CONVOLUTION ON MESH CONNECTED MULTICOMPUTERS [+]

Sanjay Ranka [*] *and Sartaj Sahni*

University of Minnesota

Abstract

Convolution is an important primitive in computer vision and image processing. In this paper, we present an efficient algorithm for convolution on a mesh connected computer with wraparound. Our algorithm does not require a broadcast feature for data values as assumed by previously proposed algorithms. As a result, the algorithm is applicable both to SIMD as well as MIMD meshes. For an $N \times N$ image and a $M \times M$ template, the previous algorithms take $O(M^2 q)$ time on an $N \times N$ mesh connected multicomputer (q is the number of bits in each entry of the convolution matrix). Our algorithms have complexity $O(M^2 r)$ where r = max {number of bits in an image entry, number of bits in a template entry}. So, in addition to not requiring a broadcast capability, our algorithms are faster for binary images.

Keywords and Phrases

Convolution, mesh connected multicomputer

1 INTRODUCTION

The inputs to the image template matching problem are an $N \times N$ image matrix $I[0..N-1, 0..N-1]$ and an $M \times M$ template $T[0..M-1, 0..M-1]$. The output is an $N \times N$ matrix C2D where

$$C2D[i, j] = \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} I[(i+u) \bmod N, (j+v) \bmod N] * T[u,v] \quad 0 \le i, j < N$$

$C2D$ is called the two dimensional convolution of I and T. Template matching, i.e., computing $C2D$, is a fundamental operation in computer vision and image processing. It is often used for edge and object detection; filtering; and image registration [ROSE82, BALL85]. Because of the fundamental nature of this problem and because of its high complexity ($O(M^2 N^2)$ on a single processor computer), much attention has been devoted to the development of efficient fine grain multicomputer parallel algorithms. For example, Chang, Ibarra, Pong and Sohn [CHAN87] have studied this problem on an SIMD pyramid computer; Fang, Li and Ni [FANG85], Fang, Li and Ni [FANG86], Prasanna Kumar and Krishnan [PRAS87], and Ranka and Sahni [RANK88a] and

[+] This research was supported in part by the National Science Foundation under grants DCR84-20935 and MIP 86-17374
* Professor Ranka's current address is: School of CIS, 4-116 Center for Science and Technology, Syracuse University, Syracuse, NY 13244.

[RANK88b] have considered it on a hypercube multicomputer; Fang, Li and Ni [FANG86] have considered perfect shuffle multicomputers; Kung and Song [KUNG81] have considered systolic arrays; and Lee and Aggarwal [LEE87], and Maresca and Li [MARE86] have considered mesh connected computers.

In this paper, a parallel algorithm for 2-D convolution is presented for a mesh connected multicomputer with wraparound. Our algorithm differs from those of [LEE87] and [MARE86] in that our algorithm does not use any broadcast of data values. While some models of SIMD multicomputers support $O(1)$ time data broadcast, others either support no data broadcast or support only a $O(logN)$ data broadcast. The algorithms of [LEE87] and [MARE86] assume an $O(1)$ data broadcast. In the case of MIMD multicomputers, $O(1)$ broadcast is not supported by any of the models. Since our algorithms use no data broadcast, the complexity of this operation is not an issue. Further, the amount of result value movement in our algorithm is an order of magnitude less than in the algorithms of [LEE87] and [MARE86]. Thus when the size of the image and template values is small (e.g, binary images and templates) as compared to the convolution values, our algorithms will be more efficient. In the case of binary images and templates the size of the result values will be $O(logM)$ bits. Thus the number of bits communicated in our algorithm will be $O(M^2)$ as compared to $O(M^2 logM)$ in previous algorithms.

Section 2 describes our computer model. In Section 3, we develop a fine grained algorithm for one dimensional convolution. This forms the basic component of our two dimensional convolution algorithm which is developed in Section 4. Througout, we assume that each processor has O(1) memory. Algorithms for the case when each processor has O(M) memory are developed in [RANK88c].

2 PRELIMINARIES

While our algorithms apply equally well to both SIMD and MIMD multicomputers, we state them explicitly for the case of SIMD multicomputers only. So, we describe only the model for an SIMD mesh connected multicomputer. The important features of such a multicomputer and the programming notation we use are:

1. There are P $\times$ P processing elements connected together via a mesh interconnection network with wraparound (Figure 1). Each PE has a unique index (0..P-1,0..P-1). The local memory of each PE can hold data only (i.e no executable instructions). Hence PEs need be able to perform only the basic arithmetic operations (i.e., no instruction fetch or decode is needed). Throughout this paper, we shall use parentheses('()') to index the PEs. So, A(i, j) refers to the A register of PE(i, j).

