International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:4, 2009

Low power and less area architecture for integer motion estimation

C Hisham, K Komal, and Amit K Mishra, *Member, IEEE*

Abstract—Full search block matching algorithm is widely used for hardware implementation of motion estimators in video compression algorithms. In this paper we are proposing a new architecture, which consists of a 2D parallel processing unit and a 1D unit both working in parallel. The proposed architecture reduces both data access power and computational power which are the main causes of power consumption in integer motion estimation. It also completes the operations with nearly the same number of clock cycles as compared to a 2D systolic array architecture. In this work sum of absolute difference (SAD)-the most repeated operation in block matching, is calculated in two steps. The first step is to calculate the SAD for alternate rows by a 2D parallel unit. If the SAD calculated by the parallel unit is less than the stored minimum SAD, the SAD of the remaining rows is calculated by the 1D unit. Early termination, which stops avoidable computations has been achieved with the help of alternate rows method proposed in this paper and by finding a low initial SAD value based on motion vector prediction. Data reuse has been applied to the reference blocks in the same search area which significantly reduced the memory access.

Keywords—sum of absolute difference, high speed DSP.

I. INTRODUCTION

Portable video devices like videophones and cameras require low power and low silicon area implementation of video compression algorithms. Motion estimation is one of the most time and power consuming block in any video compression algorithm. For example in H.264 codec implementation, integer motion estimation contributes to 74.29% of the computational complexity and 77.49% of memory access [1]. There are several algorithms for motion estimation like full search block matching algorithm (FSBMA) [2] and fast algorithms like three step search, hexagonal search etc [2]. FSBMA has simple and regular structure which makes it one of the best choices for hardware implementation of motion estimation [3]. The basic idea of block matching algorithm (BMA) is to divide the current frame in a video sequence, into smaller blocks [3], [4]. For each block we try to find the corresponding block from the search area of the previous frame, which matches the most with the current block. To find how closely two blocks match with each other, we have different approaches like, calculation of mean square error(MSE), sum of absolute difference(SAD), mean of absolute difference(MAD) etc. [2]. Out of these methods SAD is one of the most preferred algorithms, because it requires only addition and no multiplications. The SAD operation is represented by the following equation [5].

$$SAD_{P,Q}(m,n) = \sum_{i=0}^{P-1}\sum_{j=0}^{Q-1} |c(i,j) - r(i+m, j+n)| \quad (1)$$

P and Q are length and width of the block. Each can be 4,8 or 16 pixels. m and n give the maximum horizontal and vertical displacements.

Processing element (PE) array based architectures have been proposed, like the one proposed in [6], where a two dimensional array of PEs iteratively compute the partial SAD value until the final value is determined.Early termination is used in this method, so that power consumption is reduced by 30% [6]. A method by using most significant bit(MSB) first bit-serial implementation [5] is suggested to reduce the area of systolic architecture. Since the consecutive reference frames are overlapped, the reuse of reference pixels can be used to reduce the memory access [1]. For variable block size motion estimation, the results obtained from smaller blocks can be reused to compute motion vectors for larger blocks [7], [8]. Sum of absolute difference (SAD) calculation is the most repeated operation in any block matching algorithm. The main objective in a block matching algorithm is to find a block with minimum SAD [9]. Hence, we can stop the computations if the SAD being calculated is greater than the stored minimum SAD.

The authors are with the Department of Electronics and Communication Engineering, Indian Institute of Technology, Guwahati, India e-mail: akmishra@ieee.org

Most of the works done so far to reduce the area and memory access of full search is by compromising the system throughput and optimizing at the circuit level. Our work reduces the area and memory access an architecture level, without a considerable change in the throughput.

The rest of the paper is organized as follows. The Next section describes about the proposed architecture and several optimization techniques used in this architecture like alternate rows, data reuse and minimum SAD prediction. Section 3 describes about the results and the last section concludes the paper.

II. Proposed Architecture

Figure-1 shows the proposed architecture for 4×4 integer motion estimation. Here for each 4×4 block in the current frame, a 4×4 block with minimum SAD is to be found in the given search area of reference frame. There are two major blocks in this architecture, a parallel 2D SAD unit(2×4) and one 1D SAD unit. This architecture uses the help of early termination to avoid nonessential operations so that operations for a particular reference block will be stopped if calculated SAD $>$ SAD minimum. The sooner the termination occurs the more computations can be saved.

In this architecture mainly two techniques have been applied to achieve fast early termination.

