A VLSI Architecture for Advanced Video Coding Motion Estimation

Swee Yeow Yap, John V. McCanny
DSiP Laboratories, School of Electrical & Electronic Engineering, Queen's University of Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, Northern Ireland, UK.
E-mail: s.y.yap, j.mccanny@ee.qub.ac.uk

Abstract

With the advent of new video standards such as MPEG-4 part-10 and H.264/H.26L, demands for advanced video coding (AVC), particularly in area of variable block searching motion estimation (VBSME), are increasing. This has led to research into suitable flexible hardware architectures to perform the various types of VBSME. In this paper, we propose a new 1-D VLSI architecture for full search variable block size motion estimation (FSVBSME). The variable block size, sum of absolute differences (SAD) computation is performed by re-using the results of smaller sub-block computations. These are permuted and combined by incorporating a shuffling mechanism within each processing element (PE). Whereas a conventional 1-D architecture can process only one motion vector, this architecture can process up to 41 motion vector (MV) sub-blocks (within a macroblock) in a comparable number of clock cycles.

1. Introduction

There has been a growing interest in the use of advanced video coding (AVC) for temporal prediction (a) in order to obtain higher compression ratios and (b) to improve video quality in low-bit rate video systems. In particular, a video frame is segmented into smaller and variable block sizes to accommodate different changes in object movement within a video frame. One way to achieve this is by splitting the video frame using conventional fixed size macroblocks. Each macroblock is then further segmented into variable block sizes. A typical macroblock has a dimension of 16x16 pixels, with the smallest segmented block size (base block) being 4x4. In this case, a macroblock contains 16 base blocks corresponding to 16 motion vectors. Other variable blocks sizes correspond to derivatives of the base block. Newer video applications such as the MPEG-4 AVC part-10 [1] and H.264/H.26L include such schemes in their standard specifications.

The purpose of this paper is to present a new one-dimensional VLSI architecture for implementing full search variable block size video motion estimation (VBSME). An important aspect of this architecture is that it is able to perform a full motion search on integral numbers of 4x4 blocks sizes. As will be discussed, this requires a comparable number of clock cycles to previous 1-D architectures [2][3]. However this is capable of performing searches of up to 41 sub-motion displacements within a macroblock, as compared to one in previous 1-D systems. This new architecture requires an additional set of multiplexers and latches in the PE data path when compared with the PE's described in [2] and [3].

Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03)
ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE

The structure of the paper is as follows. Section 2 provides a brief overview of background work on Motion Estimation architectures and builds on this to develop a new architecture for a full search VBSME. The proposed architecture is then presented in more detail in section 3. The results of silicon design studies based on this are then given in section 4, with the main conclusions presented in section 5.

2. Background

Motion estimation algorithms exploit the temporal redundancy of a segmented video sequence, as described by Jain [4]. Among all the estimation algorithms, the full-search block-matching algorithm has been shown to produce the best results in terms of finding displacement vectors (motion vectors, MV), as depicted in Figure 1. Such algorithms are implemented in two stages, namely the calculation of the sum of absolute differences (SAD) for each displacement vector, followed by methods for finding the smallest SAD values. This is summarized by equations 1 and 2.

$$SAD(i,j) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\left|C(k,l) - R(i+k, j+l)\right| \quad (1)$$

$$SAD_{MIN} = \min\left(SAD(i,j)\right) \quad (2)$$

Here, C(k,l) and R(i+k,j+l) represent the current picture frame and search region's macro-block displacements respectively.

