ALGORITHM TRANSFORMATION TO IMPROVE DATA LOCALITY FOR MULTIMEDIA SOC

Anne Pratoomtong, Yu Hen Hu

University of Wisconsin-Madison, Madison, WI 53706

ABSTRACT

This paper is based on our previous work in [1]. In this paper we propose a more systematic approach to improve data locality of multimedia algorithm that contain nested loop while still preserve the nested loop program semantic by algorithm transformation. We introduce procedure to evaluate the reuse vector of the data access. The reuse vectors represent the input dependencies of the program in the form of index vector which reveal the information of which loop carrying reuse and thus are used as a guide to select the loops subject to transformation. We use full search block matching motion estimation (FBME) algorithm as a case study because this algorithm has 6 level nested loop and thus is a good showcase of our method.

Index Terms— Algorithm transformation, Full-search block matching motion estimation, scheduling vector, reuse vector, unimodular transformation

1. INTRODUCTION

The increasing demand in portable electronic especially in the area of real time multimedia drives the system designer to reduce the overall system power consumption, size and increase system functionality by combining the functionality into Systems-On-chip (SoC). Since they combine many functionality on chip under area and power constraints, there are very limited chip area left for on chip memory system. In fact, majority of multimedia SoC available on market today have less than 64 KB of on chip memory.

Motion estimation algorithm is one of the core algorithms in video coding and it is consider being one of the most data intensive algorithms in multimedia application. Full search block matching motion estimation (FBME) algorithm produce high accuracy motion vector. However, the accuracy come with the price of high computation and data transfer overhead which will emphasized the memory I/O bottleneck problem in SoC. Fortunately, the data access pattern of the algorithm is highly regular and deterministic which provide the opportunity for data locality optimize and enhance memory performance**.**

2. REUSE VECTOR SPACE

The computation of a nested loop with finite bounds of indices can be represented in an iteration index space. This representation is commonly used in parallel compilers and has also been adopted in designing systolic arrays [2]. In this representation, each integer coordinate corresponds to a particular set of loop indices and therefore represents a particular iteration. A regular iterative nested loop whose loop bounds are not functions of data variables then can be represented as a polytope in the iteration space.

Let us denote an index vector $\vec{i} = [i_1, i_2, ..., i_K]^T$ where i_k is the loop index of the k^{th} loop counting from the innermost loop. We denote its bounds as $L_k \le i_k \le U_k$, $1 \le k \le K$. In the FBMA algorithm, we have

$$\vec{i}\,' = \begin{bmatrix} j & i & n & m & h & v \end{bmatrix}$$

We can represent the relative data access time, $T(\vec{i})$ of the computation in each iteration in the index space as a function of loop bound as follows.

$$T(\vec{i}) = \vec{v}'\vec{i} + c$$

$$where \quad \vec{v}' = [v_1 \; v_{2 \ldots} v_K]$$

$$v_j = \prod_{q=1}^{j-1} (U_q - L_q + 1)$$

$$and \; v_1 = 1$$

$$c = \sum_{j=1}^{K} (-L_j v_j)$$

For the FBMA algorithm in figure 1 we have

$$U = \begin{bmatrix} N-1 & N-1 & p & p & N_h - 1 & N_v - 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 0 & -p & -p & 0 & 0 \end{bmatrix}$$

$$\vec{v}' = \begin{bmatrix} 1 & N & N^2 & N^2(2p+1) & N^2(2p+1)^2 & N^2(2p+1)^2 N_h \end{bmatrix}$$

$$c = pN^2 + pN^2(2p+1) = pN^2(2p+2)$$

Note that $\vec{v}$ is the nested loop sequential scheduling vector which assign a specific execution order of each index point in the nested loop, mapping each index point in the index space into a positive integer, namely, $\vec{v}'(\vec{i}) \in I^+$.

```
Do v = 0 to Nv-1
  Do h = 0 to Nh-1
    MV(h,v) = (0,0);
    Dmim(h,v) = ∞;
    Do m = -p to p
      Do n = -p to p
        MAD(m,n) = 0;
        Do i = 0 to N-1
          Do j = 0 to N-1
            currx = hN+j;  curry = vN+i;
            refx = hN+j+n;  refy = vN+i+m;
            MAD(m,n) = MAD(m,n) + |x(currx,curry)-y(refx,refy)|;
        Enddo j,i
        If Dmim(h,v) > MAD(m,n)
            Dmim(h,v) = MAD(m,n);
            MV(h,v) = (m,n);
        Endif
Enddo n,m,h,v
```

