109.878
12	109.878	109.878
16	109.878	109.878

Table 8: Maximum Operating Frequencies (MHz) For N=8 and Different P Values using Virtex FPGA

$P(N=8)$	AS1	Modified AS1
4	191.04	191.04
6	191.04	191.04
8	191.04	191.04
12	191.04	191.04
16	191.04	191.04

The maximum allowable frame rates for N=8 and different P values are calculated and depicted in the Table 9.

Table 9: Maximum CIF Frames (Frames/Sec) For N=9 and Different P Values

$P(N=8)$	Spartan II	Virtex II
4	72	125
6	38	66
8	23	40
12	11	20
16	6	12

The maximum allowable frame rates for N=8 and different P values are calculated and depicted in the Table 10.

Table 10: Maximum CIF Frames (Frames/Sec) For P=8 and Different N Values

$N(P=8)$	Spartan II	Virtex II
4	14	24
8	23	40
16	34	60

These results show the modified AS1 architecture capable to do real-time Motion Estimation with acceptable frame rate, even in high bit rate applications.

5. Conclusion

In this paper, AS1 architecture for Full Search Block Matching Algorithm and a modification of AS1 is implemented on FPGA.The results show that the FPGA is suitable for real-time motion estimation implementation. The results also show that the area occupied on a FPGA for ME is about %11 of the chip area for Virtex family and %51 for Spartan family (N = P = 8). Optimization results show that this modification decreases hardware about %20 without any effect in computation time and operating frequency. Also they show the AS1 capability for real-time Motion Estimation. Therefore, by using this technique, it is possible to implement all the video encoding system on a single FPGA chip.As a result, for N = 8 and P = 8, Modified AS1 architecturebased array processor uses %40 and %9 of FPGA resources for Spartan and Virtex, respectively. With this approach, it is possible to develop video encoder systems with high efficiency and low price in a short period of time on a single FPGA chip.

References

[1] ITU-T H.264/Advanced video coding for generic audio visual Services, Infrastructure of audiovisual services – Coding of moving video ITU-T Recommendation H.264,2005

[2] M. Ghanbari, Standard Codecs : Image Compression to Advanced Video Coding. IEE, 2003.

[3] M. Mohammadzadeh and M.Eshghi and M.M.Azadfar, "An Optimized Systolic Array Architecture for Full Search Block Matching Algorithm and its Implementation on FPGA chips", IEEE NEWCAS, 2005.

[4] M. Mohammadzadeh and M.Eshghi and M.M.Azadfar, "Parameterizable Implementation of Full Search Block Matching Algorithm using FPGA for Real-time Applications", IEEE ICCDCS, pp.200-203, NOV.2004

[5] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding tolower spatio-temporal resolutions and different encoding formats," IEEETrans. on Multimedia, vol. 2, no. 2, pp. 101–110, Jun. 2000.

[6] R. Srinivasan and K. R. Rao, "Predictive coding based on efficientmotion estimation," IEEE Trans. on Comm., vol. 2, no. 2, pp. 888–896,1985.

[7] H. G. Musmann, P. Pirsch, and H. J. Grallert, "Advances in picture coding," Proc. IEEE, vol. 73, pp. 523–548, Apr. 1985.

[8] M. Kung, VLSI Array Processors. Englewood Cliffs NJ: Prentice Hall,1987.

[9] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," IEEE Trans. on Cicuits and Systems, vol. 36, no. 10, pp.1301–1308, Oct. 1989.

[10] L. De Vos and M. Stegherr, "Parameterizable vlsi architectures for the full-search block-matching algorithm," IEEE Trans. on Cicuits and Systems, vol. 36, pp. 1309–1316, Oct. 1989.

[11] C.-H. Hsieh and T.-P. Lin, “Vlsi architecture for block-matching motion estimation algorithm,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 2, pp. 169–175, Jun. 1992.
[12] K. Tavassoli and W. Badawy, “A prototype for parallel motion estimation architecture using full-search block matching algorithm,” in International Workshop on Digital and Computational Video (DCV’02), Nov. 2002.
[13] J.-C. Tuan and C.-W. Chang, T.-S. Jen, “On the data reuse and memory bandwidth analysis for full-search block-matching vlsi architecture,” IEEE Trans. on Cicuits and Systems for Video Tech., vol. 12, no. 1, pp. 61–72, Jan. 2002.
[14] Xilinx ISE 6.1i Software Manuals, 2003.
[15] Xilinx Virtex Datasheet, Oct. 2003.
[16] Xilinx Spartan Datasheet, Sep. 2003.

M.Mahdi Azadfar received the B.E. degree in Electronics from Ferdowsi Univ. (Mashad-IRAN) in 1992. He also received his M.S degree in Electronics from Tarbiat Modaress Univ.(Tehran-IRAN) in 1995. He has worked as a researcher in Dept. of Multiplex,Network, Information Technology, at Iran Telecom Research Center (ITRC) since 1995. His research interest includes Digital Image Processing, Image and Video Coding, Analog and Digital Television, Multimedia systems, Video Broadcasting over Network. He is currently with the ITRC as a faculty member.