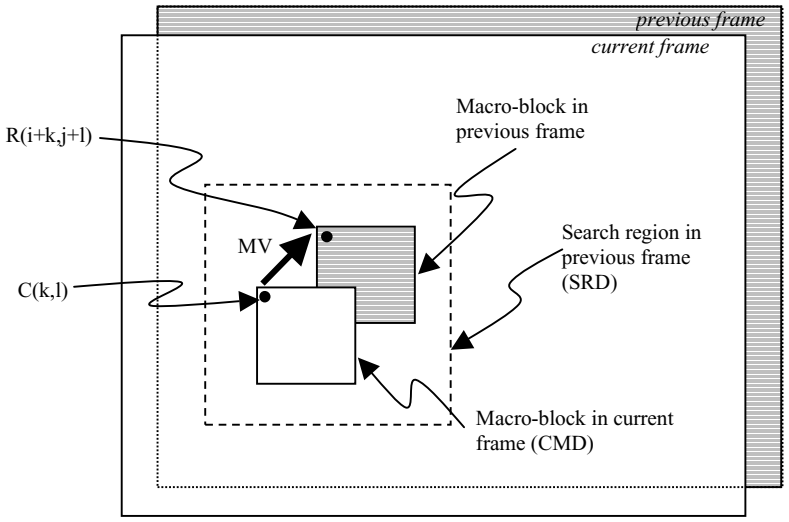

Figure 1. Block Matching

The computational requirements for block matching are high and a real-time video application usually requires a direct mapped hardware architecture. Direct mapped architectures also have important advantages in terms of reduced power dissipation. Full search algorithms, typically, can be implemented using regular 1-D or 2-D systolic or systolic-like architectures as described by Pirsch [5]. 1-D systems offer a number of attractive features over their full 2-D counterparts, in particular much less complex data scheduling and simpler structures. These architectures are also attractive for portable devices because of their lower silicon area and thus size. Kuhn's architecture [3] has

also demonstrated that flexible 1-D architectures are suitable for fast matching algorithms, such as three-step search (TSS) and pel sub-sampling.

To date, conventional VLSI architectures for computing VBSME have been based on 2-D processor systems. For example, the architecture by Vos [6] uses a 2-D array with appropriate through masking of processing elements. However, this results in low processor utilization. Shen's architecture [7] uses a smaller 2-D array with partial sum SAD calculations performed sequentially using the smallest block size, 8x8.

In advanced video coding, a macroblock is further segmented with the smallest block size being 4x4, as shown in Figure 2. This has two modes, the Macroblock mode and the 8x8 mode, as illustrated in Figures 3(a) and 3(b) respectively. Variable block sizes must be accommodated namely 4x4, 4x8, 8x4, 8x8, 16x8, 8x16 and 16x16. Referring to Figure 3b, it will be noted that there are four quarter-blocks in a macroblock, each of which contains 9 block patterns i.e. a total of 36 block patterns. However as will be observed in Figure 3a, each macroblock contains another 9 block patterns, with four of the 8x8 blocks common with the equivalent 8x8 blocks in Figure 3b. Therefore, the total number of block patterns, to be processed is 36+9-4 = 41 i.e. a total of 41 motion vectors.

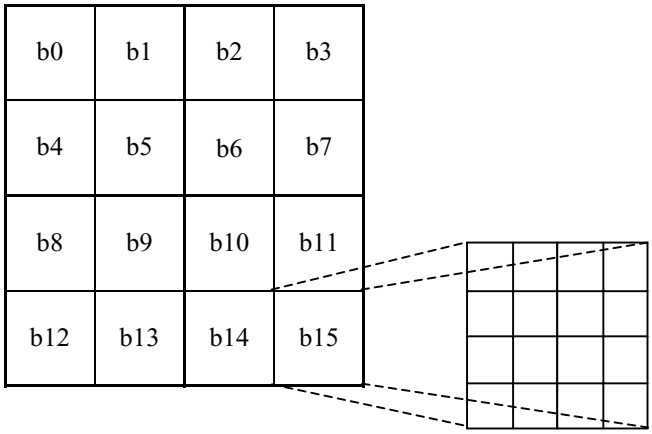

Figure 2. Segmented Macroblock

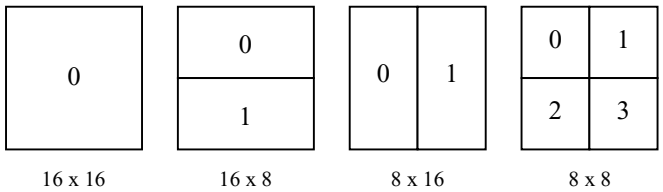

Figure 3a. MB-mode

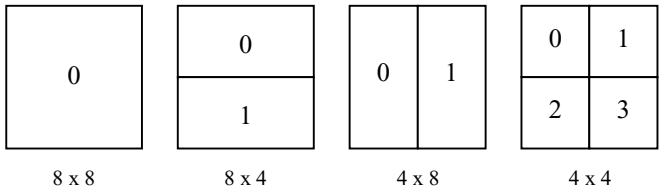

Figure 3b. 8x8-mode

Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03)
ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE

3. Proposed Architecture

The architecture presented in this paper is based on 1-D array processor. A key aspect of the approach proposed is that it incorporates within the basic processing element (PE) the means to accumulate the partial SAD through shuffling. The scheduling of the current macroblock data (CMD) and search region data (SRD) is similar to a conventional 1-D architecture [2]. However, the CMD is arranged in a raster scan sequence, as shown in Figure 4 with the SRD arranged in a dual raster scan sequence. Applying this approach to the macroblock shown in Figure 2 results in 16 SADs being computed, each with block size 4x4. The stored SADs are then re-used to compute SADs for other block sizes. This is done by permuting the sub-block SAD values and combining these in appropriate permutations to derive those for the various blocks sizes required. This approach allows the overall computational requirements to be significantly reduced by avoiding the need to derive sub-block computation values that already have been established. A total of 41 variable block size SAD values can then be processed in a single processor.

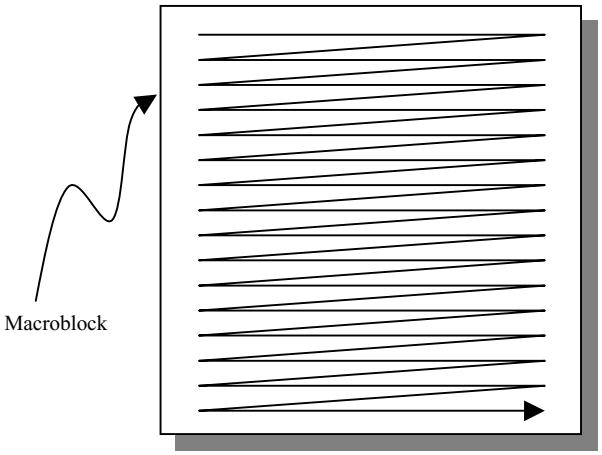

Figure 4. CMD – raster scan

Details of the proposed architecture are presented in Figure 5. The circuit operates by scheduling the CMD through a delay line and broadcasting two sets of SRD data on each clock cycle. The processing elements accumulate the absolute difference (AD) between the CMD and SRD on every clock cycle. Without taking processor latency into account, the first set of base block SADs {b0, b1, b2, b3}, shown in Figure 2, are completed on clock cycles {51, 55, 59, 63} respectively. Subsequent sets of SAD results are available on clock cycles {115, 119, 123, 127}, {179, 183, 187, 191}, and {243, 247, 251, 255}; these correspond to blocks {b4, b5, b6, b7}, {b8, b9, b10, b11} and {b12, b13, b14, b15} respectively. Other SADs are obtained through the summations of these base blocks. 16 SAD buses are therefore needed for the simultaneous and adjacent comparison of SAD values. The best vectors from each bus can be used in post-processing in various modes of operation.

