Constant-Time Algorithms for Interval Graph Problems on Reconfigurable Meshes
(Extended Abstract)

Yoojin Chung Kunsoo Park Yookun Cho
Department of Computer Engineering
Seoul National University

In this paper, we present $O(1)$ time algorithms to solve two problems in interval graphs on the reconfigurable mesh. The two problems are the domatic partition problem and the minimum coloring problem in interval graphs. These problems have not been solved in $O(1)$ time before, even on the idealistic CRCW PRAM model. These two algorithms are designed on a two-dimensional $2n \times 2n$ reconfigurable mesh, where n is the number of vertices (intervals) in an interval graph. The reconfigurable mesh consists of an array of processors and a reconfigurable bus system in a grid shape, and it is a practical model of parallel computation.

Constant-Time Algorithms for Interval Graph Problems
on Reconfigurable Meshes
(Extended Abstract)

本文では，インターバルグラフにおける domatic partition problem と彩色問題の２つの問題を可変構造メッシュ上で並列に解く定数時間のアルゴリズムを与える．これらの問題を解く定数時間のアルゴリズムは理想的な CRCW-PRAM を用いたものでさえも今までは得られていない．本文では２つのアルゴリズムを２次元の $2n \times 2n$ の可変構造メッシュ上で設計する．ここで，n はインターバルグラフの頂点数である．可変構造メッシュはプロセッサの配列と格子状の可変構造のバスからなり，並列計算の実際的なモデルである．

1 Introduction

The reconfigurable mesh (RMESH for short) consists of an array of processors connected to a reconfigurable bus system [15]. (See Figure 1.) Though the buses outside the processors are fixed, the internal connection between the I/O ports of each processor can be reconfigured by individual processors during the execution of algorithms. Efficient algorithms have been developed on the reconfigurable mesh [9, 14, 15, 18].

RMESH can be superior to the PRAM (parallel random access machine) model for some computations. For example, Furst *et al.* [5] have shown that the exclusive OR function of n boolean values cannot be computed in $O(1)$ time on a PRAM using a polynomial number of processors. Nevertheless, the exclusive OR function can be computed in $O(1)$ time on the 2-D RMESH with n^2 processors [14].

An interval graph $G = (V, E)$ is a graph defined by a collection of intervals on the real line, as follows [6]. Given a set I of n intervals, the corresponding interval graph has one vertex corresponding to each interval in I and two vertices are adjacent whenever the corresponding intervals have at least one point in common. The set I is called the interval representation of graph G. If no interval in I is properly contained in some other interval in I, G is called a proper interval graph.

Interval graphs have been studied extensively and have revealed their practical relevance for modeling problems arising in the real world. In particular, they have found applications in archaeology, genetics, ecology, psychology, traffic control, computer scheduling, storage information retrieval, and electronic circuit design [6, 17].

One of the interesting features of interval graphs is that many of the problems that are NP-complete for general graphs are solvable in polynomial time for interval graphs. Examples include the problems of computing maximal cliques, coloring, and computing maximum independent sets [1, 6, 10]. Most of these problems are solvable for interval graphs in linear time or $O(n \log n)$ time on a sequential machine.

In this paper, we present $O(1)$ time algorithms to solve two problems on interval graphs on an RMESH with $O(n^2)$ processors. The two problems are the domatic partition problem and the minimum coloring problem in interval graphs. These problems have not been solved in $O(1)$ time before, even on the idealistic CRCW PRAM model.

A *dominating set* of a graph G is a subset S of the vertices such that every vertex of G is either in S or adjacent to some vertex in S. The *domatic number* of G is the size of a maximum cardinality partition of the vertices into dominating sets and the *domatic partition problem* is to find such a partition. The domatic partition problem has an obvious application in the optimum location of facilities in a network. Determining the domatic number of an arbitrary graph is NP-complete [6]. However polynomial-time algorithms exist for interval graphs. Bertossi [2] proposed an $O(n^{2.5})$ time algorithm to determine the domatic number of interval graphs and an $O(n \log n)$ time algorithm for proper interval graphs. Linear-time algorithms for determining the domatic number of interval graphs were presented by Lu, Ho, and Chang [12] and by Rao and Rangan [16]. For the domatic partition problem on interval graphs, an $O(n)$ time algorithm was developed for the sorted case by Manacher and Mankus [13]. An $O(\log n)$ time parallel algorithm on the EREW PRAM with $O(n)$ processors to solve the domatic partition problem on interval graphs was proposed by Yu and Yang [20]. In this paper, we present a constant-time algorithm for finding a domatic partition of interval graphs on a $2n \times 2n$ RMESH.