1) **Initial SAD value prediction**
   Initial value of SAD minimum is calculated from a block which is directed by the predicted motion vector. The motion vectors are predicted from three previous motion vectors using median operation [5].
2) **Alternate rows method** In this architecture, instead of calculating SAD for the entire 4x4 blocks at a time, we are calculating the SAD in two steps. At first SAD for a strip which can represent the block more effectively, and if it is less than SAD minimum, SAD of remaining part of the block. A strip with half size of a block having alternate rows or columns can more effectively represent that block than a strip of half size with consecutive rows or columns. So if we are taking alternate rows or columns of both reference block and current block and computing their SAD(parallel SAD) first, we can disable 1D SAD unit in more percentage. Percentage of disabling 1D unit for continuous rows $[2\times4]$, continuous columns $[4 \times 2]$, alternate rows $[2 \times 4]$ and alternate columns $[4 \times 2]$ has been calculated for different test videos and it is observed that percentage of disabling 1D unit is more while taking alternate rows of both current block and reference block and computing their SAD.

Memory access has been reduced in this architecture with the help of re-use of accessed reference pixels among different processing elements .

A. 2D Parallel unit

Figure 2 shows 2D unit for 4×4 block. In this, odd rows of current block are loaded in to PE arrays and they remain fixed in PE. Odd rows of different reference blocks within the search area will be loaded to the PEs in each clock cycle. Parallel 2×4 section will compute the SAD of alternate rows in parallel within one clock cycle (parallel SAD). If parallel SAD is less than stored SAD minimum comparator 1 output will be set to '1'. If the comparator 1 output is '1', parallel SAD value and information of that block will be stored in temporarily memory, which will manage the computations of 1D SAD unit, even if comparator output=1 for few consecutive blocks.
Pixels of reference block are reused by adjacent PEs. Two horizontally adjacent 4×4 reference blocks differ only by 4 pixel values and since we are using only odd rows of these blocks, only 2 new memory access are required by this unit per reference block.

B. 1D SAD unit

1D SAD unit which is shown in figure 3 will calculate the SAD of even rows of the block by taking more clock cycles. The even rows of current block will be stored in two circular buffers. The reference frame pixels will be accessed by two PEs in this unit. Initially parallel SAD results will be added with sum of absolute difference from even rows in that cycle(SAD even) to form SAD total. In remaining cycles SAD total will be added with SAD even to update SAD total using same adder.

