A NOVEL MODULAR SYSTOLIC ARRAY ARCHITECTURE FOR FULL-SEARCH BLOCK MATCHING MOTION ESTIMATION

Hangu Yeo and Yu Hen Hu

Department of Electrical and Computer Engineering
University of Wisconsin
Madison, WI 53706

ABSTRACT

In this paper, we propose a modular systolic array architecture for the full-search block matching motion estimation algorithm(FBMA). With this novel architecture, we are able to generate a motion vector for every reference block in raster scan order while achieving 100% processor utilization and high throughput rate. Furthermore, we devised a scheme to save the pin count (I/O) by sharing memory units. This results in low memory bandwidth. This architecture is scalable in that it can easily be adapted to handle larger search ranges and different block sizes without increasing the effective latency.

1. INTRODUCTION

A large number of image transmission and storage applications such as digital storage media, video broadcast, HDTV, etc. contain images of moving objects. In the interest of reducing storage and transmission cost, it is important to exploit the temporal redundancy in successive frames by interframe video coding via motion estimation. Among the existing motion estimation algorithms FBMA is very reasonable one because of its regularity and simplicity for hardware implementation. The matching criterion of FBMA with block size $N \times N$ and search range of $\pm p$ is

$$MAD(m,n) = \sum_{i=1}^{N}\sum_{j=1}^{N} |x(i,j) - y(i+m, j+n)| \qquad (1)$$

where $-p \le m,n \le p$. $x(i,j)$ is the luminance value of the current frame and $y(i+m, j+n)$ is the luminance value of the previous frame. The corresponding coordinate of the minimum MAD value(motion vector) is determined by (2).

$$MV = arg\{minMAD(m,n)\} \quad -p \le m,n \le p \qquad (2)$$

From the matching criterion (1), for the video broadcast format (576×720 frame size, 16×16 block size, $p = \pm 8$, and 25-Hz frame rate), 8,989 million computations are required to be completed within one second. Thus the FBMA is computationally expensive, and a single processor cannot afford the real time requirement. Recently, great efforts have been made to overcome this computational bottleneck by using a systolic array architecture [1]-[5].These existing works still suffer from several serious difficulties: 1) high pin count, 2) high memory bandwidth, 3) high number of PE's on a single chip when the block size is large, 4) they don't satisfy the real time requirement for larger search ranges.

In this paper we propose a modular systolic array architecture for FBMA with a specific search range of $-\frac{N}{2}$ to $+\frac{N}{2}-1$ with the following advantages:

1. **Low pin count:** The input pin count is saved with a serial data input for the reference block data, and the reduced number of input ports for the search area data by sharing memory units.
2. **Low memory bandwidth:** By utilizing the input data dependency with a specific search range, multiple access of the search area data is avoided.
3. **High-throughput:** MSSM [6] enables overlapped execution between two successive stages, and the processors are utilized 100%.
4. **Scalable architecture:** The proposed architecture is scalable in that it can be adapted to handle block matching algorithms with different block sizes and search ranges.

2. SYSTOLIC IMPLEMENTATION

2.1. Multiple block matching architecture

The actual motion estimation algorithm requires to generate a motion vector for every reference block in raster scan order. From the matching criterion (1), the computation for choosing the best matching candidate block can be expressed as the following six-level nested *do* loops:

```
do v = 1 to N_v
  do h = 1 to N_h
    MV(h,v) = (0,0)
    D_min(h,v) = ∞
    do m = -p to p
      do n = -p to p
        MAD(m,n) = 0
        do i = (h-1)N + 1 to hN
          do j = (v-1)N + 1 to vN
            MAD(m,n) = MAD(m,n)
            +|x(i,j) - y(i+m, j+n)|
        enddo i, j
        if D_min(h,v) > MAD(m,n)
          D_min(h,v) = MAD(m,n)
          MV(h,v) = (m,n)
        endif
```