Given a set of n intervals, the *minimum coloring problem* is to assign a color to each interval such that the overlapping intervals have distinct colors. The minimum coloring problem on intervals is also called the channel assignment problem [4, 7], which has important applications in computer-aided design of integrated circuits [3, 4, 7, 8, 19]. An optimal $O(n \log n)$ time algorithm was presented by Gupta, Lee, and Leung [7]. An $O(\log n)$ time parallel algorithm on the EREW PRAM with $O(n^2)$ processors was developed by Dekel and Sahni [4]. Yu, Chen, and Lee [19] improved the

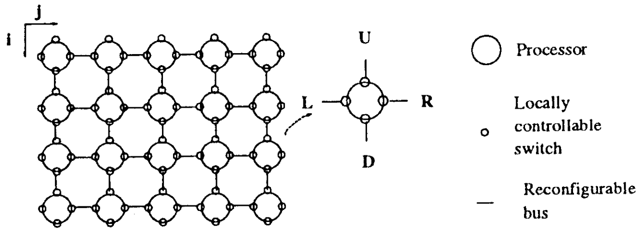

Figure 1: The reconfigurable mesh architecture

result with time complexity $O(\log n)$ and processor complexity $O(n)$ on the EREW PRAM model. The constant-time parallel algorithm on an RMESH with $O(n^2)$ processors was proposed recently by Lin [11]. In this paper, we present a constant-time algorithm to solve the minimum coloring problem on a set of intervals on a $2n \times 2n$ RMESH. Though our algorithm has the same time and processor complexities as Lin's algorithm [11], our approach is different from Lin's.

2 Reconfigurable Mesh

An $n \times n$ RMESH consists of an $n \times n$ array of processors connected to a grid-shaped reconfigurable bus system, where each processor has four locally controllable bus switches, as shown in Figure 1. Within each processor, four ports, denoted by L, R, U, and D (stand for left, right, up, and down), are provided for dynamically adjusting the local connections: These ports are able to realize any set, $A = \{A_1, A_2\}$, of connections where $A_i \subseteq \{L, R, U, D\}$, $1 \leq i \leq 2$ and the A_i's are disjoint. The processors are connected to the reconfigurable bus system through these ports. By setting the local connections properly, processors that are attached to the same subbus can communicate with one another by broadcasting values on the common subbus. At any given time, at most one processor can use the subbus to broadcast a value. When no local connection among ports is set within each processor, an RMESH is functionally equivalent to a mesh-connected computer. Each processor is identified by a unique index (i, j), $1 \leq i, j \leq n$, and the processor with index (i, j) is denoted by $P[i, j]$. In one unit of time each processor can perform basic arithmetic/logic operations on its own data, can connect or disconnect its local connections among ports , and can send (broadcast) or receive a piece of data through port L, R, U or D.

3 Constant Time Domatic Partition Algorithm

For a graph $G = (V, E)$, a vertex $v_1 \in V$ *dominates* a vertex $v_2 \in V$ if edge $\{v_1, v_2\} \in E$. A set $S \subseteq V$ *dominates* a set $S' \subseteq V$ if every vertex in $S' - S$ is dominated by at least one vertex in S. A set of vertices D is a *dominating set* of a graph $G = (V, E)$ if every vertex in $V - D$ is dominated by at least one vertex in D. Let $GP = \{P \mid P$ is a partition of V and every set in P dominates $V\}$. The *domatic number* of graph G is given by $dm(G) = max\{size(P) \mid P \in GP\}$, where $size(P)$ denotes the number of sets in P. A *domatic partition* of graph G is a partition in GP whose size equals $dm(G)$ and the *domatic partition problem* is to find a domatic partition of graph G.