2. There is a separate program memory and control unit. The control unit performs instruction

sequencing, fetching, and decoding. In addition, instructions and masks are broadcast by the control unit to the PEs for execution. An *instruction mask* is a boolean function used to select certain PEs to execute an instruction. For example, in the instruction

$A(i, j) := A(i, j) + 1, \quad (i \bmod 4 = 0)$

$(i \bmod 4 = 0)$ is a mask that selects only those PEs whose row index satisfies this property. I.e, all PEs with indices which are multiples of 4 increment their A register by 1.

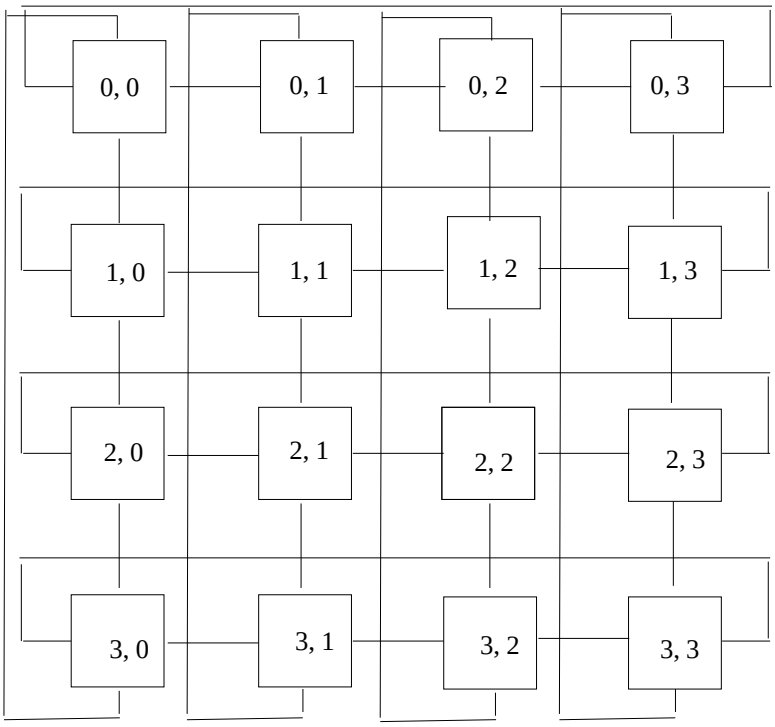

Figure 1 : A 4×4 mesh connected computer

3. In a *unit route*, data may be transmitted from one processor to another only if the two are directly connected. The cost associated with the unit route is the size of the data transmitted (in bits). Since the asymptotic complexity of all our algorithms is determined by the number of unit routes, our complexity analysis will count only these.

3 ONE DIMENSIONAL CONVOLUTION

The inputs to the one dimensional convolution problem are vectors $I[0..N-1]$ and $T[0..M-1]$. The output is the vector C1D where:

$$C1D[i] = \sum_{v=0}^{M-1} I[(i+v) \bmod N] * T[v] \quad , 0 \le v < N$$

We use the computation of C1D as a basic step in our algorithms to compute C2D. In this section, we develop algorithms for C1D on a one dimensional processor array under the assumptions that each PE has $O(1)$ memory and the controller cannot broadcast data values to the processors. There are P=N processors and the vector I is mapped onto the one dimensional processor array such that processor p contains $I[p]$. Further, there are N/M copies of T in the processor array with one copy in each block of M processors (processors $iM + j$, $0 \le j < M$ form a block for each i, $0 \le i < N/M$). Within a block, the mapping of T is the same as that of I. The case when N=16 and M=4 is given in Figure 2. The first row of this figure gives the processor index.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
T_0	T_1	T_2	T_3	T_0	T_1	T_2	T_3	T_0	T_1	T_2	T_3	T_0	T_1	T_2	T_3

Figure 2: Initial configuration for 1-D convolution

Since we only have O(1) memory per PE, we need to rotate the I and T values in a fashion that on every cycle each PE has a pair of I and T values whose product contributes to a convolution value. Figure 3 represents the terms in the C1D values required by each processor for the case M=N=8. Let us number the diagonals in this figure by assigning the main diagonal the number one. The diagonal just above this is numbered two; the next is numbered three and so on. The diagonal just below the main diagonal is numbered 8; the one below it is numbered seven; and so on. Notice that the numbers 2 through 7 are assigned to exactly two diagonal each. Also note that the number of elements on the at most two diagonals that have the same number is 8 for every assigned number. In the first step we compute the terms in the main diagonal (numbered one). The I values required by the PEs are (I_0, I_2, I_4, I_6, I_0, I_2, I_4, I_6), respectively. In the second step we compute the terms in the diagonals numbered 2. The I values required for this are (I_1, I_3, I_5, I_7, I_1, I_3, I_5, I_7), respectively. In the third step the terms on the diagonals numbered 3 are computed. For this the I values required in each PE are (I_2, I_4, I_6, I_0, I_2, I_4, I_6, I_0), respectively. This pattern of I values can be achieved by first pairing I values in the processors. The pair in processor p is (A(p), B(p)) = ($I[(q + 2k) \bmod N]$, $I[(q + 2k + 1) \bmod N]$) where $q = \lfloor p/M \rfloor M$ and $k = p \bmod M$. Figure 4 gives the initial AB pairs in each PE for the case $N = 16$, $M = 4$. This

pairing is easily obtained in M unit routes. Once the AB pairing has been obtained, the C1D may be computed by rotating the AB and T values clockwise. Throughout the algorithm, the product of A(p) and T(p) will give one of the terms needed to compute $C1D(p)$, $0 \le p < P$. B(p) will be the next I value needed. Initially, this is true for all processors except those with $p \bmod M = M - 1$.

