Single access variable block size motion estimation for multimedia SoC

Saengrawee Pratoomtong
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706, USA
pratoomt@cae.wisc.edu

Yu Hen Hu
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706, USA
hu@engr.wisc.edu

***Abstract*—To obtain an optimum rate-distortion characteristic, it is crucial to have accurate motion and residue information to guild the optimum block size selection process. In this paper, we first modified the full search block matching motion estimation(FBME) algorithm to accommodate the motion vector and sum of absolute different(SAD) calculation of each of the 7 modes when perform FBME on one macroblock without any substantial change in the original program. Then we transform the algorithm to reduce the excessive data access to reference frame. With the reduction of redundant data access and the accuracy of the FBME algorithm, the purpose algorithm provides the most accurate motion and residue information without substantially increasing the complexity and memory requirement of the variable block size motion estimation (VBME).**

Keywords—variable block size motion estimation, FBME, data access redundancy, algorithm transformation,SoC.

Topic area—image/video coding and processing, standards.

I. INTRODUCTION

Variable block size motion estimation is one of the key featured in the new video coding standard, H.264. Since only the decoder is standardized, there are many variations to implement variable block size motion estimation which is one of the most complex and demanding algorithm in the encoder. In [1] the motion vector (MV) of the current block is predicted from neighbors and then fast MV search is performed in the area pointing by the predicted MV. The drawback of this method is that the predicted MV may not yield the global minimum and the fact that the process is neighbor dependent inhibits parallel processing. [2] perform full search on 8×8 block size. The neighbor blocks are then merged if the distance between MV of those blocks is smaller than a certain threshold. The performance of this method is mainly depending on the value of the threshold which can be varied from application to application and it adds the complexity to the algorithm. [3] presents a systolic array type hardware implementation which contains 256 processing elements (PEs). Feeding data through the systolic array is a demanding task and with 256 PEs, hardware implementation consumes substantial power and chip area which is not suitable for power and area constraint platform such as system on chip (SoC).

In this paper, we extend our work in [4] and present an enhance FBME algorithm which can accommodate VBME and has lower peak memory usage and bandwidth utilization than the original algorithm while still maintaining the same accuracy. In section 2, we first review our previous work. In section 3, we present the enhance algorithm. Discussion and result are presented in section 4. Finally, we conclude this paper in section 5.

II. FBME ALGORITHM

The FBME algorithm produces a motion vector (MV), which yields minimum mean-absolute distortion (MAD) between the current block and the $(2p+1)^2$ candidate blocks within the search area, for each of the N × N block of pixel in the current frame. Fig. 1 shows the high-level language representation of FBME algorithm. It can be seen that the pixel in the current frame is used without overlapping. To complete motion estimation for one block, each pixel in a current block is used repeatedly $(2p+1)^2$ times. In other words, the life span of the pixels in each block is the time takes to complete motion estimation for one block. Therefore, as long as the cache is big enough to store one current block, the temporal locality of the current block is fully exploited, meaning each pixel in the current frame can be reused for each iteration. On the other hand, it is difficult to fully exploit the temporal locality of the pixel in a search frame since the pixel is overlapped when used in each search block. In order to complete ME of one block, $(2p+1)^2 N^2$ pixels need to be accessed but only $(N+2p)^2$ pixels exist per search area. The access redundancy gets more severe as the block size or frame size increases. In [4], under the assumption that search range equals to half of block size, we rearrange the loop to remove the reference pixel access redundancy between two adjacent search block (level 1 optimization) and search strip (level 2 optimization). We are able to achieve single reference pixel access per search strip and per search area when performing FBME on one block at level one and level two optimization respectively.

III. ALGORITHM ENHANCEMENT

A. *Full search block matching variable block size motion estimation (FBVBME)*

H.264 standard divides picture into macroblock of 16×16 pixels. Therefore in order to obtain the MV and SAD of all 41 blocks with various size (1 block of 16×16 pixel, 2 block of 16×8 and 8×16 pixel, 4 block of 8×8 pixel, 8 block of 8×4 and 4×8 pixel, 16 block of 4×4 pixel), we perform FBME on block

```
1 do v = 0 to Nv-1
2 do h = 0 to Nh-1
3   MV(h,v) = (0,0);
4   Dmim(h,v) = 8 ;
5   do m = -p to p
6   do n = -p to p
7     SAD  = 0;
8     do i = 0 to N-1
9     do j = 0 to N-1
10        currx = hN+j;  curry = vN+i;
11        refx = hN+j+m;  refy = vN+i+n;
12        SAD = SAD + |x(currx,curry)-y(refx,refy)|;
13    enddo j,i
14    if Dmim(h,v) > SAD
15      Dmim(h,v) = SAD;
16      MV(h,v) = (m,n);
17    endif
18 enddo n,m,h,v
```

Fig. 1. Six-level nested do loop FBME algorithm.