Theorem 1 [12] *The domatic number of an interval graph G equals $\delta(G) + 1$, where $\delta(G)$ is the minimum degree of graph G.*

An interval i is represented in the form of $(i.left, i.right)$: $i.left$ and $i.right$ are the left and right end-points of interval i, respectively. Without loss of generality we assume that the $2n$ end-points of n intervals are distinct. We label the intervals in the increasing order of left end-points. For each interval $i \in I$, we call $Adj[i]$ the *adjacency set* of interval i, which consists of all intervals that overlap with i. Let

$N(i) = \{i\} \cup Adj(i)$, which is called the *closed neighborhood* of i.

Let $d(i)$ denote the degree of interval i (i.e., the degree of vertex i of graph G). Assume that ORDER(i) is the order number of interval i in the increasing order of right end-points. Also for $i = 1, ..., n$, where n is the number of intervals, we define

$$\text{LINK}(i) = \begin{cases} min\{d(j) \mid j \in G \text{ such that} \\ i.right < j.left\} + \text{ORDER}(i) \\ +1 \quad \text{if such interval } j \text{ exists} \\ n+1 \quad \text{otherwise} \end{cases}$$

¿From now on, let MIN(i) denote $min\{d(j) \mid j \in G$ such that $i.right < j.left\}$. So LINK(i) is MIN(i)+ORDER(i)+1 if MIN(i) exists, and $n+1$ otherwise.

For $i = 1, ..., n$, we link interval i to interval LINK(i) if LINK(i) $\neq n+1$; otherwise we link interval i to null. In an interval graph, interval t which has the largest right end-point and does not overlap with interval n has LINK(t) $= n$ because MIN(t) $= d(n)$ and ORDER(t) $= n - |N(n)|$. All intervals in $N(n)$ link to null and every interval except those in $N(n)$ links to an existing interval.

We now show how to find dominating sets from the linking lists constructed as above. Since MIN(i) is nondecreasing as ORDER(i) increases, LINK(i) is strictly increasing and thus every interval is linked to by at most one interval. If LINK(i) $= j$, where $i, j \in I$ and $i \leq j$, then each interval in $\{i, i+1, ..., j\}$ is dominated by interval i or interval j as follows: ¿From the definition of LINK, $j =$ ORDER(i)+ MIN(i) + 1. Suppose that there exists interval k, $i < k < j$, which is not dominated by i or j. Since k is not dominated by i or j, we have $i.right < k.left < k.right < j.left$. Now we compute the degree $d(k)$ of interval k. Since there are $j - 1$ (=ORDER(i)+MIN(i)) intervals whose left end-points are less than point $j.left$ and there are ORDER(i) intervals whose right end-points are less than or equal to point $i.right$, $d(k)$ is at most MIN(i) $- 1$, which contradicts the definition of MIN(i). Therefore, the intervals in every linking list dominate all intervals from the first interval in the list to interval n. Since LINK(1) = ORDER(1)+ MIN(1) + 1 $\geq \delta(G) + 2$ is the minimum LINK value, no interval links to the intervals 1, 2, ..., $\delta(G) + 1$. Also these intervals are in $N(1)$ because $|N(1)| \geq \delta(G) + 1$ and the intervals are labeled in the increasing order of left end-points. Therefore, the $\delta(G) + 1$ linking lists which begin with the first $\delta(G) + 1$ intervals with the smallest left end-points (i.e., intervals 1, 2, ..., $\delta(G) + 1$) are dominating sets. For $1 \leq i \leq \delta(G) + 1$, give mark i to all intervals in the linking list beginning with interval i. This means that these intervals belong to dominating set P_i. Put all unmarked intervals into dominating set P_1. The details of correctness proof of this algorithm are given in [20].

Now we define several data manipulation algorithms for an RMESH. These are used to develop the algorithms for the domatic partition problem and the minimum coloring problem.

Lemma 1 [9] *The sorting of n values can be performed in $O(1)$ time on an $n \times n$ RMESH. Initially, each processor of the first row owns a value. Sorted values are stored in the first row of an $n \times n$ RMESH in order.*

Lemma 2 [18] *The prefix sum of a binary sequence of size n can be computed in $O(1)$ time on an $n \times n$ RMESH.*

For n data items $x[1], x[2], ..., x[n]$, let $smin$ [1], $smin$[2], ..., $smin[n]$ be the suffix minima of $x[1], x[2], ..., x[n]$, i.e., $smin[j]$ is the minimum of $x[j], x[j+1], ..., x[n]$.