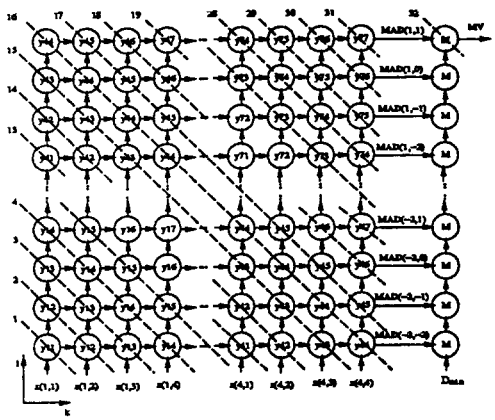

Figure 1: Two-dimensional localized DG with a new index space (k, l). ($x(i, j)$ and $y(i+m, j+n)$ are included instead of $x_s(b, l, k)$ and $y_s(b, l, k)$ for convenience)

enddo m, n, h, v

Note that the two-dimensional indices pairs h, v, or m, n, or i, j are rather symmetric. The execution order thus will be determined mainly by the data input sequence. In most video processing systems, image pixels are obtained block by block in raster scan order. With the *block scan mode* input order, the two-dimensional reference block data will be loaded column by column sequentially [7]. The observation on the image input ordering motivated us to consider to combine each of these index pairs into a composit index. In particular, $b = (v-1)N_h + h$, $l = (2p+1)(m+p) + n + (p+1)$, $k = N_h(\lceil \frac{b}{N_h} \rceil - 1)N^2 + (i - \lceil \frac{b}{N_h} \rceil)N + j$. $\lceil x \rceil$ is the smallest integer which is greater than or equal to the real number x. Then the six-level nested *do* loops can be reduced to a localized three-level nested *do* loops:

do $b = 1$ to $N_h N_v$
  $MV(b, 1) = 0$
  $D_{min}(b, 1) = \infty$
  do $l = 1$ to $(2p+1)^2$
    $MAD(b, l, 1) = 0$
    do $k = (b-1)N^2 + 1$ to bN^2
      $x_s(b, l+1, k) = x_s(b, l, k)$
      $MAD(b, l, k+1) = MAD(b, l, k) + |x_s(b, l, k) - y_s(b, l, k)|$
    enddo k
    if $D_{min}(b, l) > MAD(b, l, bN^2 + 1)$
      $D_{min}(b, l+1) = MAD(b, l, bN^2 + 1)$
      $MV(b, l+1) = l$
    endif
enddo l, b

Two-dimensional localized *dependence graph*(DG) and *signal flow graph*(SFG) for a single referene block with $N = 4$ is presented in Fig 1 and Fig 2. The search range becomes -2 to $+1$. The number dedicated to the dashed lines in the DG denotes clock cycle. The SFG is obtained by projecting the DG along a *projection vector* $\vec{d}^T = [1\ 0]$ with a

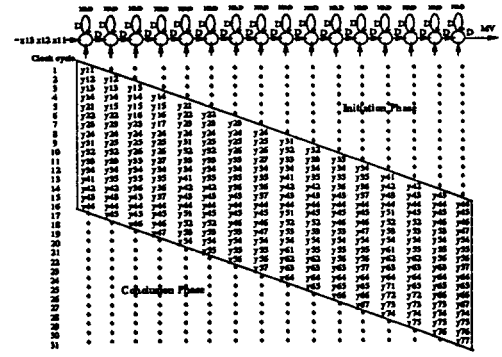

Figure 2: A SFG projected along $[1\ 0]^T$

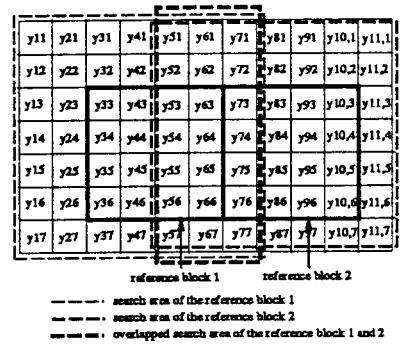

Figure 3: The search areas of the two neighboring reference blocks.

linear *scheduling vector* $\vec{s}^T = [1\ 1]$. The *pipelining period* is $\alpha = \vec{s}^T \vec{d} = 1$. Note that α is independent of block size N. The *latency*(L) for estimating a motion vector becomes $L = \{max_{\vec{p},\vec{q} \in D}\{\vec{s}^T(\vec{p} - \vec{q})\} + 1\} + 1 = 2N^2$ clock cycles. $\vec{p}$ and $\vec{q}$ are any two points in the DG index space and D is the index set of the nodes in the DG. Additional clock cycle has been added for choosing the minimum MAD value.

