Maximal Contour Algorithms on Constrained Reconfigurable Meshes

M. Manzur Murshed
Computer Sciences Laboratory
The Australian National University
Canberra ACT 0200, Australia

Richard P. Brent
Oxford University Computing Laboratory
Oxford University
Oxford, OX1 3QD, U.K.

Abstract: *The Reconfigurable Mesh (RM) attracted criticism for its key assumption that a message can be broadcast in constant time independent of bus length. To account for this limit Beresford-Smith* et al. *have recently proposed the k-constrained RM model where buses of length at most k, a constant, are allowed to be formed. Straightforward simulations of optimal RM algorithms on this constrained RM model are found to be non-optimal. This paper presents two optimal algorithms to compute the contour of maximal elements of a set of planar points on the k-constrained RM. The first algorithm solves this problem of size p in $O(\frac{p}{k})$ time on an k-constrained RM of size $k \times p$, $k \leq p$, and the second algorithm solves this problem of size n in $O(\frac{q}{k})$ time on an k-constrained RM of size $p \times q$, $p \leq q$, and $pq = kn$.*

Keywords: *Reconfigurable mesh; Parallel algorithm; AT^2 optimality; Computational geometry*

1 Introduction

It is well-known that interprocessor communications and simultaneous memory accesses often act as bottlenecks in present-day parallel machines. Bus systems have been introduced recently to a number of parallel machines to address this problem. Examples include the *Bus Automaton* [16], the *Reconfigurable Mesh (RM)* [12], the *content addressable array processor* [20], and the *Polymorphic torus* [11]. Among them RM draws much attention because of its simplicity. A bus system is called *reconfigurable* if it can be dynamically changed according to either global or local information.

In the most common model of RM, it is assumed that a message can be broadcast in constant time along any bus independent of its length. This assumption attracted criticism and cast a shadow of doubt on the implementation of RM. Although investigations of bus delays in [8–10] has confirmed that broadcast delay is very small, theoretically it cannot be correctly modelled by a constant independent of bus length. To account for this limit, Beresford-Smith *et al.* [2] has proposed the *k-constrained* RM where buses of length at most k, a constant, are allowed to be formed at any time.

In [2, 5, 6] it has been pointed out that straightforward simulations of RM algorithms on the k-constrained RM compromise with AT^2 [17, chapter 2] optimality. To address this issue, optimal algorithms have already been developed for sorting and computing convex hull [2], broadcasting [5], and multiplying sparse matrices [6] on the k-constrained RM. In this paper we explore one further problem from a similar point of view. We present here two optimal algorithms on the k-constrained RM to compute the contour of the maximal elements of a given set of planar points. The first algorithm computes the $\mathcal{M}$-contour (see Section 2.2) of p planar points on a k-constrained RM of size $k \times p$ in $O(\frac{p}{k})$ time and the second algorithm computes the $\mathcal{M}$-contour of n planar points on a k-constrained RM of size $p \times q$ in $O(\frac{q}{k})$ time, where $p \leq q$ and $pq = kn$.

The paper is organised as follows. In the next section we present the basic issues associated with RM and its various models based on message propagation delay. This section also presents definition of the $\mathcal{M}$-contour problem and its AT^2 lower bounds. Two AT^2 optimal $\mathcal{M}$-contour algorithms on the k-constrained RM are developed in Section 3. Section 4 concludes the paper.

2 Preliminaries

For the sake of completeness, here we briefly define the reconfigurable mesh and its various models based on message propagation delay in Section 2.1. We give definition of the $\mathcal{M}$-contour problem in Section 2.2 and discuss some AT^2 lower bounds in Section 2.3. Throughout the paper, we use $\Theta()$ to mean "order exactly," $O()$ to mean "order at most,"

and $\Omega()$ to mean "order at least."

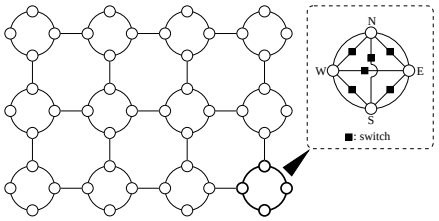

Figure 1: A reconfigurable mesh of size 3×4.