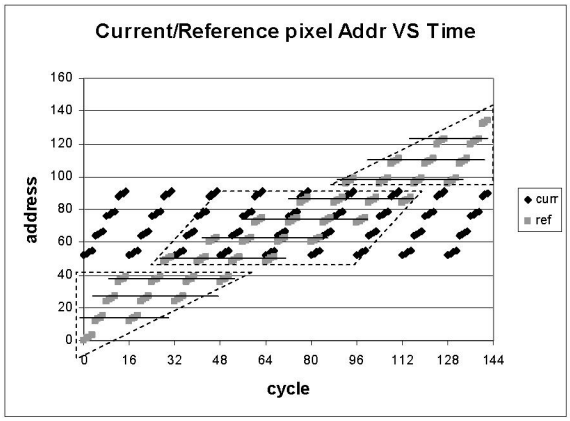

Fig. 2. Access pattern characteristic plot of current frame and reference frame pixel from FBME algorithm. Assume row major linear address mapping. Block coordinate = (1,1), Frame size = 3×3 blocks, Block size = 4×4 pixels, Search range = 4.

size of 16×16 pixel but instead of accumulating the SAD in one register, we store the SAD in 16 register. Each register represent SAD of 4×4 block. The SAD of bigger block size can be found by combining the SAD of smaller block size. Therefore indexing $SAD_{4\times4}(4\lfloor i/4\rfloor+\lfloor j/4\rfloor)$ can be used to place each absolute different in the correct position. We use raster scan order to index 16, 4×4 SAD within one macroblock. Once the 16, 4×4 SAD is know, 41 $SAD_{x\times y}[k]$ can be found using adder tree concept. Once the 41 SAD is know, the compares instruction to evaluate 41 $D_{minx\times y}[h,v,k]$ and 41 $MV_{x\times y}[h,v,k]$ can be done in parallel. Table I shows the value of x,y, and range of k with respect to each value of x and y. This simple modification allows us to obtain 41 SAD and MV when without extra iterations to perform search for each individual block size.

TABLE I. VALUE OF X,Y, AND RANGE OF K

x	y	k
4	4	0-15
4	8	0-7
8	4	0-7
8	8	0-3
8	16	0-1
16	8	0-1
16	16	0

B. Algorithm transformation

FBME is the most accurate motion estimation algorithm and its regularity makes it possible to implement in parallel architecture or parallel programming. However, one major drawback of FBME algorithm, which causes the difficulty of implementing it in SoC platform where the memory I/O is the main bottleneck, is that it is I/O bound. Therefore if we can reduce the amount of memory access, it will tremendously increase the efficiency of this algorithm. In order to be able to perform algorithm transformation to reduce the access redundancy, one must have a clear picture of the data access characteristic of the program. Fortunately, the regularity of the algorithm enables us to obtain a data access characteristic only by looking at one iteration. Fig. 2 shows that there is a reuse opportunity represented by the horizontal solid line of the reference frame pixel within the one dimension search blocks (i.e. within the n loop in figure 1). Therefore, an additional loop is required in order to eliminate the access redundancy of reference pixel within a one dimension consecutive search area. From the plot, it can be seen that there are three different patterns of indexing. The first search block (n = -p), contains 27.7% of the pixels within one search area (assume p = N) and in order to execute other iterations that use these pixels as well, an additional loop in which the index has a dependency on the index i is required. Once the loop on the first search block is completed, all iterations that use the remaining search area pixel can be completed by executing only the iteration i = 3 of the remaining iteration (i.e. n = -p+1 to n = p). The additional loop required for iteration n=-p+1 to n=0 and n=1 to n=p has a loop index that depends on the block size and index n respectively. Note that the y-coordinate of the current frame pixel changes from f(v,i) to f(v,i-k) and the $SAD_{x\times y}$ index dimension increases by one since $SAD_{x\times y}$ of more than one search block are calculated in the same n iteration and thus require extra intermediate storage. Fig. 3 shows the 3 section of the n loop. Note that Compare( ) represent the if statements that is used to compare and find $D_{minx\times y}[h,v,k]$ and $MV_{x\times y}[h,v,k]$. We call this transformation algorithm a level 1 optimization algorithm (O1). After transforming the algorithm, each pixel within the one dimension search area is acquired only once as shown in fig. 4.