P_0	$I_0T_0 + I_1T_1 + I_2T_2 + I_3T_3 + I_4T_4 + I_5T_5 + I_6T_6 + I_7T_7$
P_1	$I_1T_0 + I_2T_1 + I_3T_2 + I_4T_3 + I_5T_4 + I_6T_5 + I_7T_6 + I_0T_7$
P_2	$I_2T_0 + I_3T_1 + I_4T_2 + I_5T_3 + I_6T_4 + I_7T_5 + I_0T_6 + I_1T_7$
P_3	$I_3T_0 + I_4T_1 + I_5T_2 + I_6T_3 + I_7T_4 + I_0T_5 + I_1T_6 + I_2T_7$
P_4	$I_4T_0 + I_5T_1 + I_6T_2 + I_7T_3 + I_0T_4 + I_1T_5 + I_2T_6 + I_3T_7$
P_5	$I_5T_0 + I_6T_1 + I_7T_2 + I_0T_3 + I_1T_4 + I_2T_5 + I_3T_6 + I_4T_7$
P_6	$I_6T_0 + I_7T_1 + I_0T_2 + I_1T_3 + I_2T_4 + I_3T_5 + I_4T_6 + I_5T_7$
P_7	$I_7T_0 + I_0T_1 + I_1T_2 + I_2T_3 + I_3T_4 + I_4T_5 + I_5T_6 + I_6T_7$

Figure 3: Terms in each PE's C1D value

This situation is remedied by replacing B with I in these processors to get the first column labeled AB'. Following a rotation of AB, we get the second column labeled AB. Now, the B value in processors with $p \bmod M = M - 2$ needs to be changed to $I(p)$. With this insight, one arrives at the algorithm of Figure 5. Its correctness is easily established. The number of unit routes (including those for pairing) is at most $3M$.

4 TWO DIMENSIONAL CONVOLUTION

We assume that C2D is to be computed on an N × N mesh connected multicomputer. Further, we assume that $I[i, j]$ is initially in the I register of $PE(i, j)$, the result C2D is to be computed such that $C2D[i, j]$ is in the C2D register of $PE(i, j)$, and that N is a multiple of M. Thus the N × N array may be viewed as composed of N^2/M^2 arrays each of size M × M (Figure 6). We also assume that processor PE(i, j) contains T[$i \bmod M, j \bmod M$] in its T register.

We develop two algorithms for this case. The first is conceptually simpler but requires $4M^2 + O(M)$ unit routes. The second requires only $2M^2 + O(M)$ unit routes. For the first algorithm, we rewrite the definition of C2D as