2.1 The Reconfigurable Mesh

The reconfigurable mesh is primarily a two-dimensional mesh of processors connected by reconfigurable buses. In this parallel architecture, a processor is placed at the grid points as in the usual mesh connected computers. Processors of the RM of size $X \times Y$ are denoted by $PE_{i,j}$, $0 \leq i < X$, $0 \leq j < Y$ where processor $PE_{0,0}$ resides in the south-western corner. Each processor is connected to at most four neighbouring processors through fixed bus segments connected to four I/O ports **E** & **W** along dimension x and **N** & **S** along dimension y. These fixed bus segments are building blocks of larger bus components which are formed through switching, decided entirely on local data, of the internal connectors (see Figure 1) between the I/O ports of each processor. The fifteen possible interconnections of I/O ports through switching are shown in Figure 2. The connection patterns are represented as $\{p_1, p_2, \ldots\}$, where each of p_i represents a group of switches connected together such that $\bigcup_{\forall i} p_i = \{\mathbf{N}, \mathbf{E}, \mathbf{W}, \mathbf{S}\}$. For example, {**NS**,**E**,**W**} represents the connection pattern with ports **N** and **S** connected and ports **E** and **W** unconnected.

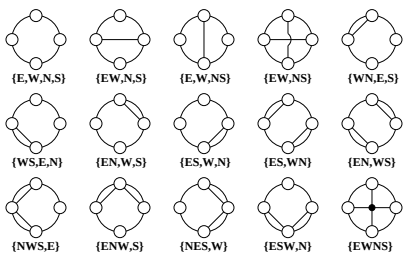

Figure 2: Possible interconnections among four I/O ports of a processor.

A reconfigurable mesh operates in the single instruction multiple data (SIMD) mode. Besides the reconfigurable switches, each processor has a computing unit with a fixed number of local registers. A single time step of an RM is composed of the following four substeps:

BUS substep. Every processor switches the internal connectors between I/O ports by local decision.

WRITE substep. Along each bus, one or more processors on the bus transmit a message of length bounded by the bandwidth of the fixed bus segments as well as the switches. These processors are called the *speakers*. It is assumed that a collision between several speakers will be detected by all the processors connected to the bus and the transmitted message will be discarded.

READ substep. Some or all the processors connected to a bus read the message transmitted by a single speaker. These processors are called the *readers*.

COMPUTE substep. A constant-time local computation is done by each processor.

Other than the buses and switches the RM of size $p \times q$ is similar to the standard mesh of size $p \times q$ and hence it has $\Theta(pq)$ area in VLSI embedding [17], under the assumption that processors, switches, and links between adjacent switches occupy unit area.

One critical factor in the complexity analysis of reconfigurable algorithms is the time needed to propagate a message over a bus. In the most common *unit-time delay* model [19], it is assumed that in any configuration any message can be transmitted along any bus in constant time, regardless of the bus length. Unfortunately this assumption, based on which a large number of algorithms with constant time complexity are developed, is theoretically false, as the speed of signals carrying information is bounded by the speed of light. This partially explains why the reconfigurable meshes have not gained wide acceptance initially. Recently some VLSI implementations of reconfigurable meshes have demonstrated that the broadcast delay, though not a constant, is nevertheless relatively small in terms of machine cycles. For example, only 16 machine cycles are required to broadcast on a 10^6 processor YUPPIE (Yorktown Ultra Parallel Polymorphic Image Engine) [9,10]. GCN (Gated-Connection Network) [8] has even shorter delays by adopting precharged circuits. Broadcast delay can further be reduced by using optical fibre for reconfigurable bus system and electrically controlled directional coupler switches as proposed in [1]. Although the above observations serve the practical purposes, unit-time delay can never be theoretically sound. To account for this theoretical limit, two different models have been introduced in the literature.

In the *log-time delay* model [13] it is assumed

that each broadcast takes $\Theta(\log s)$ time to reach all the processors connected to a bus, where s is the maximum number of switches in a minimum switch path between two processors connected on the bus.

Beresford-Smith *et al.* [2] have recently proposed the *k-constrained* model where it is assumed that in any situation any message can propagate at most k fixed bus segments and thus buses of length at most k are allowed in any step, where k is a constant. We use the notation RM_A^k to refer to a k-constrained RM of area A.