Lemma 3 *The suffix minima of n values $x[1]$, $x[2]$, ..., $x[n]$ can be found in $O(1)$ time on an $n \times n$ RMESH. Initially, each processor $P[1, j]$, $1 \leq j \leq n$, of the first row owns $x[j]$ and the computed suffix minimum $smin[j]$ is stored in processor $P[1, j]$.*

Now we describe a constant-time algorithm for the domatic partition problem with n intervals below. By preprocessing, we label the n intervals in their increasing order of left end-points in $O(1)$ time on an $n \times n$ RMESH by Lemma 1.

Algorithm Rmesh_Domatic_Partition

Input: n intervals $1, 2, ..., n$ represented in the form of $(i.left, i.right)$. Initially, $i.left$ and $i.right$ of interval i are stored at $Lcoord[i]$ and $Rcoord[i]$ in processor $P[i,1]$, $1 \leq i \leq n$, respectively.

Output: $doma[i]$ (the dominating set number assigned to interval i) in processor $P[i,1]$, $1 \leq i \leq n$.

• Step 1: Sort the $2n$ end-points of n intervals in $O(1)$ time as below.

– Substep 1-1: Assign the $2n$ end-points of n intervals in the n processors of the first column to the $2n$ processors of the first row so that each processor $P[1,j]$, $1 \leq j \leq n$, has the left end-point of interval j and each processor $P[1, j+n]$, $1 \leq j \leq n$, has the right end-point of interval j. This can be computed in $O(1)$ time as follows.

Each processor $P[1,j]$, $1 \leq j \leq 2n$, has three variables: $interval[j]$, $side[j]$, and $coord$ $[j]$. Each processor $P[1,j]$, $1 \leq j \leq n$, sets $side[j] = L$ and $interval[j] = j$ and each processor $P[1,\ n+j]$ sets $side[n+j] = R$ and $interval[n+j] = j$. Each processor $P[i,1]$, $1 \leq i \leq n$, sends $Lcoord[i]$ to $coord[i]$ in processor $P[1,i]$ as follows: After each processor $P[i,1]$, $1 \leq i \leq n$, sends $Lcoord[i]$ to processor $P[i,i]$ using row broadcasting, each processor $P[i,i]$ sends it to $coord[i]$ in processor $P[1,i]$ using column broadcasting. Similarly, each processor $P[i,1]$, $1 \leq i \leq n$, sends $Rcoord[i]$ to $coord[i+n]$ in processor $P[1, i+n]$.

– Substep 1-2: Sort the $2n$ triples ($interval[j]$, $side[j]$, and $coord[j]$), $1 \leq j \leq 2n$, according to $coord[j]$ in the increasing order. By Lemma 1, this can be computed in $O(1)$ time on a $2n \times 2n$ RMESH.

Let NUM1(i) be the number of intervals whose left end-point is less than the right end-point of interval i and let NUM2(i) be the number of intervals whose right end-point is less than the left end-point of interval i. Then $d(i)$ =NUM1(i)− NUM2(i) − 1, $1 \leq i \leq n$.

• Step 2: Calculate the degrees using prefix sum in $O(1)$ time as below.

Each processor $P[1,j]$, $1 \leq j \leq 2n$, has three variables: $num1[j]$, $num2[j]$, and $deg[j]$, where $num1[j]$ is NUM1($interval[j]$), $num2[j]$ is NUM2($interval[j]$), and $deg[j]$ is $d(interval$ $[j])$.

– Substep 2-1: Each processor $P[1,j]$, $1 \leq j \leq 2n$, sets 1 if $side[j] = L$ and sets 0 otherwise, and calculates its prefix sum in $O(1)$ time on a $2n \times 2n$ RMESH by Lemma 2. Then each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, saves this prefix sum value to $num1[j]$.

– Substep 2-2: Each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, sends $num1[j]$ to $num1[k]$ in processor $P[1,k]$ with $side[k] = L$ and $interval[k] = interval[j]$, $1 \leq k \leq 2n$, as follows.