$$C2D[i, j] = \sum_{r=0}^{M-1} CXD[i, r, j]$$

where

i	I	AB	T	*AB´*	AB	T	*AB´*	AB	T	*AB´*	AB	T	*AB´*
0	I_0	I_0I_1	T_0	I_0I_1	I_1I_2	T_1	I_1I_2	I_2I_3	T_2	I_2I_3	I_3I_4	T_3	I_3I_0
1	I_1	I_2I_3	T_1	I_2I_3	I_3I_4	T_2	I_3I_4	I_4I_5	T_3	I_4I_1	I_1I_2	T_0	I_1I_2
2	I_2	I_4I_5	T_2	I_4I_5	I_5I_6	T_3	I_5I_2	I_2I_3	T_0	I_2I_3	I_3I_4	T_1	I_3I_4
3	I_3	I_6I_7	T_3	I_6I_3	I_3I_4	T_0	I_3I_4	I_4I_5	T_1	I_4I_5	I_5I_6	T_2	I_5I_6
4	I_4	I_4I_5	T_0	I_4I_5	I_5I_6	T_1	I_5I_6	I_6I_7	T_2	I_6I_7	I_7I_8	T_3	I_7I_4
5	I_5	I_6I_7	T_1	I_6I_7	I_7I_8	T_2	I_7I_8	I_8I_9	T_3	I_8I_5	I_5I_6	T_0	I_5I_6
6	I_6	I_8I_9	T_2	I_8I_9	I_9I_{10}	T_3	I_9I_6	I_6I_7	T_0	I_6I_7	I_7I_8	T_1	I_7I_8
7	I_7	$I_{10}I_{11}$	T_3	$I_{10}I_7$	I_7I_8	T_0	I_7I_8	I_8I_9	T_1	I_8I_9	I_9I_{10}	T_2	I_9I_{10}
8	I_8	I_8I_9	T_0	I_8I_9	I_9I_{10}	T_1	I_9I_{10}	$I_{10}I_{11}$	T_2	$I_{10}I_{11}$	$I_{11}I_{12}$	T_3	$I_{11}I_8$
9	I_9	$I_{10}I_{11}$	T_1	$I_{10}I_{11}$	$I_{11}I_{12}$	T_2	$I_{11}I_{12}$	$I_{12}I_{13}$	T_3	$I_{12}I_9$	I_9I_{10}	T_0	I_9I_{10}
10	I_{10}	$I_{12}I_{13}$	T_2	$I_{12}I_{13}$	$I_{13}I_{14}$	T_3	$I_{13}I_{10}$	$I_{10}I_{11}$	T_0	$I_{10}I_{11}$	$I_{11}I_{12}$	T_1	$I_{11}I_{12}$
11	I_{11}	$I_{14}I_{15}$	T_3	$I_{14}I_{11}$	$I_{11}I_{12}$	T_0	$I_{11}I_{12}$	$I_{12}I_{13}$	T_1	$I_{12}I_{13}$	$I_{13}I_{14}$	T_2	$I_{13}I_{14}$
12	I_{12}	$I_{12}I_{13}$	T_0	$I_{12}I_{13}$	$I_{13}I_{14}$	T_1	$I_{13}I_{14}$	$I_{14}I_{15}$	T_2	$I_{14}I_{15}$	$I_{15}I_0$	T_3	$I_{15}I_{12}$
13	I_{13}	$I_{14}I_{15}$	T_1	$I_{14}I_{15}$	$I_{15}I_0$	T_2	$I_{15}I_0$	I_0I_1	T_3	I_0I_{13}	$I_{13}I_{14}$	T_0	$I_{13}I_{14}$
14	I_{14}	I_0I_1	T_2	I_0I_1	I_1I_2	T_3	I_1I_{14}	$I_{14}I_{15}$	T_0	$I_{14}I_{15}$	$I_{15}I_0$	T_1	$I_{15}I_0$
15	I_{15}	I_2I_3	T_3	I_2I_{15}	$I_{15}I_0$	T_0	$I_{15}I_0$	I_0I_1	T_1	I_0I_1	I_1I_2	T_2	I_1I_2

Figure 4: Execution Trace N = 16, M = 4

```
procedure C1D (M)
begin
   PAIRING(M); {obtain initial AB pairs}
   C1D := 0;
   for j := 0 to M-1 do
   begin
         B(p) := I(p); {p mod M = M-1-j}
         C1D := C1D + A * T;
         SHIFT(A, 1); {Shift A register data circularly clockwise by 1}
         C := B; B := A; A:= C; {Exchange A and B}
         SHIFT(T, 1);
     end;
end; { of C1D}
```