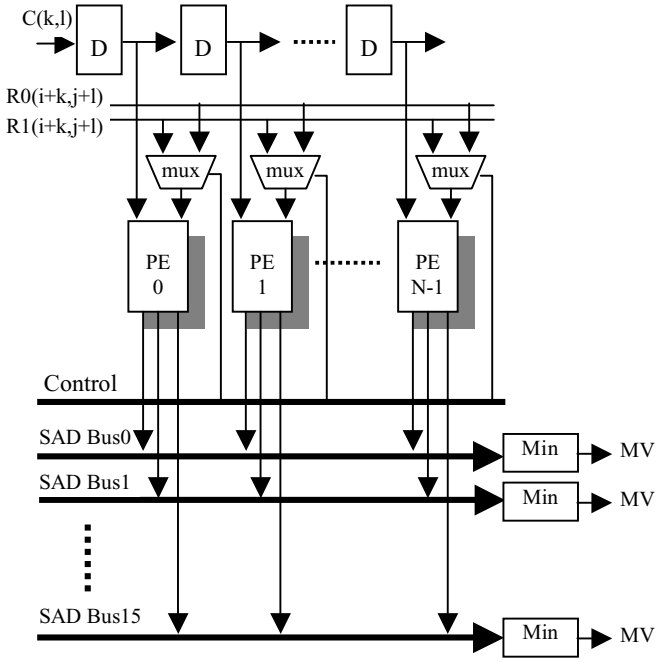

Figure 5. 1-D array VBSME architecture

With 16 PEs working concurrently, the circuit allows a total of 256 candidate MVs (16x16 search region) per sub-block to be processed in parallel. Each PE requires 256 clock cycles with up to 25 extra clock cycles for a region search (i.e. a maximum of 281 clock cycles). This is therefore comparable with existing architectures, which require a total of 256 clock cycles. Repeating this a further 16 times means that up to 4496 clock cycles are required to complete a full search. The data flow of within the array is summarized in Table 1 with the CMD being denoted by C(x,y), and the SRD data being denoted by R(u,v).

Table 1. Data flow schedule

Clk	PE_0	PE_1		PE_{14}	PE_{15}
	(t-0)	(t-1)		(t-14)	(t-15)
0	C(0,0)-R(0,0)	C(0,0)-R(1,0)		C(0,0)-R(14,0)	C(0,0)-R(15,0)
1	C(1,0)-R(1,0)	C(1,0)-R(2,0)		C(1,0)-R(15,0)	C(1,0)-R(16,0)
…	…	…	…	…	…
14	C(14,0)-R(14,0)	C(14,0)-R(15,0)		C(14,0)-R(28,0)	C(14,0)-R(29,0)
15	C(15,0)-R(15,0)	C(15,0)-R(16,0)		C(15,0)-R(29,0)	C(15,0)-R(30,0)
…					
240	C(0,15)-R(0,15)	C(0,15)-R(1,15)		C(0,15)-R(14,15)	C(0,15)-R(15,15)
241	C(1,15)-R(1,15)	C(1,15)-R(2,15)		C(1,15)-R(15,15)	C(1,15)-R(16,15)
…	…	…	…	…	…
254	C(14,15)-R(14,15)	C(14,15)-R(15,15)		C(14,15)-R(28,15)	C(14,15)-R(29,15)
255	C(15,15)-R(15,15)	C(15,15)-R(16,15)		C(15,15)-R(29,15)	C(15,15)-R(30,15)

3.1. PE Architecture

The PE cell proposed is shown in Figure 6. The major difference from this and other PE architectures is the accumulator latches. The proposed PE cell has 18 accumulator latches, with these being connected to three 18 to 1 muxes, namely muxes A, B and C. Muxes A and B are used as a feedback path to the adder and mux C is used to select which latches are output. The output is then de-muxed to the corresponding SAD bus.

The PE cell operates by shuffling the latches used during the AD accumulation operations. The first latch operates for 4 cycles, then the second for 4 cycles and so on. The four latches are used and reshuffled in this way until each has operated for 16 cycles (the time to compute a four base block SAD values). The next four latches then take over the shuffling and so on until all 16 latches are utilized. This corresponds to the processing of base block sequences {b0, b1, … , b15}.