A *linear* bus is a bus which never branches, thereby excluding the connection patterns {**NSE**,**W**}, {**NSW**,**E**}, {**EWN**,**S**}, {**EWS**,**N**}, and {**NSEW**}. Now, any RM algorithm which uses only linear buses can be simulated by the k-constrained RM by propagating signals k processors at a time.

Lemma 1 *Any message can be broadcast over a linear bus of length l in $O(\frac{l}{k})$ time on a k-constrained RM.* ■

Beresford-Smith *et al.* [2] have shown in obvious way that such a simulation loses AT^2 optimality unless the area of the mesh is reduced. To address this issue, AT^2 optimal algorithms have already been developed for sorting and computing convex hull [2], broadcasting [5], and multiplying sparse matrices [6] on the k-constrained RM.

Throughout this paper RM is assumed to be unconstrained if not stated otherwise.

2.2 Problem Definition

Let the planar point at coordinate (i, j) be defined as $P(i, j)$. Again, let for any point p, $x(p)$ denote the x-coordinate and $y(p)$ denote the y-coordinate of p, e.g., $x(P(i, j)) = i$ and $y(P(i, j)) = j$.

Definition 1 *A point p* dominates *a point q (denoted by $q \prec p$) if $x(q) \leq x(p)$ and $y(q) \leq y(p)$. (The relation "$\prec$" is naturally called* dominance.*)*

Let S be a set of N planar points. To simplify the exposition of our algorithms, the points in S are assumed to be distinct.

Definition 2 *A point $p \in S$ is* maximal *if there is no other point $q \in S$ with $p \prec q$.*

We are interested in the contour spanned by the maximal elements of S, called the *$\mathcal{M}$-contour* of S which can be obtained by simply sorting the maximal elements in ascending order of their x-coordinates (Figure 3). Let the $\mathcal{M}$-contour of a set S be denoted as $m(S)$.

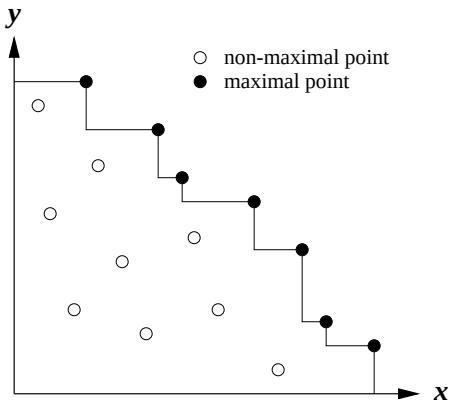

Figure 3: $\mathcal{M}$-contour of a set of planar points.

We have made two interesting observations on $\mathcal{M}$-contour in [14, 15] which are given below for the sake of completeness.

Lemma 2 *Every $\mathcal{M}$-contour is sorted in descending order of the y-coordinates.*

Proof. Suppose the contrary. Then there exists at least one pair of maximal elements p and q such that $y(p) < y(q)$ while $x(p) \leq x(q)$, which contradicts with the assumption that point p is maximal. ■

Let for any set S of some planar points functions $min_x(S)$ and $max_x(S)$ denote the *minimum* and *maximum* x-coordinates in the set respectively. Let two more functions $min_y(S)$ and $max_y(S)$ be defined similarly w.r.t. y-coordinate.

Lemma 3 *Given K sets S_0, S_1, $\ldots S_{K-1}$ of planar points such that $\forall t : 0 \leq t < K-1$, $max_x(S_t) \leq min_x(S_{t+1})$, then $\forall i : 0 \leq i < K-1$, $\forall p \in m(S_i) \wedge y(p) > max_y(m(S_j))$, $\forall j > i$, if and only if, $p \in m(\bigcup_{t=0}^{K-1} S_t)$.*

Proof. The *necessity* part can be proved by arranging a contradiction of Lemma 2. To prove the *sufficiency* part we take a point $p \in m(S_i)$, $\exists i : 0 \leq i < K-1 \wedge p \notin m(\bigcup_{t=0}^{K-1} S_t)$. Then by the definition of maximality we get $\exists q \in \bigcup_{t=i+1}^{K-1} S_t$ such that $p \prec q$, i.e., $y(p) \leq y(q)$. ■