Figure 5: Computation of C1D

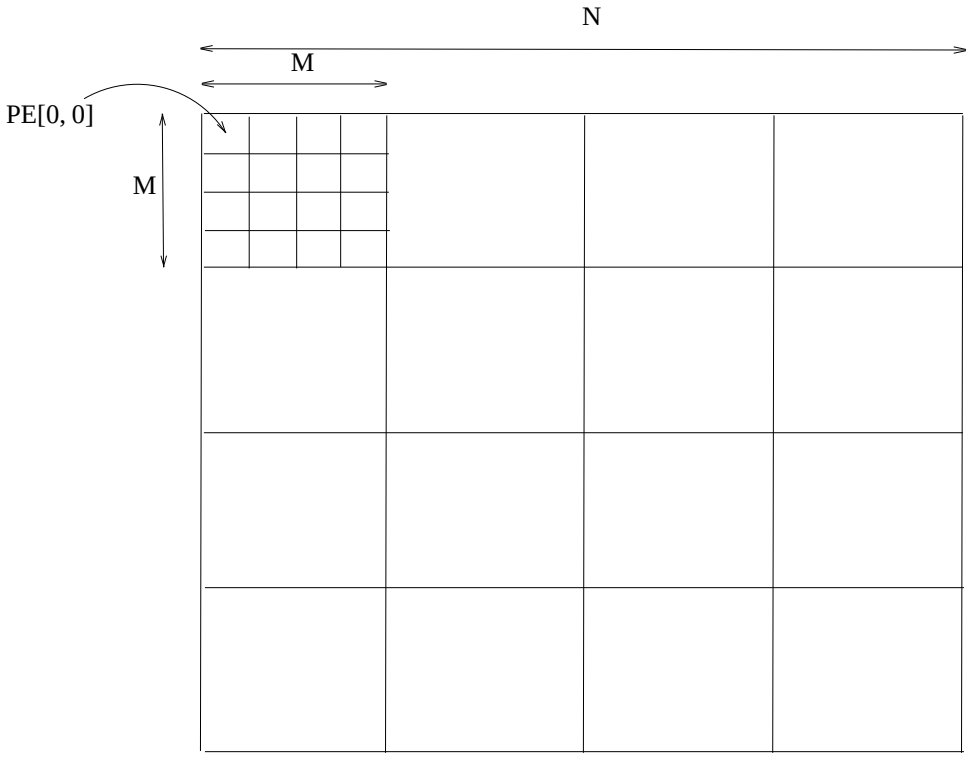

Figure 6: An N × N array viewed as N^2/M^2 M × M arrays

$$CXD[i, r, j] = \sum_{a=0}^{M-1} I[(i + r) \bmod N, (j + a) \bmod N] * T[r, a]$$

A high level description is provided in Figure 7. The number of unit routes is $4M^2 + O(M)$. In the q'th iteration of Steps 1 and 2 of this algorithm C2D(i, j) is computed for all PEs with $i \bmod M = q$. This is done by first having PE($i + r$, j). Compute $C1D(i + r, j) = CXD[i, r, j]$ for $i \bmod M = q$ and $0 \le r < M$. Next $C2D(i, j)$ is computed by summing $CXD[i, r, j]$ for $i \bmod M = q$ and $0 \le r < M$. This summation is done only in PEs (i, j) with $i \bmod M = q$ and requires shifting the CXD[i, r, j] values up along the columns. Each iteration of Steps 2 and 3 takes $4M + O(1)$ unit routes. Thus the unit route count for the entire algorithm is $4M^2 + O(M)$. Note also that this algorithm requires $M - 1$ movement of the C1D values.

The strategy for the second algorithm is similar to that used in computing C1D. We may rewrite the definition of C2D as

$$C2D[i, j] = \sum_{r=0}^{M-1} X[i, j, r] * Y[r]$$

where $X[i, j, r]$ is the 1 × M vector $I[(i + r) \bmod N, j .. (j + M - 1) \bmod N]$ and $Y[r]$ is the 1 × M

procedure C2D(N, M)

{O(1) memory C2D algorithm}

Step1: Repeat Steps 2, 3 and 4 for q := 0 to M-1

Step2: PE($i + r, j$) computes $C1D(i + r, j) = CXD[i, r, j]$ where $i \bmod M = q$ and $0 \le r < M$. This is done using procedure C1D of Figure 5.

Step3: PE(i, j) for $i \bmod M = q$ computes $C2D(i, j) = \sum_{r=0}^{M-1} CXD[(i + r) \bmod N, j]$ by repeatedly shifting the C1D values up the columns of the processor array.

Step4: $T(i, j) \leftarrow T((i + 1) \bmod N, j)$

end; {of C2D}

Figure 7: First algorithm for C2D

vector $T[r, 0 .. M-1]$. Thus C2D is viewed as the one dimensional convolution of X and Y where each term X and Y is a vector. The algorithm is presented in Figure 8. Figure 9 shows the initial AB, I1 I2, A1 A2, and B1 B2 pairs created by the first four PAIRING operation when N=8, and M=4.The total number of unit routes is $2M^2 + O(M)$. Unlike the first algorithm, there is no result movement in this algorithm.

5 CONCLUSION

In this paper, we have developed efficient algorithms for 1-D convolution and image template matching (2-D Convolution) on a mesh connected multicomputer with wraparound. None of our algorithms require a data broadcast. Hence our algorithms apply to all mesh models: SIMD, MIMD, with and without data broadcast. Further, our algorithms require less or no movement of results. Hence, these algorithms will be more efficient when the size of the image and template values is small as compared to the size of the convolution values.

6 REFERENCES

[BALL85] D. H. Ballard and C. M. Brown, "*Computer Vision*", **1985**, Prentice Hall, New Jersey.

[CHAN87] J. H. Chang, O. Ibarra, T. C. Pong, and S. Sohn, "Convolution on a Pyramid Computer", *International Conference on Parallel Processing*, **1987**, pp 780-782.

```
procedure C2D(N, M)
{assumes O(1) memory per PE}
begin
        PAIRING(M) along rows of I; {obtain AB pairs}
        PAIRING(M) along columns of I; {obtain I1 I2 pairs}
        PAIRING(M) along columns of A; {obtain A1 A2 pairs}
        PAIRING(M) along columns of B; {obtain B1 B2 pairs}
        C2D := 0;
        for a := 0 to M-1 do
        begin
                A2(i, j) := A(i, j);  ( i mod M = M-1-a)
                B2(i, j) := B(i, j);  ( i mod M = M-1-a)
                I2(i, j) := I(i, j);  ( i mod M = M-1-a)
                AA := A1 ; BB := B1;
                for b := 0 to M-1 do
                begin
                        BB(i, j) := I1(i, j);  ( j mod M = M-1-b)
                        C2D(i, j) := C2D(i, j) + AA(i, j) * T(i, j);
                        SHIFT(AA, 1) along rows; {shift clockwise by 1}
                        C := BB; BB := AA; AA := C;
                        SHIFT(T, 1) along rows;
                end
                SHIFT(A1, 1) along columns;
                SHIFT(B1, 1) along columns;
                SHIFT(I1, 1) along columns;
                C := A1; A1 := A2; A2 := C;
                C := B1; B1 := B2; B2 := C;
                C := I1; I1 := I2; I2 := C;
                SHIFT(T, 1) along columns;
        end;
end{of C2D}
```