Each processor $P[i,j]$, $1 \leq i \leq n$ and $1 \leq j \leq 2n$, establishes the local connection {L, R, U, D} if $i = interval[j]$ and the local connections {L, R}, {U, D} otherwise. Each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, broadcasts $num1[j]$ through port D and each processor $P[1,k]$ with $side[k] = L$, $1 \leq k \leq 2n$, receives data item $num1[k]$ from port U.

– Substep 2-3: Each processor $P[1,j]$, $1 \leq j \leq 2n$, sets 1 if $side[j] = R$ and sets 0 otherwise, and calculates its prefix sum in $O(1)$ time on a $2n \times 2n$ RMESH by Lemma 2. Then each processor $P[1,j]$ with $side[j] = L$, $1 \leq j \leq 2n$, saves this prefix sum value to $num2[j]$.

– Substep 2-4: Each processor $P[1,j]$ with $side[j] = L$, $1 \leq j \leq 2n$, calculates $deg[j] = num1[j] - num2[j] - 1$, and each processor $P[1,j]$ with $side[j] = R$ sets $deg[j]$ to n.

• Step 3: Calculate LINK using degrees in $O(1)$ time.

– Substep 3-1: Each processor $P[1,j]$, $1 \leq j \leq 2n$, calculates the suffix minima of $deg[j]$ (i.e., $min\{deg[i] \mid j \leq i \leq 2n\}$) in $O(1)$ time on a $2n \times 2n$ RMESH by Lemma 3 and each processor $P[1,j]$ with $side[j] = R$ saves the result to $min[j]$. Note that $min[j]$ in processor $P[1,j]$ with $side[j] = R$ equals MIN($interval[j]$) $= min\{d(interval[k]) \mid interval[k] \in G$ such that $interval[j].right < interval[k].left\}$.

– Substep 3-2: Each processor $P[1,j]$, $1 \leq j \leq 2n$, has variables $order[j]$ and $link[j]$, where $order[j]$ is ORDER($interval[j]$) and $link[j]$ is LINK($interval[j]$).

Each processor $P[1,j]$, $1 \leq j \leq 2n$, sets 1 if $side[j] = R$ and sets 0 otherwise, and calculates its prefix sum in $O(1)$ time on a $2n \times 2n$ RMESH by Lemma 2. Each processor $P[1,j]$ with $side[j] = R$ saves this prefix sum value to $order[j]$. Each processor $P[1,j]$

with $side[j] = R$, $1 \leq j \leq 2n$, sets $link[j] = order[j] + min[j] + 1$ if $min[j] < n$.

• Step 4: Make linking lists using LINK and find the dominating sets in $O(1)$ time as below.

Each processor $P[i,1]$, $1 \leq i \leq n$, has variable $doma[i]$ which is the dominating set number assigned to interval i.

– Substep 4-1: Find the first element of each dominating set.

To put all unmarked intervals into dominating set P_1, each processor $P[i,1]$, $1 \leq i \leq n$, sets $doma[i] = 1$ as initial value. Each processor $P[i,1]$, $1 \leq i \leq n$, establishes the local connection {U, D} and processor $P[1,1]$ broadcasts $min[1]$ ($min[1]$ calculated in Substep 3-1 is the minimum degree of graph G) through port D. Then, each processor $P[i,1]$, $1 \leq i \leq n$, receives $min[1]$ and sets $doma[i] = i$ if $i \leq min[1] + 1$.

– Substep 4-2: Find the other elements of each dominating set.

Each processor $P[1,j]$, $1 \leq j \leq 2n$, broadcasts $side[j]$, $interval[j]$, and $link[j]$ to all processors $P[i,j]$, $1 \leq i \leq n$, by column broadcasting. Each processor $P[i,j]$, $1 \leq i \leq n$ and $1 \leq j \leq 2n$, establishes the local connection {L, R, U, D} if ($i = interval[j]$ and $side[j] = R$) or ($i = link[j]$ and $side[j] = R$), and the local connections {L, R}, {U, D} otherwise. Then each processor $P[i,1]$, $1 \leq i \leq n$, broadcasts $doma[i]$ through port R if $i \leq min[1]+1$. Each processor $P[i,1]$, $1 \leq i \leq n$, receives the data item $doma[i]$ from port R.