The O1 algorithm can achieve 100% data reuse only within one search strip. However, the data access redundancy between two adjacent search strips still exists. In order to achieve 100% data reuse within the whole search area, the access redundancy between two adjacent search strips can be eliminate in the same manner as the elimination of access redundancy between two adjacent search blocks. Three extra loops are added, in the m = -p, m = -p+1 to 0, and m = 1 to p

```
n = -p;
do i = 0 to 15
do j = 0 to 15
do k = 0 to i
Sad1( ) { currx = hN+j;  curry = vN+i-k;
          refx = hN+j+m;  refy = vN+i+n;
          SAD(k,4⌊(i-k)/4⌋+⌊j/4⌋) = SAD(k,4⌊(i-k)/4⌋+⌊j/4⌋) +
                                    |x(currx,curry)-y(refx,refy)|;
enddo k,j,i
Compare( );
i = 15;
do n = -p+1 to 0
do j = 0 to 15
do k = 0 to 15
Sad2( ) { currx = hN+j;  curry = vN+i-k;
          refx = hN+j+m;  refy = vN+i+n;
          SAD(n+p+k,4⌊(i-k)/4⌋+⌊j/4⌋) = SAD(n+p+k,4⌊(i-k)/4⌋+⌊j/4⌋) +
                                        |x(currx,curry)-y(refx,refy)|;
enddo k,j
Compare( );
enddo n
do n = 1 to p
do j = 0 to 15
do k = 0 to p-n
    Sad2( );
enddo k,j
Compare( );
enddo n
```

Fig. 3. Level 1 optimization of FBME algorithm.

loop. The first added loop index depends on j while the second and third added loop index depends on block size and index m respectively. The y and x coordinates of the current frame pixel change from f(v,i) to f(v,i-k) and from f(h,j) to f(h,j-l) respectively. We call this level 2 optimization algorithm (O2). The O2 algorithm can be create the same manner as O1 algorithm and the details of the O2 algorithm is presented in Fig. 5. After transforming the algorithm, each pixel within the search area is acquired only once as shown in Fig. 6.

IV. Result & Discussion

According to the definition in [5], the redundancy access factor, Ra is the ratio of total number of memory accesses in the task to total pixel count in task. When Ra equals to one, the task is said to be one-access and minimum bandwidth

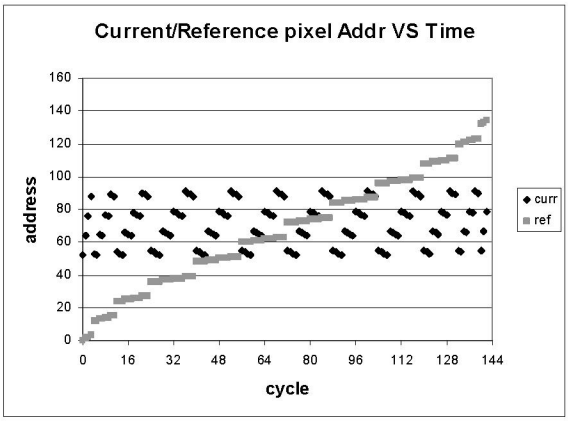

Fig. 4. Access pattern characteristic plot of current frame and refrence frame pixel from O1 FBME algorithm.

usage is achieved. In O1 algorithm, by eliminate the data access redundancy, each pixel in the search strip is accessed only once and the Ra of the task performing VBFBME on a search strip becomes one. By eliminate the data access redundancy in O2 algorithm, each pixel in the search area is accessed only once and the Ra of the task performing VBFBME on a block becomes one. In order to compare the Ra of each algorithm, we consider the task as performing VBFBME on one block. The Ra of a no optimization (O0), level1 optimization (O1), and level2 optimization (O2) algorithm can be expressed as follows.