Figure1. Six-level nested Do-loop FBME algorithm

From [3], let $\vec{f}(\vec{i}) = H\vec{i} + \vec{c}_f$ be an indexing function of an array A, two iterations, $\vec{i}_1$ and $\vec{i}_2$, reference to the same data in array $A[\vec{f}(\vec{i})]$ when $H\vec{i}_1 + c = H\vec{i}_2 + c$, that is, when $H(\vec{i}_1 - \vec{i}_2) = H\vec{r} = \vec{0}$. Therefore, the self-temporal reuse vector space of a reference $A[\vec{f}(\vec{i})]$, R_{ST}, equal to null space of H, null(H), or reuse vector space. According to [4], let n be dimension of null space of H, there is exactly n vectors in basis set of null(H) but there are an infinite number of such sets that qualify as a basis. Therefore, multimedia application involve accessing a two dimensional data in a K-level nested loop where K is greater than three will have a infinite set of reuse vector space since the dimension of null(H) will be K-2 which is greater than one. Therefore, definition one list criteria of selecting basis vector of null(H), $\vec{e}_i$.

Definition 1: In order to find a candidate basis vector of null(H), $\vec{e}_i$, that represent the reuse direction, it is necessary that $\vec{e}_i$ satisfy the following conditions.

1. $H \cdot \vec{e}_i = 0$
2. $\vec{e}_i$ is a vector with length equals to K.
3. Every element in $\vec{e}_i$ has zero or integer value.
4. Let $\vec{e}_{ij}$ represent element in j-th row of vector $\vec{e}_i$, 1<j<K.

$$\min(\Delta i_j) \le \left|\vec{e}_{ij}\right| \le \max(\Delta i_j) \Rightarrow 0 \le \left|\vec{e}_{ij}\right| \le U_j - L_j .$$

Although there is infinite number of $\vec{e}_i$ that satisfies the first condition in definition 1, the solution space is limited by condition 2-4. The second condition states that the length of vector equals to the number of level of nested loop. $\vec{e}_i$ represents the difference between two iteration so each element of $\vec{e}_i$ equals to the difference of two indexes which has integer value. Third condition implies this. The forth conditions represent the bound of each element in $\vec{e}_i$. Since the value of each element in $\vec{e}_i$ equals to the difference between two bounded indexes, it is also bounded. The lower bound equals to minimum distance between the two indexes which occurs when two indexes are the same and the upper bound equals to the maximum distance between the two indexes which occurs when one index is at the upper loop bound and the other index is at lower loop bound.

Follow the following steps to find reuse vector $\vec{e}_i$.

Step 1: Define E as a matrix consist of (K-H_m) vectors with length equals to K where K equals to number of nested loop and H_m equals to number of row of matrix H. Set diagonal elements of E to one and elements above diagonal to zero. Elements below diagonal are unknown.

Step 2: Find the unknown element below diagonal using condition one in definition two. Set unknowns to zero whenever possible. And if decision of which unknown to be set to zero need to be made, set the one located in the higher row to zero.

Step 3: After all the elements in E matrix are found, multiply the column which has non-integer element with constant to make them become integer. This is to ensure that condition 3 in definition one is met.

Step 4: Check each element of $\vec{e}_i$ and make sure that they are bounded using condition four in definition two. Only $\vec{e}_i$ that satisfy condition five is valid.