Figure 8: Second two dimensional convolution algorithm

[DEKE86] E. Dekel, D. Nassimi and S. Sahni, " Parallel matrix and graph algorithms", *SIAM Journal on computing,* **1981**, pp. 657-675.

[FANG85] Z. Fang, X. Li and L. M. Ni, "Parallel Algorithms for Image Template Matching on Hypercube SIMD Computers", *IEEE CAPAMI workshop,* **1985**, pp 33-40.

[FANG86] Z. Fang, X. Li and L. M. Ni, "Parallel Algorithms for 2-D convolution", *International Conference on Parallel Processing,* **1986**, pp 262-269.

[HORO85] E. Horowitz and S. Sahni, "*Fundamentals of Data Structures in Pascal*", Computer

AB I1 I2 A1 A2 B1 B2	$I_{0,0}I_{0,1}$ $I_{0,0}I_{1,0}$ $I_{0,0}I_{1,0}$ $I_{0,1}I_{1,1}$	$I_{0,2}I_{0,3}$ $I_{0,1}I_{1,1}$ $I_{0,2}I_{1,2}$ $I_{0,3}I_{1,3}$	$I_{0,4}I_{0,5}$ $I_{0,2}I_{1,2}$ $I_{0,4}I_{1,4}$ $I_{0,5}I_{1,5}$	$I_{0,6}I_{0,7}$ $I_{0,3}I_{1,3}$ $I_{0,6}I_{1,6}$ $I_{0,7}I_{1,7}$	$I_{0,4}I_{0,5}$ $I_{0,4}I_{1,4}$ $I_{0,4}I_{1,4}$ $I_{0,5}I_{1,5}$	$I_{0,6}I_{0,7}$ $I_{0,5}I_{1,5}$ $I_{0,6}I_{1,6}$ $I_{0,7}I_{1,7}$	$I_{0,0}I_{0,1}$ $I_{0,6}I_{1,6}$ $I_{0,0}I_{1,0}$ $I_{0,1}I_{1,1}$	$I_{0,2}I_{0,3}$ $I_{0,7}I_{1,7}$ $I_{0,2}I_{1,2}$ $I_{0,3}I_{1,3}$
AB I1 I2 A1 A2 B1 B2	$I_{1,0}I_{1,1}$ $I_{2,0}I_{3,0}$ $I_{2,0}I_{3,0}$ $I_{2,1}I_{3,1}$	$I_{1,2}I_{1,3}$ $I_{2,1}I_{3,1}$ $I_{2,2}I_{3,2}$ $I_{2,3}I_{3,3}$	$I_{1,4}I_{1,5}$ $I_{2,2}I_{3,2}$ $I_{2,4}I_{3,4}$ $I_{2,5}I_{3,5}$	$I_{1,6}I_{1,7}$ $I_{2,3}I_{3,3}$ $I_{2,6}I_{3,6}$ $I_{2,7}I_{3,7}$	$I_{1,4}I_{1,5}$ $I_{2,4}I_{3,4}$ $I_{2,4}I_{3,4}$ $I_{2,5}I_{3,5}$	$I_{1,6}I_{1,7}$ $I_{2,5}I_{3,5}$ $I_{2,6}I_{3,6}$ $I_{2,7}I_{3,7}$	$I_{1,0}I_{1,1}$ $I_{2,6}I_{3,6}$ $I_{2,0}I_{3,0}$ $I_{2,1}I_{3,1}$	$I_{1,2}I_{1,3}$ $I_{2,7}I_{3,7}$ $I_{2,2}I_{3,2}$ $I_{2,3}I_{3,3}$
AB I1 I2 A1 A2 B1 B2	$I_{2,0}I_{2,1}$ $I_{4,0}I_{5,0}$ $I_{4,0}I_{5,0}$ $I_{4,1}I_{5,1}$	$I_{2,2}I_{2,3}$ $I_{4,1}I_{5,1}$ $I_{4,2}I_{5,2}$ $I_{4,3}I_{5,3}$	$I_{2,4}I_{2,5}$ $I_{4,2}I_{5,2}$ $I_{4,4}I_{5,4}$ $I_{4,5}I_{5,5}$	$I_{2,6}I_{2,7}$ $I_{4,3}I_{5,3}$ $I_{4,6}I_{5,6}$ $I_{4,7}I_{5,7}$	$I_{2,4}I_{2,5}$ $I_{4,4}I_{5,4}$ $I_{4,4}I_{5,4}$ $I_{4,5}I_{5,5}$	$I_{2,6}I_{2,7}$ $I_{4,5}I_{5,5}$ $I_{4,6}I_{5,6}$ $I_{4,7}I_{5,7}$	$I_{2,0}I_{2,1}$ $I_{4,6}I_{5,6}$ $I_{4,0}I_{5,0}$ $I_{4,1}I_{5,1}$	$I_{2,2}I_{2,3}$ $I_{4,7}I_{5,7}$ $I_{4,2}I_{5,2}$ $I_{4,3}I_{5,3}$