$$SADtotal = SADParallel + SADevenrows \quad (2)$$

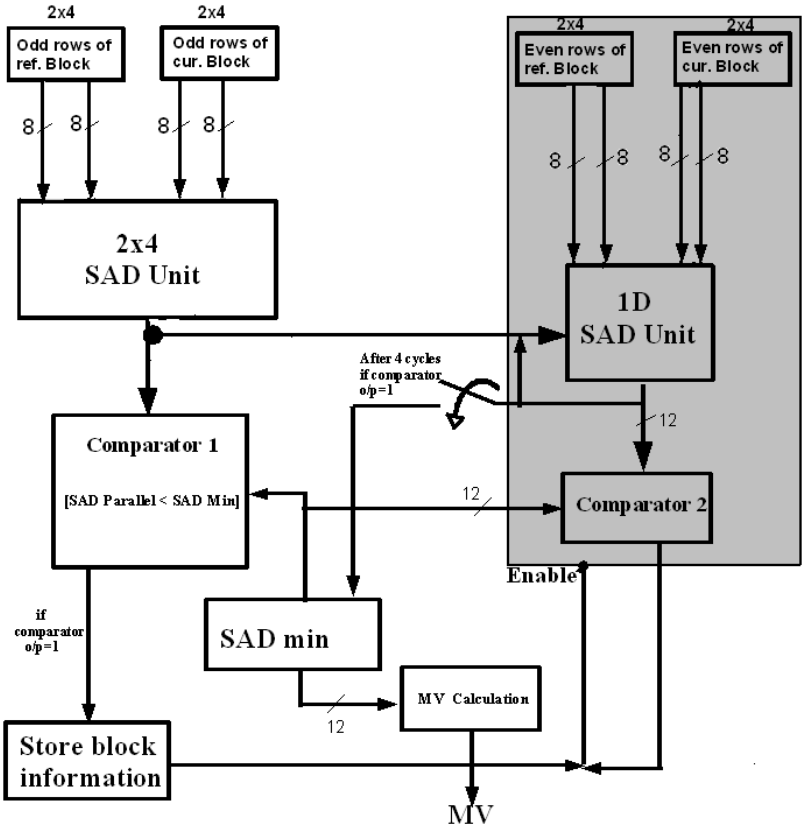

Fig. 1. Proposed architecture

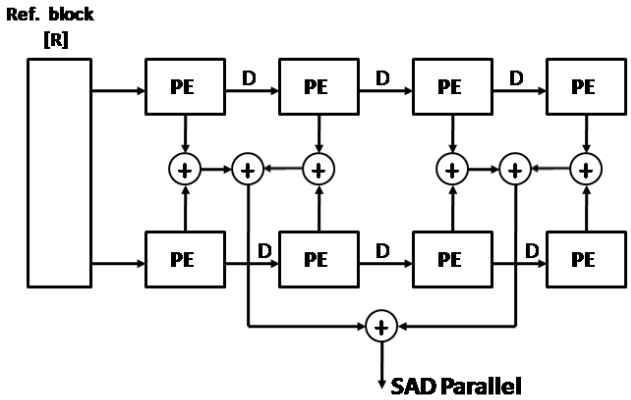

Fig. 2. Parallel 2x4 SAD unit

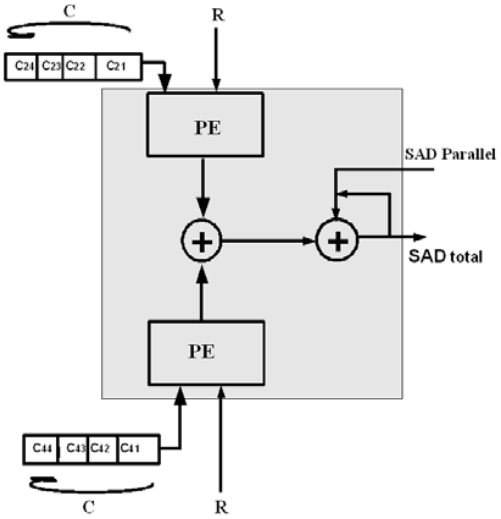

Fig. 3. 1D SAD unit (2x1)

In each clock cycle, SAD total will be compared with SAD minimum. Early termination is applied for 1D SAD unit also. Operations in the 1D SAD unit will be the stopped if the sum (SAD total) goes above the SAD minimum. Since the SAD of even rows are added with SAD parallel, there is high chance of early termination. Table 2 shows the effect of early termination in 1D unit for different video

TABLE I
COMPARISON OF PERCENTAGE OF DISABLING 1D UNIT IN DIFFERENT METHODS FOR BLOCKS OF SIZE 4X4.

test video	continuous 4x2		continuous 2x4		alternate columns		alternate rows	
	with prediction	without prediction	with prediction	without prediction	with prediction	without prediction	with prediction	without prediction
Mobile (CIF)	66.86	62.16	68.01	56.57	69.68	62.61	70.31	63.00
Tempete (CIF)	73.47	63.95	72.76	59.60	75.75	64.72	76.1	64.87
News (QCIF)	75.75	60.00	75.95	58.88	77.49	60.98	77.68	61.33
Foreman (QCIF)	79.88	67.28	79.16	66.88	81.7	68.87	82.1	69.25
Silent(QCIF)	82.60	66.20	83.32	67.17	83.55	67.70	83.82	68.02
Container(QCIF)	72.68	51.26	72.30	52.03	81.55	51.74	82.13	51.99

TABLE II
AVERAGE NUMBER CYCLES PER SAD IN 1D UNIT

test video	Average number cycles in 1D unit
Mobile	1.9
Tempete	1.95
News	2.25
Foreman	2.15
Silent	2.3
Container	1.92

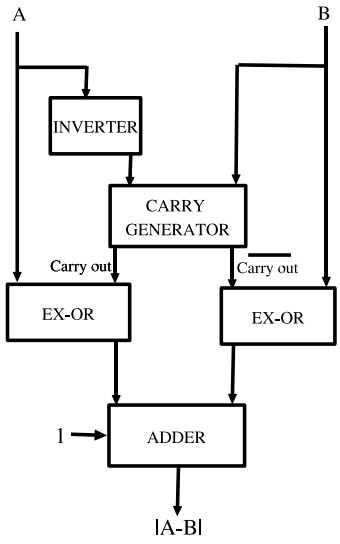

Fig. 4. Processing Element(PE)

frames. It can be observed that 1D unit finishes computations around 2 cycles instead of 4 cycles due to early termination inside the 1D SAD unit. By result it has been observed that around 75% of the cases 1D SAD block will be disabled. 1D unit operations will be completed by an average of 2-3 cycles since we are using two PE. In general, even for a fast moving video the total 1D operations can be completed while finishing parallel operations even if the number of SAD PEs in 1D unit is 4 times less than that of 2D unit. In general for $N \times N$ block,we need $N^2/2$ PEs in 2D SAD unit and $(N^2/8)$ PEs in 1D SAD unit. So totally we need only $(5N^2/8)$ PEs in this architecture instead of N^2 PEs in systolic architectures [5]. So for $N = 4$, Parallel unit will be having 8 SAD PEs and so the 1D unit must have two PEs which are working in parallel(one for each row). Total sequence of operations can be explained by the following algorithm.