The $\mathcal{M}$-contour problem is also known as finding the maxima of a set of vectors and has been extensively explored for serial computers in [7, 8, 21]. Computation of maximal elements is important in solving the *Largest Empty Rectangle Problem* [4] where a rectangle R, and a number of planar points $S \in R$, are given and the problem is to compute the largest rectangle $r \subseteq R$ that contains no point in S and whose sides are parallel to those of R. If R is divided into four quadrants then the maximal elements w.r.t. the northeast(NE), northwest(NW),

southwest(SW), and southeast(SE) directions, as depicted in Figure 4, remain the only candidates to be the supporting elements of the empty rectangles lying in all the four quadrants.

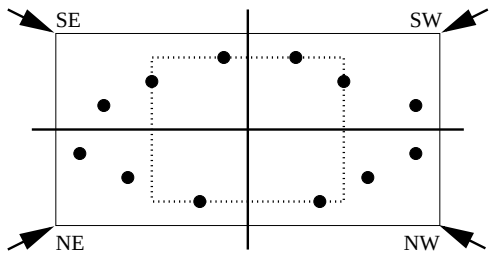

Figure 4: Importance of maximal elements in computing largest empty rectangle.

It is well known that the time complexity for computing the contour of the maximal elements of n planar points is $\Theta(n \log n)$ using a serial computer [8]. This lower boundary can be concluded from the fact that the problem of sorting can be transformed into an $\mathcal{M}$-contour problem of same size. The AT^2 lower bound of $\mathcal{M}$-contour problem of size n, as shown in Section 2.3, is $\Omega(n^2)$. Dehne [3] developed an AT^2 optimal algorithm for solving $\mathcal{M}$-contour problem on a mesh of size $\sqrt{n} \times \sqrt{n}$ in $O(\sqrt{n})$ time. The authors [14,15] recently presented three constant time $\mathcal{M}$-contour algorithms on RM of various dimensions. Using the result of optimal simulation of a multidimensional RM by a 2-dimensional RM in [18], it can easily be shown that all the three algorithms in [14,15] are AT^2 optimal.

2.3 AT^2 Lower Bounds

Let a problem $\mathcal{P}$ of size n have I *information content* [17, pages 51-54]. If this problem $\mathcal{P}$ is realised in a VLSI circuit with aspect ratio α then, by Ullman [17, page 57], AT^2 lower bound of solving $\mathcal{P}$ will be $\Omega(\alpha I^2)$.

Problem of sorting n $\log n$-bit numbers has information content $I = \Omega(n)$. As the problem of sorting n numbers can be transformed into a problem of computing the contour of maximal elements of n planar points, it can be concluded that the $\mathcal{M}$-contour problem of size n has information content $I = \Omega(n)$. Hence, the AT^2 lower bound of computing $\mathcal{M}$-contour of n planar points is $\Omega(n^2)$.

Let $\mathcal{M}$-contour of p planar points on an RM^k_{kp} of size $k \times p$, $k \leq p$, be done in $O(T)$ time. This solution will be AT^2 optimal if

$$kpT^2 = \frac{p}{k} p^2 \ ,$$

i.e., $T = \frac{p}{k}$. Similarly, to compute $\mathcal{M}$-contour of n planar points, AT^2 optimally, on RM^k_{kn} of size $p \times q$, $p \leq q$, the solution time must be $O(\frac{q}{k})$.

For the sake of simplicity, we assume $k = r^2$, $p = k^2 s^2$, and $q = k^2 t^2$ where r, s, and t are integers and $s \leq t$.

3 Optimal $\mathcal{M}$-Contour Algorithms on the k-constrained RM

We first briefly outline a few published results which are used in our algorithms.