AB I1 I2 A1 A2 B1 B2	$I_{3,0}I_{3,1}$ $I_{6,0}I_{7,0}$ $I_{6,0}I_{7,0}$ $I_{6,1}I_{7,1}$	$I_{3,2}I_{3,3}$ $I_{6,1}I_{7,1}$ $I_{6,2}I_{7,2}$ $I_{6,3}I_{7,3}$	$I_{3,4}I_{3,5}$ $I_{6,2}I_{7,2}$ $I_{6,4}I_{7,4}$ $I_{6,5}I_{7,5}$	$I_{3,6}I_{3,7}$ $I_{6,3}I_{7,3}$ $I_{6,6}I_{7,6}$ $I_{6,7}I_{7,7}$	$I_{3,4}I_{3,5}$ $I_{6,4}I_{7,4}$ $I_{6,4}I_{7,4}$ $I_{6,5}I_{7,5}$	$I_{3,6}I_{3,7}$ $I_{6,5}I_{7,5}$ $I_{6,6}I_{7,6}$ $I_{6,7}I_{7,7}$	$I_{3,0}I_{3,1}$ $I_{6,6}I_{7,6}$ $I_{6,0}I_{7,0}$ $I_{6,1}I_{7,1}$	$I_{3,2}I_{3,3}$ $I_{6,7}I_{7,7}$ $I_{6,2}I_{7,2}$ $I_{6,3}I_{7,3}$
AB I1 I2 A1 A2 B1 B2	$I_{4,0}I_{4,1}$ $I_{4,0}I_{5,0}$ $I_{4,0}I_{5,0}$ $I_{4,1}I_{5,1}$	$I_{4,2}I_{4,3}$ $I_{4,1}I_{5,1}$ $I_{4,2}I_{5,2}$ $I_{4,3}I_{5,3}$	$I_{4,4}I_{4,5}$ $I_{4,2}I_{5,2}$ $I_{4,4}I_{5,4}$ $I_{4,5}I_{5,5}$	$I_{4,6}I_{4,7}$ $I_{4,3}I_{5,3}$ $I_{4,6}I_{5,6}$ $I_{4,7}I_{5,7}$	$I_{4,4}I_{4,5}$ $I_{4,4}I_{5,4}$ $I_{4,4}I_{5,4}$ $I_{4,5}I_{5,5}$	$I_{4,6}I_{4,7}$ $I_{4,5}I_{5,5}$ $I_{4,6}I_{5,6}$ $I_{4,7}I_{5,7}$	$I_{4,0}I_{4,1}$ $I_{4,6}I_{5,6}$ $I_{4,0}I_{5,0}$ $I_{4,1}I_{5,1}$	$I_{4,2}I_{4,3}$ $I_{4,7}I_{5,7}$ $I_{4,2}I_{5,2}$ $I_{4,3}I_{5,3}$
AB I1 I2 A1 A2 B1 B2	$I_{5,0}I_{5,1}$ $I_{6,0}I_{7,0}$ $I_{6,0}I_{7,0}$ $I_{6,1}I_{7,1}$	$I_{5,2}I_{5,3}$ $I_{6,1}I_{7,1}$ $I_{6,2}I_{7,2}$ $I_{6,3}I_{7,3}$	$I_{5,4}I_{5,5}$ $I_{6,2}I_{7,2}$ $I_{6,4}I_{7,4}$ $I_{6,5}I_{7,5}$	$I_{5,6}I_{5,7}$ $I_{6,3}I_{7,3}$ $I_{6,6}I_{7,6}$ $I_{6,7}I_{7,7}$	$I_{5,4}I_{5,5}$ $I_{6,4}I_{7,4}$ $I_{6,4}I_{7,4}$ $I_{6,5}I_{7,5}$	$I_{5,6}I_{5,7}$ $I_{6,5}I_{7,5}$ $I_{6,6}I_{7,6}$ $I_{6,7}I_{7,7}$	$I_{5,0}I_{5,1}$ $I_{6,6}I_{7,6}$ $I_{6,0}I_{7,0}$ $I_{6,1}I_{7,1}$	$I_{5,2}I_{5,3}$ $I_{6,7}I_{7,7}$ $I_{6,2}I_{7,2}$ $I_{6,3}I_{7,3}$
AB I1 I2 A1 A2 B1 B2	$I_{6,0}I_{6,1}$ $I_{0,0}I_{1,0}$ $I_{0,0}I_{1,0}$ $I_{0,1}I_{1,1}$	$I_{6,2}I_{6,3}$ $I_{0,1}I_{1,1}$ $I_{0,2}I_{1,2}$ $I_{0,3}I_{1,3}$	$I_{6,4}I_{6,5}$ $I_{0,2}I_{1,2}$ $I_{0,4}I_{1,4}$ $I_{0,5}I_{1,5}$	$I_{6,6}I_{6,7}$ $I_{0,3}I_{1,3}$ $I_{0,6}I_{1,6}$ $I_{0,7}I_{1,7}$	$I_{6,4}I_{6,5}$ $I_{0,4}I_{1,4}$ $I_{0,4}I_{1,4}$ $I_{0,5}I_{1,5}$	$I_{6,6}I_{6,7}$ $I_{0,5}I_{1,5}$ $I_{0,6}I_{1,6}$ $I_{0,7}I_{1,7}$	$I_{6,0}I_{6,1}$ $I_{0,6}I_{1,6}$ $I_{0,0}I_{1,0}$ $I_{0,1}I_{1,1}$	$I_{6,2}I_{6,3}$ $I_{0,7}I_{1,7}$ $I_{0,2}I_{1,2}$ $I_{0,3}I_{1,3}$
AB I1 I2 A1 A2 B1 B2	$I_{7,0}I_{7,1}$ $I_{2,0}I_{3,0}$ $I_{2,0}I_{3,0}$ $I_{2,1}I_{3,1}$	$I_{7,2}I_{7,3}$ $I_{2,1}I_{3,1}$ $I_{2,2}I_{3,2}$ $I_{2,3}I_{3,3}$	$I_{7,4}I_{7,5}$ $I_{2,2}I_{3,2}$ $I_{2,4}I_{3,4}$ $I_{2,5}I_{3,5}$	$I_{7,6}I_{7,7}$ $I_{2,3}I_{3,3}$ $I_{2,6}I_{3,6}$ $I_{2,7}I_{3,7}$	$I_{7,4}I_{7,5}$ $I_{2,4}I_{3,4}$ $I_{2,4}I_{3,4}$ $I_{2,5}I_{3,5}$	$I_{7,6}I_{7,7}$ $I_{2,5}I_{3,5}$ $I_{2,6}I_{3,6}$ $I_{2,7}I_{3,7}$	$I_{7,0}I_{7,1}$ $I_{2,6}I_{3,6}$ $I_{2,0}I_{3,0}$ $I_{2,1}I_{3,1}$	$I_{7,2}I_{7,3}$ $I_{2,7}I_{3,7}$ $I_{2,2}I_{3,2}$ $I_{2,3}I_{3,3}$