Two neighboring reference blocks with their corresponding search areas are shown in Fig 3. The existing architectures [1]-[5] don't fully utilize the *initiation* and *conclusion phases* of computation(Fig 2), which leads to low processor utilization as well as excessive memory access. These problems can be solved by a techique called *chaining*, which enables overlapped motion estimation of the two neighboring reference blocks. MSSM [6] makes all the processors activated during the *initiation* and *conclusion phases* using the overlapped search area data. At each stage, a motion vector for the corresponding reference block is estimated using MAD as an *I/O variable* between the two successive stages. The three conditions of I/O matching(structure matching, location matching, and data flow matching) [6] are easily satisfied with *processor allocation function* $P_i = [0\ 1]$, $\vec{d}^T = [1\ 0]$, and $\vec{s}^T = [1\ 1]$.

In our proposed systolic architecture, we will initiate the processing of block 2 as soon as the processors used to process block 1 have completed their task (Fig 4). Hence no clock cycles are wasted and 100% processor utilization is accomplished. This leads to an initiation interval of N^2 clock

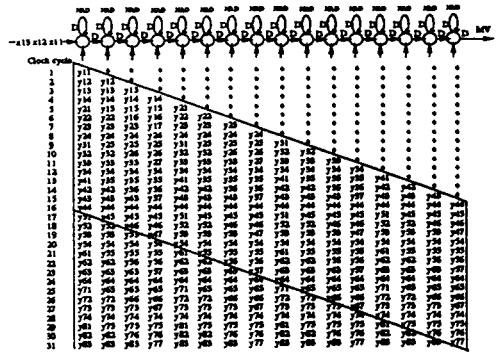

Figure 4: I/O matched array design for two successive stages.

cycles per block rather than $2N^2$ clock cycles per block. Moreover, by forwarding the data to the processors at appropriate time, the pixels in the overlapped region are used by both block 1 and block 2 without redundant memory access. As such, the low memory bandwidth requirement is achieved as well.

The *latency* of each stage (L) without chaining is $2N^2$ clock cycles as explained before. Because of the computation overlap between two successive stages, the *effective latency* of each stage becomes $L_{effective} = L - \tau_{overlapped}$, where $\tau_{overlapped} = max_{p,q \in P}(p - q) + 1$ is the overlapped execution period between the two successive stages. p and q are any two points in the processor index space P. The effective latency is 50% less than that without chaining, and our proposed architecture generates motion vectors every N^2 clock cycles.

2.2. Systolic array with reduced pin count

In the original six-level nested *do* loops, the search area data is expressed in six-dimensional index space $y(h, v, m, n, i, j)$ with the following three properties. (4) holds for the search area data in the overlapped region.

$$y(h, v, m, n, i, j) = y(h, v, m + 1, n, i - 1, j) \quad (3)$$

$$y(h, v, m, n, N, j) = y(h + 1, v, m - 1, n, 1, j) \quad (4)$$

$$y(h, v, m, n, i, j) = y(h, v, m, n + 1, i, j - 1) \quad (5)$$

Assuming that the block size is $N \times N$ with the search range of $-\frac{N}{2}$ to $+\frac{N}{2} - 1$, at a certain time instance, every N^{th} processor has the same input data by (3) and (4), and the pin count can be saved by a factor of N. Furthermore, the number of input pin count for the search area data is reduced to two and the memory unit can be shared by (5). For example, for the block size of 4×4, the four input data becomes:

1. $m = n = -2$, $j = 1$: $y(h, v, m, n, i, j)$, $y(h, v, m, n + 1, i-1, j+3) = y(h, v, m, n+2, i-1, j+2) = y(h, v, m, n + 3, i - 1, j + 1)$ by (4) and (5).
2. $m = n = -2$, $j = 2$: $y(h, v, m, n, i, j) = y(h, v, m, n + 1, i, j - 1)$, $y(h, v, m, n+2, i-1, j+2) = y(h, v, m, n + 3, i - 1, j + 1)$ by (4) and (5).