Lemma 4 *Given p items in the first row of an RM^k_{kp} of size $k \times p$, $k \leq p$, these items can be sorted in $O(\frac{p}{k})$ time, which is AT^2 optimal.*

Proof. See in [2]. ■

Lemma 5 *Given n items in the columns jk, $0 \leq j < \frac{n}{p}$, of an RM^k_{kn} of size $p \times q$, $p \leq q$, these items can be sorted in $O(\frac{q}{k})$ time, which is AT^2 optimal.*

Proof. See in [2]†. ■

Given a binary sequence, b_j, $j = 0, 1, \ldots, N-1$, the *and* computation is to compute $b_0 \wedge b_1 \wedge \cdots \wedge b_{N-1}$. Similarly the *or* computation computes $b_0 \vee b_1 \vee \cdots \vee b_{N-1}$. Adapting the technique of bus splitting [13] it is easy to show that:

Lemma 6 *Given a binary sequence of length k in the only row of an RM^k_k of size $1 \times k$, both the* and *and the* or *of the elements in the sequence can be computed in $O(1)$ time.*

Proof. It is easy to check that the constant time algorithm in [13] on an RM of size $1 \times k$ uses only horizontal buses of length at most k. Hence, the same algorithm can be used for RM^k_k of size $1 \times k$. ■

Lemma 7 *Computing $\mathcal{M}$-contour of k planar points in the first row of an $RM^k_{k^2}$ of size $k \times k$ can be done in $O(1)$ time.*

Proof. Again it is easy to check that the first algorithm in [14,15] on an RM of size $k \times k$ can compute the $\mathcal{M}$-contour of k planar points in constant time using only linear buses of length $O(k)$. Hence, the same algorithm can be used for $\mathrm{RM}^k_{k^2}$ of size $k \times k$ to compute the same problem in $O(1)$ time. ■

† In [2], n items are initially given in the first $\frac{n}{p}$ columns and in that case the proof presented in [2] is incomplete.

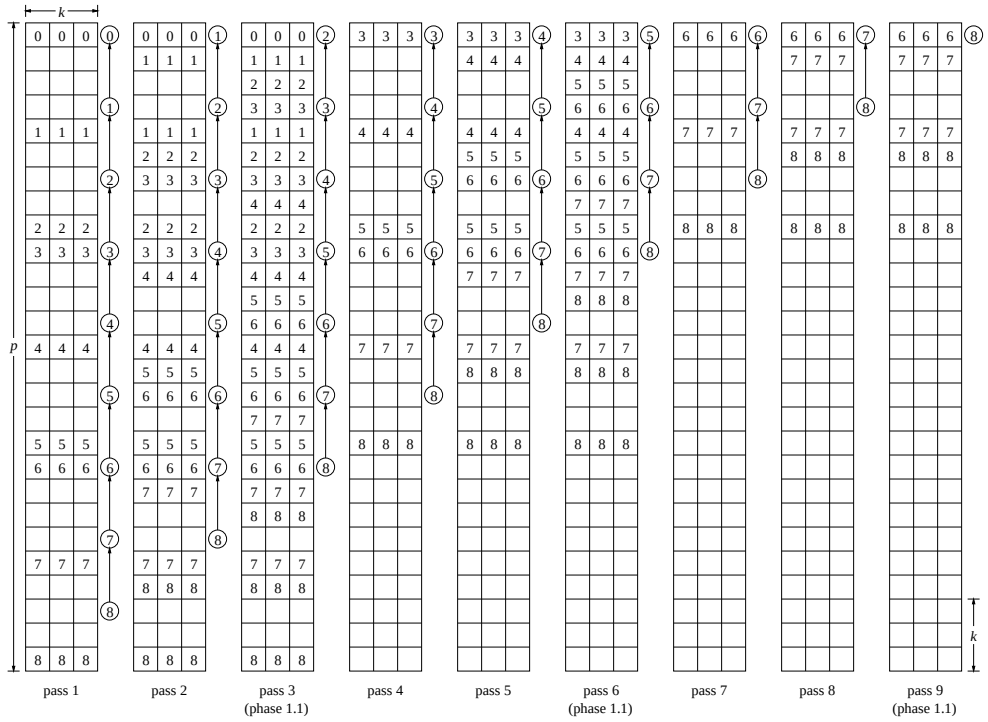

Figure 5: All the passes of the pipelining operation discussed in Section 3.1. Here $p = 27$, $k = 3$, and the propagation of $max_y(m(S_i))$ is represented by integer i for $0 \leq i < 9$. Note that for simplification, k is not considered as r^2 for some integer r as assumed.