$$
\begin{aligned}
Ra(O0) &= \frac{N^2(2p+1)^2}{(N+2p)^2} \\
Ra(O1) &= \frac{N(N+2p)(2p+1)}{(N+2p)^2} = \frac{N(2p+1)}{N+2p} \\
Ra(O2) &= \frac{(N+2p)^2}{(N+2p)^2} = 1 \\
&\text{if } p = N, \\
Ra(O0) &= \frac{(2N+1)^2}{9} \\
Ra(O1) &= \frac{2N+1}{3} \\
Ra(O2) &= 1
\end{aligned}
\qquad (6)
$$

By looking at the Ra value, one can see that the bandwidth required by the O1 algorithm is one order of magnitude less than the bandwidth required by the O0 algorithm and the bandwidth required by the O2 algorithm is less than the bandwidth required by the O1 algorithm. Table II shows the traffic generated from acquiring the reference frame pixel for each optimization level. The worst case bandwidth requirement for standard video format is calculated by assuming that the algorithm is on a single hierarchy memory system and the bus width is one byte. We also assume that all data is available in the memory. The bandwidth requirement is equal to RN_vN_hf where R is the number of reference frames read per block which equals to numerator of the first three equations in equation 6, NvNh is total number of block within one frame, and f is the frame rate. It can be seen from table II that O1 and O2 algorithm reduce the bandwidth utilization approximately by the factor of 11 and 121 respectively.

TABLE II. Bandwidth with respect to frame size

Format	None	O1	O2
	BW(Mbyte)	*BW(Mbyte)*	*BW(Mbyte)*
QCIF	827	75.2	6.84
CIF	3310	301	27.3
NTSC	11300	1030	93.2
HDTV	67700	6160	560

Eliminating data access redundancy not only reduces the memory bandwidth required, it also reduces the frequency of address and data bus transitions, which in turn reduces power consumption. Table III shows the total number of address bus transitions in number of bits (reference frame and current frame) when performing motion estimation on one HDTV

(1920×1080) and QCIF frame.

TABLE III. TOTAL ADDRESS BUS TRANSITION (BITS) PER FRAME

Format	None	O1		O2	
	Trans.	*Trans.*	*savings*	*Trans.*	*savings*
QCIF	1.14×10^8	6.45×10^7	43.59%	6.17×10^7	46.06%
CIF	4.77×10^8	2.60×10^8	45.52%	2.47×10^8	48.24%

V. CONCLUSION

In this paper, we are able to modify and enhance FBME such that when perform FBME on one current block, 41 MV and SAD can be found without increasing number of accumulate and absolute difference instruction and each reference pixel is accessed only once. We have focused on software techniques to improve locality of data accesses because this technique can optimize both memory performance and power consumption which is the goal of memory optimization for SoC. Improving the data locality results in a reduction of the effective cost of a memory access, a reduction of memory bandwidth requirement and is also beneficial in term of energy. We eliminate the access redundancy of the data acquired by the VBFBME algorithm with a customized algorithm transformation. We transform the algorithm by splitting up the iteration, which does not induce a complicated index transformation as other techniques that involve loop folding to reduce nested loop levels ([5], [6]) or use the block algorithm [7]. By eliminate the data access redundancy to the reference frame pixel; the number of pixels read per block was reduced substantially, thereby reducing the memory bandwidth requirement and power consumption.