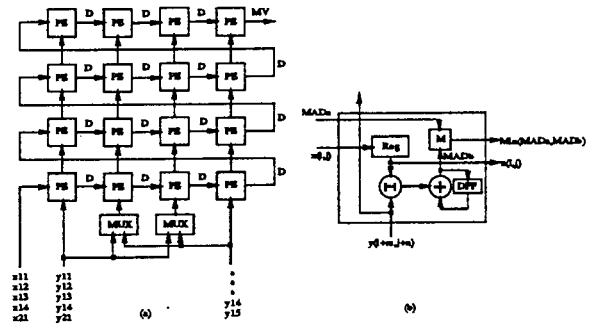

Figure 5: (a)The proposed array processor with reduced pin count.(b)The detailed PE.

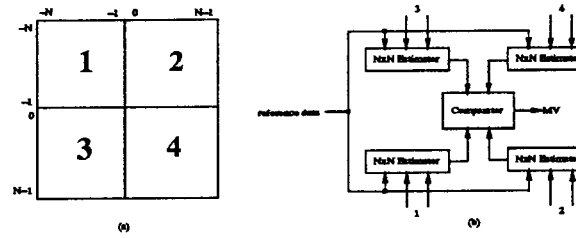

Figure 6: (a) $4N^2$ search points with search range of $-N$ to $+N - 1$. (b) An architecture for the search range of $-N$ to $+N - 1$.

3. $m = n = -2$, $j = 3$: $y(h, v, m, n, i, j) = y(h, v, m, n + 1, i, j - 1) = y(h, v, m, n + 2, i, j - 2)$, $y(h, v, m, n + 3, i - 1, j + 1)$ by (4) and (5).
4. $m = n = -2$, $j = 4$: $y(h, v, m, n, i, j) = y(h, v, m, n + 1, i, j - 1) = y(h, v, m, n + 2, i, j - 2) = y(h, v, m, n + 3, i, j - 3)$ by (4) and (5).

The resulting array processor with reduced pin count and the detailed PE is presented in Fig 5(a) and (b).

2.3. Design for larger search ranges and various block sizes

If the search range is doubled so that the search range is enlarged to $-N$ to $+N - 1$, the $N \times N$ motion estimator proposed above can easily be adapted to handle such case. In particular, we partition the search points into four 2×2 subregions and designate an $N \times N$ motion estimator to handle each subregion (Fig 6). These four processor arrays will work in parallel to estimate the motion vector within each region. The results then will be forwarded to a comparator to pick the motion vector which minimizes the four candidate MAD values. The motion vector for each reference block is generated every N^2 clock cycles.

The block size of $2N \times 2N$ with search range of $-N$ to $+N - 1$ can be easily handled by cascading four $N \times N$ motion estimators (Fig 7). The reference block data are serially delivered through the motion estimators while the search area data are delivered in parallel. The results are forwarded to the comparator, and the motion vector corresponding to the minimum MAD value is chosen.

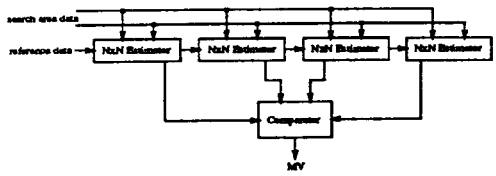

Figure 7: An architecture for the block size of $2N \times 2N$ with search range of $-N$ to $+N-1$.

It generates motion vectors every $(2N)^2$ clocks cycles. However, the latency for estimating one frame is still same, because the number of blocks in a frame is $\frac{1}{4}$ compared to the block size of $N \times N$. As such, the block size of $N \times N$ can be handled by cascading four $\frac{N}{2} \times \frac{N}{2}$ motion estimators. Thus, for the video broadcast format with block size of 16×16, we can avoid implementing 16^2 PE's on a single chip by cascading sixteen 4×4 motion estimators.

3. COMPARISON

Type	Input pin count	# Clock cycles /block	/frame	f_{max}
Komarek *et al*	16	544	881,280	45
Heieh & Lin	2	1,030	1,668,600	23
Jehng *et al*	16^2	289	468,180	85
Yang *et al*	3	4,096	6,635,520	6
Chan *et al*	16	816	1,321,920	30
Yeo and Hu	3	256	423,936	94

Table 1. Block size: 16×16, search range: ± 8.