3.1 Optimal Computing of the $\mathcal{M}$-Contour of p Planar Points on RM^k_{kp} of Size $k \times p$, $k \leq p$

First we sort the points w.r.t. x-coordinate by Lemma 4 in $O(\frac{p}{k})$ time. Let the RM^k_{kp} of size $k \times p$ be divided into $\frac{p}{k}$ submeshes of size $k \times k$ each and the given p planar points in the first row be distributed in such a way that each processor $PE_{i,jk}$, $0 \leq i < k$ and $0 \leq j < \frac{p}{k}$, receives a point. It is obvious that such a redistribution of elements can be carried out in constant time using a column broadcast followed by a row broadcast.

Let the points residing in column jk be denoted by the set S_j, $0 \leq j < \frac{p}{k}$. Clearly these $\frac{p}{k}$ sets of planar points follow the condition of Lemma 3, i.e., $\forall j : 0 \leq j < \frac{p}{k} - 1$, $max_x(S_j) \leq min_x(S_{j+1})$. The $\mathcal{M}$-contours $m(S_j)$, $0 \leq j < \frac{p}{k}$, are now computed in parallel using a submesh of size $k \times k$ for each computation. By Lemma 7 this operation takes only $O(1)$ time. Now, we transfer the $max_y(m(S_j))$ values to the first row of the RM^k_{kp} in the following single RM step using Lemma 2:

1. b: Any processor in column j containing a point $\in m(S_j)$ disconnects all port interconnections while the rest of the processors connect port **N** with **S** for all $0 \leq j < \frac{p}{k}$.

   w: Any processor in column j containing a point $\in m(S_j)$ now writes the y-coordinate of the point to port **S** for all $0 \leq j < \frac{p}{k}$.

   r: Every processor in the first row reads port **S** in.

Here, the substeps are labelled as "b:", "w:", "r:", and "c:" to denote the BUS, WRITE, READ, and COMPUTE substeps respectively. As the buses formed in the above RM step are of length at most k, this RM step can also be used as a single k-constrained RM step.

Now, the $\mathcal{M}$-contour of the entire p points can be computed in the following RM phases:

1. Iterate the following for $t = 0, 1, \ldots, \frac{p}{k^2} - 1$:

   1.1 Copy $max_y(m(S_{tk+v}))$ to processors $PE_{i,v+rk}$, $0 \leq i < k$, $r = 0, 1, \ldots, tk + v$, using a column broadcast then a row broadcast and finally a column broadcast for all $v : 0 \leq v < k$.

   1.2 Copy the y-coordinate of the point residing in processor $PE_{i,kj}$ to the processors $PE_{i,kj+r}$, $0 \leq r < k$, using a row broadcast, for all $0 \leq i < k$ and $0 \leq j < k(t+1)$.

1.3 Now in the jth submesh of size $k \times k$, the ith row contains k max_y values paired with the y-coordinate of a particular point, say d. Now, apply Lemma 3 to eliminate d by computing the *and* over the comparison values for all $j : 0 \leq j < k(t+1)$.

The bus length in the column broadcasts in phase 1.1 is at most k while the same in the row broadcast is at most p. Hence, by Lemma 1 phase 1.1 takes $O(\frac{p}{k})$ time on the k-constrained RM. Phase 1.2 uses buses of length at most k and thus it can be done in constant time. By Lemma 6 phase 1.3 takes $O(1)$ time. It can then be concluded that the above iteration takes $O(\frac{p}{k^2}\frac{p}{k})$ time which is non-optimal.

Now, the ideas of *pipelining* can be applied to the above iteration to achieve optimal time. Observe that in the t-th iteration, data are moved from the u-th submeshes of size $k \times k$ each, $kt \leq u < k(t+1)$, to all the v-th submeshes of size $k \times k$ each, $0 \leq v < k(t+1)$. So, we can start all the iterations simultaneously without making any bus-access conflict by pipelining data as shown in Figure 5. Obviously such pipeline emulation of the above iteration takes only $O(\frac{p}{k})$ steps as buses of length at most k are configured. As $\frac{p}{k^2} \leq \frac{p}{k}$, it can be concluded that