C. Processing Element(PE)

The processing element(PE) will calculate the absolute difference between two pixel values. For calculating the absolute difference between two pixel values, the 2's complement of smaller pixel value is added with the bigger pixel value. In this work, to calculate absolute difference between two pixel values A and B, find $\bar{A}$ and we are generating carry of $\bar{A}+B$ by using a carry generator(shown in

Algorithm 1 Find the motion vector for a particular current block

Calculate the initial SAD minimum from the predicted block.
if SAD Parallel $\geqq$ SAD minimum **then**
  disable 1D operations for that block and load next reference block to 2D SAD unit.
else if SAD Parallel $<$ SAD minimum **then**
  Complete SAD calculation of remaining even rows with 1D unit and load next reference block to 2D SAD unit.
  if SAD total $\geqq$ SAD minimum **then**
    stop the computation of 1D unit.
  else if SAD total $<$ SAD minimum **then**
    continue the computations.
    if output of second comperator=1 after all computations **then**
      Replace SAD minimum by SAD total.
    else if output of second comparator $\neq 1$ after all computations **then**
      Take the next inputs to the 1D unit.
    end if
  end if
end if
Calculate the motion vector from the reference block with minimum SAD.

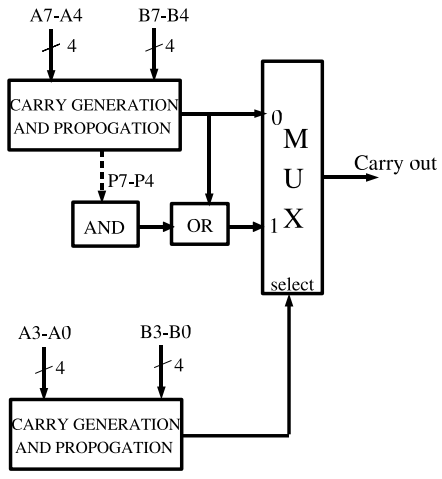

Fig. 5. Carry generator

figure 4). If carry is not generated, B is the smaller one. Find $\bar{B}$ (of the smaller number).
The concept used here for absolute difference calculation is $\bar{B} = 2^n - 1 - B$, so $|A-B| = A+\bar{B}+1$ and neglect $n+1^{th}$ bit to get absolute difference between A and B.

1) ***Carry generator****:* Carry generator is shown in fig.5. In this work instead of finding carry out of 8^{th} bit, we are finding carry of first nibbles and second nibbles separately to reduce the delay. In this work area is reduced by grouping all the common logical functions together and by storing them. For two numbers A and B, possibility of carry generation on i^{th} bit will be $G = A(i) * B(i)$, and possibility of propagating previous carry is $P = A(i) + B(i)$. So carry out of i^{th} bit is $G + P * carryout(i-1)$.

III. Results and discussions

From table-1 it can be observed that alternate pixel method proposed in this architecture gives better results than consecutive pixel methods. Among alternate pixel methods, alternate row method gives a slightly better result. From table 1 it can also be observed that prediction of minimum SAD by predicting motion vectors improved the disabling of 1D unit by more than 10%. Variation of disabling of 1D unit with respect to correlation between reference frame and current frame has been studied and given in table 3. Table-3 shows even if two frames are highly uncorrelated, around 60% of disabling of 1D unit can be achieved.
From table 4 it can be observed that the proposed method have less hardware (about half) compared to the parallel architecture.

Bandwidth of memory access is reduced in this method compared to parallel architecture. Table 5, 6 shows that this architecture achieved speed of 2D systolic array [5] architecture with less memory access. Table 6 shows the number of PEs and memory access/cycle for N=16, m=24 and n=16.