Figure 9: Configuration after the four PAIRING operations in Figure 13 for the case N = 8 and M = 4

Science Press, **1985**.

[KUNG82] H. T. Kung and S. W. Song, "A Systolic 2-D Convolution Chip", *Multicomputers and Image Processing: Algorithms and Programs, editors: Preston and Uhr (Academic Press, New York),* **1982**, pp 373-384.

[LEE87] S. Y. Lee and J. K. Aggarwal, "Parallel 2-D convolution on a mesh connected array processor", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **July 1987**, pp 590-594.

[MARE86] M. Maresca and H. Li, "Morphological Operations on Mesh-connected Architecture : A generalized convolution Algorithm", *Proceedings of 1986 IEEE Computer Society Workshop on Computer Vision and Pattern Recognition* ,**1986**, pp 299-304.

[PRAS87] V. K. Prasanna Kumar and V. Krishnan, "Efficient Image Template Matching on SIMD Hypercube Machines", *International Conference on Parallel Processing*, **1987**, pp 765-771.

[RANK88a] S. Ranka and S. Sahni, "Image Template Matching on an SIMD hypercube multicomputers", *Proceedings 1988 International Conference on Parallel Processing*, Vol III, Algorithms & Applications, Pennsylvania University Press, pp 84-91.

[RANK88b] S. Ranka and S. Sahni, "Image Template Matching on MIMD hypercube multicomputers", *Proceedings 1988 International Conference on Parallel Processing*, Vol III, Algorithms & Applications, Pennsylvania University Press, pp 92-99.

[RANK88c] S. Ranka and S. Sahni, "Convolution on SIMD mesh connected multicomputers", *Proceedings 1988 International Conference on Parallel Processing*, Vol III, Algorithms & Applications, Pennsylvania University Press, pp 212-217.

[ROSE82] A. Rosenfeld and A. C. Kak, "*Digital Picture Processing*", Academic Press, **1982**