Since each step in the above algorithm takes $O(1)$ time, the overall time complexity is $O(1)$. Therefore, we obtain one of the main results of this paper as follows.

Theorem 2 *The domatic partition problem in interval graphs with n intervals can be solved in $O(1)$ time on a $2n \times 2n$ RMESH.*

4 Constant Time Minimum Coloring Algorithm

We use the same notations, assumptions, and preprocessings as in Section 3. So intervals $1, ..., n$ are labeled in the increasing order of left end-points.

Two intervals are *independent* (or nonoverlapping) if their intersection is empty. A set of intervals are independent if they are pairwise independent. A partition $\{P_1, P_2, ..., P_k\}$ of an interval set $\{1, ..., n\}$ is a *coloring* if every P_i, $1 \leq i \leq k$, is an independent set. The *minimum coloring problem* is to find a coloring with the minimum number of sets in it.

For a set of intervals, a subset of it forms a *clique* if each pair of intervals in this subset has a nonempty intersection. A clique with the maximum number of elements is called a *maximum clique*. For a set of intervals I, let $\omega(I)$ be the number of intervals in a maximum clique of I and let $\chi(I)$ be the minimum number of colors needed to color intervals of I. Since an interval graph is a perfect graph, we have $\omega(I) = \chi(I)$ for a set of intervals I [6].

To solve the minimum coloring problem, we divide the available colors into two types: colors that have been used (queue OLD) and colors that have not been used (stack NEW). NEW stores the colors which have not been assigned to intervals and OLD stores the colors which had been assigned to some intervals but are now available again. Initially, NEW contains the colors $\{1, 2, ..., \omega(I)\}$ and OLD is empty. Scan the end-points in the increasing order as follows: When scanning the left end-point of an interval, assign a color to the interval by getting a color from OLD if OLD is not empty and getting a color from NEW otherwise, and when scanning the right end-point of an interval, return its color to queue OLD. The correctness of this scheme was proved in [19].

For each interval i, let NEXT(i) be the next interval that receives the same color as i after the above approach is completed. To solve the minimum coloring problem on an R-MESH, first we compute NEXT(i) for each interval i, $1 \leq i \leq n$. Then we assign a suitable color to the first node of each list, and propagate this color to the other nodes of each list.

Now we describe a constant-time algorithm for the minimum coloring problem with n intervals below.

Algorithm Rmesh_Minimum_Coloring

The same input in the Rmesh_Domatic_Partition algorithm.

Output: $color[i]$ (the color assigned to interval i) in processor $P[i,1]$, $1 \leq i \leq n$.

• Step 1: Sort the $2n$ end-points of n intervals as in Step 1 in the Rmesh_Domatic_Partition algorithm. As a result, each processor $P[1,j]$, $1 \leq j \leq 2n$, has variables $interval[j]$, $side[j]$, and $coord[j]$ in the increasing order of $coord[j]$.
• Step 2: For each interval i, $1 \leq i \leq n$, find NEXT(i). For this, each processor $P[1,j]$, $1 \leq j \leq 2n$, has variable $used.color[j]$ and each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, has variables $total.color[j]$, $shift[j]$, $local.next[j]$, and $global.next[j]$.
– Substep 2-1: $used.color[j] + 1$ is the number of colors currently used when scanning $coord[j]$ with $side[j] = R$.

Each processor $P[i,j]$ with $side[j] = L$, $1 \leq j \leq 2n$, sets $used.color[j] = 0$. Each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, sets $used.color[j] = \Sigma_{1 \leq t \leq j} k[t]$, where $k[t] = 1$ if $side[t] = L$ and $k[t] = -1$ if $side[t] = R$, which is the number of intervals that intersect $coord[j]$. This can be computed in $O(1)$ time as follows: Each processor $P[i,j]$, $1 \leq i \leq n+1$ and $1 \leq j \leq 2n$, establishes the local connections {L, U}, {R, D} if $side[j] = L$ and the local connections {L, D}, {R, U} if $side[j] = R$. Then processor $P[n+1,1]$ broadcasts 1 through port L on this configuration. Each processor $P[i,j]$ with $side[j] = R$, $1 \leq i \leq n+1$ and $1 \leq j \leq 2n$, which receives 1 from port R, sends $(n+1) - i$ to $used.color[j]$ in processor $P[1,j]$ using column broadcasting.
– Substep 2-2: $total.color[j]$ with $side[j] = R$ is $max_{1 \leq k \leq j}(used.color[k])$. $total.color[j] + 1$ is the total number of colors used till now when scanning $coord[j]$ with $side[j] = R$.