IV. Conclusion

This paper proposes an architecture for full search integer motion estimation in which a 2D parallel unit and one 1D unit works in parallel. Data reuse has been employed for reference blocks in a given search area to reduce the memory access. The

TABLE III
VARIATION OF PERCENTAGE OF DISABLING 1D UNIT FOR DIFFERENT FRAME NUMBERS

Test video	frame nos reference., current.	Correlation	Percentage of disabling 1D unit	Avg. number of cycles in 1D unit
Tempete	9,10	0.96	76.12	1.87
	8,10	0.92	74.19	1.82
	7,10	0.87	74.84	1.85
	6,10	0.82	74.56	1.87
	5,10	0.79	73.8	1.85
	1,100	0.20	60.01	2

TABLE IV
COMPARISON OF FUNCTIONAL UNITS WITH 4X4 ARCHITECTURE

Block size	2D Systolic Architecture [5]	Proposed Method
4x4	16 absolute diff. PEs 15 adders, 1 comparator	10 absolute diff. PEs , 9 adders, 2 comp.

TABLE V
COMPARISON WITH DIFFERENT ARCHITECTURES

Architecture	Zhang and Gao [10]	Chen et al [11]	A.M. Campos et al. [12]	Proposed Method
Num. of abs.diff PEs	N^2	N^2	N^2	$5N^2/8$
Bytes/cycle	$2n+N$	$N+1$	$2N$	$(N/2+E^*)$

Here N is the block size m, n are maximum horizontal and vertical displacements and E can be N/2 or 0 depends on whether 1D unit is enabled or not.

TABLE VI
COMPARISON WITH DIFFERENT ARCHITECTURES FOR N=16, M=24, N=16

Architecture	Zhang and Gao [10]	Chen et al [11]	Campos [12]	Proposed Method
Num. of abs.diff PEs	256	256	256	160
Bytes/cycle	55296	30816	55296	$Bestcases$: 25673* $Normalcases$: 28862* $Worstcases$: 31644*

$*$ Based on best, normal and worst situations from table 1

alternate rows method proposed in this architecture and prediction of motion vector helps to achieve early termination at the earliest . Our results shows that this method significantly reduced the hardware and memory access without reducing the speed compared to 2D systolic architectures.

REFERENCES

[1] 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 Circuits and Systems for Video technology*, vol. 17, MAY. 2007.

[2] I. Richardson, *H.264 and MPEG-4 Video compression*. John Wiley and Sons Ltd, 2003.

[3] Z.Wujian and Z. Runde, "A high-throughput systolic array for motion estimation using adaptive bit resolution," *IEEE Trans. on Circuits and Systems for Video technology*, pp. 378–381, Mar. 2001.

[4] J. Olivares, J. Hormigo, J. Villalba, and I. Benavides, "Minimum sum of absolute differences implementation in a single fpga device," *Lecture Notes in Computer Science, Springer*, vol. 3203, pp. 986–990, Aug. 2004.

[5] M.H.Brian and H.W.Leong, "Serial and parallel fpga-based variable block size motion estimation processors," *Journal of Signal Proc.Systems*, vol. 51, pp. 77–98, Aug. 2007.

[6] K. Lam and C. Tsui, "Low power 2-d array vlsi architecture for block matching motion estimation using computation suspension," *IEEE Workshop on Signal Proc. Systems*, pp. 60–69, 2000.

[7] S. Lpez, F. Tobajas, A. Villar, V. de Armas, J. F. Lpez, and R. Sarmiento, "Low cost efficient architecture for h.264 motion estimation," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 412–415, May 2005.

[8] Z. Liu, Y. Huang, Y. Song, S. Goto, and T. I. IPS, "Hardware-efficient propagate partial sad architecture for variable block size motion estimation in h.264/avc," *Great Lakes Symposium on VLSI*, March 2007.

[9] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, and L.-G. Chen, "Survey on block matching motion estimation algorithms and architectures with new results," *Journal of VLSI Signal Processing*, vol. 42, no. 8, pp. 297–320, 2006.

[10] L.Zhang and W. Gao, "Improved fsbm algorithm and its vlsi architecture for variable block size motion estimation of h.264," *International Symposium on Intellig. Signal Pro.Comm. Syst.*, pp. 445–448, 2005.

[11] C. Chen, S. Chien, Y. Huang, T. Chen, T. Wang, and L.G.Chen, "Analysis and architecture design of variable block-size motion estimation for h.264/avc," *IEEE Trans. on Circuits and Systems*, vol. 53(2), p. 578593, 2006.

[12] A. Campos, F. Merelo, M.A.M.Peiro, and J. Esteve, "Integer-pixel motion estimation h.264/avc accelerator architecture with optimal memory management," *Microprocessors and Microsystems in Elsevier*, 2007.