FBME algorithm is shown in figure 1. The data input come from two arrays which stored pixel value of two video frame, current frame (x) and reference frame (y). The indexing function of array x , $\vec{f}_X(\vec{i})$ and indexing function of array y , $\vec{f}_Y(\vec{i})$ can be written in the format describe in [9] as follows.

$$\vec{f}_X(\vec{i}) = H_x\vec{i} + c$$

$$\vec{f}_Y(\vec{i}) = H_Y\vec{i} + c$$

$$Where\ H_X = \begin{bmatrix} 1 & 0 & 0 & 0 & N & 0 \\ 0 & 1 & 0 & 0 & 0 & N \end{bmatrix}$$

$$H_Y = \begin{bmatrix} 1 & 0 & 1 & 0 & N & 0 \\ 0 & 1 & 0 & 1 & 0 & N \end{bmatrix}$$

$$\vec{i} = \begin{bmatrix} j & i & n & m & h & v \end{bmatrix}^T$$

$$c = 0$$

We use the four steps describe above to derive the reuse vector, $\vec{e}_i$ of reference and current frame as shown in figure 2.

3. ACCESS TEMPORAL LOCALITY PERIOD(ATLP)

ATLP represent the time duration between two iterations that use the same data and it can be represented as a dot product of scheduling vector and reuse vector.

$$Step\,1\,\&\,2:\ E_X=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\-\frac{1}{N}&0&0&0\\0&-\frac{1}{N}&0&0\end{bmatrix},\ E_Y=\begin{bmatrix}-1&0&0&0\\0&-1&0&0\\1&0&-1&0\\0&1&0&-1\\0&0&\frac{1}{N}&0\\0&0&0&\frac{1}{N}\end{bmatrix}$$

$$Step\,3:\ E_X=\begin{bmatrix}-N&0&0&0\\0&-N&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{bmatrix},\ E_Y=\begin{bmatrix}-1&0&0&0\\0&-1&0&0\\1&0&-N&0\\0&1&0&-N\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$Step\,4:\quad E_X=[\vec{e}_{X1},\vec{e}_{X2}],\ E_Y=[\vec{e}_{Y1},\vec{e}_{Y2},\vec{e}_{Y3},\vec{e}_{Y4}]$$

$$E_X=\begin{bmatrix}0&0\\0&0\\1&0\\0&1\\0&0\\0&0\end{bmatrix},\ E_Y=\begin{bmatrix}-1&0&0&0\\0&-1&0&0\\1&0&-N&0\\0&1&0&-N\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

Figure2. Reuse vector of current and reference frame access in FBME algorithm

Since there are two and four basis vectors in the current frame and reference frame access reuse vector space respectively, there are two and four ATLPs of current and reference frame access respectively. The ATLPs of current frame (x) and reference frame (y) access of the program in figure 1 can be derive as dot product of nested loop sequential scheduling vector and basis vector of current frame and reference frame access reuse vector space as follows.

$$ATLP_{X1}=\vec{v}'\vec{e}_{X1}=N^2$$
$$ATLP_{X2}=\vec{v}'\vec{e}_{X2}=N^2(2p+1)$$
$$ATLP_{Y1}=\vec{v}'\vec{e}_{Y1}=N^2-1$$
$$ATLP_{Y2}=\vec{v}'\vec{e}_{Y2}=N^2(2p+1)-N$$
$$ATLP_{Y3}=\vec{v}'\vec{e}_{Y3}=N^2(2p+1)^2-N^3$$
$$ATLP_{Y4}=\vec{v}'\vec{e}_{Y4}=N^2(2p+1)^2N_h-N^3(2p+1)$$

Current frame pixels are reused every time the search area changes, therefore there are two ATPLs for the current frame pixel. Accesses to reference frame pixels are more complicated since the boundary of the search area overlaps among many different iterations and as a result there are four ATLPs for the reference frame.