Each processor $P[1,j]$, $1 \leq j \leq 2n$, calculates the prefix maximum of $used.color[j]$, $1 \leq j \leq 2n$, in $O(1)$ time on a $2n \times 2n$ RMESH as in Lemma 3, and each processor $P[1,j]$ with $side[j] = R$ saves it to $total.color[j]$.
– Substep 2-3: $shift[j]$ with $side[j] = R$ is $total.color[j] - used.color[j]$. $shift[j]$ is the number of available colors in queue OLD when scanning $coord[j]$ with $side[j] = R$.
– Substep 2-4: $local.next[j]$ with $side[j] = R$ is the smallest interval k with $k.left > coord[j]$. As all $coord[k]$, $1 \leq k \leq 2n$, are sorted in Step 1, and intervals are labeled in the increasing order of left end-points, $local.next[j]$ is $interval[t]$, where t is the smallest one with $t > j$ and $side[t] = L$. This can be computed in $O(1)$ time as follows.

Each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, establishes the local connection {L, R}. Then each processor $P[1,j]$ with $side[j] = L$, $1 \leq j \leq 2n$, broadcasts $interval[j]$ through port L. Each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, receives a data item $local.next[j]$ from port R.
– Substep 2-5: $global.next[j]$ with $side[j] = R$ is NEXT($interval[j]$). If $shift[j]$ (the number of available colors) is 0, the color used for $interval[j]$ which becomes available at $coord[j]$ will be used for the next interval $local.next[j]$, i.e., $global.next[j] = local.next[j]$. If $k = shift[j] > 0$, the k colors available will be used for the first k intervals starting after $coord[j]$ and thus the color used for $interval[j]$ will be used for the $(k+1)$st interval, which is $local.next[j] + shift[j]$. Therefore, $global.next[j] = local.next[j] + shift[j]$.

Each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, sets $global.next[j] = local.next[j] + shift[j]$ if $local.next[j]$ exists.
• Step 3: Color the intervals. This step can be computed in $O(1)$ time as below.
• Subtep 3-1: Find and color the first node of each list (coloring from stack NEW).

Each processor $P[i,1]$, $1 \leq i \leq n$, sets $mark[i] = 1$ as initial value. Each processor $P[i,j]$, $1 \leq i \leq n$ and $1 \leq j \leq 2n$, establishes the local connection {L, R, U, D} if ($i = global.next[j]$ and $side[j] = R$) and the local connections {L, R}, {U, D} otherwise. Then each processor $P[1,j]$ with $side[j] = R$, $1 \leq j \leq 2n$, broadcasts $global.next[j]$ through port D if $global.next[j]$ exists. Each processor $P[i,1]$, $1 \leq i \leq n$, which receives i from port R, sets $mark[i] = 0$.

Each processor $P[i,1]$, $1 \leq i \leq n$, calculates the prefix sum of $mark[i]$ in $O(1)$ time on an $n \times n$ RMESH by Lemma 2. Each processor $P[i,1]$ with $mark[i] = 1$, $1 \leq i \leq n$, saves this prefix sum value to $color[i]$.
• Subtep 3-2: Color the other nodes (coloring from queue OLD).

Each processor $P[i,j]$, $1 \leq i \leq n$ and $1 \leq j \leq 2n$, establishes the local connection {L, R, U, D} if ($i = interval[j]$ and $side[j] = R$) or ($i = global.next[j]$ and $side[j] = R$), and the local connections {L, R}, {U, D} otherwise. Then each processor $P[i,1]$ with $mark[i] = 1$, $1 \leq i \leq n$, broadcasts the color value $color[i]$ through port R. Each processor $P[i,1]$, $1 \leq i \leq n$, receives the data item $color[i]$ from port R.