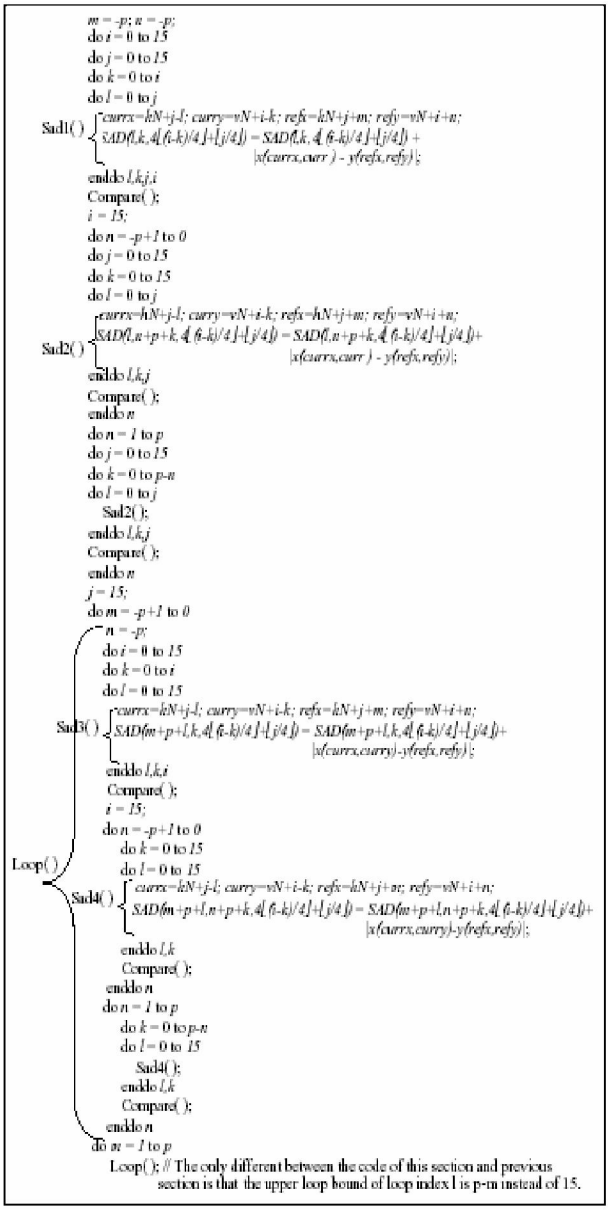

Fig. 5. Level 2 optimization of FBME algorithm.

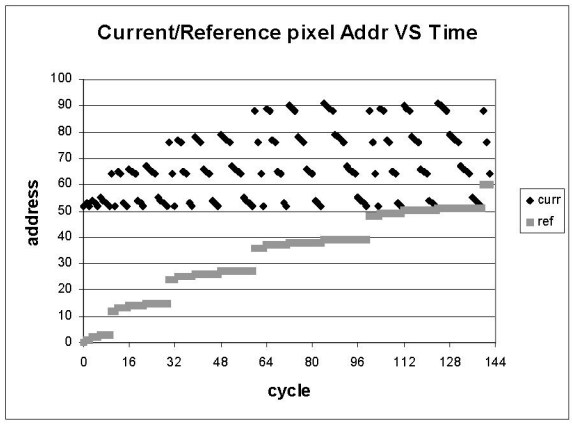

Fig. 6. Access pattern characteristic plot of current frame and reference frame pixel from O2 FBME algorithm.

REFERENCES

[1] Zhi Zhou, Ming-Ting Sun, and Yuh-Feng Hsu, "Fast variable block-size motion estimation algorithms based on merge and split procedures for H.264/MPEG-4 AVC," *Int Symposium on Circuits and systems,* Vol3, p. III-725-8, May 2004.

[2] Yu-Kuang Tu, Jar-Ferr Yang, Yi-Nung Shen, and Ming-Ting Sun, " Fast variable-size block motion estimation using merging procedure with an adaptive threshold," *Int Conf. on Multimedia and Expo,* Vol2, p. II-789-92, July 2003.

[3] Yu-Wen Huang, Tu-Chih Wang, Bing-Yu Hsieh, Liang-Gee Chen, "Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264," *Int Symposium on Circuits and systems,* Vol2, p. II-796-9, May 2003.

[4] Anne Pratoomtong, Yu Hen Hu, "On-chip cache algorithm design for multimedia SoC," *Accepted in IEEE Int. Conf. On Acoustics, Speech, and Signal Processing*, May2005.

[5] J.-C. Tuan, T.-S Chang, C.-W. Jen, "On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture," *IEEE Trans. Circuits and Systems for Video Technology*, pp. 61–72, Jan. 2002.

[6] S. Kittitornkun, Y.H. Hu, "Frame-Level Pipelined Motion Estimation Array Processor," *IEEE Trans. Circuits and Systemsfor Video Technology*, pp. 248-251, Feb. 2001.

[7] Monica S. Lam, Edward E. Rothberg, Michael E. Wolf," *The Cache Performance and Optimizations of Blocked Algorithms*," Procs pf 4th int. conf. on Architectural support for programming languages and operating systems, pp 63-74, April 1991.