4. ALGORITHM TRANSFORMATION

The pixel in the current frame is used without overlapping. Therefore, as long as the cache is big enough to store one current block, the temporal locality of the current block is fully exploited. On the other hand, it is difficult to fully

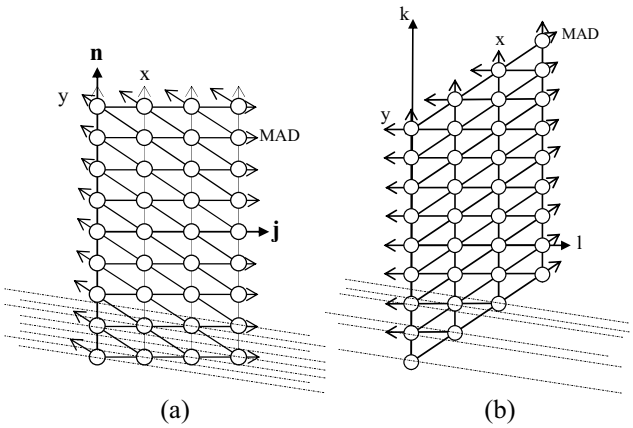

Figure3. (a) Index space of [n,j] loop of FBME algorithm for N = p = 4. (b) Transformed index space.

exploit the temporal locality of the pixel in a search frame since the pixel is overlapped when used in each search block. Therefore, our goal is to improve the temporal locality of the pixel in a search frame by reducing the $ATLP_Y$. We will reduce the $ATLP_{Y1}$ which is the smallest one. $ATLP_{Y2}$ can also be reduced in a similar manner as reducing $ATLP_{Y1}$. We will not reduced $ATLP_{Y3}$ and $ATLP_{Y4}$ because they are fairly large since they involve the two most outer loops carrying reuse and therefore is not cost effective to reduce since it will create a lot of intermediate storage.

In $ATLP_{Y1}$, loop n and j carrying reuse. Therefore, to reduce the complexity of the algorithm transformation procedure, we first consider only the two loops, n and j which represent one dimension motion estimation. The data locality behavior of the one and two dimension motion estimation is similar and thus the result of the algorithm transformation of the one dimension motion estimation can be extended to the algorithm transformation of the two dimension motion estimation. Figure 3a shows the index space of loop n and j with block size (N) equals to search range (p) equals to four along with the hyper plan of nested loop sequential scheduling vector. The indexes along [n j] = [0 1] direction execute in consecutive iteration. To preserve the nested loop program semantic, we need to find a unimodular transformation that change the reference frame access reuse vector from [1 -1]′ to [0 -1]′ or [0 1]′. This will guarantee that the indexes that use the same reference pixel, which is the node along direction [1 -1]′, are executed in consecutive iteration with nested loop sequential scheduling vector. The unimodular matrix $U=\begin{bmatrix}1&1\\0&1\end{bmatrix}$ change the reference frame access reuse vector from [1 -1]′ to [0 -1]′. The unimodular transform index [n j]′ to index [k l]′ where k = n+j and l = j. Figure3b shows the new index space results from unimodular transformation along with the same hyperplan shows in figure3a. Figure4 shows only the four

```
Do m = -p to p
  Do i = 0 to N-1
    Do k = -p  to p+N-1
       Do l = max(0,k-p) to min(N-1,k+p)
          currx = hN+l;  curry = vN+i;
          refx = hN+k;  refy = vN+i+m;
          MAD(m,k+p-l) = MAD(m,k+p-l) +
                  |x(currx,curry)-y(refx,refy)|;
Enddo l,k,m,i
```

(a)

```
Do r = -p  to p+N-1
  Do k = -p  to p+N-1
    Do s = max(0,r-p) to min(N-1,r+p)
      Do l = max(0,k-p) to min(N-1,k+p)
         currx = hN+l;  curry = vN+s;
         refx = hN+k;  refy = vN+r;
         MAD(r+p-s,k+p-l) = MAD(r+p-s,k+p-l) +
                  |x(currx,curry)-y(refx,refy)|;
Enddo l,s,,k,r
```

(b)