Theorem 1 *Given p planar points in the first row of an RM^k_{kp} of size $k \times p$, $k \leq p$, the $\mathcal{M}$-contour of these points can be computed in $O(\frac{p}{k})$ time, which is AT^2 optimal.* ■

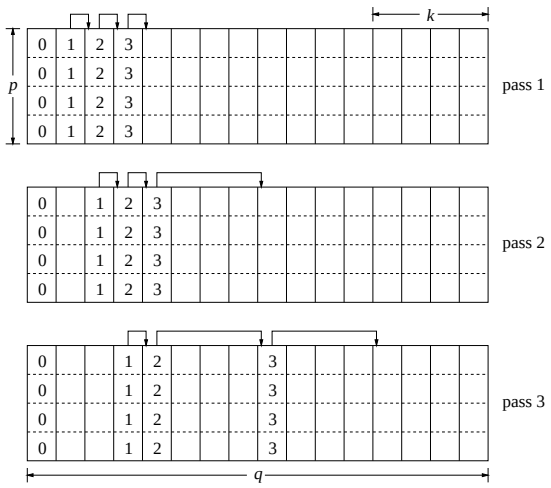

Figure 6: All the passes of the pipelining operation discussed in Section 3.2. Here $p = 4$, $q = 16$, and $k = 4$ and C_j is represented by a column of integer j for $0 \leq j < 4$.

3.2 Optimal Computing of the $\mathcal{M}$-Contour of n Planar Points on RM^k_{kn} of Size $p \times q$, $p \leq q$

Let the RM^k_{kn} of size $p \times q$ be divided into $\frac{q}{k}$ submeshes of size $p \times k$ each and the given n planar points in the first $\frac{n}{p}$ columns be distributed in such a way that each processor $PE_{i,jk}$, $0 \leq i < p$ and $0 \leq j < \frac{q}{k}$, receives a point. It can easily be shown that such a redistribution can be carried out in $O(\frac{n}{p}) = O(\frac{q}{k})$ time using the ideas of pipelining in the following way:

For $0 \leq j < \frac{q}{k}$, let C_j denote the contents of column j in the initial condition. As we assume, for the sake of simplicity, $k = r^2$ and $q = k^2t^2$ where r and t are integers, we may conclude that $\frac{q}{k} = kt^2$. In the first pass, every C_j moves from column j to column $j+1$, for $1 \leq j < \frac{q}{k} - 1$ and $C_{\frac{q}{k}-1}$ moves from column $\frac{q}{k} - 1$ to column kt^2 as shown in Figure 6(pass 1). In the second pass, every C_j moves from column $j+1$ to column $j+2$, for $1 \leq j < \frac{q}{k} - 2$, $C_{\frac{q}{k}-2}$ moves from column $\frac{q}{k} - 1$ to column kt^2 and $C_{\frac{q}{k}-1}$ moves from column kt^2 to column $k(t^2+1)$ as shown in Figure 6(pass 2) and so on. There should be at most $\frac{q}{k} - 1$ passes and whenever any of C_js reaches its destination column in some pass, it will not take part in any of the remaining passes.

Now, we sort the points w.r.t. x-coordinate in column-major order by Lemma 5 in $O(\frac{q}{k})$ time.

Let the points residing in column jk be denoted by the set S_j, $0 \leq j < \frac{q}{k}$. Clearly these $\frac{q}{k}$ sets of planar points follow the condition of Lemma 3, i.e., $\forall j : 0 \leq j < \frac{q}{k} - 1$, $max_x(S_j) \leq min_x(S_{j+1})$. The $\mathcal{M}$-contours $m(S_j)$, $0 \leq j < \frac{q}{k}$, are now computed in parallel using a submesh of size $p \times k$ for each computation. By Theorem 1 this operation takes only $O(\frac{p}{k})$ time.

Now taking very similar steps as used in Section 3.1, the following can be concluded:

Theorem 2 *Given n planar points in the first $\frac{n}{p}$ columns of an RM^k_{kn} of size $p \times q$, $p \leq q$, the $\mathcal{M}$-contour of these points can be computed in $O(\frac{q}{k})$ time, which is AT^2 optimal.* ■

4 Conclusion

In [2, 5, 6] it has been pointed out that straightforward simulations of RM algorithms on the k-constrained RM compromise with AT^2 optimality. To address this issue, we have developed two optimal algorithms to compute the contour of maximal elements of a set of planar points. The first algo-

rithm solves this problem of size p in $O(\frac{p}{k})$ time on an k-constrained RM of size $k \times p$, $k \leq p$, and the second algorithm solves this problem of size n in $O(\frac{q}{k})$ time on an k-constrained RM of size $p \times q$, $p \leq q$, and $pq = kn$.