Since each step in the above algorithm takes $O(1)$ time, the overall time complexity is $O(1)$. Therefore, we obtain one of the main results of this paper as follows.

Theorem 3 *The minimum coloring problem in interval graphs with n intervals can be solved in $O(1)$ time on a $2n \times 2n$ RMESH.*

References

[1] A. Aboelfotoh and C. Colbourn: "Efficient algorithms for computing the reliability of permutation and interval graphs,", *Networks*, vol. 20, pp. 883–898, 1990.

[2] A.A. Bertossi: "On the domatic number of interval graph,", *Information Processing Letters*, vol. 28, pp. 275–280, 1988.

[3] K. Chaudhary and P. Robinson: "Channel routing by sorting,", *IEEE Transactions on Computer-Aided Design*, vol. 10, no. 6, pp. 754–760, 1991.

[4] E. Dekel and S. Sahni: "Parallel scheduling algorithms,", *Operations Research*, vol. 31, no. 1, pp. 22–49, 1983.

[5] M. Frust, J. Saxe, and M. Sipser: "Parity, circuits and polynomial time hierarchy,", *Proc. of IEEE Foundations of Computer Science*, pp. 260–270, 1981.

[6] M.C. Golumbic: *Graph theory and perfect graphs*, Academic Press, New York, 1980.

[7] U.I. Gupta, D.T. Lee, and J.Y.-T. Leung: "An optimal solution for the channel assignment problem,", *IEEE Transactions on Computers*, vol. C-28, no. 11, pp. 807–810, 1979.

[8] A. Hashimoto and J. Stevens: "Wire routing by optimizing channel assignment within large apertures,", *Proc. of 8th Annual Design Automation Workshop*, pp. 55–169, 1981.

[9] J. Jang and V. Prasanna: "An optimal sorting algorithm on reconfigurable meshes,", *Proc. of International Conference on Parallel Processing*, pp. 130–132, 1992. Also *Technical Report*, IRIS 277, UCS, 1991.

[10] P.N. Klein: "Efficient parallel algorithms for chordal graphs,", *Proc. of Foundations of Computer Science*, pp. 150–161, 1988.

[11] S.S. Lin: "Constant-time algorithms for the channel assignment problem on processor arrays with reconfigurable bus systems,", *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems*, vol. 13, no. 7, pp. 884–890, 1994.

[12] T.L. Lu, P.H. Ho, and G.J. Chang: "The domatic number problem in interval graphs,", *SIAM Journal on Discrete Mathematics*, vol. 3, no. 4, pp. 531–536, 1990.

[13] G.K. Manacher and T.A. Mankus: "Determining the domatic number and a domatic partition of an interval graph in time O(n) given its sorted model,", Submitted *SIAM Journal on Discrete Mathematics*, 1991.

[14] R. Miller, V.K. Prasanna Kumar, D. Reisis, and Q. Stout: "Data movement operations and applications on reconfigurable VLSI arrays,", *Proc. of International Conference on Parallel processing*, pp. 205–208, 1988.

[15] R. Miller, V.K. Prasanna Kumar, D. Resis, and Q. Stout: "Meshes with reconfigurable buses,", *Proc. of 5th MIT Conference on Advanced Research in VLSI*, pp. 163–178, 1988.

[16] A.S. Rao and C.P. Rangan: "Linear algorithm for domatic number problem on interval graphs,", *Information Processing Letters*, vol. 33, pp. 29–33, 1989/90.

[17] F.S. Robert: *Graph theory and its application to problems of society*, SIAM, Philadelphia, 1978.

[18] B. Wang, G. Chen, and H. Lin: "Configurational computation: a new computation method on processor arrays with reconfigurable bus systems,", *Proc. of International Conference on Parallel processing*, vol. 3, pp. 42–49, 1991.

[19] M.S. Yu, C.L. Chen, and R.C.T. Lee: "An optimal parallel algorithm for minimum coloring of intervals,", *Proc. of International Conference on Parallel Processing*, vol. 3, pp. 162–168, 1990.

[20] M.S. Yu and C.H. Yang: "An optimal parallel algorithm for the domatic partition problem on interval graphs,", *Proc. of International Conference on Parallel Processing*, vol. 3, pp. 146–150, 1992.