Figure4. (a) O1 FBME algorithm (b) O2 FBME algorithm

most inner loop of the transformed FBME algorithm. The other part of the transformed algorithm is the same as the original algorithm shows in figure1 and thus is omitted due to limited space. In figure4a, loop k and l is created by applying unimodular transformation to loop n and j in figure1 and results in reducing $ATLP_{Y1}$ to one. After transforming the algorithm, each pixel within the one dimension search area is acquired only once. We call this new algorithm level 1 optimization algorithm (O1). The O1 algorithm can achieve 100% data reuse only within one search strip (n loop). However, the data access redundancy between two adjacent search strips (m loop) still exists. In order to achieve 100% data reuse within the whole search area, the m and i loop need to be transformed in the same manner as n and j loop. The unimodular matrix that use to transformed index n and j is used to transform the index i and m result in loop r and s shown in figure4b. The transformed algorithm in figure4b achieves 100% data reused within the search area (O2 algorithm). That is each pixel within the search area is acquired only once.

5. PERFORMANCE

Equation 1 shows the redundancy access factor, Ra[5] of a no optimization (O0), O1, O2 algorithm and equation two shows ratio of number of reference frame pixel acquired when perform FBME on one block in FBME optimized algorithm (L1, L2) and origin algorithm (L0). It can be conclude from equation one that both O1 and O2 algorithm has smaller Ra than the original algorithm which means that the algorithm transformation result in reducing the reference frame redundancy data access. In fact O2 achieve the Ra of one which means that there is no reference frame redundancy data access when performing FBME on one block. In equation 2, the percentage of number of reference pixel acquired in O1 and O2 algorithm with respect to number of reference frame pixel acquired in the original algorithm decrease substantially especially for larger block. Consider when block size equal to 16 or 32; O1 and O2 algorithm acquires almost 95% and 100% less data than the original algorithm respectively.

$$Ra(O0) = \frac{N^2(2p+1)^2}{(N+2p)^2}$$

$$Ra(O1) = \frac{N(N+2p)(2p+1)}{(N+2p)^2} = \frac{N(2p+1)}{N+2p} \qquad (1)$$

$$Ra(O2) = \frac{(N+2p)^2}{(N+2p)^2} = 1$$

$$\frac{L1}{L0} = \frac{N(N+2p)(2p+1)N_vN_h - 1}{N^2(2p+1)^2N_vN_h - 1} \approx \frac{N+2p}{N(2p+1)} \qquad (2)$$

$$\frac{L2}{L0} = \frac{(N+2p)^2N_vN_h - 1}{N^2(2p+1)^2N_vN_h - 1} \approx \left(\frac{N+2p}{N(2p+1)}\right)^2$$

6. CONCLUSION

In this paper, we are able to increase temporal locality of the reference frame data access by reducing the ATLP induced by the two and the four most inner loops of FBME algorithm to one. Usually when ATLP is large, there is a high possibility that the data get replaced before it is used again and thus induce redundancy bandwidth to reload the data. Reducing ATLP to one guarantee that the data will remains on-chip when it is use again since ATLP equals one means that the data is acquire in consecutive iteration. As a result, it eliminate the redundancy access induce in those loop subject to transformation, thereby reduce memory bandwidth utilization. This also reduces power consumption, mainly due to a reduction of the on chip bus usage since data is still available in the pipeline. By applying unimodular transformation to the loop index subject to ATLP reduction, the program schematic does not change. Our method does not induce a complicated index transformation or increase the depth of loop nest as other techniques that involve loop folding to reduce nested loop levels [5] or loop tiling [3].

7. REFERENCES

[1] S. Pratoomtong, Y. H. Hu, "On-Chip Cache Algorithm Design for Multimedia SOC," *IEEE int. Conf. Acoustics, Speech, and Signal Processing*, pp. 337-340, Mar. 2005.

[2] S.Y.Kung, *VLSI Array Processors,* Prentice Hall, 1988.

[3] Michael E. Wolf, Monica S. Lam, "A Data Locality Optimizing Algorithm," *ACM SIGPLAN,* ACM Press, pp. 442-459, Jun. 1991.

[4] John S. Bay, *Fundamentals of linear state space systems,* McGraw-Hill, 1998.

[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.