Type	Input pin count	# Clock cycles /block	/frame	f_{max}
Komarek *et al*	16	1,584	2,566,080	15
Heieh & Lin	2	2,310	3,742,200	10
Jehng *et al*	16^2	1,089	1,764,180	22
Yang *et al*	6	4,096	6,635,520	6
Chan *et al*	16	1,856	3,006,720	13
Yeo and Hu	9	256	423,936	94

Table 2. Block size: 16×16, search range: ± 16.

Type	# Data accesses /block	Total # data accesses/sec
Komarek *et al*	8,960	362,880,000
Heieh & Lin	1,280	51,840,000
Jehng *et al*	74,240	3,006,720,000
Yang *et al*	12,288	497,664,000
Chan *et al*	8,960	362,880,000
Yeo and Hu	768	31,104,000

Table 3. Block size: 16×16, search range: ± 8.

With the proposed architecture, $2N^2$ clock cycles are required to estimate the first reference block of each row of blocks in the frame because of the N^2 initial delay. Due to the overlapped execution, the initiation interval of N^2 has been achieved rather than $2N^2$ clock cycles. Assuming that each *slice* consists of one row of blocks, the number of clock cycles required for estimating one slice is $Clock_{slice} = (N_h + 1)N^2$. For a image of size $N_h \times N_v$ blocks, the number of clock cycles required for estimating one frame is $Clock_{frame} = N_v \times Clock_{slice}$. Hence this architecture can estimate upto $\frac{1}{ClockPeriod \times Clock_{frame}}$ frames per second.

The comparison of the proposed architecture with the other existing architectures is presented. The video broadcast format with search ranges of ± 8 and ± 16 and 25ns clock period has been considered. For the proposed architecture and Yang's architecture, search ranges of -8 to $+7$ and -16 to $+15$ are considered. In Table 1 and Table 2, the input pin count, the number of clock cycles required to estimate a motion vector and a frame, and the maximum number of frames estimated per second(f_{max}) have been compared. Table 3 shows the total number of data accesses per second including reference block data and the corresponding search area data.

4. CONCLUSION

In this paper, a modular multiple block matching architecture for FBMA has been described. By employing a technique called chaining and choosing a specific search range, low pin count, low memory bandwidth, and high throughput have been achieved with the 100% utilization of the processors.

This architecture is scalable, and hence it can easily handle various block sizes and search ranges. The comparison shows that this chip can be a optimal one for the video broadcast format with larger frame rate and search range.

5. REFERENCES

[1] T. Komarek and P. Pirsch, "Array Architectures for Block Matching Algorithms", *IEEE Transactions on circuits and systems*, Vol. 36, No. 10, pp. 1301-1308, October 1989.

[2] C. H. Hsieh and T. P. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm", *IEEE Transactions on circuits and systems*, Vol. 2, No. 2, pp. 169-175, June 1992.

[3] Y. S. Jehng, L. G. Chen, and T. D. Chiueh, "An Efficient and Simple VLSI Tree Architecture for Motion Estimation Algotithms", *IEEE Transactions on signal processing*, Vol. 41, No. 2, pp. 889-899, February 1993.

[4] K. M. Yang, M. T. Sun, and 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.

[5] E. Chan and S. Panchanathan, "Motion Estimation Architecture for Video Compression", *IEEE Transactionson Consumer Electronics* Vol. 39, No. 3, pp. 292-297, August 1993.

[6] Y. Hwang and Y. H. Hu, "MSSM-A Design Aid for Multi-Stage Systolic Mapping", *Journal of VLSI Signal Processing*, 4, pp. 125-145, 1992.

[7] L. D. Vos and M. Stegherr, "Parameterizable VLSI Architectures for the Full-Search Block-Matching Algorithm", *IEEE Transactions on circuits and systems*, Vol. 36, No. 10, pp. 1309-1316, October 1989.

[8] S. Y. Kung. *VLSI Array Processors*, Englewood Cliffs, NJ, Prentice Hall, 1988.