References

[1] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster. The power of reconfiguration. *Journal of Parallel and Distributed Computing*, 13:139–153, 1991.

[2] B. Beresford-Smith, O. Diessel, and H. ElGindy. Optimal algorithms for constrained reconfigurable meshes. *Journal of Parallel and Distributed Computing*, 39:74–78, 1996.

[3] F. Dehne. $O(n^{1/2})$ algorithms for the maximal elements and ECDF searching problem on a mesh-connected parallel computer. *Information Processing Letters*, 22:303–306, 1986.

[4] F. Dehne. Computing the largest empty rectangle on one- and two-dimensional processor arrays. *Journal of Parallel and Distributed Computing*, 9:63–68, 1990.

[5] O. Diessel, H. ElGindy, and L. Wetherall. Efficient broadcasting procedures for constrained reconfigurable meshes. Technical Report 96-07, Department of Computer Science, The University of Newcastle, Australia, 1996.

[6] H. ElGindy. On sparse matrix multiplication for constrained reconfigurable meshes. In *Proceedings of the 3rd Workshop on Reconfigurable Architectures*, Honolulu, Hawaii, April 1996.

[7] H. T. Kung. On the computational complexity of finding the maxima of a set of vectors. In *15th Annual IEEE Symp. on Switching and Automata Theory*, pages 117–121, Oct. 1974.

[8] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22:469–476, 1975.

[9] H. Li and M. Maresca. Polymorphic-torus network. *IEEE Transactions on Computers*, 38:1345–1351, 1989.

[10] R. Lin, S. Olariu, J. Schwing, and J. Zhang. A VLSI-optimal constant time sorting on reconfigurable mesh. In *Ninth European Workshop Parallel Computing*, pages 1–16, Spain, 1992.

[11] M. Maresca. Polymorphic processor arrays. *IEEE Transactions on Parallel and Distributed Systems*, 4:490–506, 1993.

[12] R. Miller, V. K. P. Kumar, D. I. Reisis, and Q. F. Stout. Data movement operations and applications on reconfigurable VLSI arrays. In *Proc. International Conference on Parallel Processing*, pages 205–208, 1988.

[13] R. Miller, V. K. Prasanna-Kumar, D. I. Reisis, and Q. F. Stout. Parallel computations on reconfigurable meshes. *IEEE Transactions on Computers*, 42:678–692, 1993.

[14] M. M. Murshed and R. P. Brent. Constant time algorithms for computing the contour of maximal elements on the reconfigurable mesh. In *Proceedings of the 1997 International Conference on Parallel and Distributed Systems*, pages 172–177, Seoul, Korea, November 1997. Korea University, IEEE Computer Society.

[15] M. M. Murshed and R. P. Brent. Constant time algorithms for computing the contour of maximal elements on a reconfigurable mesh. *Parallel Processing Letters*, 8:351–361, 1998.

[16] J. Rothstein. Bus automata, brains, and mental models. *IEEE Trans. Syst. Man Cybern*, 18:522–531, 1988.

[17] J. D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, Rockville, Maryland, 1984.

[18] R. Vaidyanathan and J. L. Trahan. Optimal simulation of multidimensional reconfigurable meshes by two-dimensional reconfigurable meshes. *Information Processing Letters*, 47:267–273, 1993.

[19] B.-F. Wang and G.-H. Chen. Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems. *IEEE Transactions on Parallel and Distributed Systems*, 1:500–507, 1990.

[20] C. C. Weems et al. The image understanding architecture. *Internat. J. of Comput. Vision*, 2:251–282, 1989.

[21] F. F. Yao. On finding the maximal elements in a set of plane vectors. Technical report, Comput. Sci. Dep. Rep., U. of Illinois at Urbana-Champaign, 1974.