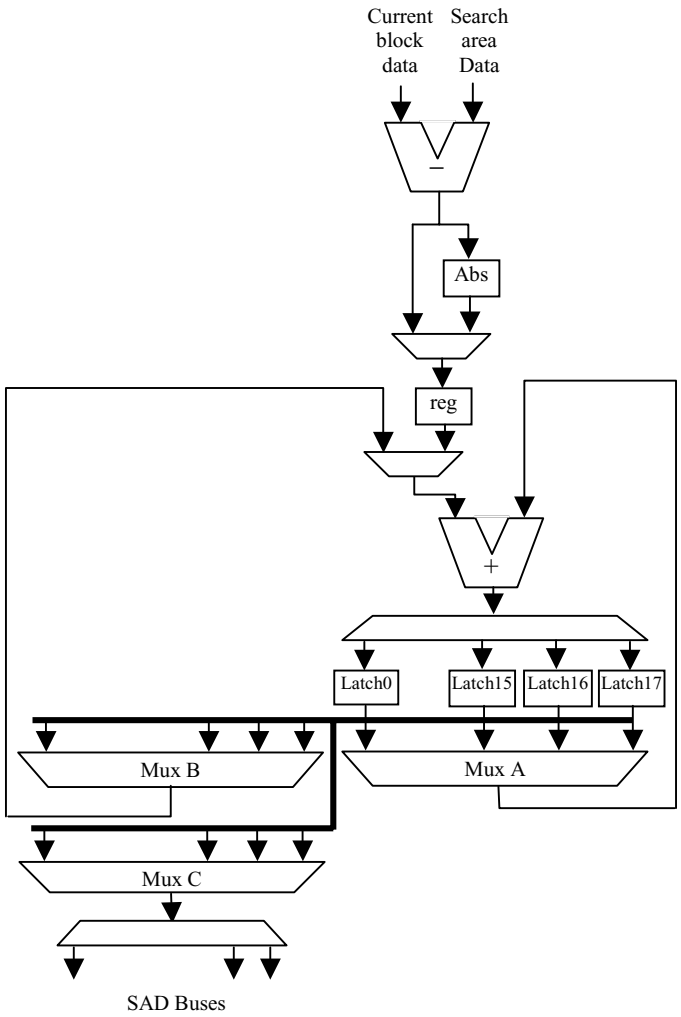

Figure 6. Processing element

Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03)
ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE

When this has been completed, the next SAD sequences are processed on an 8x8 block basis (i.e. there are four 8x8 quarter blocks in each macroblock). In order to optimize latch utilization, the sub-blocks of each 8x8 quarter block are computed in the sequence of two 4x8, two 8x4, and one 8x8 blocks. The results for the first quarter block are then summed in the order {b1, b5} → latch16, {b0, b4} → latch16, {b0, b1} → latch16, {b4, b5} → latch17, {{b0, b1}, {b4, b5}} → latch0. The other three quarter blocks are processed in a similar fashion with the final results stored as {{b2, b3}, {b6, b7}} → latch1, {{b8, b9}, {b12, b13}} → latch2, and {{b10, b11}, {b14, b15}} → latch3. By this time, a full 8x8 mode motion estimation has been completed. A further 5 clocks are required to complete the full MB-mode estimation, in this case, in the sequence of two 8x16, two 16x8, and one 16x16 blocks. The corresponding summations are then summations {latch0, latch2} → latch4, {latch1, latch3} → latch5, {latch0, latch1} → latch6, {latch2, latch3} → latch7, and finally {latch12, latch13} → latch8. The whole sequence is summarized in Table 2.

Table 2. Bus lines allocation

	Bus line	Blocks (b)	Block size	Latch		Bus line	Blocks (b)	Block size	Latch
1	0	0	4x4	0	22	5	3,7	4x8	16
2	1	1	4x4	1	23	6	2,6	4x8	16
3	2	2	4x4	2	24	7	2,3	8x4	16
4	3	3	4x4	3	25	8	6,7	8x4	17
5	4	4	4x4	4	26	9	2,3,6,7	8x8	1
6	5	5	4x4	5	27	10	9,13	4x8	16
7	6	6	4x4	6	28	11	8,12	4x8	16
8	7	7	4x4	7	29	12	8,9	8x4	16
9	8	8	4x4	8	30	13	12,13	8x4	17
10	9	9	4x4	9	31	14	8,9,12,13	8x8	2
11	10	10	4x4	10	32	15	11,15	4x8	16
12	11	11	4x4	11	33	0	10,14	4x8	16
13	12	12	4x4	12	34	1	10,11	8x4	16
14	13	13	4x4	13	35	2	14,15	8x4	17
15	14	14	4x4	14	36	3	10,11,14,15	8x8	3
16	15	15	4x4	15	37	4	0,1,4,5,8,9,12,13	8x16	4
17	0	1,5	4x8	16	38	5	2,3,6,7,10,11,14,15	8x16	5
18	1	0,4	4x8	16	39	6	0,1,4,5,2,3,6,7	16x8	6
19	2	0,1	8x4	16	40	7	8,9,12,13,10,11,14,15	16x8	7
20	3	4,5	8x4	17	41	8	full macroblock	16x16	8
21	4	0,1,4,5	8x8	0					

4. Implementation

The architecture described has been captured using VERILOG and used to synthesize an ASIC demonstrator design. This is based on a 130nm CMOS technology (1.2V) and has been implemented using TSMC's standard cell library. The circuit design is based on a 16 PE 1-D array, has a search range of 16x16 ({-8,7}) and can handle the variable block sizes listed in table 3. If a wider search range of 32x32 ({-16,15}) is required then a 4 times search readily can be performed. The input word-lengths used were 8

bits. This is consistent with common video standards. The memory scheme used is similar to that described in [2] and [3]. The design created contains 108K gates and can operate at frequencies of up to 100 MHz, i.e. up to 13 frames per second (fps) in 4CIF video resolution.

The results obtained also show that, for QCIF video resolution at 15 fps, such a circuit, clocked at 6.68 MHz, has a power consumption of only 9.1 mW, which is very attractive for low power portable applications. A comparison between this circuit and previous VBSME circuits is presented in table 4. An exact comparison is complicated by the fact that these have been implemented with different technologies and exhibit variations in their specifications and capabilities. Nevertheless, it will be noted that the design presented exhibits the highest level of flexibility in terms of block sizes catered for. It offers the highest clock rates and has a gate count of around 40% of the most flexible alternative - that of Vos [6]. In addition, it should be pointed out that the flexibility of the architecture presented means that it is easy to re-program the latches to cater for other block sizes, should these be needed in future video standards.

Table 3. Performance

Algorithm	Variable block size full search motion estimation
Number of PE	16 (1-D array)
Searching range	16 x 16
Block size	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16
Technology	TSMC 130nm CMOS std. cell lib.
Gates count	108k
Max frequency	100 MHz
Example circuit	6.68 MHz, 15 fps, 9.1 mW, QCIF

Table 4. Comparison of some VBSME core

	Vos '95[6]	Fujita '97[8]	Shen '01[7]	This work
PE number	16x16	---	64	16
Search range	16x16	---	16x16, 32x32	16x16, 32x32
Block size	2nx2n, n>=1, (masking)	16x16, 8x8	8x8, 16x16, 32x32	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16
Process	0.6um	0.35um	0.6um	0.13um
Voltage	---	---	5V & 2.5V	1.2V
Frequency	72MHz	15MHz	60MHz	100MHz
Gate count	263k	12k	67k	108k

5. Conclusion

In this paper, a new 1-D VLSI architecture for full search, variable block-size motion estimation (FSVBSME) is presented. This architecture can process up to 41 variable blocks motion vectors in a macroblock in a similar number of clock cycles to other conventional 1-D architectures. A key aspect is the shuffling, permutation and combination of partial SAD values within each PE. Design studies show that this is very suitable for the next generation of advanced video coding, particularly devices requiring small silicon area, a high temporal compression ratio at low bit rates and low power dissipation. The concepts presented can be extended to half and quarter pixel motion estimation for FSVBSME. Research on this is currently underway and will be discussed in a future paper.

6. References

[1] ISO/IEC 14496-10, "Coding of Moving Pictures and Audio", 2002.

[2] K. M. Yang & L. Wu, "A Family of VLSI Designs for the Motion Compensation Block-Matching Algorithm", IEEE Transactions On Circuits and Systems, vol. 36, no. 10, pp. 1317-1325, October 1989.

[3] P. M. Kuhn, "Fast MPEG-4 Motion Estimation: Processor Based and Flexible VLSI Implementations", Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, vol. 23, pp 67-92, October 1999.

[4] J. R. Jain & A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding", IEEE Transactions on Communications, vol. COM-29, no. 12, pp. 1799-1808, December 1981.

[5] P. Pirsch, "VLSI Architectures for Video Compression – A Survey", Proceedings of the IEEE, vol. 83, no. 2, pp. 220-246, February 1995.

[6] Luc de Vos & M. Schobinger, "VLSI Architecture for a Flexible Block Matching Processor", IEEE Transactions on Circuits and Systems for Video Technology, vol. 5, no. 5, October 1995, pp.417-428.

[7] J.F. Shen et al, "A Novel Low-Power Full-Search Block-Matching Motion-Estimation Design for H.263+", IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 7, July 2001, pp.890-897.

[8] G. Fujita et al, "A new motion estimation core dedicated to H.263 video coding", Proceedings of 1997 IEEE International Symposium on Circuits and Systems in the Information Age ISCAS '97, Part vol. 2, 1997, pp. 1161-4 vol. 2.

ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE